CS148 Homework 1

Homework Due: **July 2nd**. There is live grading for this homework assignment. Please complete this handout, prepare for the live grading assessment, and also submit on **Gradescope**.

Assignment by Sarah Jobalia, Kevin Li, and Ron Fedkiw, with modifications by Kate Baker.

0.1 Assignment Format

This assignment has **4** `TODO` s of varying lengths, each with their own instructional tutorials. The TODOs are covered within the first 6 pages of the document. The rest of the handout consists of the instructional tutorials. We recommend reading through the TODOs first, then the instructional tutorials for more context.

You will need to download these source files (NOTE this is a HYPERLINK, and there will be many hyperlinks throughout the document!) for the homework. Please save and unzip the files in some course directory `$CS148_DIR` on your machine.

0.2 Collaboration Policy

You may work with one other partner and attend the grading session together as one group. **Both partners must attend the grading session though, whether it be together or individually, as each partner will be asked a different quiz question.** Discussion of the homework and quiz questions across groups on Ed, etc is allowed as long as the answers (i.e. any **code**, numbers, screenshots, explanations, etc) are not explicitly posted or given out publicly. **Sharing code between partners is allowed.**

If you need to make a private post that includes your code to the course staff, then please DO NOT post screenshots of your code; INSTEAD, always post code in PLAIN TEXT in case a CA or instructor wants to sanity check it on their own computer!

0.3 Office Hours

Office Hours start Week 2, and the logistics will be posted on the course website. There is a mix of both in-person and online office hours, though the online office hours will prioritize the remote students.

For in-person office hours, you simply need to show up to the location listed on the website at the scheduled time. The instructor(s) or CA(s) for those office hours will give further instruction at the location.

For online office hours, enter the Zoom call listed on the calendar within the scheduled time. You'll be placed in a waiting room within the Zoom call, and the instructor(s) or CA(s) will pull you into the main session when it is your turn. They'll be keeping track of students in first come first serve order.

0.4 Grading Session

Please see the website and lecture 1 slides for information about live grading times, Zoom links, and queues. This homework will have live grading held on **Thursday, Jul 2**.

If you can not attend the recurring time blocks, then make a post on Ed about your circumstances, and we will try to accommodate with a different time. The actual grading process should

take only 5 minutes, but depending on when you show up, there may be a line of other students. You can check the queue to see your status. Feel free to turn off your camera and mute and do other work until we call your name or you get notified that it's your turn in the queue in case there is a wait.

Quiz 1

The first quiz will be held in conjunction with the live grading. After you show your homework results, you will be randomly asked one of these questions:

- What are two reasons for why we often use triangle meshes to model our objects over other polygonal meshes?
- What does it mean to subdivide a mesh, and why do we need to move all the vertices after inserting new vertices each step?
- Why do we want to use cubic splines when editing a mesh as opposed to a simpler scheme like linear interpolation?
- What is the purpose of homogeneous coordinates for geometric transformations?
- What is the difference between applying a geometric transformation onto an object around its local axes vs the global axes?

The quiz will be graded on a 3 pt scale where 0 pt corresponds to not showing up, 1 pt to an inadequate answer, 2 pt to an incomplete answer, and 3 pt to a complete answer.

1 Assignment

1.1 TODO 1: Write an .obj file for a cube and import it into Blender.

In order to get an intuitive understanding of .obj files, you'll start by writing your own from scratch. For this, you'll place the vertices and specify which of those vertices are connected by faces.

1. Open the starter file `$CS148_DIR/HW1/myCube.obj` in any TEXT EDITOR you prefer.
2. Fill the file with all the necessary vertex and face lines to describe a cube with side length 1 and a vertex located at the origin (0, 0, 0). There are some example lines already in the .obj file – feel free to delete these as you see fit.
3. Open up a new scene in Blender and delete the default cube. Import your own cube .obj file and check to see that your cube has all its vertices and faces.

See section 3 for more details.

Show us: (1 pt) Your completed .obj file and the import process within Blender.

1.2 TODO 2: Write a Python script that generates a sphere in Blender.

This will be your first exercise in scripting in Blender as well as an exercise in geometric intuition. The goal is to generate a complete sphere object from scratch by computing all the points and faces that would make up the mesh.

In order to make a sphere out of a set of points, it is common to use a set of rings stacked on top of each other with radii that get smaller as you reach the top or bottom, eventually converging to a single top or bottom point.

We divide the surface into horizontal vertex rings called **stacks** (red) and vertical vertex arcs called **sectors** (blue) as shown in Figure 1. We choose the number of stacks and sectors when we tessellate a sphere this way. There should be two triangle faces (as shown in Figure 1) in every square of 4 vertices. (If you have another idea of how to break a sphere into vertices and faces, then feel free to try it! However, the walkthrough below may not be able to give as much help should you need to debug your own process.)

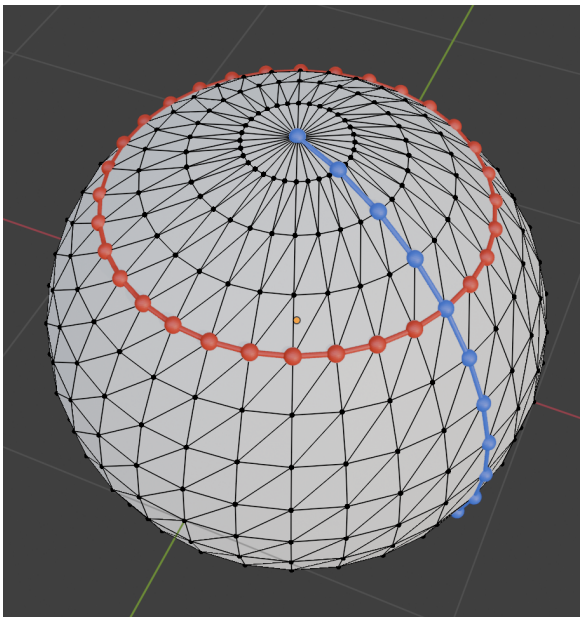
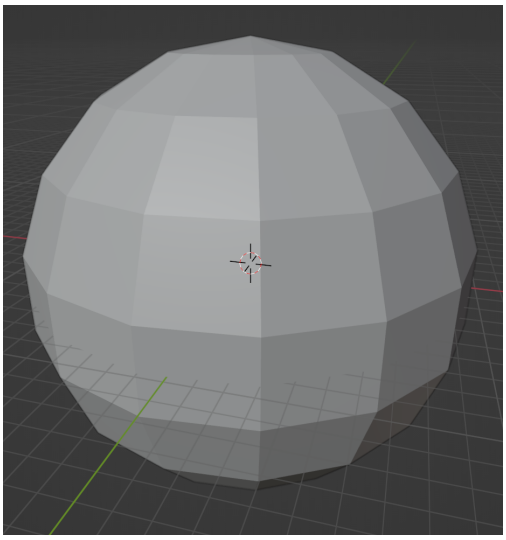


Figure 1: A visualization of how we divide a sphere into red stacks and blue sectors.

1. Open up the starter file `$CS148_DIR/HW1/mySphere_problem.blend` as described in Section 2. It should automatically open to the scripting tab with starter code already provided. Edit the code in this file so that it generates a complete sphere when run.

2. A correct implementation that keeps the default number of stacks and sectors as provided in the starter code will yield the crude-looking sphere below. You should continue to see this shape even as you rotate it in the Blender GUI. Feel free to increase the number of stacks and sectors if you want a smoother sphere.

   Tip: When debugging, you might want to lower the stack and sector count so that you have fewer vertices and faces to deal with. Be sure to also make use of print statements to the command line or terminal of your operating system!



See sections 2 and 4 for more details.

Show us: (1.5 pt) Your code running and generating a complete sphere in Blender.

1.3 TODO 3: Appropriately apply geometric transforms to an object.

In this section, you will be applying the concept of geometric transformations **(e.g. translating, rotating, and scaling points in a coordinate space)** as discussed in lecture. To see how these transformations interact, you will apply them to a simple cone in alternating orders.

1. Open up the starter file `$CS148_DIR/HW1/myTransformations.blend`. It should automatically open to the scripting tab with starter code already provided.

2. Complete the functions to return a correct translation matrix given a translation amount and axis to translate on, and a correct rotation matrix given a rotation in degrees and axis to rotate around.

3. Uncomment the lines of code at the end to produce all four of the following sets of transformations to the four cones:

   (a) rotate a cone about the x-axis by 45 deg, then rotate about the y-axis by 45 deg.

   (b) rotate a cone about the y-axis by 45 deg, then rotate about the x-axis by 45 deg.

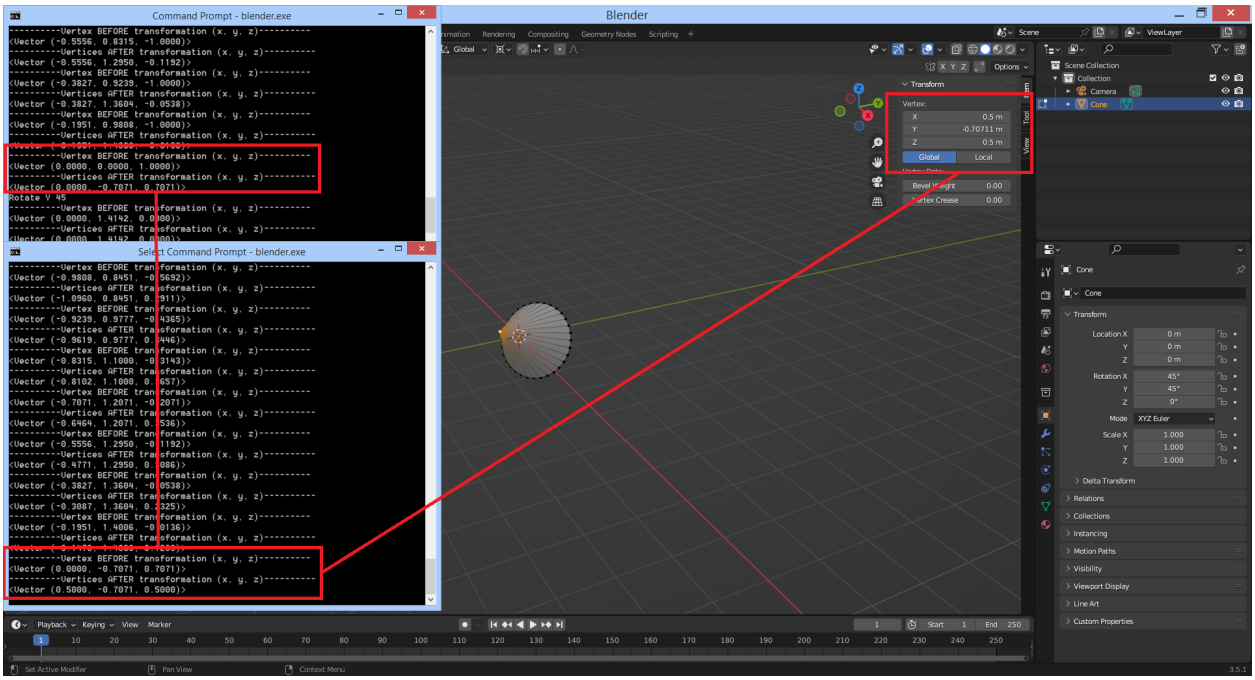
   (c) translate a cone along the x-axis by 10 units, then rotate about the y-axis by 45 deg.

   (d) rotate a cone about the y-axis by 45 deg, then translate along the x-axis by 10 units.

4. **NEXT, in a different instance of Blender**, create a new scene, delete the default cube, and instead add four cones `Add → Mesh → Cone` (or `Add → Cone` depending on version). Apply each of the above 4 pairs of transformations to a new cone **using the Blender hotkeys** for translation and rotation to recreate the final scene from Part 3.

   Tip: Use print statements as well as the Blender vertex view shown below to track the tip of the cone. The tip starts at (0,0,1), and you can manually compute where it should end up after each transformation process to sanity check your results. Make sure that you are also

rotating around the **global axes** in the GUI! You should NOT be getting the same results for any two sets of transformations.



See section 5 for more details.

Show us: (1.5 pt) All 4 cones in their respective configurations as a result of both your script and via performing the transformations via the Blender GUI. They should match!

1.4 TODO 4: Model a unique piece of geometry in Blender.

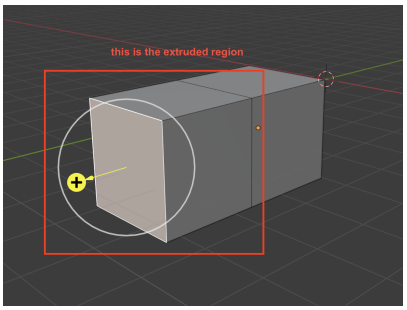


Figure 2: Extruding a cube in Edit Mode

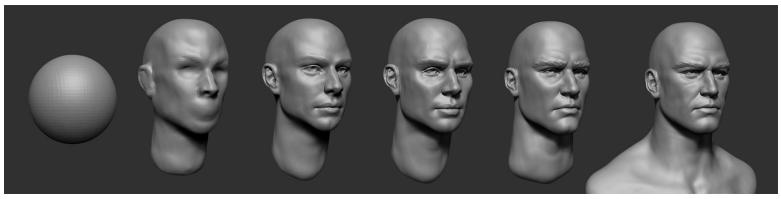


Figure 3: Creating a face in Sculpt Mode

For this part of the assignment, we want you to experiment with all the modeling tools that Blender has to offer. This document includes various tutorials on the matter in the later pages that you can reference. You don't have to try all of them – we just want you to pick 1 or 2 of the methods

that you see and think might be fun for you, and try it out by making your own unique piece of geometry.

This part of the homework is extremely open ended. We want you to try modeling an object you're interested in, whether it'd be something in real life or from a CGI movie, etc. This is a great place to start thinking about what objects you want for your final project, and what tools you will like the most to make them. Here are some common ways of editing geometry (you can choose to do 1, or a combination of them):

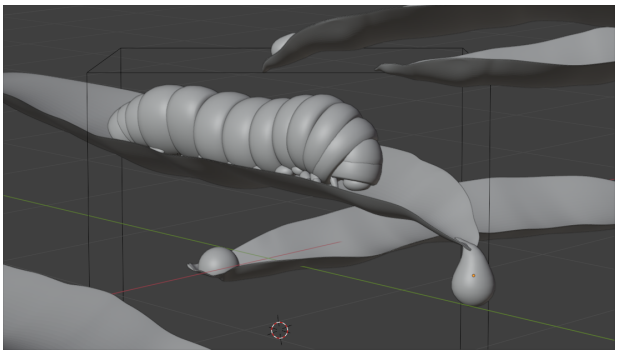
- Create a basic shape and then go into Edit Mode to move and add vertices, edges, and faces. If you like engineering, this might be for you, since you'll be placing new points, connecting them, and deciding where to move and manipulate faces. This can feel similar to a CAD tool.
- Create a basic shape and then go into Sculpt Mode to push and pull the mesh around, as if it's clay. If you like drawing/sculpting, this might be for you, since the mesh will automatically move to match your strokes.
- Create a basic shape and manipulate it with Geometry Nodes. If you like coding or math, this might be for you since you will can use a series of node components like code commands to change the shape and create something procedural.

We encourage people to try something that seems fun but is out of their comfort zone, because the best way to make things in Blender is through not just specializing in one of these techniques, but a combination of them. Getting comfortable with them all eventually will benefit you in the long run, especially as you work towards your final project.

See section 6 for more details.

Show us: (1 pt) A unique piece of geometry that you made in Blender. Be prepared to discuss the inspiration behind what you modeled and how you did your modeling. We're looking for geometry that is non-trivial (e.g. you cannot model a cube and turn that in) and has artistic merit. The most original models will be showcased in class!

As a motivating example, the caterpillar in this image from the Fall 22 showcase by Kate Eselius and Jamie Ullman was modeled entirely with a combination of basic spheres and torii! As this is the first assignment, you of course are not expected to assemble a model this complex so early. But we'd like to see you take some first steps in assembling something that may become even more complex down later in the quarter.



That covers all the TODOs! Everything else in this document is a mixture of review from lecture as well as instructional (Blender) tutorials for you to reference. For the last 2 TODOs especially, you'll probably want to take a look below.

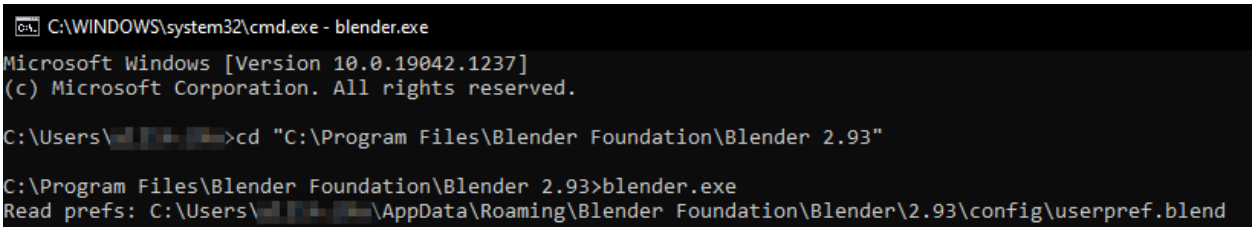
2 Starting Blender from the Command Line (for Debugging Scripts!)

When doing any sort of heavy-duty Python development in Blender, we want to launch Blender from the command line. Doing so will let Blender output any bugs or any information you explicitly print via the Python print() function to the command line.

- Windows: Press the **Windows** + **R** keys on your keyboard to open the "Run" box. Type **cmd** and hit enter to open the command prompt.

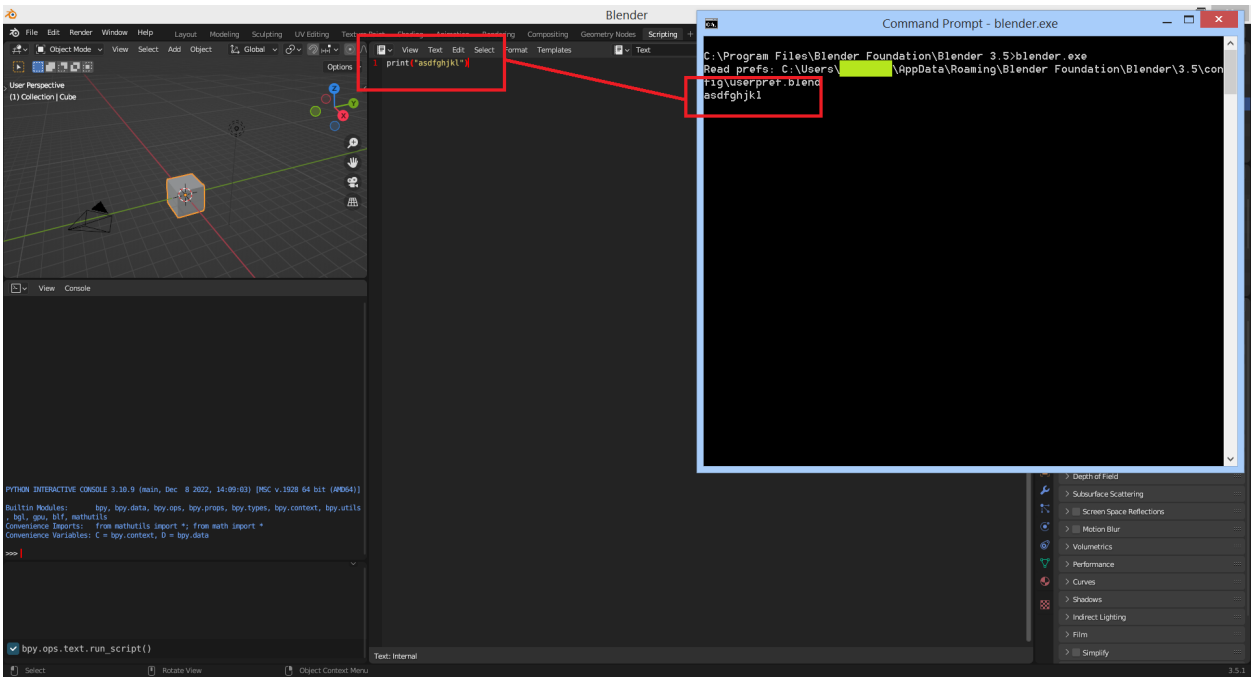
  cd to the Blender installation directory. The default location is **C:\Program Files\Blender Foundation\Blender 3.5**.

  In the Blender installation directory, type **blender.exe** to launch Blender. The following image shows the above steps in action launching a past version of Blender.



- Mac: Please refer to the official document linked here.
- Linux: Please refer to the official document linked here.

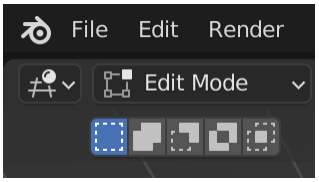
Blender will launch as usual. From there, you can open the .blend files that came with this homework from the menu bar, i.e. **File** → **Open**. Try adding **print()** statements to the scripts, like **print("asdfghjkl")**. You should see the print out pop up in your command line.



3 The OBJ file format

As discussed in lecture, one of the most commonly used file formats to store graphics data is the Wavefront .obj format. To get a sense of how the file format works, let's examine an .obj file firsthand. Open up a new Blender scene, and delete the default cube. Import the **myCube.obj** file that came with the homework files (**File** → **Import** → **Wavefront (.obj)** and navigate to the example file, e.g. **$CS148_DIR/HW1/myCube.obj**).

The cube .obj should now appear as a mesh in the Blender GUI. Select the cube, and enter **Edit Mode** . You should see an assortment of vertices and faces along the surface of the cube. To look at the .obj file at an even lower level, open up the sphere example file in your favorite text editing program (e.g. Notepad++, TextEdit, Sublime Text, etc). Scrolling through the file, you may notice the following file structure:



v x1 y1 z1
v x2 y2 z2
v x3 y3 z3
...
v xm ym zm
f face0v1 face0v2 face0v3
f face1v1 face1v2 face1v3
f face2v1 face2v2 face2v3
...
f facenv1 facenv2 facenv3

This is an example of the most basic form of the .obj format, which contains a list of vertices and a list of triangular faces that together specify the geometry of a 3D object **mesh**. The coordinate space in which the vertices are specified is the **object space** of the object as discussed in lecture.

The lines that begin with a v contain **vertex** data, while the lines that begin with a f contain **face** data. Each "vertex line" starts with a v and follows the v with three floating point numbers: the x, y, and z coordinates (in that order) of a vertex in the described 3D model. Each "face line" starts with a f and follows the f with three integers specifying the three vertices that make up a triangular face in the 3D model. For example, the following line:

```
f 1 8 37
```

specifies a face that is made up of the first, eighth, and thirty-seventh specified vertices in the file. Note that this means the vertices are 1-indexed; i.e. the first specified vertex in the file is referred to as the first vertex, rather than the zeroth vertex. This is simply due to convention.

3.1 More Complex OBJ Files

You may notice while importing the cube example file that there was also an export option in Blender. If you try exporting e.g. the default cube as an .obj file and then examining it in a text editor, then you will see a lot more data in the file than just the vertices and faces.

We will discuss this other data later in the class. But if you're curious, then you might find this website a good reference.

4 Tessellating a Sphere

We divide the surface into horizontal vertex rings called **stacks** and vertical vertex arcs called **sectors** as shown in Figure 4. We choose the number of stacks and sectors when we "tessellate" a sphere this way.

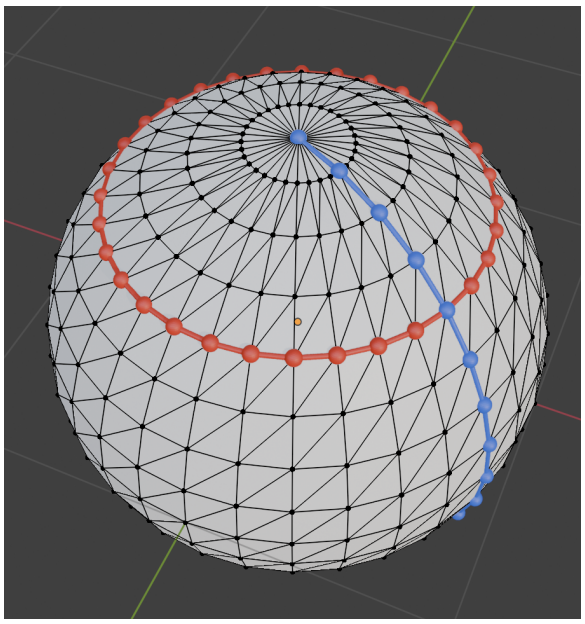


Figure 4: We can divide the vertices of a sphere up into latitudinal rings called stacks and longitudinal arcs called sectors.

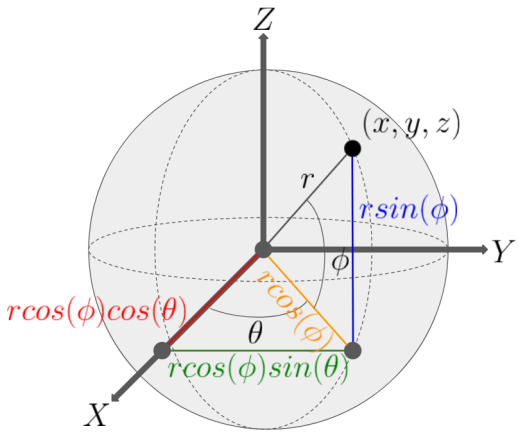


Figure 5: A Cartesian point (x, y, z) on the surface of a sphere can be computed using spherical coordinates (r, θ, ϕ).

Let I be the number of stacks minus one (because the top vertex is 0 indexed), and let J be the number of sectors; then, let i index the stacks from $[0, I]$ with the 0^{th} stack being the point at the top and the I^{th} stack being the point at the bottom, and j index the sectors from $[0, J]$, with the sector indices increasing as you move counter clockwise around the sphere. Note because there are J sectors, sector 0 and sector J are the same.

To create a sphere, for every stack that's not the top or bottom (`for i in (1, I)`), and for every sector (`for j in J`), we want to use i and j to compute `(x, y, z)` coordinates in 3D space, then connect them to form faces.

4.1 Tessellating a Sphere: Vertices

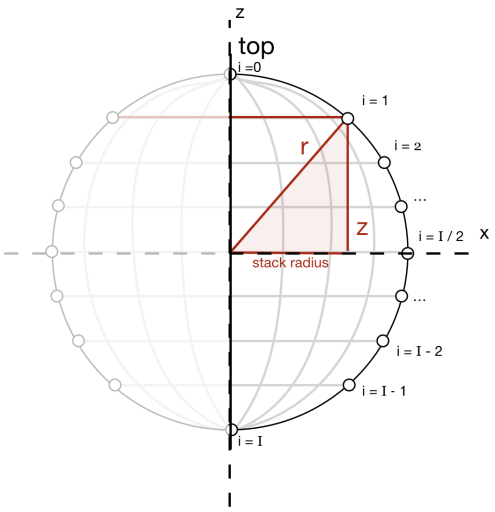
Before continuing, think about how you might do this. Start with the middle stack where the stack radius is at it's widest, r. The z coordinate is constant for the whole stack (because it's flat), so how would you go about finding the x and y coordinates? How do you use the sector number to calculate θ within the stack? Similarly for the sectors, for any given sector, how would you determine the height? How would you determine ϕ? **Give it some thought before reading more!**

Here are two possible ways to do the vertex computations:

1. Use the stack number to find the height (z coordinate) and the stack radius at that height. Then, we can use the stack radius to find the x and y coordinates around the circle. This has

one big limitation that we'll discuss in a bit.

2. Use the stack number to find the angle ϕ from the xy plane to the stack height. Then use the sector number to find the angle θ from the x axis to the point that we're looking for. Then use the two angles to calculate x, y, and z.



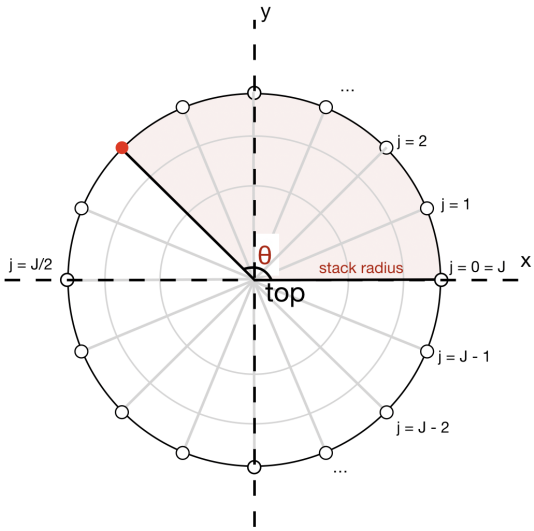
For the first method, we can figure out the z coordinate by dividing the entire height, $2r$, by the number of sections between stacks, I, to get the height between any two stacks. We then multiply by the stack number of our vertex, i, and subtract the result from r since the first stack ($i = 0$) starts at the top, $+r$. This gives us:

$$z = r - (i * 2r/I) \tag{1}$$

Once we have z, we can use the triangle in the diagram to the left to get:

$$\text{stack radius} = \sqrt{r^2 - z^2} \tag{2}$$

Note how the stacks are evenly spread on the z-axis here, but not spread out evenly along the curve of the circle. We want the difference in stack height to get smaller as we move towards the top and bottom. This is still a fine method for this homework, but think about how we might address it!

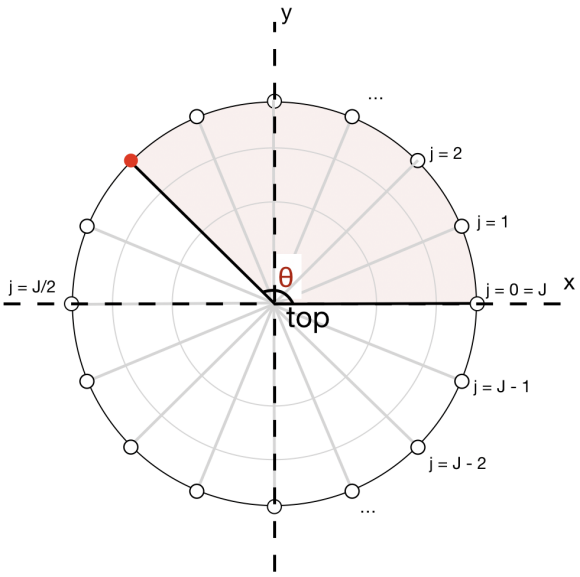


Now we only have to think about the stack with our vertex. Knowing the radius of that stack, we can calculate x and y using

$$\begin{aligned}
\theta &= 360° * \text{percentage of sectors in } \theta \\
&= 2\pi * \frac{j}{J} \\
x &= \text{stack radius} * cos(\theta) \\
y &= \text{stack radius} * sin(\theta)
\end{aligned}$$

Taking z from before, we finally have an x, y, and z to define a vertex.

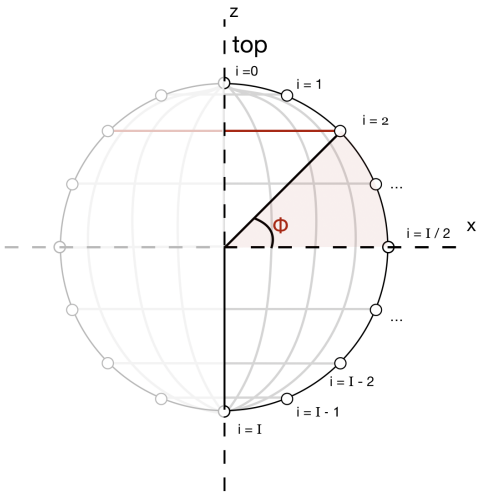
That was the first method as mentioned above. It's good for an intuitive demonstration, but the following, second method is preferred.



For the second method, similarly to the first, let's use the sectors to calculate θ, which is the angle to the correct sector when looking down from the top. Let's say sector 0 is at $\theta = 0$. We can calculate θ by observing what percentage of the J sectors are within θ. By this logic,

$$\begin{aligned}\theta &= 360° * \text{percentage of sectors in } \theta \\ &= 2\pi * \frac{j}{J}\end{aligned}$$

Stop for a second and reference the diagram to the left to make sure that this logic makes sense to you.



Now, let's calculate ϕ, which is the angle up from the xy plane. Since the top of the sphere is at $\frac{\pi}{2}$, we can use the same logic to calculate

$$\phi = \frac{\pi}{2} - \pi\frac{i}{I} \tag{3}$$

Again, look at the left diagram, and make sure this logic makes sense to you.

Notice how θ ranges from $[0, 2\pi]$ and ϕ ranges from $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

With θ and ϕ, we can compute the Cartesian coordinates x, y, z of a point on the surface of the sphere as shown in Figure 5:

$$x = r\cos\phi\cos\theta \tag{4}$$

$$y = r\cos\phi\sin\theta \tag{5}$$

$$z = r\sin\phi \tag{6}$$

Thus, we have a set of equations to find the (x, y, z) location of any vertex from its stack and section numbers.

And so we've covered 2 different ways to think about this problem. We also provided some starter code to help you decompose the problem. You're welcome to code up whichever method you find easier to understand. If you have your own way to do this, then that is also fine as long as you get the right result in the end!

4.2 Tessellating a Sphere: Faces

Now that we have our vertices computed, let's index them from 1...n to define our faces. Pause here! Once you have all the vertex indices, how would you fill each square in the diagram below with two triangles? For each vertex, v, how would you add 2 triangles assuming v is the top-left point of the square that you're trying to fill?

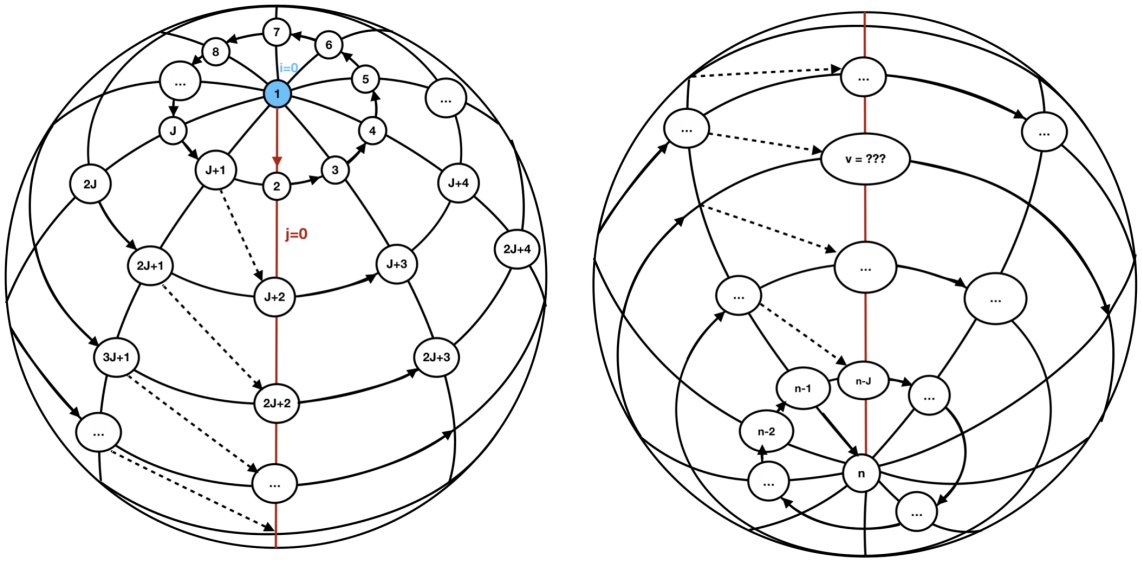


Figure 6: The order in which you store your vertices to an array will decide how the vertices on the sphere correspond to the indices. For example, if you add vertices to your array starting from the top stack, then your sphere might look like the one in this diagram. But if you store the vertices at the bottom first, then the indexing would be different. **In this case, for any vertex, v, given its stack and sector as well as the total number of stacks and sectors (I, J, i, and j), how would you calculate the index of v?**

Iterating through these vertices, for any vertex, v, notice that we can make two triangles from $(v, v+1, v+J)$ and $(v+1, v+J, v+J+1)$ unless v is the last vertex in the stack. If v is the last vertex in the stack, then $v = aJ + 1$ for any multiple, a, and our triangles would be $(v, v + 1 - J, v + J)$ and $(v + 1 - J, v + J, v + 1)$. Think about how your sphere is indexed - how would you find the triangle faces?

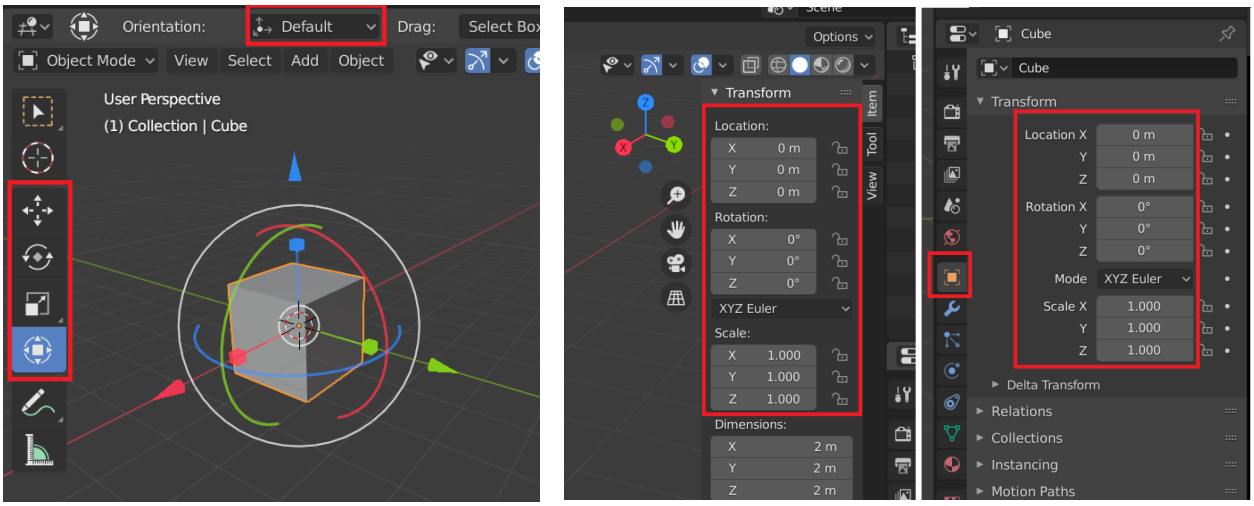
Note though that the top-most and bottom-most stacks only have one triangle each. These should be handled by a separate case, i.e. use if-statements: if $v = 1$ or $v > n - J$, then what?

5 Transforming Objects in Blender

To transform an object in Blender, you can use the buttons in the left toolbar of the 3D viewport: Move , Rotate , Scale , and Transform . Selecting an object with one of these buttons enabled will prompt an orange dot to appear to indicate the object origin, as well as red, green, and blue lines to show the degrees of freedom in which you can move the object.

The keyboard shortcuts are `g` (for grab/translate), `r` (for rotate), and `s` (for scale). You can press `x`, `y`, or `z` afterward to lock the transform to a specific axis. For example, if you press `g`, then `x`, and then move your mouse, then the object will only translate along the X-axis. You can also type in numbers after pressing the keys to specify the exact values you want to move, rotate, or scale. For example, if you press `g`, `z`, 2, then hit Enter, then the object will translate 2 units along the Z-axis.

Note: make sure your cursor is in the 3D viewport area when pressing the keys. Blender uses the cursor location to determine which area you want to send your keystrokes to.



5.1 Local Transformations

You can view and alternatively change the **local transform** of an object in two other places in the Blender interface. The first is the sidebar of the 3D viewport by pressing `n`. The second is in the **Properties Editor** in the bottom right under **Object Properties**, represented by an orange square. Under **Transform**, you should see the local location (translation), rotation, and scale of the selected object.

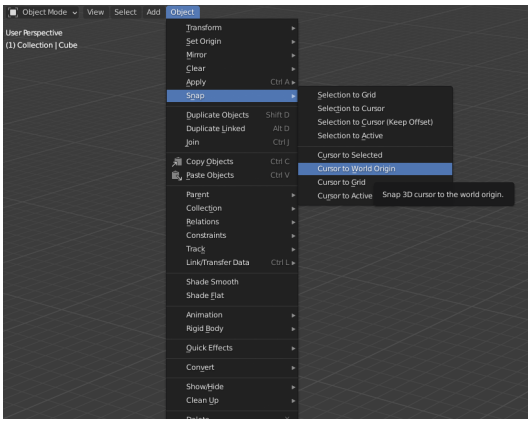
REMINDER: Any changes in the Properties Editor are LOCAL to the object's own coordinate space. While the Properties Editor is often the most convenient part of the GUI to use for transforming your objects, keep in mind that it is only for local transformations, NOT GLOBAL transformations.

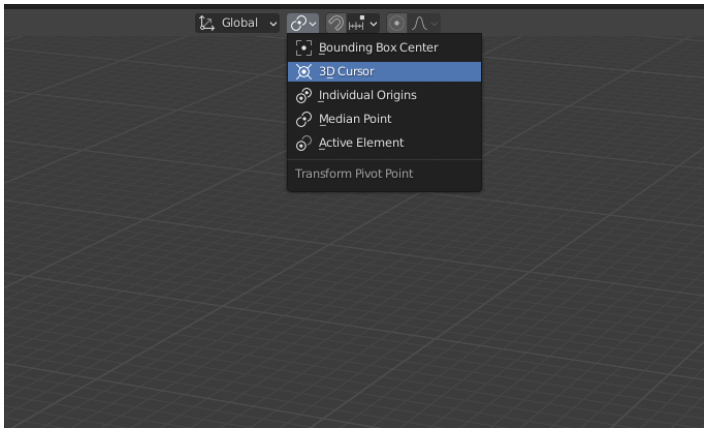
5.2 Rotating Objects about the Global World Origin

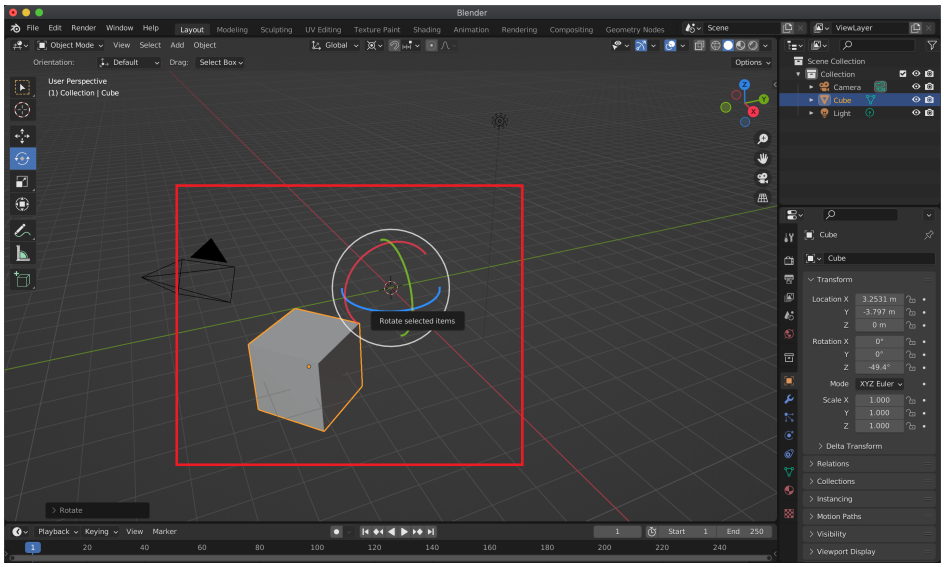
You may notice that if you try to rotate an object using aforementioned tools in Blender, then it only rotates around its own origin point rather than the world space origin. For instance, if you translate the default cube e.g. 5 units along x, then try to rotate about z, then the cube ends up rotating about its own center, not the global z-axis. This is because by default, Blender does **local transforms** about the object's own origin.

To make it rotate about the **global** axes, first go to the **Object** dropdown menu near the **Object Mode** indicator, then select **Snap** → **Cursor to World Origin**. If you're in **Edit Mode** instead, then the option will be under the **Mesh** dropdown menu instead. Then, regardless of

what mode you're in, set the transform pivot in the dropdown menu next to the **Global** indicator to **3D Cursor**. Now, if you use the rotate tool or the **r** keyboard shortcut, you'll see the rotations go about the world origin.

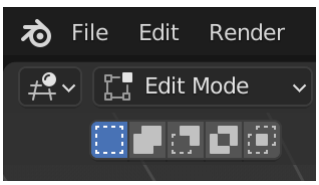


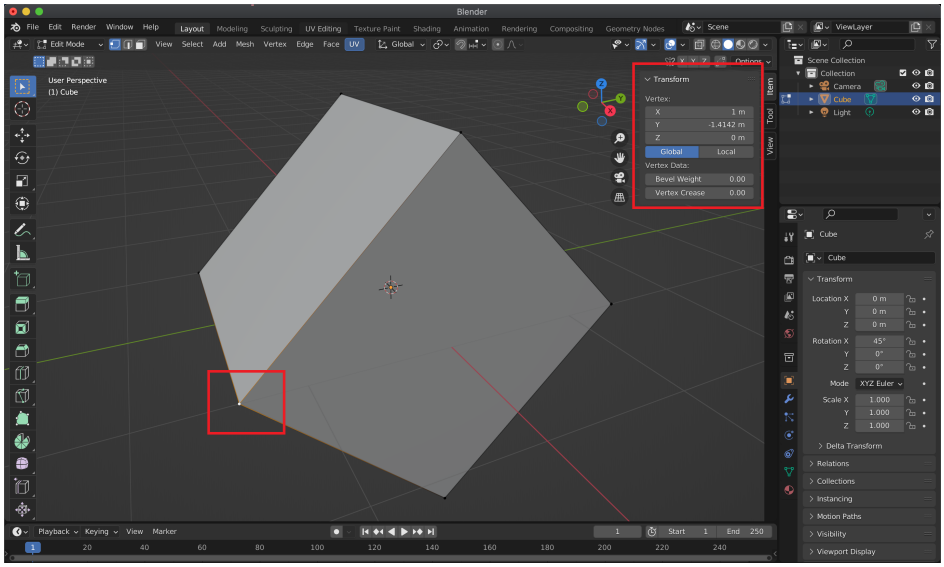




5.3 Viewing the Global Coordinates of a Vertex

Enter **Edit Mode** and select a vertex. From there, press the **n** key-board shortcut to open the side-bar of the 3D viewport. This by default displays the local coordinates of the vertex, relative to the object origin. Toggle to **Global**, and you'll see the global coordinates of the vertex, relative to the global origin.





6 Modeling Geometry in Blender

Now that you're familiar with the .obj format, you can try your hands at modifying object meshes in Blender to create anything you'd like! Three approaches that we'll discuss are polygon modeling, sculpting, and geometry nodes. We'll also mention subdivision as discussed in lecture as a way to add more geometry for an object with too few vertices and faces.

We also encourage you to look at the wide array of resources online. The best way to learn Blender is honestly through video tutorials. If there are any particular objects that you want to learn how to model, then usually a simple Google or Youtube search will yield some results. For instance, here's a Blender tutorial for making a flower.

6.1 Polygon Modeling

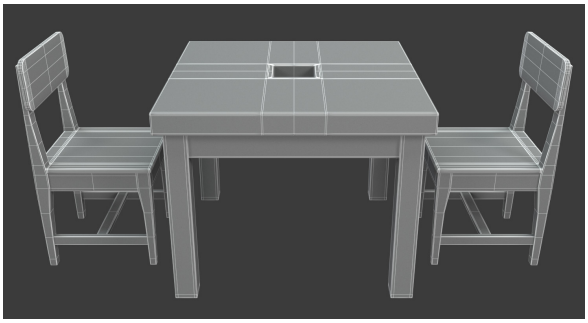


Figure 7: Dining table object made from modeling. Source: ArtStation

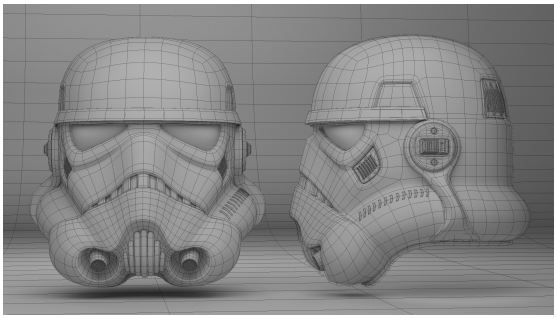
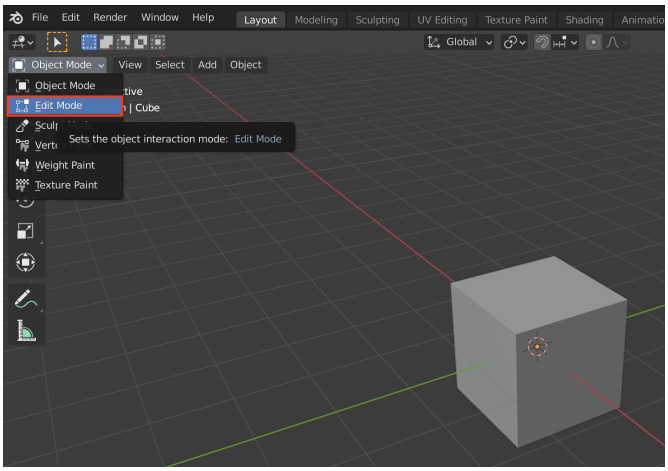


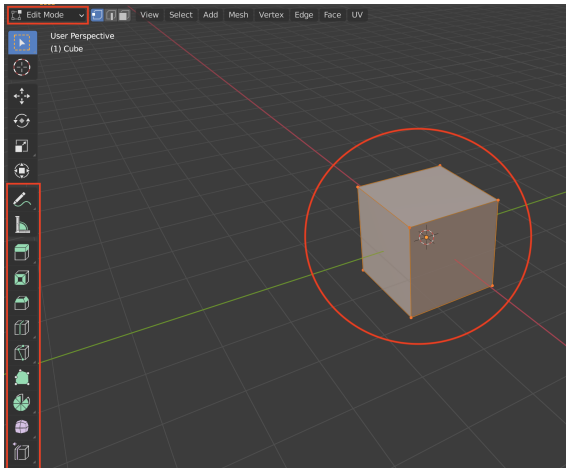
Figure 8: Stormtrooper helmet object made from modeling. Source: ArtStation

Polygon modeling is the basis of every 3D modeling package. With modeling, we manipulate polygonal meshes by moving around vertices, edges, and faces to create more complex and interesting

objects. Modeling is ideal for creating hard-surface objects (i.e. man-made objects with sharp angles) due to the precision of selecting and moving individual vertices. On the flip side, modeling is not so great for organic objects - if you're hoping to create a more organic effect, then sculpting might be the better technique to use.

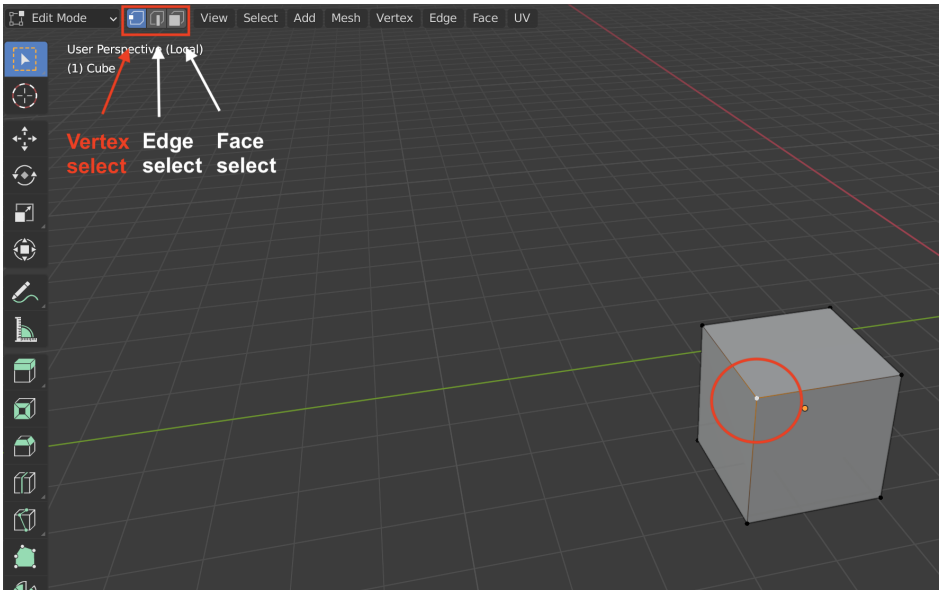
To access the modeling tools in Blender and make changes to objects at the vertex/edge/face-level, we will want to be in **Edit Mode**. Once in **Edit Mode**, you will notice that 1) new tools have appeared in the sidebar, and 2) you can now see individual vertices, edges, and faces.





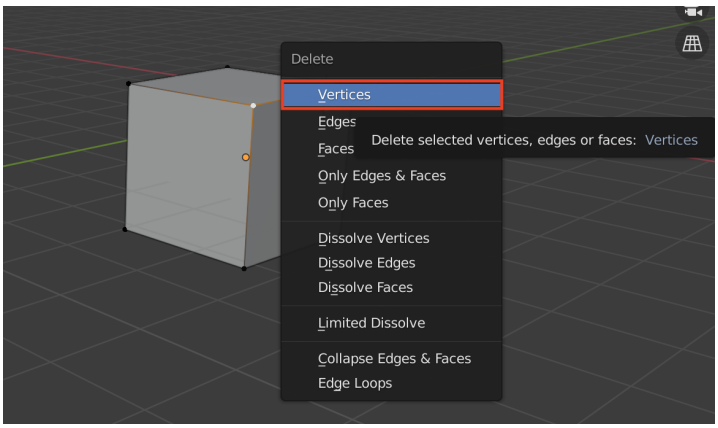
6.1.1 Selection

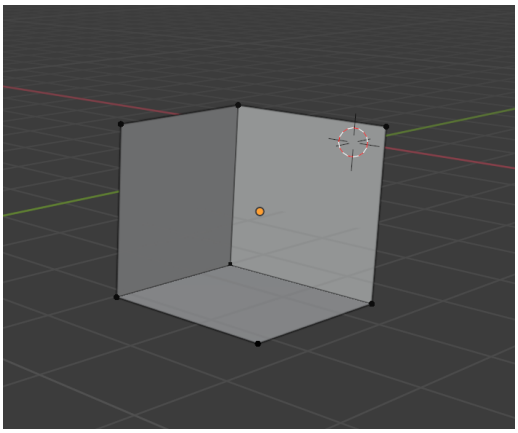
Simply click on a vertex to select it. If you want to select multiple vertices, hold **Shift**, and click on other vertices that you want to select. You can also toggle between **Vertex Select**, **Edge Select**, and **Face Select** in the upper-left, or alternatively with the **1**, **2**, and **3** shortcut keys.



6.1.2 Deletions

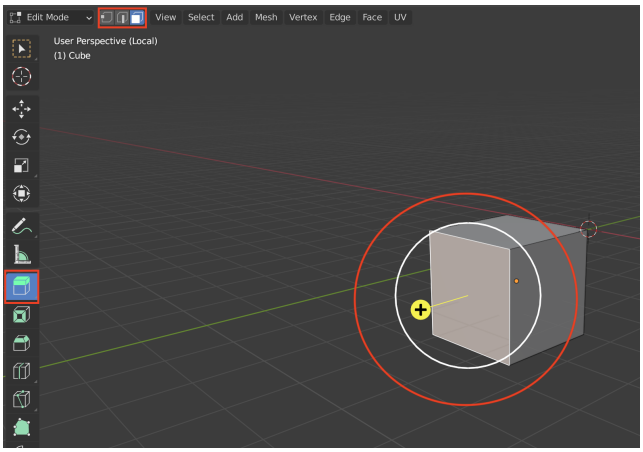
To delete a vertex / edge / face, simply press **X** while the vertex / edge / face is selected. This will bring up the "delete" menu, which you can use to specify exactly what you want to delete.

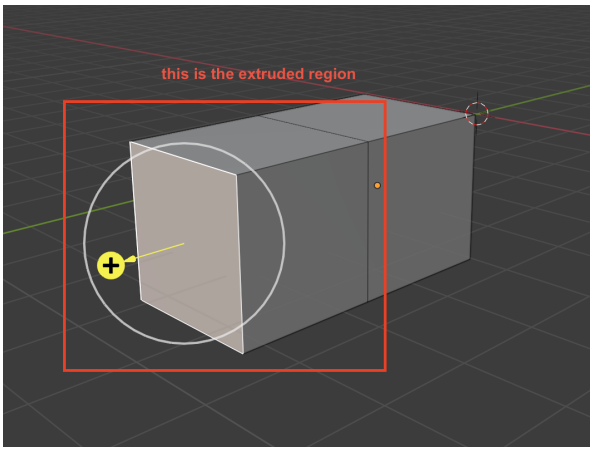




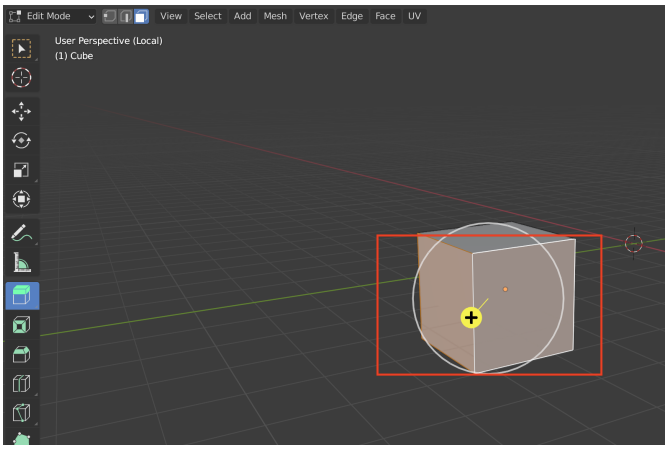
6.1.3 Extrusions

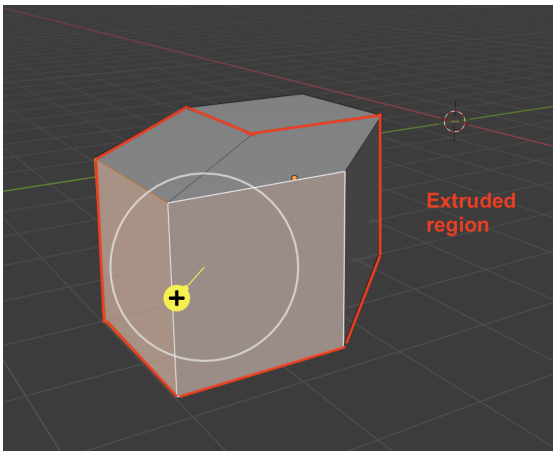
A basic but powerful mesh editing tool is the **Extrude Region** tool. To extrude a face, first select the **Extrude Region** tool in your toolbar. Then, select the face you want to extrude (switch to **Face Select** and click on the face). Then, click and drag the yellow + symbol. A keyboard shortcut is also available by pressing **E** after selecting the face to extrude.





We can also extrude an entire region composed of multiple faces. Simply select the faces you want to extrude (hold **Shift** and click to multi-select), and drag the yellow + symbol.





6.1.4 Resources

- This Youtube video series is a great basic intro series to modeling on Blender's official channel. It was created as part of Blender's 2.8 fundamentals series, but works in later versions as well.
- Other common techniques for modeling are with booleans and bevels - see this beginner's Youtube video tutorial.
- Blender has a free add-on for performing boolean operations. It already comes with Blender, but see here for how to enable the **Bool Tool** and configure its preferences plus keyboard shortcuts.
- For more in-depth information on how to use any of Blender's modeling tools, see the full Blender documentation on editing meshes.
- There are many, many modeling tutorials on Youtube, so if there's a specific object you're interested in creating, try searching on Youtube for a video tutorial that can help get you started. Donut tutorials for instance tend to be a very popular start for students.

6.2 Sculpting

Sculpting is best suited for organic shapes, and uses brushes to deform the object. Note that the resulting mesh from sculpting will likely have a high polygon count, so you will need decent computing power if you plan to sculpt fine details.

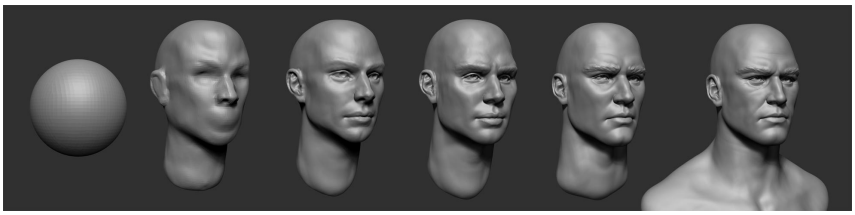
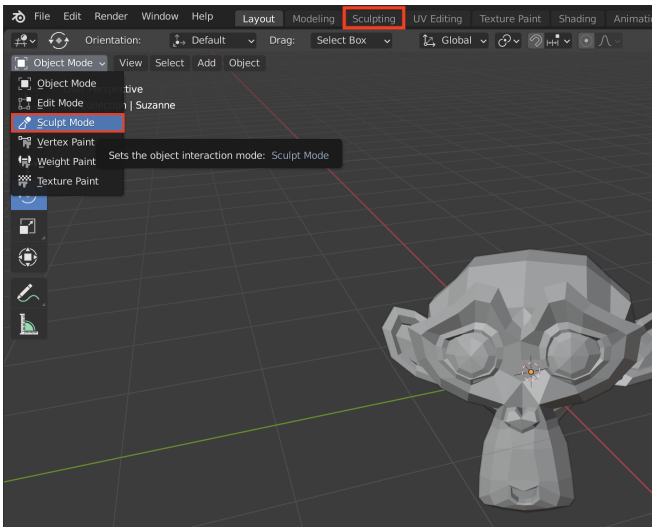
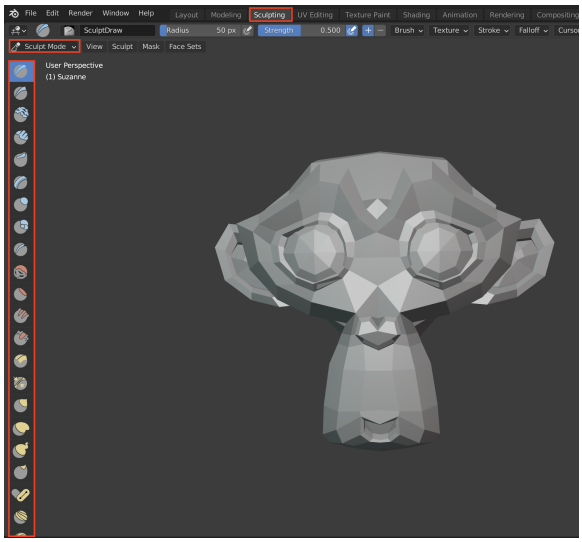


Figure 9: Scuplting a realistic face. Note that human faces are really hard to get right! Source: FlippedNormals

Figure 10: Sculpted creature. Source: ArtStation

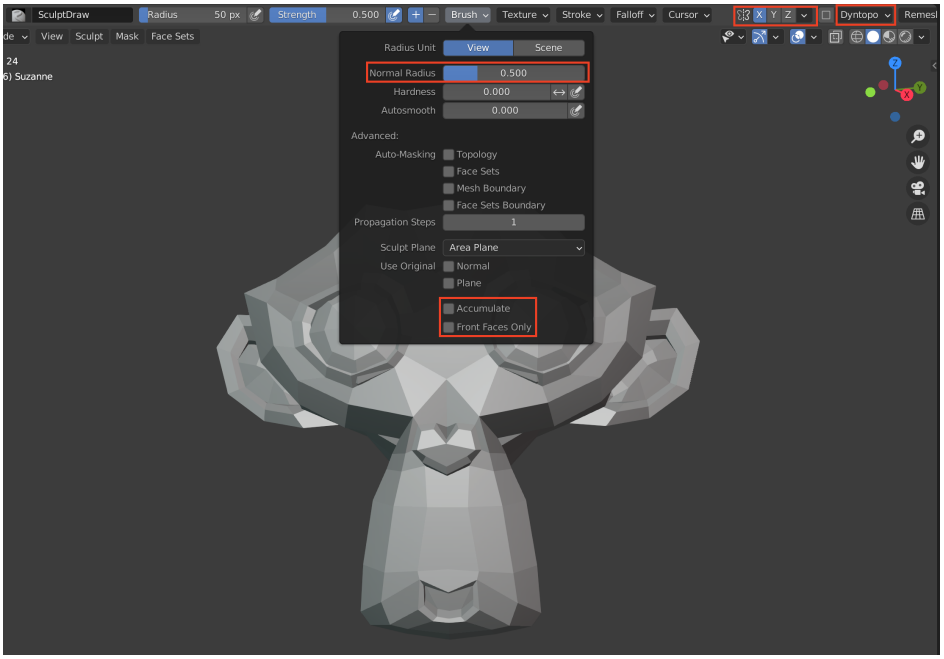
Access **Sculpt Mode** by directly clicking the **Sculpting** tab on the top or through the **Object Mode** menu. Once in **Sculpt Mode**, you will notice that new tools have appeared in the sidebar.





6.2.1 Brush Settings

The brush settings are on either the top or side toolbar. You can change the size of the brush using **Normal Radius** and the intensity using **Hardness**. On the top right, you can change symmetry between x,y,z. The brush, at default, is symmetrical around x.

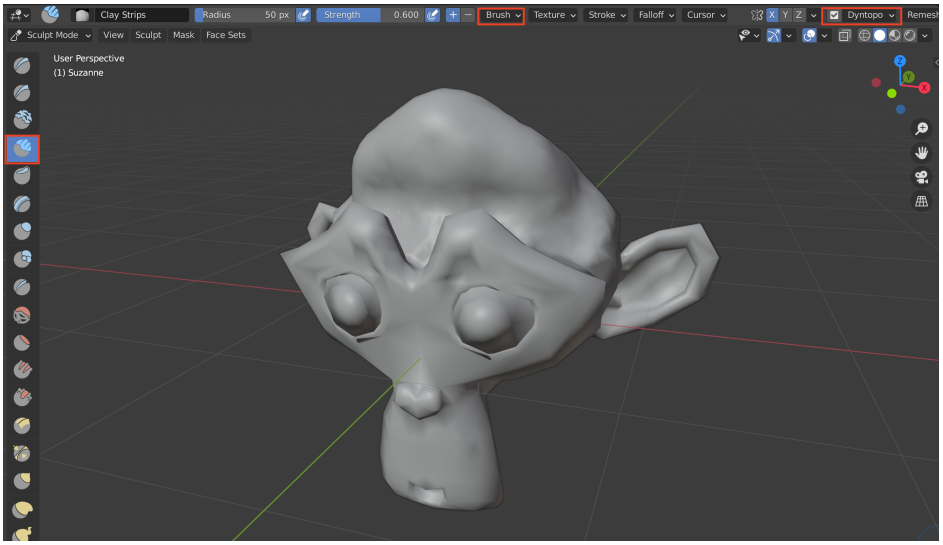


Two useful settings to know are the **Accumulate** and **Front Faces Only** options. Toggling **Accumulate** on causes brush strokes to stack on top of each other, while toggling **Front Faces Only** on makes it so that the brush only affects the vertices that you see from your viewpoint.

Also note that toggling ctrl/cmd while using the brush will change the direction that the brush affects the geometry. Furthermore, holding down ctrl/cmd subtracts from the object while using the brush shape.

6.2.2 Dyntopo

Dyntopo (short for Dynamic Topology) is a useful tool that can help you add or remove geometry, allowing you to construct more complex shapes out of a simple mesh. You can find it in the upper right when in **Sculpt Mode**. If you need more detail in a model, then enabling Dyntopo will cause Blender to dynamically tessellate your mesh into more vertices and faces for finer sculpting.



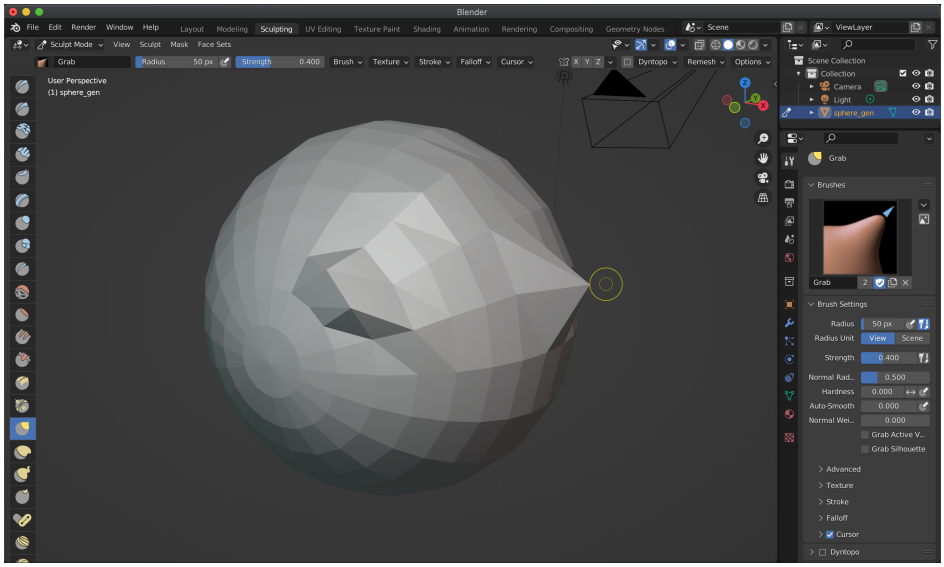
The above edit to the default monkey mesh (**Object Mode** → **Add** → **Mesh** → **Monkey**) was done with a **Clay Strips** brush with **Accumulate** and **Dyntopo** enabled to make the top of the head larger.

6.2.3 Resources

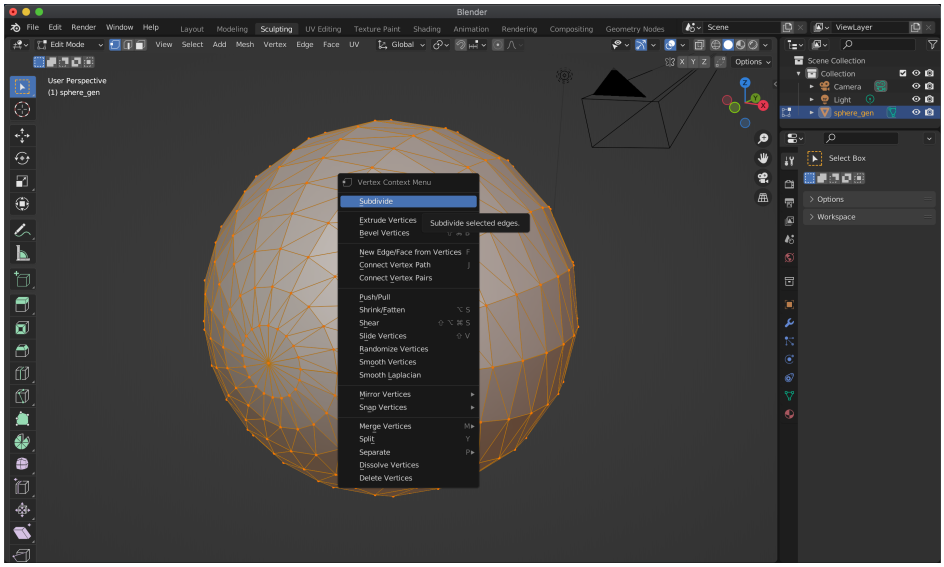
- For an introduction to sculpting and an explanation of the various brushes, see the official tutorials on Blender Sculpting Fundamentals and Sculpting in Blender with Complex Models.
- There's also this post for first-time sculptors if you're into character modeling.
- The Youtube channels YanSculpt and CGBoost are good places to find advanced sculpting videos.
- For more in-depth information on how to use the sculpting tools, see the full Blender documentation on Sculpting.

6.3 Subdivision

If you tried sculpting on top of your generated sphere from the assignment, then you will notice that the brushes have very jagged effects, and grabbing a vertex will deform a large portion of the sphere into a big spike. This is because our generated sphere is still relatively low-poly, even with 20+ stacks and sectors apart. To generate more faces, i.e. more geometry, to work with, we can **subdivide** the faces on the sphere into more faces as mentioned in class.

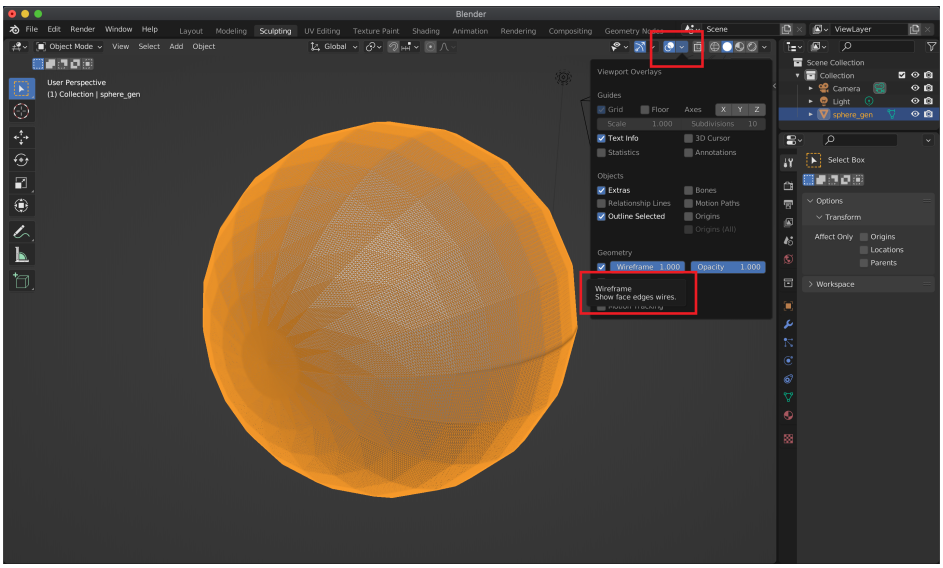


Go into **Edit Mode** and right click over the object you want to subdivide (in this case, the generated sphere .obj). One of the first options in the drop down menu will be **Subdivide**. For the sphere, try sudividing 3-4 times. With each subdivision, you will see more triangles litter the surface of the sphere.

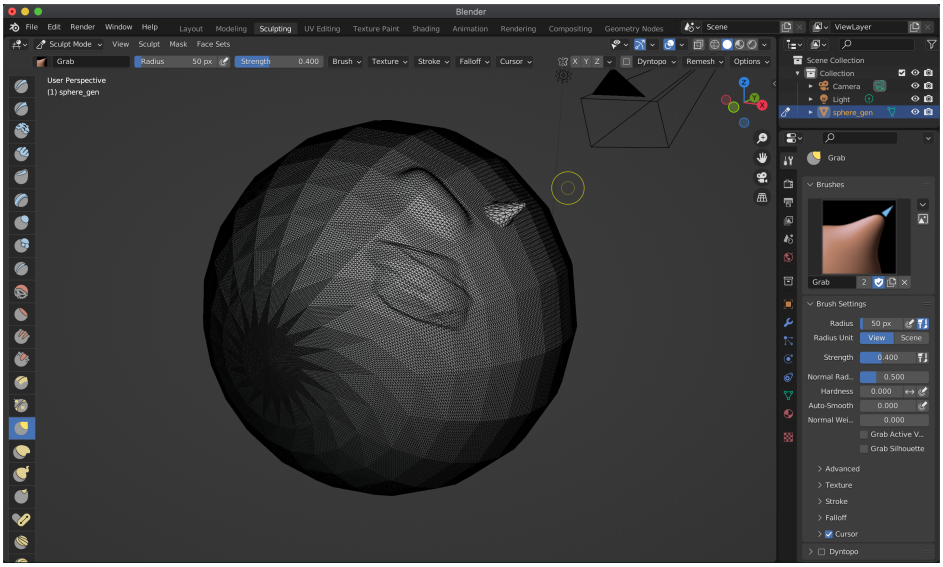


Note that if you go back to **Object Mode** or **Sculpt Mode**, you might see the sphere revert back to its original low-poly form. This is merely Blender displaying the original mesh. The added

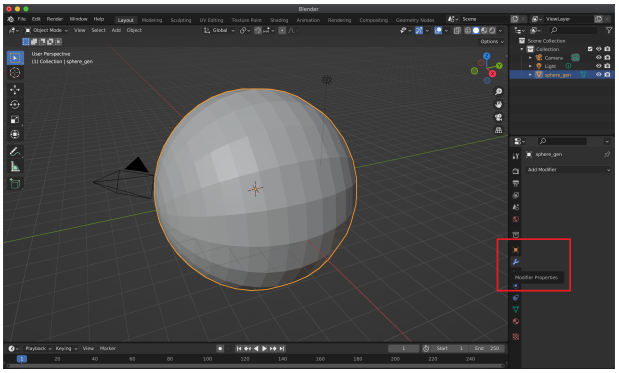
faces from subdivision are still there. To view them, we need to toggle on **Wireframe** under the **Viewport Overlays**.

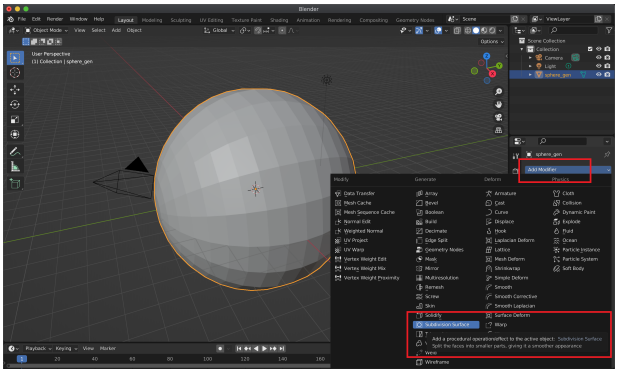


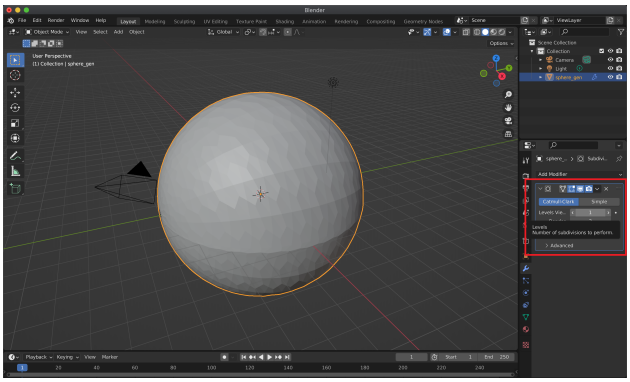
Now if you try to sculpt over the sphere, then you should see much more finer brush edits.



Another way to subdivide is to use the **Subdivision Surface Modifier**. This can be found from the **Properties Editor** on right-hand side under the **Modifier Properties** tab, labeled with a wrench icon. There, you can add a **Subdivision Surface Modifier**, which uses a different subdivision algorithm from the one we discussed in class. Increasing the levels will add additional subdivisions to the mesh.

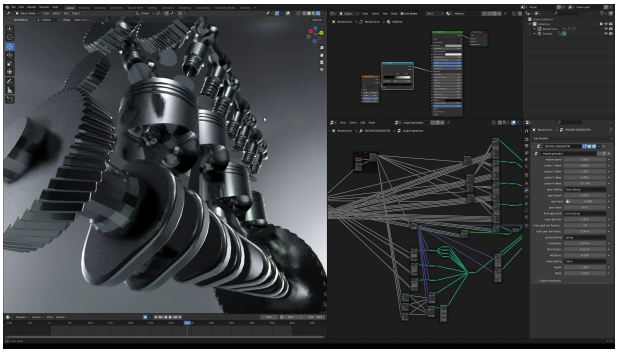






6.4 Geometry Nodes

Geometry nodes are a more recent Blender feature that allows you to do procedural modeling (similar to writing a script) with nodes. You can add and manipulate many objects at once based on location, change their size and shape, and also join them together. Since there are a lot of different nodes with different capabilities, the best way to learn is by following a tutorial! Here are some of the things you can do with geometry nodes:



6.4.1 What is a node?

A node is effectively a function with various inputs and an output. One of the geometry nodes for example is the **Transform** node, which works similarly to the transform functions you need to code up for this assignment. The node takes in geometry that you wish to transform as well as parameters for translation, rotation and scaling. The output is the transformed geometry.

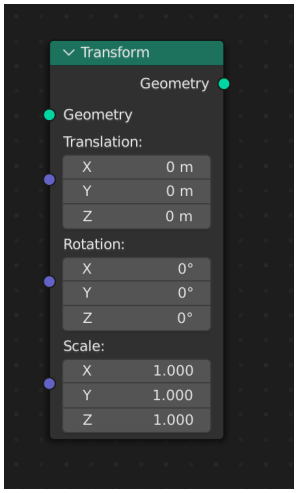


Figure 11: The small circles on the left represent connection points for inputs, and the circle on the right represents a connection point for where to send the output. The numbers in the node itself can either be typed in or generated by another node that's connected to the blue circle(s).

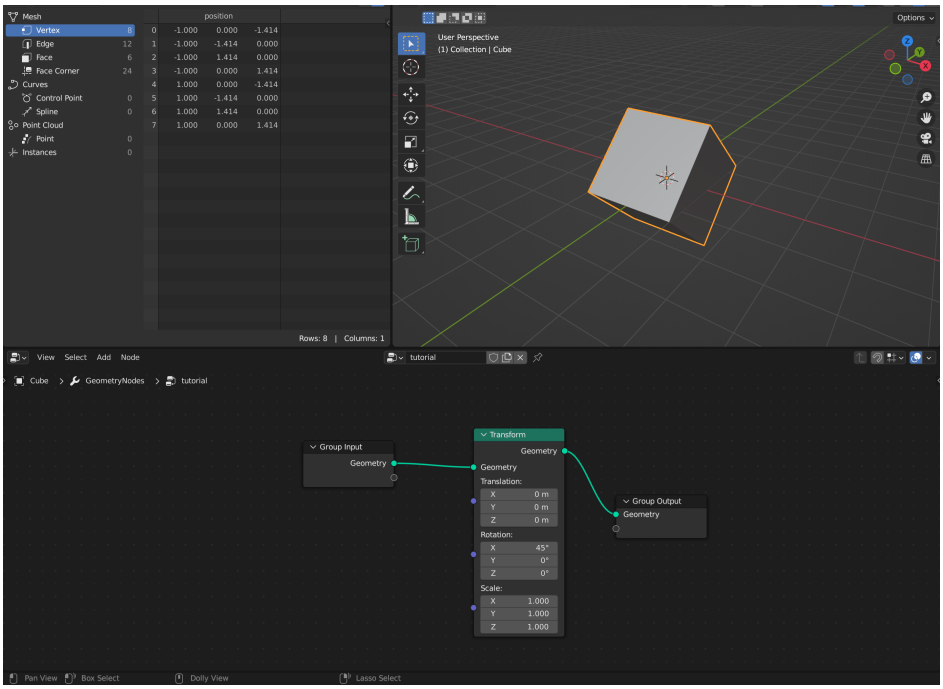


Figure 12: Here, the input (Group Input: Geometry) sends the target geometry to be transformed. Then the transformation node sends the new vertex locations to the geometry output node.

At the beginning of our node graph, we have to have an input, and at the end, we must have an output. The input is initial geometry to be transformed and edited, and the output is the new edited geometry that is pushed back into our scene.

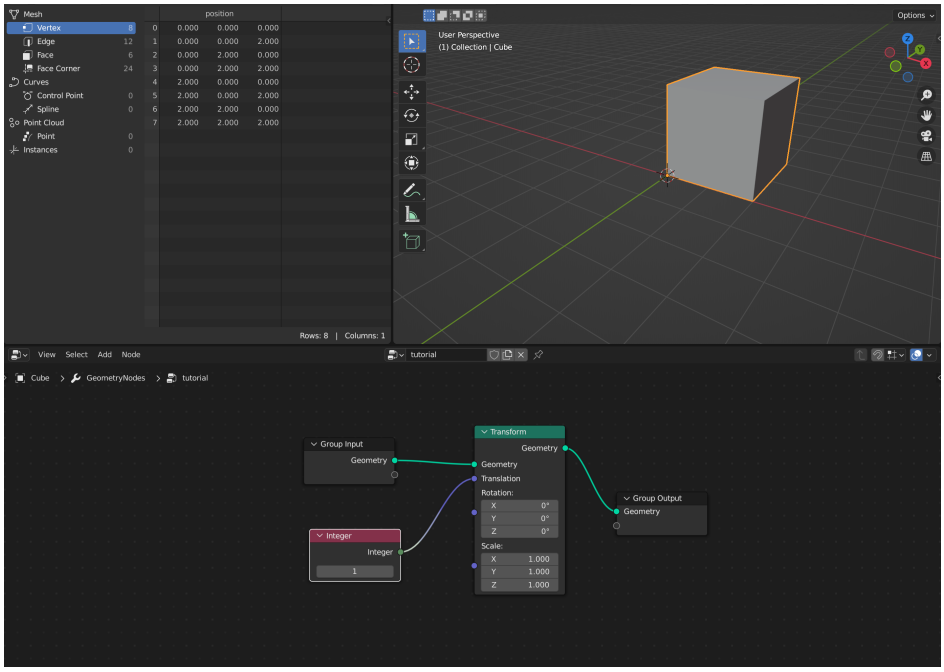
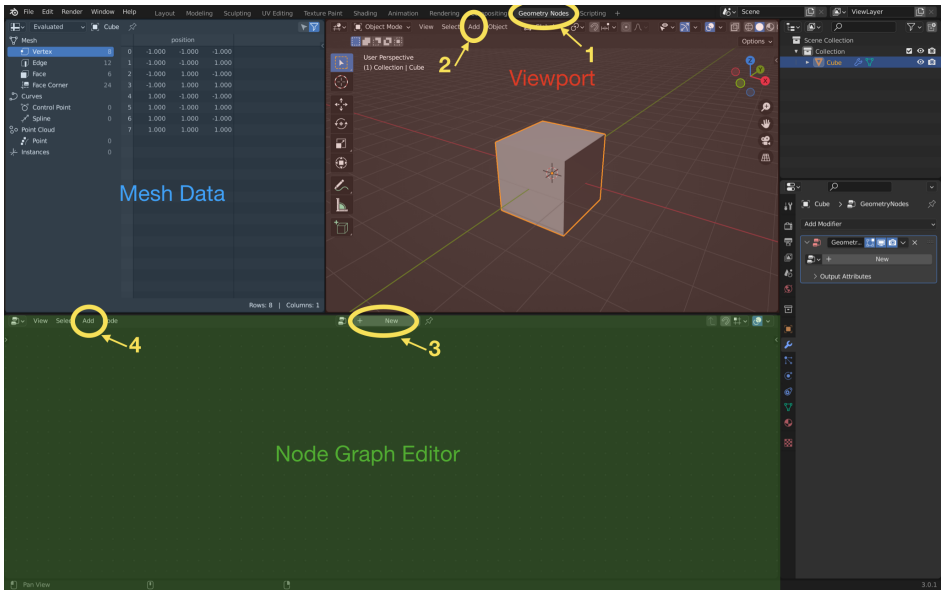


Figure 13: You can replace the numbers in a node by connecting another node with numerical input.

6.4.2 Getting Started

1. Click on the "Geometry Nodes" tab at the top of the Blender window. This tab consists of three windows.



2. Add a simple geometry, say a cube, and select it. Now the top left panel should show you all the mesh data of your object, starting with the vertex locations. The top right panel is your standard viewport, and the bottom panel is your node graph.

3. Click the button that says "+ New" in the center of the node graph panel. This will add a new Geometry Node Graph with just an input and output.

4. To add nodes to the node graph, click the add button in the top left of the Node Graph Editor panel. Try experimenting with the different nodes!

6.4.3 Resources

Now that you know the basics of what a node is and how to start, check out these resources for more ideas of what you can do with this tool:

- This is a great beginning tutorial that's short and moves fairly slowly on how to make a sugar coated candy: Beginner's geometry node tutorial.

- This tutorial on how to make an altar with pillars is a little harder to follow, but it does a good job of showing how to combine mesh editing with geometry nodes. This will be helpful if you want to make some objects to procedurally edit and transform: Pillars tutorial

- This grassy field tutorial might be helpful for anyone that is planning on making an outdoor scene: Grassy field tutorial.