CS148 Homework 1

Homework Due: Jul 2nd at 11:59 PM PST
Quiz Date: Tuesday Jun 28th

1 Assignment Outline

This assignment has 6 TODOs of varying lengths. These are marked with the **Action:** indicator in this PDF as well as TODO in the section headers. We recommend first reading through the TODOs and see which you can do immediately with your prior knowledge. Then, for any TODOs that you're unsure about, read through the notes in the same section.

- **TODO 1**: Write a (short) Python script to tessellate a sphere and generate the vertices and faces data of a simple sphere .obj file.
- **TODO 2**: Screenshot your simple sphere .obj imported into Blender.
- **TODO 3**: Make as simple (or as complex) of an object as you want in Blender with polygon modeling or sculpting.
- **TODO 4**: Screenshot your created object in the Blender interface.
- **TODO 5**: Write short snippets of Python code to transform a cube object 4 different ways and compute the resultant coordinates of the vertices.
- **TODO 6**: Repeat **TODO 5** in Blender.

Please download **HW1.zip**, save it in some course directory **$CS148_DIR** on your machine, and unzip the file in this directory. All the work will be done in the **$CS148_DIR/HW1/HW1.ipynb** Jupyter Notebook.

When you are finished, print and save your Notebook as a PDF and submit to Gradescope, similar to what you did for HW0.

1.1 Policies

The policies regarding usage of Piazza, Honor Code, Collaboration Policy, and Late Assignments are as listed in HW0, and will stay the same across the quarter. In particular, please keep in mind that no sharing of code is allowed!

2 Quiz 1

The first quiz will be held Tuesday Jun 28th, and the various quiz sessions throughout the day are posted on the Canvas syllabus and calendar. To join a quiz session, **join BOTH the Zoom call link on Canvas for that time AND QueueStatus**. You should be able to make a QueueStatus account with your Stanford email if you don't have one already.

Quizzes are meant to be done solo. While you can work with a partner on the homeworks, you each must take your own quiz session.

When you join a quiz session, the CA or instructor will admit you from the waiting room into the main room once they reach you on QueueStatus. They will then ask **randomly one of the following questions**:

- What are two reasons for why we often use triangle meshes to model our objects over other polygonal meshes?
- What does the .obj file format store (at the minimum) to keep track of our geometry data for an object?
- What is the point of subdivision and why do we need to move all the vertices after inserting new vertices?
- Why do we want to use cubic splines when editing a mesh as opposed to linear interpolation?
- What is the purpose of homogeneous coordinates for transforming objects?

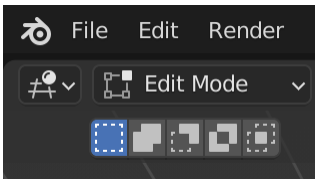
3 Generating a Basic Sphere OBJ File

As a refresher of Python and a way to get hands-on experience with the .obj file format, you will be writing a short script to generate an .obj file for a simple sphere mesh. To do so, you will need to tessellate a sphere with the appropriate vertices and faces, and then write each vertex and face to a file according to the .obj format.

3.1 The OBJ file format

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 (**File** → **New** → **General**), and delete the default cube (e.g. select it and press **x**). Import our example .obj file (**File** → **Import** → **Wavefront (.obj)** and navigate to the example file) located in **$CS148_DIR/HW1/sphere_example.obj** .

The sphere .obj should now appear as a mesh in the Blender GUI. Select the sphere, and enter **Edit Mode** . You should see an assortment of vertices and faces along the surface of the sphere. To look at the .obj file at an even lower level, open up **$CS148_DIR/HW1/sphere_example.obj** 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.

3.2 More Complex OBJ Files

You may notice while importing the sphere example file in Section 3.1 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 future lectures and homeworks. But if you're curious, then you might find this website a good reference.

3.3 Tessellating a Sphere: Vertices

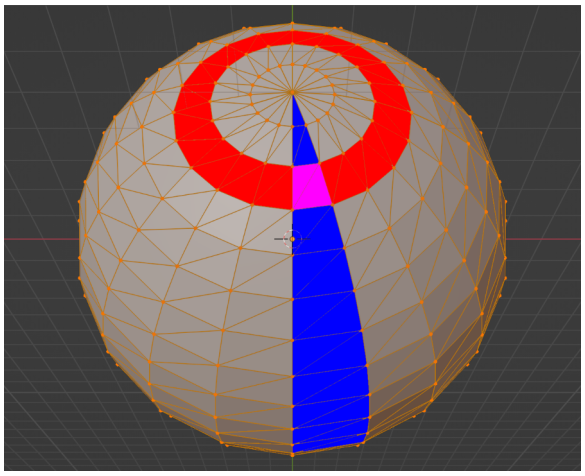


Figure 1: We can divide the surface of a sphere up into latitudinal faces called stacks (in blue) and longitudinal faces called sectors (in red).

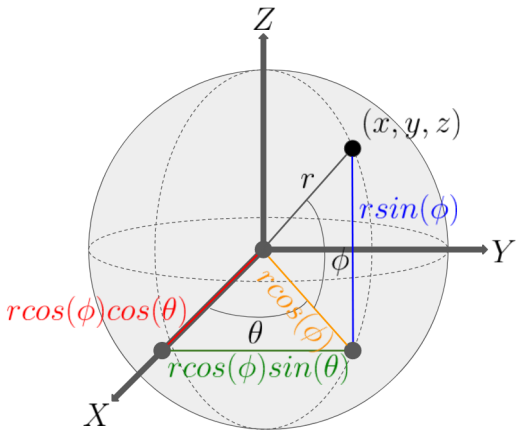


Figure 2: A Cartesian point (x, y, z) on the surface of a sphere can be computed using spherical coordinates (r, θ, ϕ).

We can tessellate a sphere by dividing its surface into latitudinal faces called **stacks** and longitudinal faces called **sectors** as shown in Figure 1. We choose the number of stacks and sectors when we tessellate a sphere this way.

Let I be the number of stacks, and let J be the number of sectors; then, let i index the stacks from $[0, I]$ and j index the sectors from $[0, J]$. We can compute the spherical coordinates necessary

to compute a point or vertex on the surface of a sphere as shown in Figure 2 using stacks and sectors as follows:

$$\theta = 2\pi \frac{j}{J} \tag{1}$$

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

Notice how θ ranges from $[0, 2\pi]$ and ϕ ranges from $[-\frac{\pi}{2}, \frac{\pi}{2}]$. This lets us define each i-th "stack step" in $\frac{\pi}{I}$ increments and each j-th "sector step" in $\frac{2\pi}{J}$ increments.

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 2:

$$x = r \cos\phi \cos\theta \tag{3}$$

$$y = r \cos\phi \sin\theta \tag{4}$$

$$z = r \sin\phi \tag{5}$$

3.4 Tessellating a Sphere: Faces

Suppose we compute the coordinates of all the vertices on our tessellated sphere using the method in 3.3 and index them from $1...n$. For a given stack or sector face, we can represent it as 2 triangle faces. If we let k_1 be the index of the upper-left vertex of the face and k_2 be the index of the lower-left vertex, then we can express each triangular face as a set of 3 vertices $(k_1, k_2, k_1 + 1)$ and $(k_1 + 1, k_2, k_2 + 1)$ as shown in Figure 3.

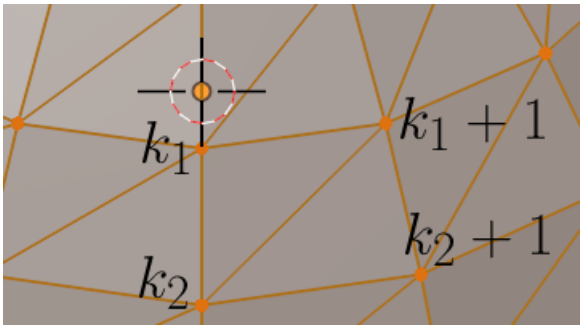


Figure 3: We can express triangular faces on a tessellated sphere for a particular stack or sector face as sets of 3 vertices $(k_1, k_2, k_1 + 1)$ and $(k_1 + 1, k_2, k_2 + 1)$.

Using the same indexing for I stacks and J sectors as in 3.3, we can compute k_1 and k_2 for a given pair of triangle faces as follows:

```
1: for i ← 0, ..., I − 1 do
2:     k1 ← i ∗ (J + 1)
3:     k2 ← k1 + J + 1
4:     for j ← 0, ..., J do
5:         Write_Faces()
6:         k1 ← k1 + 1
7:         k2 ← k2 + 1
```

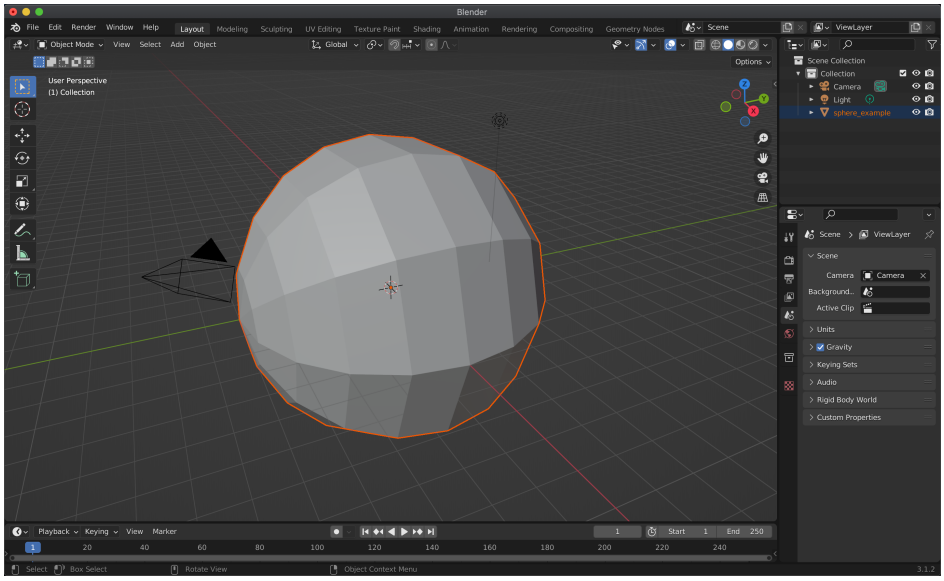
Note though that the top-most and bottom-most stacks only have one triangle each. This edge condition can be handled by having appropriate if-statements in however we decide to write the faces to output.

3.5 Your Task (TODO Parts 1-2)

Action:

1. Open up the Jupyter Notebook in `$CS148_DIR/HW1/HW1.ipynb`, and find the `TODO Part 1` cell. Here, write a Python script that computes and prints the contents of an .obj file for a sphere with radius $r = 5$, $I = 20$ stacks, and $J = 20$ sectors to standard output. Copy and paste your output into a blank text file and give it the .obj extension. Name it however you want, as you won't be turning this file in.

2. Open up a new default scene in Blender and delete the cube as you did when examining the example sphere file in Section 3.1. Import your new generated sphere .obj file in Blender and take a screenshot of the Blender GUI. Save the screenshot it as:
   `$CS148_DIR/HW1/images/sphere_gen_screenshot.png`
   and then run the `TODO Part 2` cell to make sure your screenshot shows up correctly. It should look similar to the following, but with a smoother sphere.



4 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! Two approaches that we'll discuss are polygon modeling and sculpting. 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.

4.1 Polygon Modeling

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.

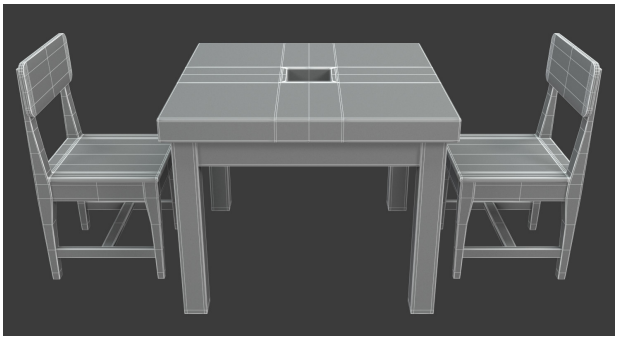


Figure 4: Dining table object made from modeling. Source: ArtStation

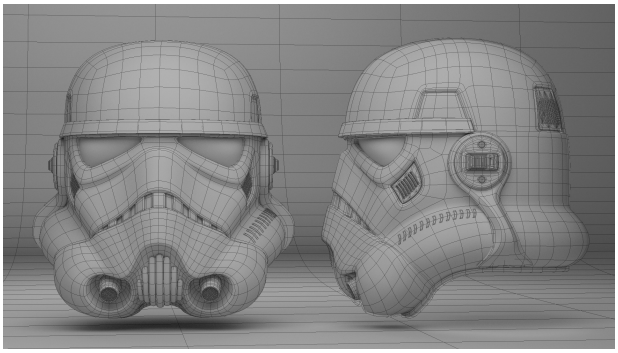
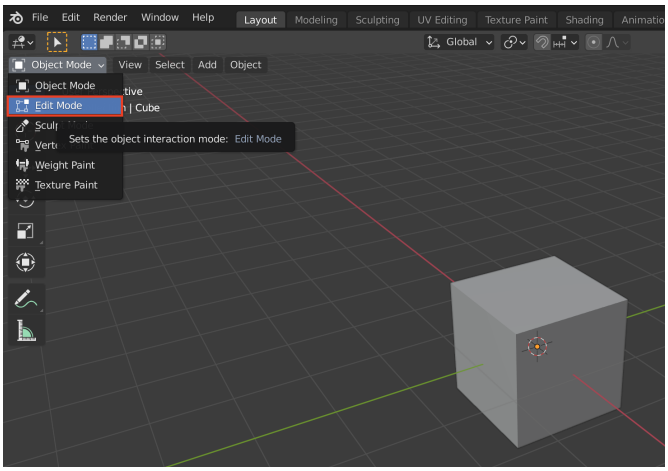
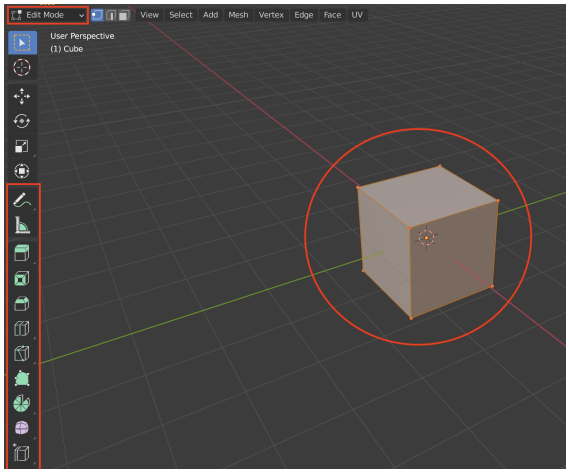


Figure 5: Stormtrooper helmet object made from modeling. Source: ArtStation

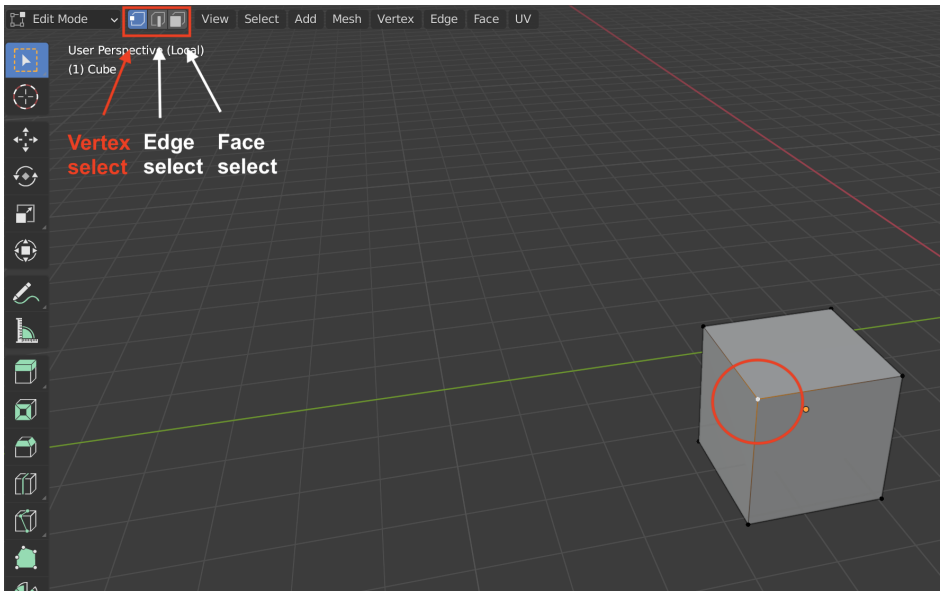
To access the modeling tools in Blender and make changes to objects at the vertex/edge/face-level, we will want to be in **Edit Mode**. Once in **Edit Mode**, you will notice that 1) new tools have appeared in the sidebar, and 2) you can now see the individual vertices, edges, and faces that make up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





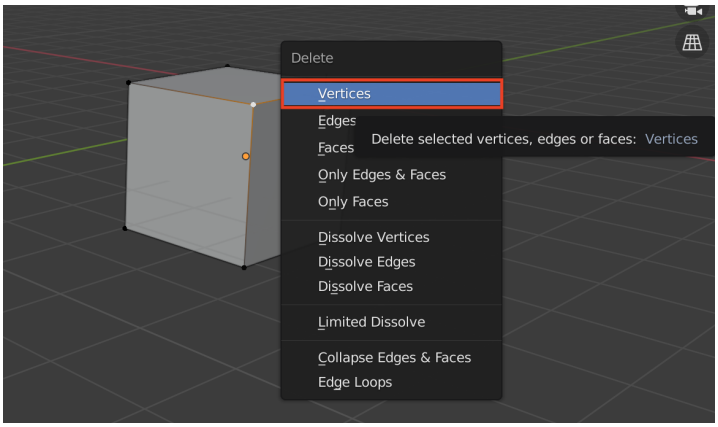
4.1.1 Selection

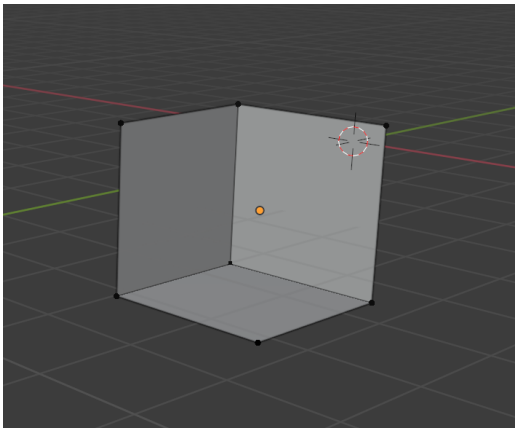
Simply click on a vertex to select it. If you want to select multiple vertices, hold **Shift**, and click on other vertices that you want to select. You can also toggle between **Vertex Select**, **Edge Select**, and **Face Select** in the upper-left, or alternatively with the **1**, **2**, and **3**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



4.1.2 Deletion

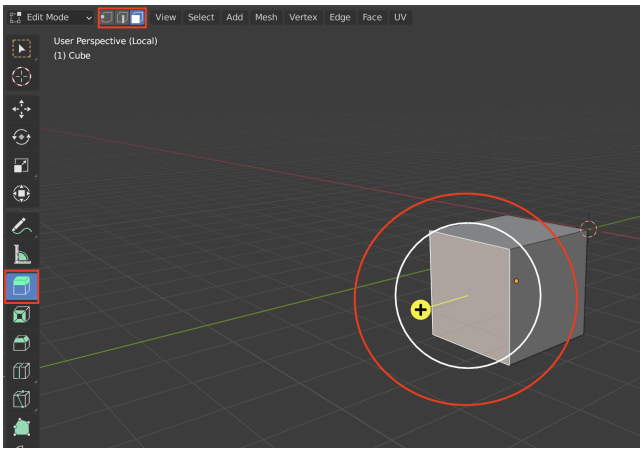
To delete a vertex / edge / face, simply press **X** while the vertex / edge / face is selected. This will bring up the "delete" menu, which you can use to specify exactly what you want to delete.

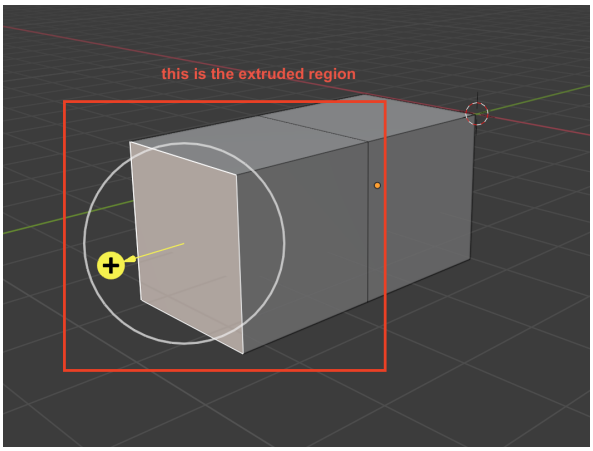




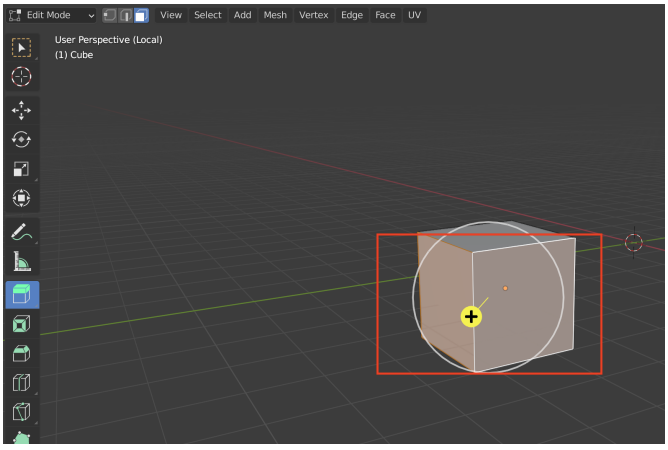
4.1.3 Extrusions

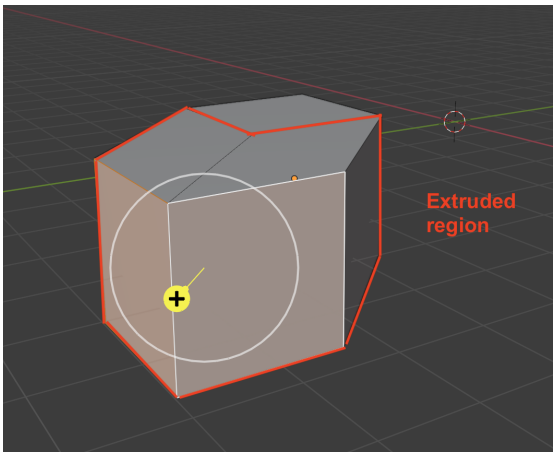
A basic but powerful mesh editing tool is the **Extrude Region** tool. To extrude a face, first select the **Extrude Region** tool in your toolbar. Then, select the face you want to extrude (switch to **Face Select** and click on the face). Then, click and drag the yellow + symbol. A keyboard shortcut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





4.1.4 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.

4.2 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.

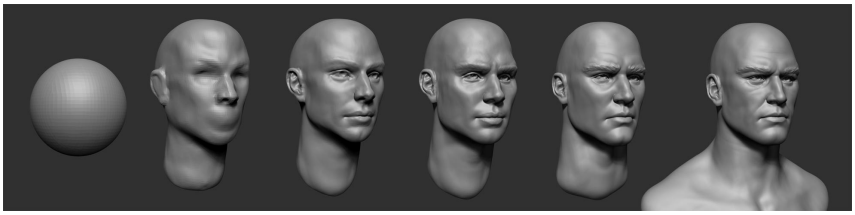
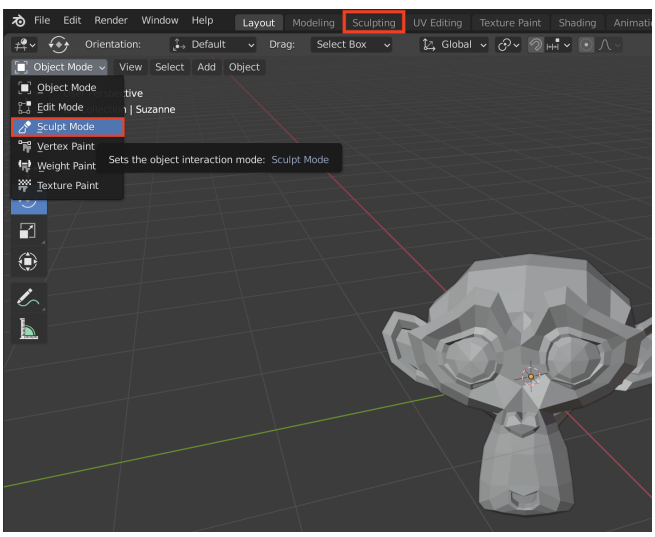
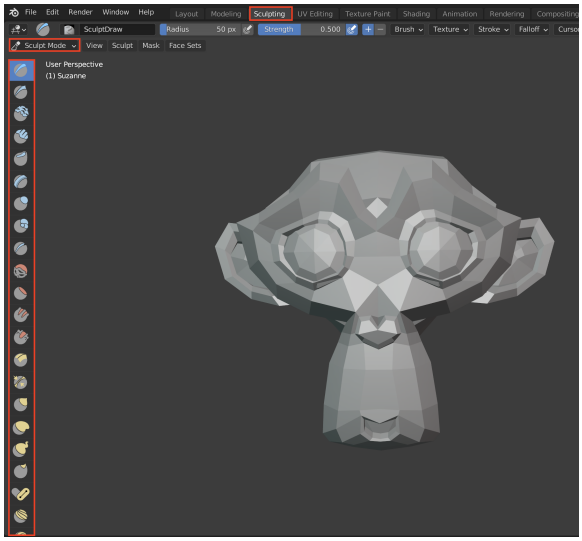


Figure 6: Scuplting a realistic face. Note that human faces are really hard to get right! Source: FlippedNormals

Figure 7: 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.



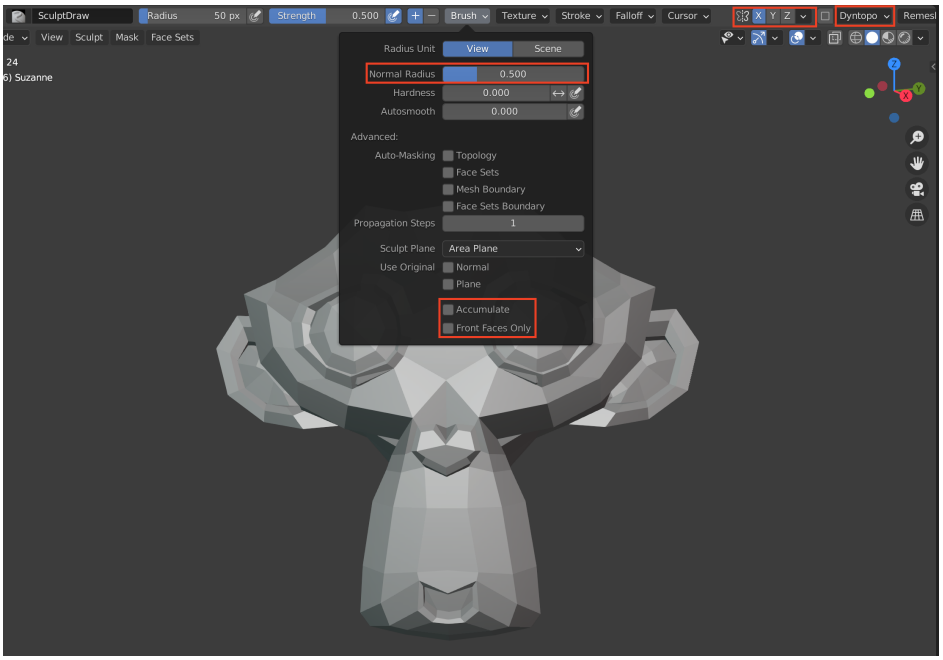


4.2.1 Brush Settings

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.

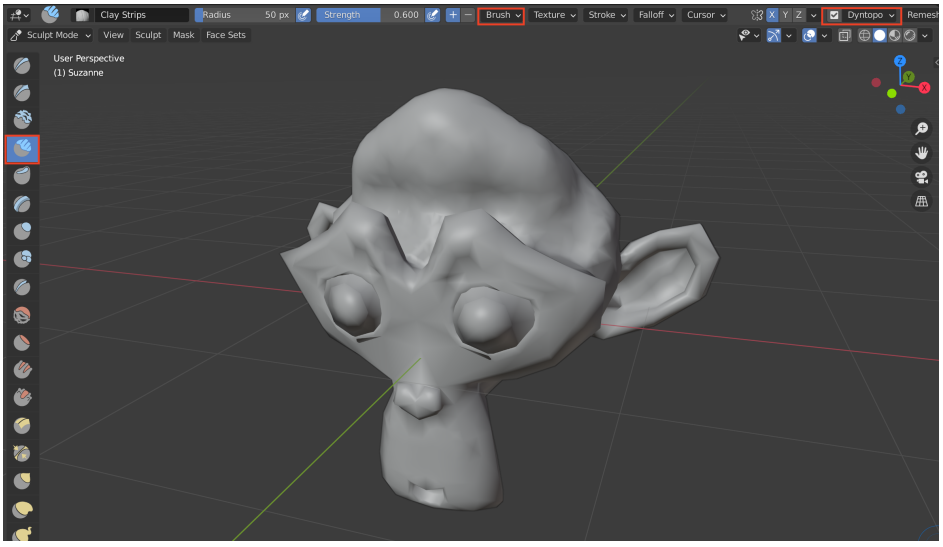
Two useful settings to know are the **Accumulate** and **Front Faces Only** options. Toggling **Accumulate** on causes brush strokes to stack on top of each other, while toggling **Front Faces Only** on makes it so that the brush only affects the vertices that you see from your viewpoint.

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.



4.2.2 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.



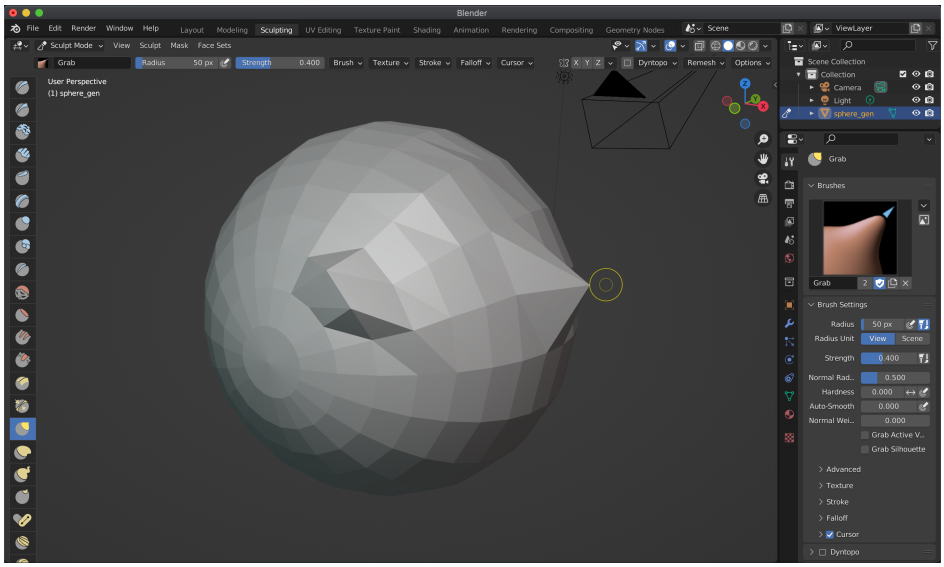
The above edit to the default monkey mesh (**Object Mode** → **Add** → **Mesh** → **Monkey**) was done with a **Clay Strips** brush with **Accumulate** and **Dyntopo** enabled to make the top of the head larger.

4.2.3 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.

4.3 Subdivision

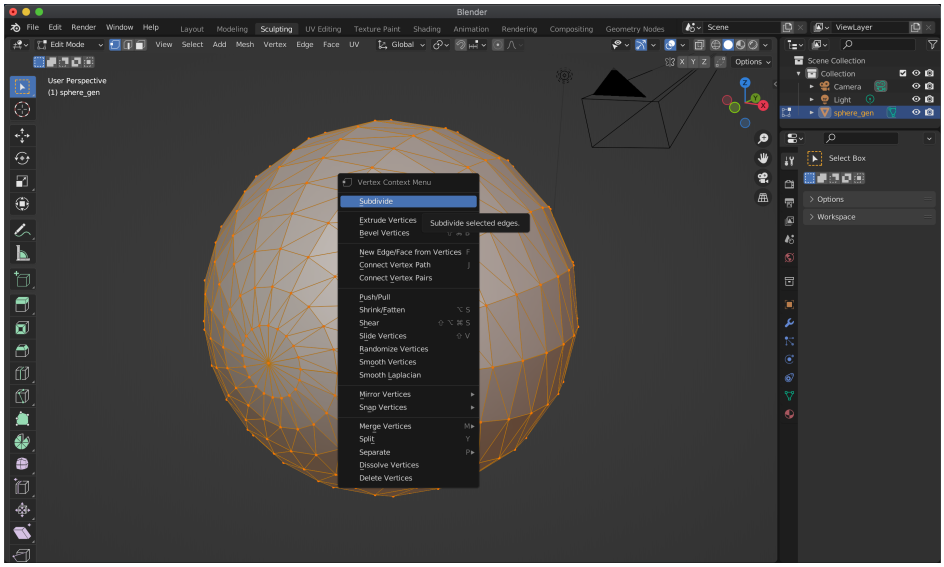
If you tried sculpting on top of your generated sphere from **TODO Part 1**, then you will notice that the brushes have very jagged effects, and grabbing a vertex will deform a large portion of the sphere into a big spike.

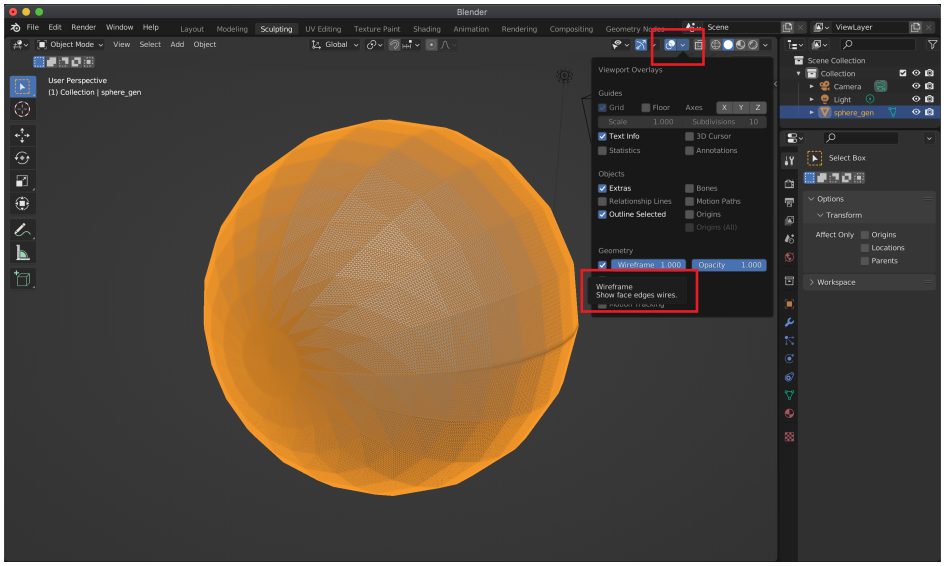


This is because our generated sphere is still relatively low-poly, even with 20 stacks and sectors apart. To generate more faces, i.e. more geometry, to work with, we can **subdivide** the faces on the sphere into more faces as mentioned in class.

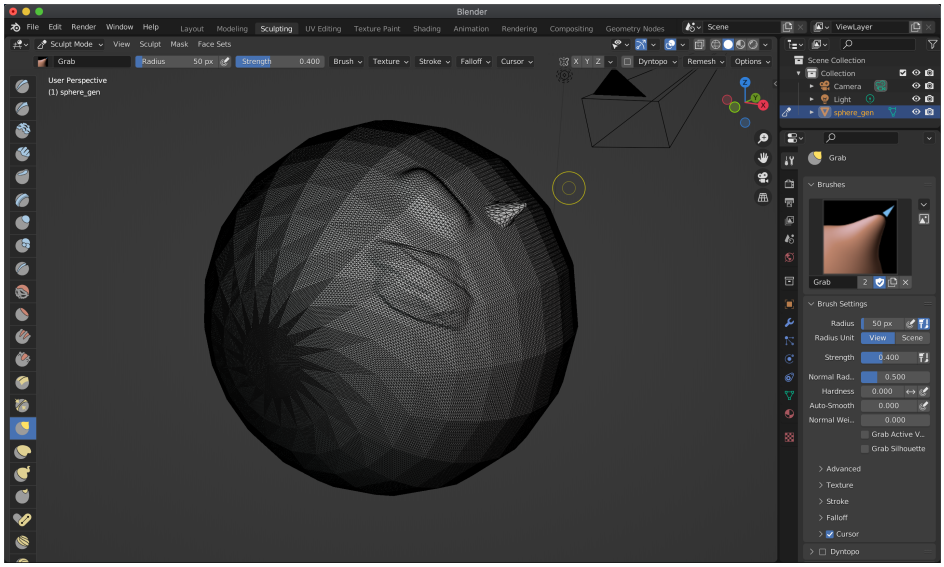
Go into **Edit Mode** and right click over the object you want to subdivide (in this case, the generated sphere .obj). One of the first options in the drop down menu will be **Subdivide**. For the sphere, try sudividing 3-4 times. With each subdivision, you will see more triangles litter the surface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

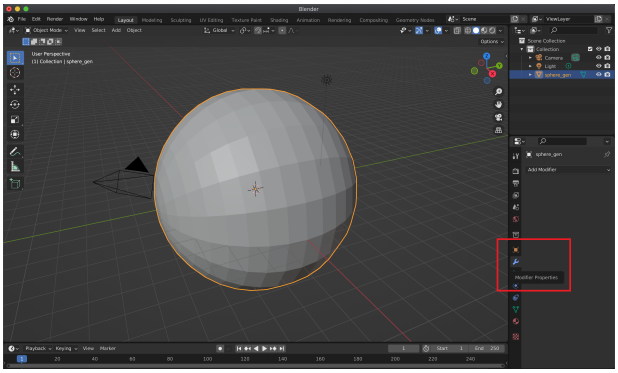


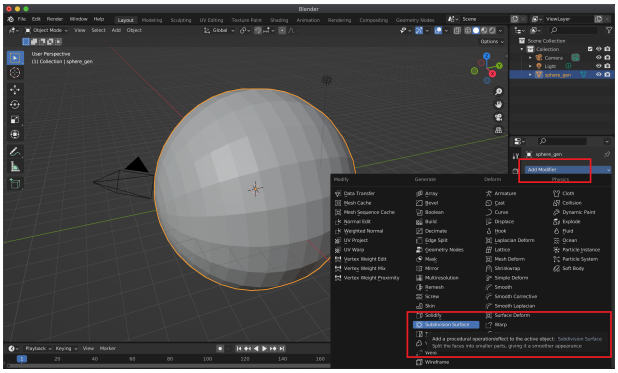


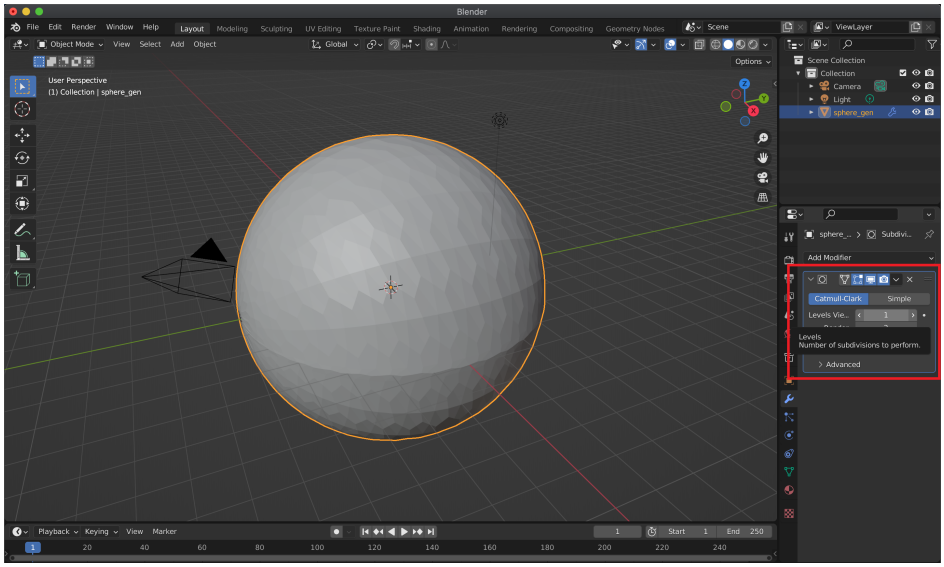
Now if you try to sculpt over the sphere, then you should see much more finer brush edits.



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.







4.4 Your Task (TODO Parts 3-4)

Action:

1. Use any of the mentioned polygon modeling or sculpting methods to make as simple (or as complex) of an object as you want. Save a screenshot of your Blender interface with your model as `$CS148_DIR/HW1/images/my_object.png`. Then run the `TODO Part 3` cell in `$CS148_DIR/HW1/HW1.ipynb` to make sure your screenshot shows up correctly.

2. Write 2-3 sentences describing what you did to make your object in the `TODO Part 4` cell.

Your object does not have to be too complicated. We just want to see you experiment a bit with the tools. The object can be as simple as just a few brush strokes on a sphere. Make sure that your screenshot shows the geometry (i.e. the vertices and faces) clearly!

5 Working with Geometric Transformations

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 with each other, you will apply them to a simple cube in alternating orders. This will be done in both code and Blender.

5.1 Your Task (TODO Part 5)

Action:

In `$CS148_DIR/HW1/HW1.ipynb`, find the `TODO Part 5` cell. We have provided you the vertices and faces of a simple cube mesh centered at the origin in world space. Using these vertices and faces, generate the following 4 .obj files of:

1. the cube rotated about the x-axis by 45 degrees, then rotated about the y-axis by 45 degrees.

2. the cube rotated about the y-axis by 45 degrees, then rotated about the x-axis by 45 degrees.

3. the cube translated along the x-axis by 1 unit, then rotated about the y-axis by 45 degrees.

4. the cube rotated about the y-axis by 45 degrees, then translated along the x-axis by 1 unit.

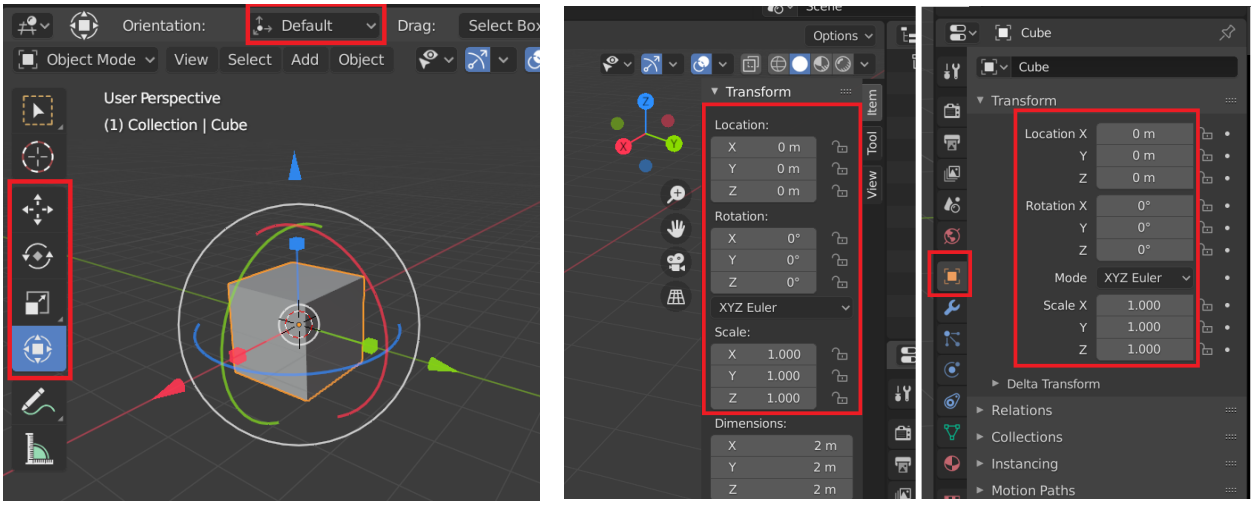
5.2 Transforming Objects in Blender

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.

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 **Property 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.

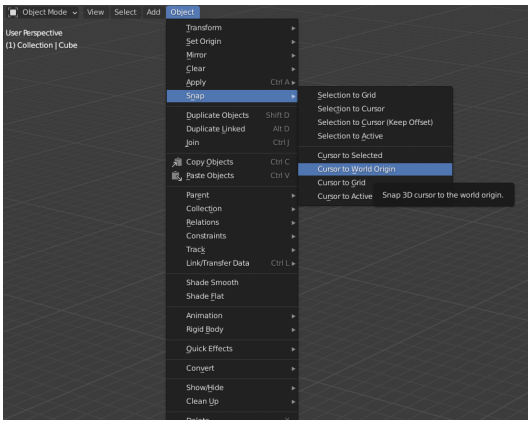


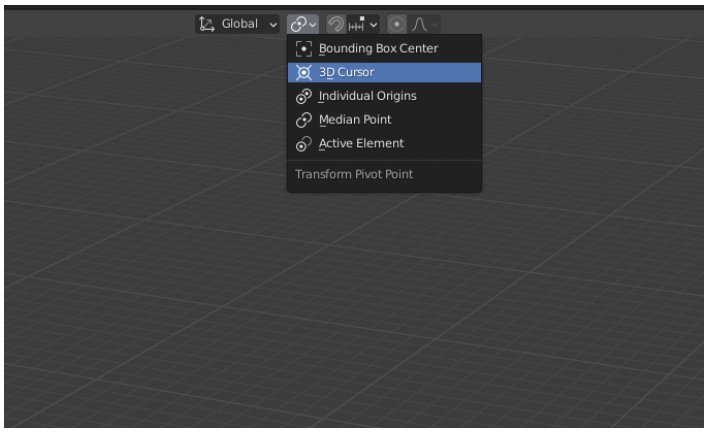
5.2.1 Rotating Objects about the World Origin

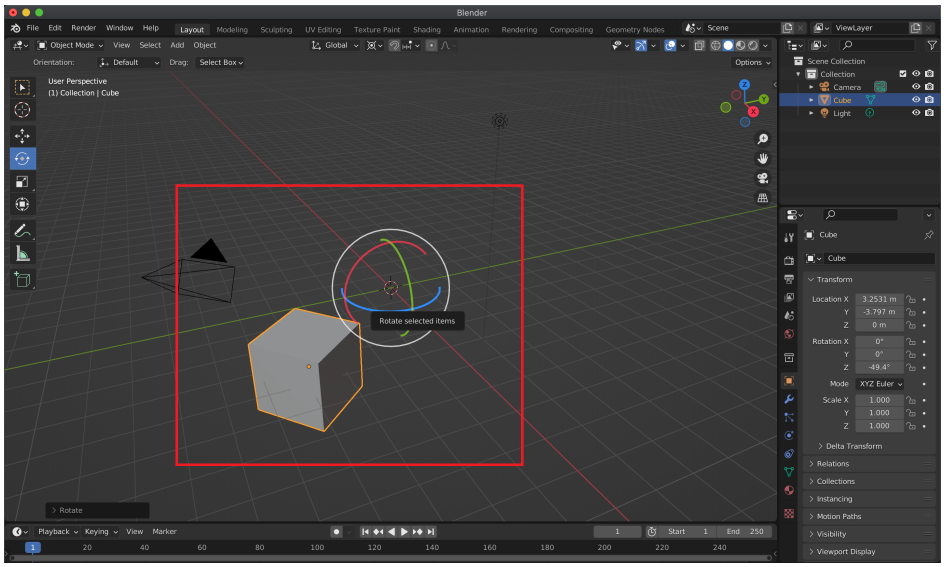
You may notice that if you try to rotate an object using aforementioned tools in Blender, then it only rotates around its own origin point rather than the world space origin. For instance, if you translate the default cube e.g. 5 units along x, then try to rotate about z, then the cube ends up rotating 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, go to the **Object** dropdown menu near the **Object Mode** indicator, then **Snap** → **Cursor to World Origin**. If you're in **Edit Mode** instead, then it will be

under the Mesh dropdown menu instead in the same spot. From there, 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.

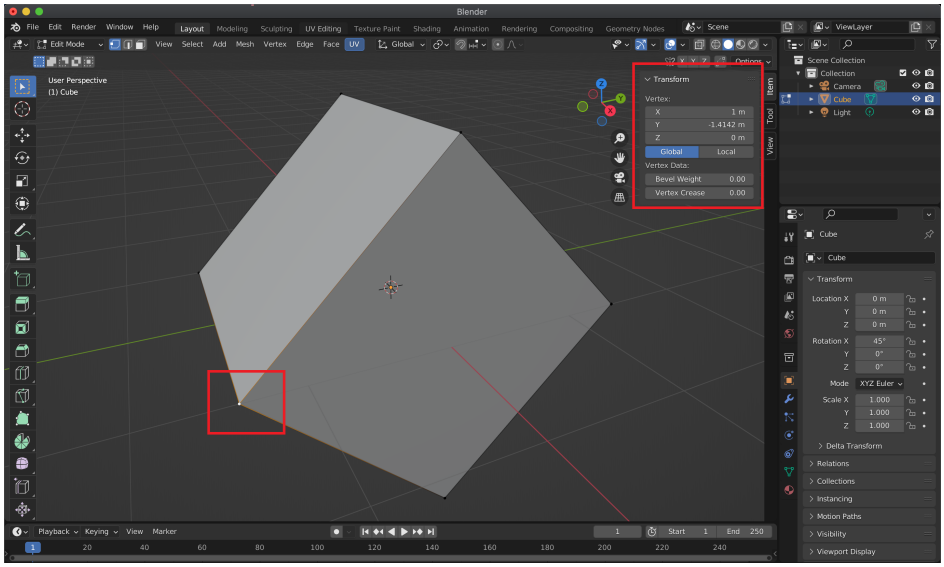






5.2.2 Viewing the Global Coordinates of a Vertex

Enter Edit Mode and select a vertex. From there, press the n keyboard shortcut to open the sidebar 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.



5.3 Your Task (TODO Part 6)

Action:

Create a new Blender scene with the default cube, and find the vertex at $(1, -1, 1)$ in **Edit Mode**. Then pull up the viewport sidebar with the **n** hotkey, and toggle to show the global coordinates of the vertex.

For each of the following transformations, take a screenshot of the vertex's **global coordinates** at the end as shown in Blender. You will likely want to do the transforms in **Object Mode** first, then swap to **Edit Mode** to view the vertex. For sanity's check, the transformed coordinates of this vertex should match those you computed in **TODO Part 5**.

1. the cube rotated about the **global** x-axis by 45 degrees, then rotated about the **global** y-axis by 45 degrees.

2. the cube rotated about the **global** y-axis by 45 degrees, then rotated about the **global** x-axis by 45 degrees.

3. the cube translated along the **global** x-axis by 1 unit, then rotated about the **global** y-axis by 45 degrees.

4. the cube rotated about the **global** y-axis by 45 degrees, then translated along the **global** x-axis by 1 unit.

Name the screenshots **$CS148_DIR/HW1/images/transform1.png** through 4 as specified in the **TODO Part 6** cell in **$CS148_DIR/HW1/HW1.ipynb**. Then run the cell to make sure your screenshots show up correctly.

Note: Those of you who imported the .obj files of the cube from **TODO Part 5** might notice that those looked different from the resultant cubes for this TODO. This is because whenever you import an .obj file, Blender adds a 90 degree rotation about the object's local x-axis to account for .obj files exported from other software that might use different coordinate systems. Removing this 90 degree rotation will give you the same results for both TODOs in Blender.