For the project, I wanted to choose a scene that I knew I could replicate but was also a challenge. When I saw the photo of the pumpkin, it was perfect. The image had a sphere, cylinders, cubes, and planes. First, I developed the table. Each part of the table is made up of cubes and connected planes. I built the vertices for all the different parts of the table. Having separate vertices allowed me to add different textures to different parts of the table. Next, made from planes, I built the walls, floor, and ceiling. Then, for the pumpkin, I chose to use a sphere. Online, I found and used a sphere file made by a developer named Song Ho Ahn. The main reason why I chose to use this sphere is because its smoothness can be changed. Pumpkins aren’t naturally smooth, they have ridges around the border, so having a non-smooth sphere adds to the pumpkin look. Next, for the cylinder, I also used a cylinder file made by Song Ho Ahn. I chose to use this cylinder file because the bottom and top radiuses can be different. This was perfect for making the pumpkin’s stem. Originally, I made the pumpkin stem out of a pyramid, but it didn’t look as natural. Though, for the Song Ho files, the shape was rendered using VBO and IBO only. So, to get the shapes to render properly in my program, I had to bind the IBO to a VAO. For the cloth around the pumpkin, I made my own shape. The cloth’s shape is like a cube, but the top is made from four triangles that connect in the middle. The top of the cloth indents, making it seem like pumpkin is sitting on it. Originally, I had a plan to use a torus for the cloth, but I could not get the shape to work properly. Then, I also used cylinders for the plant pots. In the photo, there are four plants, two taller plants, and two shorter plants. The short plants are hard to notice in the image because one is almost completely out of the photo on the left, and the other looks like it is part of the taller plant on the right. To make the pants, I just created a lot of triangles close together. Furthermore, just to add to the room, I added a rug. The rug is made from the same vertices as the floor, but it is scaled smaller and rotated. Then, for the window, I used a bunch of crisscrossing planes. A lot of calculations and drawing was involved to get the window planes positioned right. Lastly, I chose to have the point lights be spheres. I ultimately made the lights spheres to replicate the sun and to have a ceiling lamp in the room. The sun is tinted with a gold color and mainly uses diffuse and specular lighting. Then the ceiling lamp only adds slight white ambient lighting to the scene.

Going over the program’s navigation, I followed the rubrics to figure out what to include. The main navigations are the WASD keys and the mouse movement. The WASD movements are that the W-key moves the camera forward, A-key moves the camera to the left, S-key moves the camera backwards, and the D-key moves the camera to the right. Next, the moving the mouse allows you to move the camera’s view and to look around the 3D scene. Per the rubric, there were additional navigation criteria that had to be met. So, I also added QE key movements. The Q-key makes the camera go up and the E-key moves the camera down. Also, the P-key switches the projection of the 3D scene to be either perspective or ortho. Perspective projection allows you to move through the scene. On the other hand, ortho changes the view to look like a 2D image. Lastly, I changed the scroll callback to allow the user to speed-up or slow-down the speed of the camera’s movement. To speed-up, the user would have to scroll up on the mouse wheel or pull their fingers apart on a mouse pad. To slow-down the camera, the user would scroll down on the mouse wheel or pull their fingers together on a mouse pad.

One function that I used that can be reused in other programs is a key callback function. I used this function for switching between the projection and perspective modes. Originally, when the P-key was hit, it would switch between the two projections very fast. This is because OpenGL iterates the key press while the key is pressed down, even for a moment. So, sometimes it would land on the right projection, sometimes not. Using the class discussion board, a classmate pointed me in the right direction to solve this issue. Thus, creating the key callback function. With this function, OpenGL only calls the key press once, eliminating the continuous projection switch. This function could be used in a lot of situations, like games. For example, if a key needed to be pressed to shoot balls in a game, the key callback function could be used to shoot one ball at a time.

**References:**

Ahn. S. H. (2019). OpenGL Cylinder, Prism & Pipe. *Song Ho*. Retrieved from [OpenGL Cylinder, Prism & Pipe (songho.ca)](http://www.songho.ca/opengl/gl_cylinder.html)

Ahn. S. H. (2018). OpenGL Sphere. *Song Ho*. Retrieved from [OpenGL Sphere (songho.ca)](http://www.songho.ca/opengl/gl_sphere.html)