**Development Choices**

When determining the objects I wanted to use for this scene, I of course first had to ensure that the objects would meet the given requirements. The objects needed to be diverse enough to be represented through various primitive shapes, and at least one of them needed to be able to be represented using two primitive shapes. Knowing that the course was only eight weeks long, I tried to select objects that could be very basically recreated but also allowed room for more complex aspects to be added based on how well I grasped the concepts. I also looked at previously submitted projects so I could understand how proficient past students were by the end of this course.

I also knew that the examples we were given used cubes and our assignments required pyramids. I wanted to use those shapes since I knew there would be plenty of practice with them. Lastly, I knew that we would be applying textures to our objects, so I tried to select objects with contrasting designs that would be easy to replicate. Through a combination of using resources from LearnOpenGL, the tutorials provided for the class, and source code to render a cylinder, I was able to recreate these objects.

**Navigation**

As per project requirements, a user can navigate forward, backward, left, right, up, and down in the scene using the W, S, A, D, E, and Q keyboard keys, respectively. This is achieved using GLFW’s keyboard input and is managed in the processInput function. When an applicable key press is registered, this function manipulates the camera position vector accordingly. Additionally, the mouse cursor can be used to change the virtual camera’s orientation, and the mouse scroll can be used to increase and decrease the sensitivity of the camera. The calculations to make this possible take place in the mouse\_callback and scroll\_callback functions, respectively. GLFW records the pitch and yaw of the mouse cursor movements and the mouse scroll direction, and this information is used to manipulate the camera orientation and movement speed.

**Custom Functions**

The code contains functions to handle keyboard and mouse interactions with the scene since they are lengthy and do not need to interact directly with any variables in the main function. I was also provided with a cylinder class that contains all the calculations necessary to render a cylinder. This way, only the cylinder’s desired measurements need to be passed within the main code to generate this object.