For simplicity’s sake, I initially chose a desk with some objects on top (a lamp, a coffee mug, and a coaster) because I thought they would be fairly easy to put together using primitive shapes. The desk worked well as it was simple to put together using just a single primitive that was scaled in various ways (cube). I initially drew the legs individually but they were simple enough to allow me to create a loop to draw all of them – which improved the readability of the code by cutting down on the overall number of repetitive lines. I added a floor as a plane to meet the requirements for including at least 4 primitive shapes and am happy with the additional realism that it brings to the scene along with being a good foundation to build everything else off of.

After implementing lighting into my scene with a lamp on the desk, I noticed that the light from the lamp on the desk was not blocked by the desk and shined directly down onto the floor. This looked very unnatural, so I decided to move the lamp off the desk and to the side – which turned out to be an improvement. Initially, I planned on having the lamp stand connect to a light box directly under the lamp shade (like a light bulb; the lamp shade was going to be a pyramid without a base). But, due to time constraints, I decided not to include the light box and went with a more artistic representation of what a lamp could look like (with the stand connecting directly to the bottom of the pyramid (with a base).

I was going to have a laptop on the desk and use a series of planes on the laptop to emulate the look of a keyboard but due to fearing I would not have enough time to make it look realistic, I substituted the laptop for a simple chair in front of the desk.

The user has been provided with all of the controls necessary to view any object in the scene from any angle. Using keyboard controls (WASD), the user is able to move to different X and Y positions in the scene. Using Q and E, the user is able to change the cameras Z position. Just using those controls, the user is able to move to any XYZ position in the scene. But in order to view objects from different angles, the mouse was chosen as an input device to change the cameras tilt and direction. In order to move around the scene faster or slower, the user can use the scroll wheel to adjust movement sensitivity (forward for increased speed and back for decreased speed). Finally, by pressing P, the user is able to toggle between perspective and orthographic viewing modes.

My program includes four reusable functions that make the code more modular. The functions are related to processing user input (keyboard controls, mouse controls, scroll wheel controls, and window resizing). Each of these functions can be applied to other projects with no modifications in order to implement movement controls and window resizing.

Additional custom functions that could have been included, given additional time, are functions for binding vertices to individual VAO’s. I used a separate VAO for each of my primitive shapes (pyramid, plane, and cube). The code to bind the vertices to their respective VAO’s was nearly identical except for the name of the unique vertices object and the name of the VAO (each of these could have been made into parameters for a function that binds vertices to a VAO). The same thing could be done for loading our textures (I had four textures with nearly identical code for loading each one). I believe this function would have been fairly reusable in other programs that require loading textures for a 3d scene using opengl. Finally, although I believe it would have been the most complicated and the least re-usable of all the custom functions in my program, I could have created a function for drawing the objects. As parameters, I could have used the texture to be applied, the transformations to be applied, the VAO to bind, and the number of vertices to draw.