**CS 330 Final Project Reflection**

The reason I chose my selected objects is because I wanted something that not only adhered to the requirements by way of number of objects but also objects that were straightforward in their basic shapes. I felt by trying to go after overly complex objects or a profound scene would only complicate my ability to accurately convey the scene with only a basic knowledge of OpenGL. Another reason for my choice was because the objects in the scene are relevant to me and crucial to my success as a student. I rely on my view, my desk, my headphones, etc. not only for school, but also for work, so I tend to spend most of my time in this spot.

How I was able to program the required functionality came down to breaking each object in my scene into its most fundamental 3d objects. Also, figuring out which shapes work best in combination with another shape. For instance, the mouse was difficult to portray because I couldn’t configure a shape well enough to simulate the buttons for each control, so I did the best I could with adding in two box meshes and condensing them down and contouring them over the mouse to act as a left and right button. The keyboard was also another major component, as I needed to add each individual key on top of the keyboard in a way that made sense with the scene. This took quite a bit of time to figure out the best way to do it.

I set up the virtual camera with the ability to quickly zoom in and out of the scene and navigate around the scene easily. By using the mouse wheel, you can increase or decrease the speed as needed. This allows you to gain control of navigation at a precise detail when studying distinct aspects of the scene. This also allowed me to enhance some of the details of the objects during creation, like getting certain shapes to fit together just right. Included in the controls is also the orthographic and perspective views. By pressing ‘O’ on the keyboard, the camera will automatically shift into an orthographic view that allows the viewer to scene from a single angle, as if they were viewing the scene directly in front of the desk. The perspective view can be viewed using the ‘P’ key on the keyboard and is immediately shifted into the view of the scene where each object is viewed slightly above and at an angle to give the presence of depth. Another additional detail that is included is the ability to move the camera with pitch and yaw, allowing the viewer to slightly move the camera’s head in a multi-directional format. Lastly, the ‘W’, ‘A’, ‘S’, ‘D’, ‘E’, ‘Q’, keys all allow the viewer to adjust the location of the camera for optimal viewing. A viewer can freely float around the scene as needed and raise or lower, move right or left, or even zoom in and out of the scene to gain the best view possible when inspecting each meticulous detail.

This aspect is perhaps the one I struggled with the most, as it required looking through most of my code to see where improvements could be made. I realized the object that required it the most was the keyboard as at first, I had added each key individually via copy and paste. Eventually I settled on implementing a for loop that would add the specified number of keys at the starting position of the designated key. This made the code much more modularized and customizable, as I could specify any number of keys I wanted to include and the position. After figuring out how to implement this, it made the entirety of my code much more succinct and digestible. It also opened my mind about ways in which a for loop could be implemented for other details of different objects, but for the rest of the scene I chose to stick with the meshes I had originally drawn since they may not benefit as much from utilizing a for loop. Overall, this modularization increased the organization and readability of my code for others to inspect and allowed me to determine how to reduce redundancy in future projects.