Project Reflection

When it came to building my 3D scene out, I chose 4 objects, my monitor, 2 cans of soda, and my headset that I use. I chose the objects purely out of chance, the assignment at the beginning of the course called for us to take a picture of a scene, so I looked around my room and decided on my desk. I feel like the selected items were rendered as well as they could be with the meshes we had, and look like the objects or closely resemble them. They are also placed in the exact positions that they were taken in, and I tried to add color and textures to each to the best of my ability to portray the object, and with some parts like the cans and bottom of the monitor and headset mic, portray their reflectiveness in the light. Programming to the required functionality wasn’t extremely difficult, however I had not used C++ in quite some time, and this library was entirely new to me. The tutorial videos and assignments, the OpenGL documentation, as well as your help were extremely large parts in me being able to progress and understand how to work with the shapes and create a scene.

In my scene the user can navigate using the WASD keys as well as the Q and E keys. These keys allow a perspective movement through the scene, allowing the user to fully view each part of the scene and move through and around objects. The user can also use the mouse to change their views, flicking and using the mouse to change the speed of the camera as they need. I set up control using different input devices through several functions, like UProcessInput, and the UMouseScroll. These functions use the Camera.H classes definitions and in function definitions to allow the user to move the camera. Also through several functions there are calls to the cameraMatrix, which sets the views and perspectives. Finally, the user can use the O and P keys to toggle their perspectives, showing a basic view, as well as an orthographic view which shows shapes as they are on a 2d plane. This allows the user to fully grasp the setup of the scene as well as completely maneuver it.

Custom functions such as UCreateMesh allow for the shapes details and vertexes in the meshes.h file to be properly rendered into the scene. Without this function, custom vertexes would need to be made for every single shape, which would cause quite a bit of hassle. The header file also allows for the shapes to be resized to fit the needs of each object inside the scene. For example, there was no arching shape I could use to fully and accurately display the top of the headset. So I used a torus and stretched it incredibly thin over the top of the headset, so It could at least display the large arching fit the headset has, as well as connect to the ground, because I couldn’t have the headset floating in the air with just using cylinders. Many of the functions in the URender functions are set to be modularized away from each other, and commented clearly in chronological order so they can be properly understood and read as well.