Dominic Clapper

CS330

June 15, 2023

OpenGL Scene Reflection

So, the 2D scene that I decided to create in 3D using OpenGL was an image of my Nintendo Switch gaming setup. I really love this setup a lot, and I have been using it a decent bit recently to play Legend of Zelda: Tears of the Kingdom. This setup is pretty much my low table, my switch with the docking station, my wooden coaster, my laptop, my storage bins, and my pokeball figure. This setup is in my bedroom, where I have an overhead light inside of my ceiling fan, as well as some ambient light from my lamp and window.

When it came to selecting which objects I wanted to make in my scene, I was given a lot of options to select from since we only had to make four objects. My main thought when it came to selecting which objects I wanted to make was how easy I can replicate the scene, while also keeping it a bit simple in terms of shapes and workload. With this in mind, my first choice of the objects was the table. This table acts as the main starting point of my scene. All of the objects I had the option to make were going to be on this table, so I knew I had to include it. Plus, the table was going to be easy to make since it was just an array of differently shaped cubes and a plane. The next object I chose to make was my monitor. The monitor in the 2D scene I selected was somewhat the centerpiece of the scene, and I felt the scene would be incomplete without it. The plus side to adding the monitor was that it was not too hard to make in terms of the shapes needed. (Cubes and some planes) However, it did require a lot of shapes to make since it was a complex object. The last few objects I chose to make were my pokeball figure, my coaster, and my Nintendo switch + docking station. All of these objects were mostly personal preference because I felt they went well with the scene, and they are personal to me. They helped me hit the four object minimum as well. And even though the sphere for the pokeball and the cylinder for the coaster were not easy to make, I still enjoyed making them quite a bit. (I am glad I selected for them to be in my 3D scene) Also, using the pokeball and coaster in my scene made it easy to meet the four shape types requirement.

From here, there was a lot of coding involved in getting everything set up to make the project look good, but I was able to get it done after a lot of work. Some of the hardest things for me in regard to the setup were getting the lights and camera to work properly. I found more difficulty with the lighting setup because of how tedious and exact it had to be, which made me spend a lot of time trying to figure out what the heck I was doing. To be specific, setting up the diffuse lighting for my light objects was by far, the most annoying and difficult thing I had to do in the entire course. Moving on from lighting though, the next hardest thing for me was trying to setup my camera for the scene. Setting up the code for the camera took a lot of trial and error, and was overall, just a very tedious task. Trying to configure the input handlers was honestly very annoying because it required me editing the Camera.cpp class and the application.cpp class. (I had to configure my moveCamera method in the Camera.cpp class and the handleInput method in the application class) It was also difficult trying to set up the callbacks for the camera movement.

In terms of how users can navigate my 3D scene, the camera is pretty much the only way they can do so. The camera can move using the WASD and QE keys. These keys let users move up, down, left, right, forwards, and backwards. The user can also move the camera a bit via the mouse. By using the scroll wheel, a user could zoom in and out in the scene. And by moving the mouse cursor in any direction, the user can change where they are looking at.

Besides my camera and lighting difficulties, there was also a bit of a struggle when it came to setting up the organization scheme for my project. My 3D scene project utilizes a lot of different shapes, objects, and textures, which made it quite hard to keep organized. Heck, my one file where shape vertices and elements are stored is over 2000 lines long. Thus, I had to make a way which would keep things organized and reusable with ease. I had to divide things up in my include and src directories to keep the files mostly organized. This led me to making three different directories in the src and include directories for different files: core, game\_objects, and rendering. Beyond this, there were a good number of functions in my classes that were used to keep things more organized. Such as in my mesh class. In this class, I had constructors that took in vector objects of vertices, and vector objects of uints to make it easier to store the shapes vertex and construction data. There was also an initializer method in the mesh class that allowed for the vectors to be loaded and generated. Another good example of how a class that used functions to simplify my work is the material class. The material class really came in handy when setting up the lights and shaders for each of the objects in my scene. It had functions that made it easy to add a texture to an object. There are also some functions that I used in the game\_objects directory files that made it easier to create objects and alter them individually.