CS 330

Zach Beachy

Final Project Design Decisions

Justify development choices for your 3D scene:

In selecting the objects for my scene, I honestly didn’t put much thought into it. I tried to find something in my home that would be simple to build out on and that I could take some creative liberties on. Some of the more complex objects in my original picture didn’t make the cut, mostly because it would have been a difficult process to code in the various shapes that would be needed to create them. For example, I had a heart-shaped lamp that was in the original picture; it would have taken a lot of very intricately placed cylinders in order to replicate the shape. In order to program for the required functionality, I mostly relied on the SNHU-defined code logic that supplemented this course. Without it, I undoubtably would not have gotten to this point. I have taken this course before and failed it dramatically due to my confusion with the old course material, which required a lot more user-defined code. This was especially difficult as I was struggling to understand the concepts of 3D visualization on top of trying to learn OpenGL code. The SNHU-defined supplement made it possible for me to perform the necessary actions in the project and the entire course.

Explain How a user can navigate your 3D scene:

The user can navigate the scene using the keyboard inputs ‘W’, ‘A’, ‘S’, and ‘D’ to move about the x-axis/zoom-in and out of the z-axis. With the ‘Q’ and ‘E’ buttons, the user can move up or down about the y-axis. Additionally, the user can press the ‘O’ key to enable an orthographic view of the scene. The mouse wheel allows for the user to adjust the camera movement speed, making the keyboard inputs pivot the camera faster or slower. Finally, the user can use the mouse to look around the scene from the current camera position. All of the logic for this was programmed within the ViewManager.cpp file. Essentially, glfw has a function that checks for input from the user. Once an input is received, if the key pressed (or mouse movement/scroll) has defined logic for it, that defined logic will execute. Otherwise, the program continues to operate as before. This is possible with the glfwGetKey() function.

Explain the custom functions in your program that you are using to make your code more modular and organized:

The codebase relies very heavily on custom functions, however, they are not of my design. These are the various SNHU-designed helper functions that simplify the process of setting up the OpenGL environment. The caveat to this is the custom code that I have defined within these functions. These sections dictate where to load the textures from, how to define the different materials and how they should interact with the lighting sources, and the lighting sources themselves. The most impressive aspect of this code to me is how modular and reusable the functions that were defined by SNHU are here. For example, I am confident that I could not handle rendering properly without the SceneManager.cpp file. One of the benefits of OpenGL is its vast configurability, but this also can be detrimental to a new user. The code in SceneManager removes a lot of the headache that comes with configuring each low-level setting and allows the developer to focus on the more fun part of the OpenGL process, which is of course building the scene out. In a way, the SNHU-defined code organizes the OpenGL processes in such a way that the developer doesn’t need to focus on the low-level features very much at all if they don’t want to. This reminds me of something like a game engine, where a lot of the low-level configuration is obfuscated so that the developers can focus purely on the design of their game rather than the complex interworking pieces of the 3D graphics rendering pipeline.