Sarah Deleppo

Professor Holbert

CS 330

April 14, 2021

Design Decisions

When initially choosing the real-life scene to recreate in a virtual 3D environment, I carefully considered how this scene would translate to primitive shapes. I analyzed the primitive shapes that should appear in my creation VIA the project guidelines and chose a scene that would lend itself to more simple polygonal shapes rather than extremely complex objects. Therefore, I ended up choosing a scene that utilizes cubes, cylinders, pyramids, and planes to accurately recreate. I broke down the real-life scene into the following complex and simple shapes: the couch in my scene was broken down into a series of cubes to recreate its look, I simplified each mirror as a pair of cylinders with different textures to differentiate the mirror from its frame, each pillow on the couch is a pyramid-like structure, and lastly, I used planes for the floor and wall.

Keeping in mind how these objects appear in real-life allowed me to accurately place them in my scene relative to one another. Additionally, I utilized multiple different textures to give my 3D scene a similar look and feel to my real-life scene as well as to differentiate the objects from one another. This ended with my code having multiple different texture objects being used simultaneously on the different objects I was drawing. I had to carefully discern which objects would receive which texture within my URender function.

The user can navigate my scene in a few different ways. Using the keyboard, the keys WASD allow the user to move the camera forward, left, backward, and right to navigate the scene. Additionally, Q can be used for upward movement and E can be used for downward movement. The viewport display of my camera can be changed by pressing P, this switches between a perspective view and an orthographic view with each key press. Most keyboard functionality is processed within my UProcessInput function which utilizes GLFW to process which key is being pressed. The user can also navigate my scene using their mouse, where the user aims their cursor is the direction in which my camera will face so the user can easily look around. If the user wants to change the speed at which the camera travels, they can do so by scrolling up to increase camera movement speed or scroll down to decrease said speed. This functionality was programed using a series of callbacks to determine the mouse position and scroll direction and adjusting the camera based on this data.

One of the functions I developed for this project was my fragment shader source code and my vertex shader source code which allows for multiple lights on one scene or object. Rather than using one of the fragment and vertex shaders we are provided, I chose to write my own set of shaders for my scene. Within my fragment shaders, I have two lights, a directional light and a purple point light but more lights can be added using the set-up variables and Phong calculations within my source code. The code written within those two shaders can be reused and implemented in different projects to conduct calculations for specular, ambient, and diffuse lights as well as multiple lights being used on one scene or object.