Southern New Hampshire University

7-1 Project

Jehu A. Domenech Ortiz

CS-330-T2829 Comp Graphic and Visualization 22EW2

12/10/2022

During the development of this project, I had to research various ways of drawing the shapes that I used. While these shapes are simple drawing them in OpenGL is difficult. For my scene I decided to use simple shapes, for the floor I decided to go with a plane since it is flat, and it can easily be stretched. For the light fixtures I decided to go with a cube for the one on the right and a pyramid for the one on the left I believe these fit quite well for the scene. The counter is a combination of three different cubes which combined gives us a 3D rectangular shape. For the cubes I found that I was not able to use an EBO alongside textures or normal therefore I had to instead render the cube using 12 triangles with each vertex point written down with their normal, texture, and color. I was able to find a pyramid which allowed me to use an EBO alongside the normal and textures. The plan was an adaptation of the cube, and it works with no issues, I needed to change the y values to a smaller number so that flattened up like a plane and I also had to increase the x and z values to make the plane bigger.

To navigate the scene I am using W, A, S, D to move forward, back, and to the sides and Q and E to move up and down. I also integrated a callback for the mouse which allows me to navigate the scene depending on how I move the mouse. I can change the sensitivity of the mouse in the code and with the scroll-wheel callback I am able to speed up the camera or slow it down I had to set up a maximum and minimum for this value so that the speed wouldn’t infinitely increase making it impossible to control. While the default for viewing the scene is in perspective you can change between perspective and orthographic by using O and P keys.

I made a VAO, VBO, and EBO file for making the process of setting up the object simpler with function to bind, unbind, deleting, and linking. There are various shader files in my program. While originally, I was only going to only have one vertex shader and one fragment shader, I found that I needed to have a separate one for the light sources that are in the scene since they work differently than one for an object. I also made a separate file named shaderClass.cpp which have five different functions. We have get\_file\_content which reads a file and turns everything in that file into a string which is used in Shader constructor which takes in a vertex file and a fragment file for processing. There’s also a Activate() function to active the shader and a Delete() function for deleting the function. Finally there is a compileErrors() function that checks whether there was an issue during shader compilation.