Daniel Behmer

12/9/2022

CS-330

Prof. Rodriguez

CS 330 Project

Throughout this course, we have been learning about the essentials for OpenGL. We have learned how to compose and create objects, transpose those objects to a desired place within the 3D space, apply texture to those objects, and create lighting for the 3D space. Within my 3D space, I have included a cube, a sphere, and a cylinder, as well as camera controls that will allow the user to move around the space.

Because I chose more of a Christmas type theme for my project, I decided to have these objects because they would be able to capture the items that are typically used during Christmas time. The cube is the shape of a typical present, I have a shorter cylinder that represents a log pit for a fire, and I have a sphere that represents an ornament that can be found on a Christmas tree. We were also given supplemental videos during this course that were able to provide guidance and help when it comes to designing, building, and implementing our shapes.

With these shapes within my project, I was also able to provide textures to each of the shapes, although only one of the shapes came out the way it was intended, the log pit. The other shapes do have textures applied to them, but they do not look as intended, in particular the sphere. By using the camera controls, the user can traverse the world and notice that the texture is wrapped around the sphere, but not as intended.

Within the 3D space, we were also tasked with implementing camera controls. In my controls, I included both keyboard and mouse controls. The W, A, S, and D keys will control the camera along the X and Y axis. Each key from the keyboard will do something different. The W key will bring the camera forward along the Y axis in the 3D space. We do this by creating if statements for each key.

The W key’s algorithm is as follows:

if (glfwGetKey(window, GLFW\_KEY\_W) == GLFW\_PRESS)

cameraPos += cameraSpeed \* cameraFront;

The A key’s algorithm is as follows:

if (glfwGetKey(window, GLFW\_KEY\_A) == GLFW\_PRESS)

cameraPos -= glm::normalize(glm::cross(cameraFront, cameraUp)) \* cameraSpeed;

The S key’s algorithm is as follows:

if (glfwGetKey(window, GLFW\_KEY\_S) == GLFW\_PRESS)

cameraPos -= cameraSpeed \* cameraFront;

And lastly, the D key’s algorithm is as follows:

if (glfwGetKey(window, GLFW\_KEY\_D) == GLFW\_PRESS)

cameraPos += glm::normalize(glm::cross(cameraFront, cameraUp)) \* cameraSpeed;

If we take a look at the difference between each key, we can notice a pattern between them. We notice that the W and S keys are opposite in a sense that the W key increments positively along the Y axis, while the S key traverses along the Y axis negatively by decrementing. The A and D keys follow the same pattern, but along the X axis instead.

The mouse will control which angle the camera points for its direction. Imagine the mouse controls a pair of eyes. When moving the mouse up, the camera angle also points upwards. If you move the mouse downwards, the camera angle will now be looking down upon the scene, and so on for the left and right movements. These mouse movements will control the pitch and yaw of the camera. The pitch refers to the “side-to-side” axis, or the X axis, while the yaw refers to the vertical, or Y axis.

Within this project, there are multiple functions that were developed to help keep the code modular and organized. One of these functions is the ourShader.use(); function. The point of this function is that it tells OpenGL which texture unit it belongs to. Another function that is used is C.render();. The point of this function is to render and “draw” the cylinder in the correct position. And lastly, we have the function S.Draw();. This function draws and renders the sphere that is used within the 3D space. The point of these functions is to keep our code from being cluttered. We are able to create multiple C++ files within one project, keeping the code clean, dry, and efficient.