Project Reflection

Matt Smith

06/13/2023

I had no experience with OpenGL prior to this class so I wanted to pick objects that would cover all the basic primitives from the list in the original prompt. I also wanted to make sure the objects I selected were simple so I wouldn’t overwhelm myself trying to create them throughout the course. The objects I ended up selecting were a hard drive, a pencil, a toy ball, and a toy donut because they were close by and fit the requirements for the project. I picked the pencil as my complex object because it required multiple primitives to complete and seemed simple enough at the time. I later figured out it was a little more difficult than I originally thought but I was able to get it to look how I wanted.

I was able to program for the required functionality because of the many resources provided to me during the course as well as reading the tutorial code to see how everything worked together. When it comes to the 3D objects, the provided meshes were a huge help, considering they provided the vertex positions, normals, and texture coordinates. The only thing I had to genuinely concern myself with was getting the correct textures for each of my objects.

A lot of the textures I used were from the Demes, L. (n.d.) website. The textures that I had to create myself were the ones for the pencil body and the pencil ferrule. To create these textures, I used my phone to take a picture of my pencil. From there, I transferred the picture to my computer and used image editing software to crop the body of the pencil and the ferrule to create two separate texture images. To achieve my complex texturing, I separated the yellow part of the ferrule from the green part so I could overlay them on the ferrule.

When it came to lighting the scene, I positioned one of the lights approximately where it was in the real-life scene in relation to the objects. To try to match the color of the light, I downloaded a picture of the product box for the light and used a color picker to determine the RGB value for the real light. I converted it to its float value and assigned those values to the main light color. The lighting for the scene at the point still felt like it was missing something, so I added a light directly above the plane and set the specular intensity low enough to deal with some unwanted shadows.

As far as placement of my objects is concerned, I first worked on getting the pencil to a reasonable looking size and used its size to create a plane that seemed to match the size of the real-world table using the pencil as a reference. From there, I rendered the rest of my objects into the scene and again scaled them using my pencil object as a reference as well as my picture of the scene. I also chose to make the world color a grey color so that if a user switched to ortho mode, they would be able to see the pencil in the scene.

A user can navigate my scene using in multiple ways. They can move the camera forward, backward, left, and right using the keyboard buttons W, S, A, and D. They can also move the camera upward and downward using the keys Q and E. There is also control of the speed the camera moves by moving the mouse wheel forward to speed it up, or backward to slow it down. The user can also change the camera orientation by moving their mouse. To achieve this, I have a few functions in the program that actively check for the user presses specific keys, moving the scroll wheel, and moving the mouse around. The camera.h header file had most of the movement functionality required but I had to implement some changes such as what happens when a user presses Q, E, or P. I was able to allow the camera movement speed to be adjusted by using the existing MovementSpeed variable inside the file and having it adjust depending on how the user moves the mouse scroll wheel.

The main functions that are modular and organized are contained in the meshes.cpp file of my program. It creates the meshes for all primitive shapes and can be easily implemented into other OpenGL projects that require them. I also customized a few functions for the camera movement and those functions would be UProcessInput and UProcessScrollCallback. In the UProcessInput function, I added statements for when the user presses Q, E, or P and I added those keys into the camera.h file. I edited the UProcessScollCallback to handle when the user scrolls the mouse wheel forwards or backwards and I used the existing MovementSpeed variable from the camera.h file to do it. These functions are reusable because they can also be easily added to an OpenGL program to handle camera movement with minor code updates in the camera.h file.

References:

Demes, L. (n.d.). *3Dassets.One - the asset search engine*. 3Dassets.one - The asset search engine. https://3dassets.one/#order=latest