Final Project Reflection

I had a significant amount of ups and downs during this project considering I have never worked with anything other than console applications. Every week brought on new challenges that I had to overcome with week 3 or 4 being the worst due to overestimating my understanding of how the information was processed on the GPU side. In hindsight I’m happy with the objects that I chose for this project.

More specifically I’m happy that I chose cylinders to represent some of my objects due to it’s complexity. During week 3 or 4 when we were tasked with creating our complex object made up of 2 or more shapes mine happened to be the ballpoint pen. I chose to create my object out of a cylinder for the main body and a pyramid for the head. The pyramid creation was simple enough considering I manually mapped out each vertex, normal, and texture coordinates, but the cylinder proved to be more difficult. Luckily, I was able to source an algorithm for vertex creation, and then I modified it further to get the normal and texture coordinates.

The camera manipulation was one of my favorite additives to the program. Using matrix manipulation, I gave the sense of a camera moving in the 3D environment. The user can freely move about the scene using WASD for forward, left, down, and right camera movements respectively with the arrow keys giving the same functionality. The user is also able to control the speed at which the camera moves around using the scroll wheel on the mouse, scrolling forward slows the camera down while scrolling backwards speeds it up. The last user input ability will be using the Q and E keys which moves the camera up and down respectively.

I actually created multiple classes and functions to help cut down on the code in the main cpp file with more to do in the future for a continuation project. One function I created, that doesn’t really help with reusability per se, is my toggleInput function. This function allows the user to toggle between looking at the scene orthogonally or perspective. I did this because I noticed that when I placed the key input inside of the processInput function it would “jitter” between the 2 view points and settle on one randomly. I realized this was due to the processInput being inside of the main loop and it was being called as fast as my processor would allow. I’m sure there are better ways of creating the same functionality without creating a whole new function that just monitors key press events, but this is what I came up with.

I also updated the original URender function to allow more parameters to passed in and took some functionality away. The only thing left inside the URender function now is the assignments of the matrices for the model, view, and perspective views. This allows me to pass in matrix manipulation properties for each of my objects in the scene individually rather than have them all manipulated the same way. I also created a new URenderLight function to pass in the physical representation objects for the lights.

My object creation functions allow for the best reusability at the moment. Especially the cylinder function which gives you the option of providing a radius, height, and number of triangular sections to make it more or less defined based on your choosing. I currently have it set at 100 slices to provide a very smooth circular appearance.

Mesh, VAO, VBO, EBO, Shader, and Texture classes were also created to assist with a more modular design to the project. Although the main cpp file is still crowded, in my opinion, there is much more modularity than there once was. I was unable to complete the full design that I had in mind for the whole project due to time constraints, but I will discuss how I will finish this project later on. The next update I want to implement is creating a camera class. This won’t clean up the code significantly, but it will do fairly well. The next big update will clean up quite a bit which will be implementing classes for basic object creation. My cylinder, pyramid, cube, and plane functions will be made into separate classes, and the vertex, normal, and texture coordinates will be auto generated with just a few user input parameters needed.