Mauricio Bautista

CS 330: Computational Graphis & Visualization

Southern New Hampshire University

Prof. Wabara

February 20, 2022

**Reflection: Design Decisions**

I selected the objects in my scene for two reasons. First, they were the less complex primitives to draw. Although I said less demanding, they were complicated to draw on the plane. Second, I could not find or figure out how to draw other primitives like spheres, torus, or cylinders. Also, since the week we were assigned to draw a pyramid, I was confused with the coordinate system. As a result, I could not get the pyramid to stand straight. It wasn’t until this week that I took extra time to play around with the pyramid and finally figured out how to rotate it to stand straight. This extra time spent rotating the pyramid was essential to build a smaller one in my final scene. The pyramid is supposed to represent a cone. The webinars and following along with them was a necessary part of the process to accomplish the result.

The user can navigate using different inputs like the mouse and the WASD and QE keys in the scene. For example, the W key moves the camera towards the screen, and the S key moves the camera away. Similarly, the A key moves the camera to the left and the D key to the right. Finally, the Q key moves the camera on top of the objects, and the E keys move the camera below the objects. In addition, if the user presses the P key, the camera will project an orthographic view and a perspective projection. This option allows for a different perspective of the scene. Another way the user can navigate the scene is by pressing the mouse’s left key and holding the Alt key simultaneously. This option allows the user to navigate the scene from all angles. In addition, the scroll wheel on the mouse allows the user to zoom in and out of the scene.

Several costume functions in the program make the code more modular and organized—for example, the glBindVertexArray function used in the program. I used the function before drawing a primitive with a different parameter every time. Other functions similar to glBindVertexaray were the glGenBuffers and the glGenVertexArrays. The functions were reusable and only required the change of one parameter depending on which VBO, EBO, or VAO I was creating. Of course, there are many more functions that we can reuse to keep the program organized, like the glVertexAttribPointer and glEnableVertexAttribArray that specify the attribute location and layout to the GPU. To conclude, I learned that most of the code could be reusable and effective by modifying it as you go.