Hatcher Blair

CS-330 Comp Graphic and Visualization

Project Reflection

8/13/2023

Most of the objects in the scene are cubes, there are 63 cubes in the scene. However, they are different widths and sizes. Each row of the keyboard has a different angle on the top of the key and there are keys that are different sizes for certain keys. To avoid having several thousand lines of vertex arrays I created a function that would make one row of keycaps at a time. The function would edit the vertices depending on the scale of the keycap and then place them in the scene. Once complete it returns a 2D array containing the vertices for all the keycaps in the row. The code for the cylinders and tori used was modified from the code that was provided to allow me to use my already existing class structure for storing the data related to objects. I modularized my code as much as possible, creating classes to handle each individual element of an object and to render objects. This meant that almost all the logic was outside of the main function and the purpose of the main function was to link it all together.

To navigate the scene, I implemented simple WASD controls to move the camera as well as mouse controls to allow the user to look around the scene. You are also able to use the Q and E keys to move the camera up and down in the Y direction. To switch between a 3D projection and a 2D projection you can press the P key. Finally, there is the ability to click mouse 1 to enable to cursor and detach the mouse input. This allows you to lock the camera in one position as well as makes it easier to navigate through different windows during the debugging process. By default, mouse control is enabled but with a simple left click it can be disabled to allow you to have control of your cursor again.

There are a lot of custom functions throughout the code, most are modular, but there are a couple that are not very modular and are only used for a specific purpose. In the mesh class there are several draw functions that are used to render an object on the screen. These functions take a shader object and a camera object as inputs and render the object that was used to call the function. There is a function for drawing an object with indices, without indices, using triangle strips, and using triangle fans. With these four functions you can render a range of different objects using different methods, in this project it was used to render cubes, cylinders, tori, and a plane.

Within the keyboard class there are several functions that are used to create the keycaps and the base of the keyboard. The createKeyCaps function takes no inputs and returns a vector of type mesh containing all the keycaps for the keyboard. This function could be modular with some modifications but is currently not modular so that you do not need to pass anything into it. It calls the function createKeyCapRow that was talked about in the first paragraph five times and then creates the index array and texture array for all the keycaps and puts them into one vector. This function also calls another custom function called RotateKeyCap that rotates a keycap to be parallel with the top of the keyboard base. This function applies the rotation matrix to the vertex that is input and returns the new vertex position. This function is also not very modular but could be made more modular if it was modified to input the rotation axis and the rotation amount instead of using predefined values. There is another implementation of this function in the torus class. There are also functions for creating the keyboard base and the lighting for the keyboard. Both functions use predefined arrays of vertices and indices to create the base of the keyboard and the light on top of the keyboard respectively.