Charles French

CS 330

Final Project Reflection

I chose the given objects for the scene – a cube, two planes, a circle, and a torus – as a good mix of different object types without being too complex. After having issues with the freeGLUT library as well as issues determining a custom setup for a thicker torus, I decided to go with a torus made of a series of planes, which ended up working well with the chosen number of sides for my circle. I chose to keep the polygon count for it and the circle rather low (8 sides) since for one part, I was hard coding the vertices for them, and two, I liked the look of fewer polygons with the other objects in the scene, giving it sort of a “Playstation One” style of look. I was able to texture everything appropriately and add lighting effects per the requirements, and employed WASD style camera movement with up and down controlled by Q and E, respectively, by utilizing the appropriate keypress callback events.

In addition to the WASD movement, the camera direction is controlled by steering the mouse, utilizing mouse movement callback events to change the yaw and pitch of the camera as appropriate per the given mouse movements per frame. If given the option, I would have set the camera to move up via pressing spacebar, and camera downward movement to X, as that is a more standard layout for programs such as games that allow for movement along all three axes, and would be more intuitive for the average user.

Custom functions I used to tidy up the code include a loadTexture and a InitObject function to reduce redundant code blocks that would otherwise be copied for the eight textures and six objects in the scene. The loadTexture function takes a string for the filepath of a given image, and generates a 2D texture assigned to an unsigned int declared along with the string. The InitObject function takes an array of vertices and the sizeof() value of the given array, generates a VAO for the given vertices, along with vertex attribute pointers for x y z coordinates, normal coordinates, and texture coordinates. After creating the VAO, it then adds it to a VAO array to help make rendering the object easier later on. Both of these functions work to make as many objects or textures as needed, and build them in a uniform way to help reduce errors from possible mistyping, as well as save on upwards of a few hundred lines of redundant code.