Katie Stapleton

CS 330

M7 Final Project

07/06/2021

Final Project

My scene is an indoor view of a living room. I choose the couch, side dressers, lamp, balloon, door, painting, floor, and wall. The floor and wall are needed to define the dimensions of the living room. Two planes were defined and used repeating textures. The couch is the primary object of the scene. Due to time constraints, the couch in the OpenGL scene is simplified to a basic couch structure. The lamp serves as a source of light. Due to lack of the proper lessons, I could not figure out how to perform the draw pi method. The top and bottom were calculated with a circle-calculation software and then converted into OpenGL coordinates. I charted out the vertices by hand. The balloon was not necessary, but helped complete the requirements for the project. Like the lamp, the balloon is a sphere that is calculated using a circle calculator and calculating the connections by hand. The painting and door were not a necessity, but were included to create a more realistic scene. The painting and door are transparent images of a painting and a door. It was more practical to create planes than try to reconstruct a door and painting with OpenGL.

A user can navigation through the 3D scene using a mouse and keyboard. Controllers and other devices were not programmed for usage. The mouse and keyboard are the best selection because the devices are common, usually considered necessary to operate a computer. The keyboard and mouse navigation are like the standard controllers of a PC gamer, but on a more basic level. The keys of ADWS move the scene to the left, right, forward, and backward. The keys of QE control the upward and downward movement. The P key switches projection between perspective and orthographic. The mouse movement changes the orientation of screen with the ability to loop up, down, right, and left. The mouse scroller adjusts the speed of the movement when navigating.

For this project, I found it challenging to develop custom functions. The book used a more procedural/functional approach rather than object-oriented, which is the programming style I use. It was hard figuring out what methods and variables needed separate instances for each object versus reusing the same function. The project is not as modular as I was hoping for, but it is still modular. A large amount of coding is all in one file instead of split into multiple files. Unfortunately, understanding how to separate the code into separate files was introduced in tutorials too far into the semester, making it unpractical to reset the project.

I created multiple reusable custom functions for this project. According to a post on stack overflow, the vertex shader only needed to be defined once rather than for each fragment shader. The fragment shaders used the same base code, but were separated and customized to fit the requirements of each object. The navigation functions are in separate classes from the rest of the code because the overall purpose of the methods are the same; it is important to prevent conflicts or missing navigation features. change or add/remove navigation features. Each object has its own mesh and draw function. The draw function only requires a new set of attributes per class instead custom coding within the function. The largest benefit is the mesh and draw functions can be called upon, edited or even deleted without affecting the functions used for other objects.