Cameron Winningham

Southern New Hampshire University

4/18/2021

Design Decisions

For my 3D scene, I decided to put together a various assortment of objects on my table all with different shapes and sizes for variety. The first object is a globe which I created from a pyramid for the base, a sphere for the globe, and a cylinder for the axis. The other objects are created from cubes, a cylinder, and a torus.

A user can navigate my scene using standard WASD keys to move and using the mouse to look. These controls are standard in the industry so anyone can run the application and know exactly how to move around in the scene. I also added the ability to move up or down relative to the direction the user is facing using the Q and E keys. Lastly I added a “run” feature where you can hold the shift key to move faster or use the mouse wheel to increase or decrease movement speed.

In the source code, I made all of the components as modular as I could by separating different functionality into different classes and files. For example, all of the code to create and render a cube is contained within the “Cube” class located in the “cube.cpp” and “cube.h” files. Functionality that is shared between different classes such as Cube and Pyramid was further modularized by the “Model” and “Mesh” classes located in the “model.cpp” and “model.h” files. I have designed these classes in a way which you could copy the files and paste them directly into a different project to re-use them. For example, if I wanted my “Cube” class to be in a different project I would copy the Cube class files (cube.cpp and cube.h) plus its dependent files (texture.cpp, texture.h, model.cpp, model.h) and paste them right into the new project’s source code directory.

Overall, designing my 3D scene was a very fun and enlightening experience. I have learned more about OpenGL while creating clean reusable code that I could easily use in future projects. I also proved to myself I can visualize a 3D scene in my head and translate that vision directly to code which renders the scene.

Scene Photograph:

A computer on a desk

Description automatically generated with low confidence

Final Rendered Scene:

A picture containing text, indoor, electronics, display

Description automatically generated