Kyle Lund

CS-330

12/9/2023

# Reflection

## Design Decisions

When choosing my objects to create a 3D scene with, I wanted to choose items that would be complex enough to make a scene interesting without being too difficult to create. I chose four different objects three basic shapes and one that is more complex. The objects I chose were a Rubik’s cube, glass marble, rubber ring, and a glasses case. Each object was simple with the case being a complex object consisting of triangles and rectangles. When arranging the items, I wanted them to be arranged in a more complex way than simply sitting next to each other. I decided to put the marble inside the ring and have the case lean on top of the Rubik’s cube.

### Requirements

The programs required functionality includes, having multiple rendered objects that are textured along with a textured plane for the objects to sit on, camera movement, and lighting. To help with organization and modification of the code I used modular code that was easy to edit to allow change such as adding objects or transforming the objects easier. Throughout the development process we were introduced to new concepts and techniques that were added over time, each new step created a new function outside of the main function that could be change and edited without having to rewrite code.

## 3D navigation

Navigating the 3D scene requires both a mouse and keyboard. To begin with I set up a standard view where the camera is placed on the scene, all movements affect the way the camera moves and the speed it moves. The first movement implemented was the mouses control that will rotate the camera along the x and y axis, in other words movement like turning your head. The WASD keys are used to move the camera forward, left, backwards, and to the right. The QE keys will move the camera up and down. The mouse scroll wheel adjusts the speed at which the camera moves through the scene. The code looks for these specific key presses and moves or rotates the camera along the axis the keys are assigned too. An additional orthographic camera view was implemented by pressing the P key, the camera will jump to an orthographic view of the scene. The controls for this view are the same as above with more limited movement as the camera is locked into this view state, pressing the P key again will revert to the first view.

## Custom Functions

The development of the program involved adding new features over time, such as camera movement, textures and lighting. Each of these functions were created to be modular and could be edited at any time without having to rewrite or effect the main function that runs the program. An example would be the lighting function or shaders that contain all the methods the program would use to determine how light should interact with the objects. Then in the render function I would input the lights strength, size, and location without having to rewrite the lighting function for each light I want to use, in other words each light can refer to the function and use the information there to keep the code looking more clean and well organized. Another function I have contains all me meshes along with the data for them such as the normal and texture coordinates. In the render function when I want to create a shape, I call to the mesh function to pull the objects I want, this makes the render function cleaner and freer from congestion of vertex data. Overall, the project and program at first were difficult to understand and implement but once an understanding of the program’s functions were achieved, creating what I needed became much easier.