**Project Reflection**

Michael Duteau

Southern New Hampshire University

CS-330: Comp Graphic and Visualization

Dr. Scott Gray

February 25, 2024

**Justification of Development Choices**

When considering options for the proposal image, there was the option to go for an already-made image of a scene. However, I thought it would be more interesting and thoughtful to take a picture of my own scene that included a few objects of varying complexity that mean something to me in some regard. I’m quite fond of the Rubik’s cube as it’s nice for fidgeting and I have an affinity for puzzles requiring logical thinking. The duck and fox are two of my favorite animals, and the Psyduck Pokémon in particular is my favorite Pokémon. The plant is also a nice addition as I have a fondness for nature and all of its creations.

Beyond what emotional connection I have to these objects, there was also the consideration of object complexity and texture variation. The Rubik’s cube in particular is incredibly simple, as it only requires one cube mesh in order to properly construct. On the opposite end of the spectrum, the plant required a lot more work to create an accurate representation of. After placing the base cylinder that represents the pot, multiple other cylinders were required at varying rotations and translations in order to properly convey that it is, indeed, meant to be a plant. I also ended up developing a secondary plane mesh that would simulate leaves by altering the z values of certain vertices in order to add another dimension to these objects, making them more convincing.

**Navigating with the Camera**

In order to provide intuitive and detailed movement throughout the scene, WASD controls were implemented in order to produce forward, backward, and side movement similar to how many PC games control. The mouse movement is also used in order to control the camera’s angle, while the mouse wheel is used either to speed up or slow down the movement of the camera. Q and E keys were also utilized in order to implement up and down movement, respectively. Lastly, P and O keys alter the projection of the scene, where pressing O will trigger orthographic projects and P will retrigger perspective projection. When run, the program defaults to perspective projection, and these mouse and keyboard controls allow ease of movement to anywhere within the scene.

**Custom Functions**

Many of the user-defined functions revolve around initializing and rendering the scene, as well as creating/destroying textures, meshes, and shader programs. Likewise, there are also functions that implement the aforementioned mouse and keyboard controls. Most of these functions are borrowed from prior assignments and tutorials, being modified to suit the needs of my particular scene. However, when making the switch from the OpenGLSample assignments to a standalone project, I felt it necessary to create headers and associated .cpps in order to better organize mesh and object creation and declutter the Source.cpp file. BuildMesh.h handles all of the UCreateMesh functions, and BuildObject.h handles creating the individual scene objects by constructing multiple meshes and scaling, rotating, and translating them in order to accurately construct the respective object.

The functions for building meshes and building objects both have potential for re-usability. Each mesh creation function pertains to a specific type of 3D object or plane; these functions can be reused in projects where construction of these meshes are applicable. The object building functions are also reusable in that multiples of a specific object can be built in order to further populate the scene. However, further development would be required to properly do this; currently, the objects are not held as a type of structure or variable, so their scale, rotation, and translation are fixed and cannot be modified outside of the function itself in the current state.