**Final Project Reflection**

Ryan P. Gingery

Southern New Hampshire University

CS330: Computational Graphics and Visualization

Professor Philip Enkema

June 20, 2023

**Final Project Reflection**

**Justify development choices for your 3D scene**. As you write, think about why you chose your selected objects. Also consider how you were able to program for the required functionality.

I chose the selected objects and scene because I am an active Dungeon Master for Dungeons & Dragons (D&D) with my friends. I chose these items as I thought it would be a topic that would keep me engaged and active if the 3D scene was something I really enjoyed. While I did not include everything that could be found behind the DM Screen in my project, the basics I felt would not provide an overwhelming task but cover the basics of what I need to learn OpenGL.

Programming the required functionality was another aspect I was able to accomplish. While the overall look of the scene was to me personal preference and design, I still needed users to be able to look around the 3D scene with their cursor and be able to move with their mousepad. Another functional design aspect that I needed was lighting. The “ceiling” light was a cubed light source placed above the scene, much like in my actual flat when I took the picture. This helped provide that nice glow to the object as you move about. I also needed to implement the ability to change the perspectives within the scene.

**Explain how a user can navigate your 3D scene**. As you compose your thoughts, discuss how you set up to control the virtual camera for your 3D scene using different input devices.

As I mentioned above, the user can navigate through the 3D scene using two input devices. The first is the computer mouse. Using the computer mouse to move the cursor around the screen while the program is running will allow you to look around the 3D scene in a complete 360 degree radius in all directions. The user can then move around the area using WASDQE. W is forward, S is backwards, A is left, D is right, Q is up, E is down. P would change the viewing perspective. The scroll bar on the mouse for mine scene allowed you to zoom in while I initially needed it to change the speed of the cursor.

**Explain the custom functions in your program that you are using to make your code more modular and organized**. Ask yourself, what does the function you developed do and how is it reusable?

A few steps I took to simplify my code was first to take each major objects within my scene and create its own .cpp and .h file and also, place all of the vertices and elements (indicies) in their own class labelled “shapes”. In the shapes.h file I was able to create as many shapes as I needed for the project. Here I could create a single pyramid or cylinder to use multiple times throughout the code. I also create vertices and elements specific to certain items based on their shapes. This would be the DM Screen for example. I created specific arrays just for that. In the item specific files, I would create their own specific methods, functions, and other variables and aspects needed. In these is where I would call the vertices and elements for the applicable items. If I needed to reuse them for any reason I could also do that. If you looked at the D4 dice, they used a pyramid (that has more than 4 sides but it was not easy to make a normal D4) that is called multiple times under each method with its own specific coordinated within the scene.

If you looked at the DM Screen for example, in there I was able to create not only the boards to the screen but also the straps, pyramids for the bolts, the pages that are pinned onto the screen for the DM to view, and the cylinders representing the magnets. That same cylinder from shapes.h was used for both magnets and the pencil on the table. Once all of the pieces of the scene were assembled within their own classes, I could call them to the application class to actually execute and setup the scene when the code is ran. While I did over create some shapes since I did not initially realize I could reuse the vertices and elements arrays, I believe that it would be easy to jump into the code to add more objects to the scene. The amount of classes and layout could cause some to be lost initially but the use of the comments within the code should explain the areas out enough for someone to simply hop in and start making additions.