Pomai Ahlo

SNHU 21EW2

CS 330 – Professor Alnaji

December 9, 2021

Final Project: Design Decisions

For this project, I modeled a scene containing a tissue box, a bottle of hand sanitizer, a stress ball, and a container of aloe. I chose these items because they all could easily be made from simple shapes. The hand sanitizer is comprised of a cylinder and a pyramid, the tissue box is a cube, the stress ball is a sphere, and the container of aloe is a cylinder. This design choice allowed me to stay true to the original photo.

As far as my code is concerned, I had many options, and I made choices that made things easier. For example, for my non-circular objects, I kept the vertex, normal, and texture data separate from each other. In theory, I could have kept the data in one large array, and utilized the stride to still create correct buffer objects. By the time I understood the concept, I had already created two complex objects. I opted to keep things consistent, and kept everything separate. The only exception to this was the cylinder and sphere. I used premade code by Song Ho Ahn, which had the three types of data already together in one array.

I made a decision only to use one shader. I could have made several shaders, one for each object, or I could have had passed in a variable for the specular value so that each item had different shininess. I did this for ease of use, and relied heavily on my textures to give the illusion of shininess.

Additionally, I could have had a mesh with more vertices in my cube. This would have made the diffuse light more realistic (Kilgard, 2000). I did not know out about this fact until well after I finished working on my project, and as a result I did not find it worth the extra time and effort to change my code for such a subtle increase in realism. This also applies to my pyramid, which only had 5 vertices (6 triangles).

My camera moves as outlined by the rubric. Pressing W and S moves the camera forward and backward, pressing A and D moves the camera left and right, pressing Q and E moves the camera up and down. You can look around with the mouse, control how fast you move by scrolling up and down. Lastly, you can press P to toggle between orthogonal and perspective points of view.

It should be noted that the navigation of the scene is not perfect. Looking around with the mouse will cause the camera to rotate about its original position, instead of moving in a way that perfectly mimics looking around with our own eyes. This phenomenon occurred in the tutorials as well, although not in a way as extreme as mine.

There are several functions in this program that makes the program more modular and organized. First, there is the *initializeWindow* function, which initializes glew, glfw, and the window. It’s used only once, but it helps organize the code. I also have a *LoadShaders* function, which loads the vertex shader and the fragment shader files. Again, this function is only used once, but in previous iterations of my project, I used it several times, having several shaders. The other main custom function is *createTexture,* which takes a file name and a texture ID, and creates a texture based off of the image linked to the filename, binding it to the texture ID. This function uses another function, *flipImageVertically*, which flips the image so that it lines up with how OpenGL process images. While it’s only used once, *flipImageVertically* would be particularly useful if I wanted to process images in any other way. *createTexture* was used once per texture I wanted to load, so it saved a lot of time and space in my code. The other methods used in my program relate to the processing of input from the mouse and keyboard. These methods also improve the modularity of my code.

References

Kilgard, M. J. (2000) *Avoiding 16 Common OpenGL Pitfalls.*

<https://www.opengl.org/archives/resources/features/KilgardTechniques/oglpitfall/>