Final Project Reflection

Al Houghton

**Development Choices**

My original goal was to create a behind the screen type of scene that shows what a Game Master sees. I found certain aspects of it more difficult than anticipated, namely the creation of cubes. I realized that the vertices I was trying to use was being shared with the plane that serves as the tabletop, so I placed those vertices in their own vector called planeVertices and redid the generic vertices vector. When I created the cube object, I ended up getting a mishmash of triangles. That tells me that my vertices are not connected. After reviewing the code, I realized my mistake: I set the stride to eight and it should have been five. Because I had it at eight the code was trying to draw triangles with mismatching vertices. Once I corrected the error, my cube came out perfect; I just had to use a scaling method to reduce the size so that I appears to be the size of a six-sided die (known as a d6).

Next, I opted to use a sphere to define a 20-sided die (known as a d20, the primary die used in all Dungeons and Dragons games). A d20 is not a perfect sphere, so I reduced the number of sectors slightly which created a sphere with that isn’t perfectly round.

For the markers I simply took two cylinders and rotated them 90-degrees so they would lay flat on their side. I considered using a series of cylinders that got smaller stacked on top of each other to provide a point, but decided that would create too many extra lines of code, so instead I created another cylinder which had a slightly larger radius to function as a cap.

Lastly, I chose to use a soda bottle as my complex object. Some sort of caffeinated beverage is always behind my game master screen, and I decided on a soda bottle. I used a series of cylinders for that as well, opting to merge a sphere to represent the rounded top of the bottle. Just like with the marker, I felt that was a cleaner option than going with a series of shrinking cylinders.

**Navigation**

The user should be able to control the camera with the WASD keys, allowing them to move forward, backwards and side to side. The WASD keys are the popular keys used for movement through a scene in a game. The Q and E keys are used to allow the user to go up and down. Lastly, the scroll up and scroll down on the mouse allows you to adjust camera speed (up for faster and down for slower).

When writing the code for the scroll wheel I noticed that after the speed multiplier value reached zero, any further decrease would end up being a positive number. That would mean as you further decrease the speed, you’ll hit a point where you’ll be increasing the speed using the decrease speed command. I applied a simple fix to that: I simply added a condition that only allowed you to decrease the speed while the speed multiplier is positive.

**Custom Functions**

One of my favorite custom functions is the loadTexture function. It really clears up the texture loading process. I ended up replacing all texture loading processes using that function which reduced my overall number of lines of code and made it easier to find any errors or issues.

While not a custom function, I did also include renaming the VBO and VAO variables to make sense with what I was adding. It made it a lot easier to make adjustments to my markers because I gave them names like blackMarkerVBO and blackMarkerVAO.