Johnathan McBrayer

SNHU CS-330 Computer Graphics and Visualization

Final Project – 8/13/2023

Reflection

For my development choices for my 3D scene, I utilized several sets of cylinders, a plane, sphere, hemisphere, and pyramids. The first object I worked on was a can of beer. This was simple to set up three different cylinders to represent the can, and top and bottom. I also added another circle and texted it to represent the pop can top. The other objects in my scene were a bit more difficult to create as they involved spheres and hemispheres along with some cylinders and pyramids. The second object I worked on was lime. This was by far one of the most difficult parts of finishing this project. I struggled on how to get all the vertices and render the object correctly. I also had to search the internet a lot to figure out how to render the sphere. The same bits of code provided in the tutorials worked great for all other shapes but would not render a sphere. I was finally able to find an example of how to draw a sphere and once I was able to make sense of the code, I was able to apply it to my application. After I figured out the sphere, creating a hemisphere was a simple task. After building the lime out of one main sphere and two hemispheres on each side I was able to scale it wider to look like a lime. Finally, the most complicated portion was the bowl of salsa. This had a cylinder for the base, a hemisphere for the main bowl, another hollow cylinder for the top rim (to give it some depth as the hemisphere did not have any thickness), a circle for the salsa, and then the flat pyramid for the chip.

In my 3D scene the user can navigate around in all directions to explore the scene. This is accomplished by the use of both keyboard controls and the mouse. The W, A, S, and D keys are used on the keyboard to move forward and side to side. Q, and E are used to move up and down. The mouse is used to look around side to side and up and down. There is not roll options as it would not make sense in this type of scene. Roll controls would really only make sense in a scene base in space where there is no up/down or top and bottom. Also, the mouse scroll wheel is used to speed up or slow down the movement in the scene. I also included some additional keyboard controls to toggle between orthographic (2D) using the O key and perspective (3D) using the P key. The left and right arrow keys are used to switch between a filled object and just the lines. Then finally there is an orbiting light around the scene that can be paused and resumed using the L and K keys. With this combination of keyboard and mouse inputs, the user can explore all parts of the scene, move around objects, and see everything from every angle if desired.

For the custom functions in the application almost everything is modular. Starting with the examples provided in the tutorials splitting each component into smaller reusable section. I was also able to find an example online where another user configured to render common shapes and allow for an easier way to build multiples and place them around the scene. As this provided some of the functionality, I needed I also added several other functions to generate my spheres and hemispheres. Each shape has its own function, each shape function calls a function to build all the vertices, scale translate, etc. for each object. This information is then stored in an array of structures so that when it comes to time to draw everything on the screen the render loop can get each object and draw the information to the screen. The shape building can reuse the same functions to get the vertices and store the information so that each object does not need its own section of code. As the light presented a bit of an issue with the way my application would render objects, I did have to set up a separate function just to draw the light cube. This utilized the same basic code in tutorial 6.3 that was used to draw both the main cube and the light cube. If I had more time, I’m sure I could have done away with the separate light cube mesh function altogether, but I ran out of time. This would be something I would return and refactor to make everything more efficient.