Carmen Kingery

Brian Battersby

Comp Graphic and Visualization

Mod 7 – Project Reflection

02/17/2023

**Justify development choices for the 3D scene:**

In this scene we have a plate of cake with a cherry on top, along with a spoon on the side of the plate and a cup of tea just behind it. Starting with the teacup, at first I tried making it out of a straight cylinder, with the top cut off. It worked out just fine but I thought it looked more like a coffee mug than a teacup. Therefore, after looking at the reference picture more, I decided to go with a tapered cylinder instead. This helped imitate the delicate curved edges of the teacup better. I used the same shape for the tea inside the cup so that it would fit properly with no edges clipping out the sides of the cup. I stuck with a short cylinder for the base of the cup because they look fairly square in the reference picture with not much curve to them, unlike the walls of the cup. Then I used a small, stretched torus to replicate the little delicate handle of the teacup. This made the most sense as none of the other shapes have a natural hole in the middle. The teacup was the most complex object in the scene, being made up of four different shapes. All the others were made up of two or less shapes.

The next item I worked on was the plate. It is made up of half a sphere because the reference plate has a bit of a round curve to it. This had to be rotated just right so that it would sit flat on the table top. Similarly, I made the head of the spoon out of half a sphere as well but tilted it up slightly as the spoon in the reference does not sit squarely on the plate. I went back and forth with what shape to use for the spoon’s handle. A thin tapered cylinder could work as the handle does have a bit of a curve to it and does indeed taper down where it connects to the head. However, I decided to keep it simple and went with a long, thin cube. The cake was made with two separate cubes because the top face needed to have a different texture than the rest of it. Therefore, I put a thin cube on top of the main cube. The top one received the frosting texture while the bottom cube got the cake texture. Finally, the cherry on top was created with a simple sphere with a thin cylinder for the stem.

**Explain how a user can navigate the 3D scene:**

The inputs I have set up for this project include a keyboard and mouse. You can move around the scene using the WASD keys. W and S move you forward and backward, and A and D move you left and right. On top of that, you can use E and Q to move up and down. Your view is controlled by the mouse. Moving your mouse around will change the angle at which you are viewing the scene. And lastly, the scroll wheel on your mouse can adjust the speed at which you fly around the scene.

**Explain the custom functions in your program:**

I have a few user defined functions in my program that were made to be very universal. They can be used in any OpenGL project because they pull all of their source material from various libraries instead of from the project itself. You could get by without separating all these processes into their own functions by simply throwing all the process code into the main function. However, this would make for a very messy project and would be much harder to read and follow along with. Therefore, I have separated them into various functions to help keep my project clean and organized.

The first set of functions we created deal with creating and updating the window. UInitialize() creates the viewing window for your scene. UResizeWindow() does just what it says and resizes the window if and when the situation arises that it needs to do such a thing. UProcessInput() pulls the information coming from the setup input devices, such as the keyboard and mouse for this project. If the escape key is pressed the window automatically closes.

The next set of functions we have deal with rendering the scene we have created. URender() draws the images to the window. UCreateShaderProgram() initializes the program used to color and shade the images and UDestroyShaderProgram() frees up that memory space once the shader program is no longer needed. UCreateTexture() finds and stores the textures to be used on each object, it also initializes how those textures are to be applied to the objects. UDestroyTexture() does just that, it destroys the texture by freeing up the memory space once that texture is no longer needed.

The final set of functions we have deal with processing the mouse input. UMousePositionCallback() updates the view position. When the mouse moves, it tells the screen to move the camera in the corresponding direction. UMouseScrollCallback() updates the movement speed for the camera. And UMouseButtonCallback() handles all the additional buttons on the mouse, which we don’t have programed to do anything in this project.