Becky Scott

CS 330

Project Submission

2/19/2023



**Development Choices**

I selected the photo of the hydrangeas with the teacup because I thought it would be a fun challenge, and it included a lot of interesting primitive shapes that I could create in OpenGL. Hydrangeas are also one of my favorite flowers! The scene has planes such as the table and backdrop, cylinders for the vase and teacup, spheres for the hydrangea bulbs, and a torus for the teacup handle. For the textures, I wanted to keep it as close as possible to the original, while also playing with the light and colors to make a fresh and fun scene. The scene has 2 lights, the main key light has a soft yellow hue, to give the scene a warm and welcoming ambiance. The secondary light is set to white to add brightness.

**Navigation**

I implemented the camera movement to use the keyboard and mouse, independently of each other. The W and S keys will move the camera forward and backwards, the A and D keys will move the camera left and right, and the Q and E keys will move the camera up and down. Mouse movement will change the camera’s pitch and yaw. The mouse scroll will increase or decrease the camera movement speed within the range of 1 to 20 units. The O key will set the camera to orthographic mode, and the P key will set it back to perspective mode.

Aside from moving the camera, you can also move the position of the main light using the directional keys to control the forward and right axes, and the page up and down keys to move the camera up and down.

**Custom Functions**

Since the scene has several repeated primitive shapes, I wanted to create functions that could make creation of primitives reusable. Each of the primitives has its own create function which takes in values specific to that primitive. For example, the sphere function takes in the radius and the number of vertical and horizontal slices. This means you can use the same sphere creation function for a small low-poly sphere and a large high-poly sphere. Each primitive function has a helper function which creates the vertices, normal, and texture mapping matrix, and a function which creates the triangles for the primitive.

Another way my code modularizes mesh creation is by using a struct for the mesh data, which includes the VAO, VBO, and color data within. This allowed me to save that data to the mesh object and recall it easily, rather than using separate variables for each. I did the same with the lights and their position, scale, color, and intensity. Although my scene only has 2 lights, this would make it very easy to add several more lights.