OpenGL Scene Reflection

Chris Austin

CS-330-T3696

Prof. Gene Rodriguez

February 20, 2022

**Scene Justification:**

For the 3D scene project, we were asked to select a collection of objects that could be modeled using geometric primitives. In addition, at least one object needed to be modeled as a complex or compound object. I attempted to select some interesting objects that would satisfy theses requirements. The objects in my original screen were:

* A “bubble gum machine” toy container.
* A coffee cup.
* A small flow holder.
* A Ruler.

The plastic toy container that is a hemisphere sitting on top of a cylinder. All of the objects were chosen to meet the projects requirements.

I was able to build most of the objects using a combination of cylinders, sphere, and plains. By building induvial objects and rotating and translating these objects I was able to build more complex shapes.

**Scene Navigation:**

Users can navigate the 3D scene in much the same way a person familiar with First Person video games would expect to navigate. The keys W, A, S, and D are used to move front, back, left, and right respectively. The mouse is used to control the pitch and yaw of the view. So, for example, using the W key one can move forward in the scene while focusing on the object in the scene using the mouse. Also, the Q and E keys can be used to translate up and down across the scene. Finally, the scroll wheel on a mouse can be used to speed up and slow down movement in the scene.

**Custom Functions:**

The code uses a few custom functions to make the code reusable and editable. Some of these functions are as follows:

* procesInput() is very simple and very extensible for adding support for different movement methods.
* UConstructTextures() provides a clear and extensible approach to load textures. As it is presently, it’s not highly reusable but can be modified to receive an array of texture information rather than a list of arguments to be more reusable.
* cylinderHelper() is probably the most used custom function. It’s a helper method that receives and struct describing a cylinder and it’s position and then renders the cylinder. This is reusable function but could be further generalized to render any shape.