# **7-1 Assignment: Project Reflection**

Cory Remick

CS 330 Comp Graphic and Visualization

Professor Malcolm Wabara

10/16/2022

The project's purpose was to take a two-dimensional image and render it as a three-dimensional scene. The three-dimensional scene should have Phong lighting, and the camera should be able to be moved and face any direction so the scene can be explored from any angle.



Figure Reference Image

Primitive shapes were selected that would reflect the reference image above. For the newspaper surface, a plane was chosen. For the wood box, a rectangular cube was chosen. For the glass candle holders, tori with cylinders inside to represent the votive candles were chosen. The matchbox is a rectangular cube. The copper candle is a cylinder, and the spray bottle is two cylinders stacked on each other.

Textures for each shape were sourced from the internet. Attention was paid to ensuring the images were free for copyright purposes. This made it difficult to find a texture closely resembling the reference image. A less accurate picture was often used because the more detailed one would have required licensing.

One of the main requirements for the project was to keep the polygon count low. Early on, an attempt was made to create a shape using a widely used open-source 3d drawing application. Upon examining the output files for the shape, it was clear that it would not be possible to hand-tweak the geometry to reduce the detail. The low polygon requirement led to the need to either hand code the vertices for the shapes or use a procedure to generate the vertices for the shape. Hand-coded geometry was used for planes and cubes, and procedures were used for tori and cylinders.

Various keyboard and mouse controls were captured and used to manipulate the camera to navigate the scene. The "W" key will move the camera position vector forward by adjusting the z-position in the positive direction. The "S" key will move the camera backward along the negative z-position. The "A" and "D" keys will slide the camera left and right, respectively, by adjusting the position along an orthogonal vector to the plane formed by the up and camera-facing vectors. The "Q" and "E" keys will slide the camera up and down, respectively, by adding or subtracting the "up" vector. Mouse movement will change the direction the camera faces; left, right, up, and down. The mouse wheel will increase or decrease the camera speed as it moves through the scene.

There are a couple more keyboard keys used for controlling the scene. The "P" key will toggle between the projection modes, perspective, and orthogonal (several attempts may be needed for the keypress to toggle the mode). Keys "1" and "2" can switch between surface and wireframe. Lastly, the "ESC" key will exit the application.

During implementation, functions needed to be refactored to prevent them from becoming too complex to read. These special functions handled shape creation, shape drawing, and shader creation and use. The idea to have a shader class came from another open-source project. The implementation of the Shader class came from a combination of the course lessons, course tutorials, and project requirements. The best practice is to load shader programs from files, but this project's implementation opted for static constants. This allowed quick tweaks to shader programs without needing to hunt for external files. Also, different static programs allowed for quickly switching between different shaders for different shapes as not all the geometry texture coordinates and normal were specified simultaneously. Helper methods in the Shader class help set various Uniforms and control when the shader is used for drawing.

A screenshot of the final scene appears below. One lesson learned that will be applied to future projects is that texture loading should be separate from geometry creation. It would be optimal to declare a primitive shape in unit coordinates and load it into a vertex buffer once. Then any shape that requires that primitive could rotate, scale, translate and supply a different texture and shading program separately before drawing. Alas, refactoring this approach would have been too much for this project, but the lesson will be incorporated into future projects.

A picture containing letter

Description automatically generated

Figure 2 3D Scene