Design Decisions

For the final project, I was able to render a 3D deck card holder, a pyramid, a small rectangular cube, and an octahedron. All of the 3D objects mentions above was able to stay on the plane with a light cube on top (directional light). I think this final output compromise the 3D scene I was imagined from Module 2 assignment, because I wasn’t able to render a sphere, a cylinder. From the assignment span from Module 2 to Module 6, I was able to incorporate majority of the requirement that needed for Module 7. I was able to learn the basic of OpenGL from Module 2 by creating a 2D pyramid, a skeleton of 3D scene from Module 3, camera movement from Module 4, interactivity from Module 5, and lighting from Module 6. All of them are very useful for my final scene.

To navigate the 3D scene, the default camera view is in perspective, and user can use keyboard and mouse to control the upward, downward movement of the scene by using Q, E and use W, A, S, D to go top, left, down, right, respectively. To control the speed, user can hold the middle mouse button to increment the speed. To make the speed back to default, user only need to release the middle mouse button. Also, user can hold the left mouse button and use Q, E, W, A, S, D to rotate, translate the scene without any problem. In addition to that, user can press P or O to change the scene view from perspective to orthographic, and press ESC key to exit the program.

All of the code has been organized from VAO, VBO, EBO to Texture, Shader, Camera and Mesh. Every class all have header and functions of their own for main usage. For example, we can create vertices and indices of a cube in main, then utilize Shader class to create a cube shader, and Mesh class to create a 3D cube because Mesh class is the one holding the important VAO, VBO, EBO to do the render.

Not to mention that, Mesh class will require fragment shader and vertex shader to do the rendering. Therefore, we have fragment shader text and vertex shader text to control the coordinates, colors, normal and texture. Fragment shader will output the color and light of shader and vertex shader will take care of the assign vertex data and output the position of all of the coordinates. After that, we update the camera view and draw the object, the delete the shader we just created in main.