The 3D scene that I developed consisted of a Rubik’s Cube, a pyramid, a plane, and a lamp effect light. The chosen objects were done based off of items I had sitting around me when I was developing a basic model. These objects were a Rubik’s Cube and a Beyblade along with a wooden desktop. To simulate the Rubik’s Cube, I created a 3D cube using triangles and set a for loop to replicate the cube with a slightly offset position each time. I also put a slightly large cube in the very center position to give the Rubik’s Cube a feeling that the object was one whole.

For the Beyblade, I ended up choosing to just place a pyramid down since the Beyblade would look unnatural flipped the other way. This in the end made the shape look more like a pyramid and a Beyblade but it is what it is. The next item created was the plane/desktop. To create this item, the basic idea was to create a large square platform and place both objects on top of it. Lastly, the lamp effect light. I placed a light above the Rubik’s Cube and pyramid at an angle to give the effect of a lamp light shining down on them. Overall, the added ambience given by this light effect was a massive success.

The only major problem encountered after all of this was placing textures on multiple objects. I could get two textures to load into the program but for some reason only one actually shows in the scene with the other displaying the textures color but not the actual texture. I have spent a plethora of hours on this current situation and hopefully I manage to solve it before the due date.

To navigate my 3D scene, you can use the normal gameplay keys of w, a, s, and d to maneuver forward, left, right, and backwards. There is also q and e for going down and up respectively. There is also a perspective switch which allows you to view the scene orthogonally. To look around the 3D scene, you can move your mouse and the program tracks your mouse position to allow you to change directions with precise movements. The last movement item is using the mouse scroll wheel to slow down and speed up your movement speed.

Modularizing/organizing my program was simple yet also gave me a headache at times. Basically, if the feature was a key component that needed could be split, I split it. This means there are many different methods to help organize code into their respective places for later calls. These methods include a main function, flipImageVertically, CreateMesh, CreateShader, CreateTexture, DestroyMesh, DestroyShader, DestroyTexture, MousePosition, MouseScroll, ProcessInput, MouseButton, Render, ResizeWindow, Initialize. Main function contains the main loop and calls upon all the other functions in the list. CreateMesh sets all the objects being created in the scene. CreateShader creates the shader for the scene while CreateTexture creates the texture for the scene. All destroy methods allow for object destruction to help cleanup the program. MousePosition, MouseScroll, ProcessInput, and MouseButton are for all input updates to allow maneuverability in the 3D scene. Render places all the objects onto the scene. Initialize and Resize Window are for creating and editing the OpenGL window. These functions all allow for the program to feel neat and also help to dot i’s and cross t’s with program clean up.