**Design Decisions**

1. **Justify development choices for your 3D scene. Think about why you chose your selected objects. Also consider how you were able to program for the required functionality.**

I chose the objects because they fit the assignment criteria without being overly complicated. This allowed me to focus on understanding and mastering the objectives without feeling overwhelmed. I could examine the code given to us for this project, the sample we were given at the beginning of the term, and our weekly reading assignments to determine how to implement the required functionality. I had some problems with the movement speed adjustment using mouse scrolling but was able to come up with a solution by consulting my professor. Ultimately, I could not find a texture for the Rubik’s cube, so I had to enlist a different texture.

1. **Explain how a user can navigate your 3D scene. Explain how you set up to control the virtual camera for your 3D scene using different input devices.**

The user can use the W and S keys to move the camera forward and backward, the A and D keys for left and right, and the Q and E keys to move up and down. The mouse scrolling function controls the speed of camera movement. Some movement functionality was already in place, provided in the base code. I used the same principles to add the camera up and down using the Q and E keys. Movement speed was more difficult. It required consulting my professor for a solution, as I found nothing on it in our reading.

1. **Explain the custom functions in your program that you are using to make your code more modular and organized. Ask yourself, what does the function you developed do, and how is it reusable?**

There are many examples of using modular design in the code. For example, there are separate functions for lighting, textures, and drawing shapes. The function void SceneManager::SetupSceneLights() is dedicated solely to setting up each lighting source. void SceneManager::PrepareScene() utilizes the modularity of functions like SetupSceneLights(), LoadSceneTextures(), DefineObjectMaterials(), and functions like m\_basicMeshes->LoadPlaneMesh() that draw the objects. LoadSceneTextures(), to be expected, loads the textures you mean to use in the scene. DefineObjectMaterials() sets up how the objects will react to lighting, such as their reflectivity. The modularity of the basic meshes was particularly useful as some objects, such as a torus, would have been challenging to write from scratch.