**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 did not want the scene to be too complicated as I thought it would require too much trial and error to position the shapes. I also thought it would be cool to have some cohesion of the scene. The scene is about solving the Rubik's cube. There are miniature Rubik's cubes along with note taking of the solving process. I used the provided template code along with some extra custom helper functions to model the scene.

**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.**

There are standards and libraries that can make it easier for developers to implement computer input without worrying too much about the different operating systems and hardware. However, there still can be some incompatibilities rendering some functionalities unreliable. The keyboard can be quite a reliable input device and the standard "WASD" was used to move the camera forward, backward, left, and right. However, the mouse may have, for security reasons, limited programmatic control. For this reason, it is probably best to only program for a single machine design for critical applications.

**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?**

The template code only provided a single global rotation representation for each shape. This meant that if you wanted an object to be in a different orientation, each shape would have to be reoriented. You could not just rotate the entire object. It is tedious to model an object and then remodel the object just to put it in a different position. A solution to this problem might be to use a "3D orthonormal basis". This is a set of unit vectors that can be used to add up many rotations that could represent an object's orientation. The final orientation can then be retrieved from the vector angles. I actually did not use this method for modeling my scene because it was not necessary given the limited rotational requirement of each component. Instead, I added a helper function to give a rotational representation for each component which allowed adding a limited extra repositioning of the object. This helped to align my code with the given template code.