Humans have five basic senses: touch, sight, smell, taste, and hearing, and of those, sight is widely considered the most important. Sight is how we navigate the world and how we make sense of it all. Just seeing an object can evoke powerful memories, such as seeing a child’s play toy can bring back memories of one’s own childhood. It is with this in mind that I tackled designing a scene to be made in the 3D graphical API program known as OpenGL. Choosing a scene was quite an interesting task because there were a few things that needed to be balanced to achieve my goal.

First, I wanted my scene to contain objects that would be familiar to not only myself but to others. Second, I needed the objects to be built from primitive shapes that I could recreate in OpenGL without making things overly complex to code. Finally, I needed to light the scene to invoke the emotion I was trying to achieve, which was familiarity. The objects I chose were a Rubik’s cube, which is a simple yet familiar object that almost anyone will immediately recognize, a laptop computer, an item that has become nearly ubiquitous in everyday life, a pen a simple yet complex object when broken down into basic primitive shapes, and finally an old reliable coffee mug. I placed the objects onto a countertop that may look like anyone’s work-from-home space and the items were lit with a soft white light with spotlights from above.

Coding the scene started much like any coding project, simple at first and growing ever more complex. First, I needed to build the window and have all the error checking before ever considering building a 3D object. Next, I had to turn what was a simple 2D plane into a nearly limitless 3D space to place the objects. This was accomplished with the use of perspective which placed the camera into a 3D space and determined the near and far points that the viewer could see. Then it was time to place a 3D object into the space, something simple, like a cube or a pyramid. 3D objects are great and all, but not very exciting if they are colorless/lifeless objects, which is why we use texture mapping to bind images to the object to make them feel more real. I achieved this using the object's vertices and indices to tell the program how to layer the texture onto the object. Before I knew it I had my first 3D object, this processes was repeated, with varying complexity, with all other objects being placed in the scene.

It is at this point where the scene takes shape; the objects are textured and placed into their respective spots based on the initial scene photo and things start to come to alive. The final step is to set the lighting, which plays a very important role when building a scene. Lighting is so important in movies and tv because it can set the tone and mood of a scene, which is why almost all scary movies are set at night because the darkness ratchets up the tension of the scene, very few horror films could pull off full daylight scares.

The scene is slowly coming together, but without the ability to move around the space, it feels very bland, like looking at a copy of a picture, which is why the ability to move around the scene is important. To do this, I wrote code that would allow the program to take inputs from both a mouse and keyboard to control the camera inside the 3D scene, the user could move forward, backward, left, right, up, and down using the keyboard controls and tilt and pan the camera with the mouse. The mouse scroll mechanisms allowed users to control the speed at which the camera moved, scrolling forward to speed up movement, and backward to slow it down.

With all the functionality of the program one would think that the code would become overly complex (you’d be right), but luckily C++ uses object-oriented programming that allowed me to separate blocks of code to make it more modular and easier to read. I separated the camera controls, the vertex shaders, and the image loaders, I have also seen instances of people making their programs even more modular by separating the vertex buffer object, vertex array object, and elemental array buffer objects into their own files.

This project was a very difficult task, but one that I found rewarding as it slowly came together week after week and though my skill set may not allow me to get the accuracy, composition, and quality of some professional, ultimately I’m happy with how it turned out.