* 1. Project Reflection

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

The scene I picked was my desk where I was originally going to do my monitors, desktop tower, keyboard, headphones, microphone, and mouse. As I started working on my final scene, I decided to scrape a few of these since I didn’t have enough time to implement them and I decided to take out my headphones, speaker, and mouse since re-creating my keyboard took way longer than expected since it needed a lot of keys to create.

Most of my items were using the same vertices but just modified to match what I was making, this helped me to finish my keyboard in time due to the fact that I didn’t need to keep creating keys for the keyboard, I did have to modify a few keys to be bigger to match the larger keys but they were using the same code so it didn’t take a lot of time to create.   
  
The most amount of time was getting my keyboard setup to match the correct shape which took a lot of back and forth to have them put into the correct shape, unfortunately I ran out of time due to how long it took to create the keyboard that I was unable to get everything textured but thankfully I was able to get the plane and computers textured which met the requirements of the project but I will work after this to get them completed since I already invest so much time into this that I want it to be matching my vision.

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

It’s easy to move around the scene by using WASD as the movement keys in which W is forward, A is to the left, S is back, and D is to the right. These keys move at a steady constant pace that isn’t linked to framerate since powerful computers will be able to move super-fast while older computers will be moving a lot slower  
  
To move up and down we will be using QE as the movement keys in which Q is to move up, and E is to move down which move at a steady constant pace that isn’t linked to framerate since powerful computers will be able to move super-fast while older computers will be moving a lot slower  
  
To change the camera perspective you can press P which will change the viewpoint from perspective and orthographic in the 3D space.  
  
Finally we have the scroll wheel which will allow you to zoom in and zoom out at a much faster pace than moving in which scrolling up zooms you in and scrolling down will zoom you out till you are back to the original starting point.

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?

I watched a tutorial which helped me into breaking down into a modular project layout in which I broke almost everything up which helped by knowing which source files are for and keeping the code separated so that I can easily debug and work on different parts of the code and know if something breaks which source file the issue could be.   
  
With how I setup my vertices and indices they were able to easily be reused in which I created a keyboardkey indices which I used for all the modified key vertices, this helped to not have to keep creating more code but also helped save time.

Another thing that helped by separating my code and making it more modular like I said before is that it made it much easier to read which helped in the case of my application.cpp file in which it was bloated to the point that at times I could not see where everything was. Thanks to making the code modular you can see in the application.cpp file where I commented out sections of code that were no longer needed.

The custom functions that were used were quite a few in which I had a texture function, light function, mesh function, model function, shader function, plane function, and monitors functions. Each of these custom functions helped with making it so that I didn’t need to keep writing the same code for different things and by having these functions I could just call them making it easier to read and faster to create new objects related to them.

The shader function helped with calling in the vertex and fragment shader to the GPU without having to type out the code ever time for each instance of me using them.

The texture function made it easier to read where the textures are stored and then calling it to the GPU to be applied (ran into some bugs with this and ran out of time to get it fixed.)  
  
The shape functions such as first monitor, second monitor, and plane were used to create the functions and where they are placed in the 3D environment making it more manageable to read the code in them instead of them bloating up the application.cpp or main.cpp source files. If any issues were run into it also made it easier to find out which one of them was causing it for me to fix it.

This goes on with all the other functions that were created since they helped reduce the need for code in other areas since they would be called in and I could then call them into the functions that needed them without having to type everything out every time when I needed to something with them.