After nearly eight weeks, I have finished my CS-330 Comp Graphic and Visualization project. It has led me down many paths of exhaustion, exhilaration, and eventually success. I am proud of my work on this project, even if it isn’t a hyper-realistic 3D scene. I originally chose a house, fence, mailbox, and ground because I thought of where I am right now. I thought doing my computer setup would be interesting, but there are way too many objects to narrow down, especially writing vertices, indices, and managing normals and UV coordinates by hand. As such, I thought bigger: the house! The house in the 3D scene does not exactly represent the size and shape of my house, but it is generic enough to be easily workable. In my scene, I have multiple objects that are using planes and cubes. Given the way I set the project up in the beginning, migrating to a different solution (such as implementing GLUT) would be more of a hassle than it’s worth, and take up more time than necessary. As such, I went through the painstaking process of drawing the scene and memorizing vector locations relative to each other. I would say that the hardest part of designing the scene was determining what normals to use. I had never had an in-depth understanding of normals before this course, and as such, it was an interesting time getting a hold of it. After getting a grasp on normals and the Phong lighting system, I am more comfortable building 3D scenes like this one.

Navigating the 3D scene is rather simple given the standardized controls used in most 3D games and scenes. Use the W, A, S, and D keys to navigate forward, left, backward, and right respectively. The user may also use Q to go upwards and E to go downwards. Moving the camera is simple as the mouse is captured in the window once the scene starts. Escaping the window is as simple as hitting escape, though keep in mind that this will also close the window. I believe the mouse’s sensitivity in the 3D scene makes sense, and no adjustments had to be made on that front. During prototyping for the scene, I implemented a button (P) which would toggle the view between orthogonal and perspective, which is left on perspective view by default. There was also a prototype (now gone) where scrolling would change the FOV of the camera instead of speeding it up, but it made more sense to use the scroll wheel to speed up the movement of the camera in a large scene such as this one.

Throughout the development of the application, I have overcome struggles of determining the best way to organize my code. Given the small scale of the project, I did not believe that the code needed to be split into separate files, though for scalability, this is a viable option if I were to continue expanding on the scene. I noticed that I had to scroll a lot to get from one mesh-creation function to another, and would sometimes confuse myself with which one I was looking at. This leads into one issue that I was not able to overcome: genericizing the mesh-creation process. I tried moving the code that is at the bottom of each mesh-creation method to a separate method, passing in the vertex and index arrays by reference, but they would simply not render in the 3D scene–it would remain black (or, off-white blue at the end of development). If I could figure that out, the mesh creation functions would only hold the vertices and indices, making them smaller and more portable. Regardless, the rest of the code is structured well enough that it can be copy and pasted into another project relatively pain-free. When I say “relatively,” I mean the project being moved into already having the vertices set up where it’s the coordinates followed by normals followed by U/V coordinates. OpenGL itself is fairly strict on the way the application is structured, but libraries such as GLUT exist to make it easier to work with.

In the end, I had lots of fun working on the project and would like to optimize and re-organize it if I had the spare time. I would like to use GLUT (GL Utility Toolkit) to simplify the process of rendering meshes so that I do not have to draw each vertex by hand.