3-2 Milestone Two: Enhancement One: Software Design and Engineering

Carl Allen

CS-499 Computer Science Capstone 23EW2

6/1/2025

**Introduction:**

For my Enhancement One: Software Design and Engineering, I chose a project where I created a 2D vector rendering made up of triangles. I originally built it in Visual Studio, which is an integrated development environment (IDE), and used OpenGL—an application programming interface (API)—along with C++, which is my preferred programming language. I worked on this artifact during my CS-330 Computer Graphics and Visualization course (CS-330).

I picked this project to enhance because OpenGL is still a widely used and well-supported API, even after all these years. It’s compatible with modern hardware and remains popular in the industry, so having a solid grasp of it can open a lot of doors. The original project already demonstrated my C++ and OpenGL skills well, but I wanted to take it a step further. I upgraded the rendering from a 2D shape to a 3D pyramid, cleaned up and restructured the comments for better readability, and added error handling. I made these changes because graphics technology continues to evolve, pushing for higher realism with more powerful hardware. I also wanted to make sure that any future developers looking at my code would find it easier to understand.

**Overview:**

While working on enhancing my original 2D triangle rendering, I learned that creating 3D objects involves building more complex shapes using basic geometry. For example, to create the base of my 3D pyramid, I needed a square—so I formed it by combining two triangles. The sides of the pyramid were a bit more challenging, but I figured out how to calculate and place each point by creating specific coordinates. This process helped me demonstrate my ability to define vertices and build the shape I wanted.

Once the pyramid was complete, I decided to add texture to the sides. Just like when building the shape, I had to determine the right coordinates to map the texture correctly. To make sure the texture displayed properly, I used a technique called texture wrapping. Through this, I learned about different wrapping modes like GL\_CLAMP\_TO\_EDGE, GL\_CLAMP\_TO\_BORDER, and GL\_REPEAT, which are also shown in Example 1. These options gave me control over how the texture behaved when coordinates fell outside the usual range.

Finally, I needed a way to load the texture image into my project. Since C++ doesn’t have a built-in way to do this directly, I learned to use an external library called stb\_image. By getting familiar with how stb\_image works, I was able to successfully load images from local files into my OpenGL project (CS-330).

**Figure 1**

Example of texture wrapping from Milestone\_Two.cpp

A white text with blue text

Description automatically generated

Now that the artifact is a fully textured 3D pyramid, I added functionality to control the scene using the W, A, S, and D keys, along with mouse input. I used the MODELVIEW matrix to handle these interactions and control the pyramid's movement and rotation within the scene. While setting up the functions and parameters for key presses and releases, I discovered that not all keys are handled the same way in GLUT. For instance, keys like F1 through F12 are considered special keys. Instead of using glutKeyboardFunc like with standard keys, you need to use glutSpecialFunc to detect those inputs. I also learned that many standard keys can be accessed using their ASCII values (CS-330).

**Figure 2**

Example of programmed keys in Milestone\_Two.cpp.

A computer code with blue text

Description automatically generated with medium confidence

**Conclusion:**

Taking this artifact from a simple 2D object to a fully enhanced 3D pyramid allowed me to demonstrate a range of skills and knowledge tied directly to the course outcomes. One major outcome I achieved involved designing and evaluating computing solutions that solve specific problems using algorithmic principles and established computer science practices. This applied when I worked through lighting calculations by computing surface normals—an essential step in determining how light interacts with the pyramid's surface. This connects to the algorithms outcome, as well as to data structures, since I used vectors to store and manipulate coordinate data that helped calculate light positioning (CS-330).

Another outcome I met was demonstrating the ability to apply reliable and innovative techniques, tools, and skills in real-world computing scenarios. Through software design and engineering, I followed C++ programming standards and wrote clear, well-structured, and fully commented code that could be easily understood by other developers. Throughout the process, I expanded my understanding of both C++ and OpenGL. I learned to use helpful libraries to handle callback functions, texture wrapping, and camera setup. I also figured out how to use processInput to connect keyboard inputs to scene interactions. My grasp of C++ grew stronger, especially in areas like using arrays of vertices and managing vertex buffer objects (VBOs), which are crucial for handling large amounts of vertex data efficiently.

One of the most difficult challenges was working with shaders and buffer setups. When something went wrong, I was often met with a blank screen that would open and close instantly, making it hard to identify the issue. I eventually resolved these bugs by adjusting my input and output settings. Enhancing this artifact for the Software Design and Engineering category was a meaningful challenge, and completing it gave me a real sense of pride and boosted my confidence in my growing skills in the computer science field (CS-330).