CS 499 Milestone Two: Enhancement One

Jeffrey Tung

Southern New Hampshire University

The artifact included for enhancement one is the final project form the CS 330 course. This artifact was originally created on August 11, 2020 during the final weeks of the CS 330 course. It is an OpenGL C++ project which uses different libraries like freeglut, opengl32, glew32, glu32, and SOIL. The artifact is a low polygon recreation of a Powerade bottle with features of switching between orthogonal (2D) and perspective (3D) view. This artifact’s main source file showcases various different software development skills and abilities. These skills include modular coding with different functions for each of the separate sections of the code, proper commenting and indentation, as well as utilization of a C++ plugin like OpenGL. Overall, with the length of the main source file being 1259 lines long, having proper inline documentation is important for the maintenance of the artifact.

My implementation to this artifact was to include a zoom feature to zoom in and out as well as creating a rotational view feature. Through the process of implementing these features, I recognized how important it was to include meaningful comments so I could quickly understand what each part of the program did in the grand scheme of the OpenGL program. Since I had already create separate features within my program of detecting mouse inputs and position as well as keyboard inputs, all I had to do was to figure out the changing of different pre-established variables to obtain the outcomes I desired for the enhancements.

For the zoom enhancement, I ended up with the design of holding down the right mouse button which activated the zoom logic, allowing for the object in the viewport to be zoomed in or out; moving the mouse up would zoom out, and moving the mouse down would zoom in. For the rotating enhancement, I ended up with the design of two hotkeys to toggle a Boolean logic of rotating the object or having the object be still; pressing ‘r’ would rotate the object, pressing ‘s’ would make the object stop rotating.

After my zoom implementation, I recognized a potential flaw with how the logic was implemented. This falls within the orthogonal and perspective view. Under the perspective view (hotkey of ‘p’), the zoom logic works perfectly since the camera is able to move closer or farther away depending on the location of the mouse pointer. However, under the orthogonal view (hotkey of ‘o’), the zoom logic is unneeded and faulty. This is because under the orthogonal view, the object is portrayed as a 2-demensional object, which zooming in and out on a planar object does not actually make the object bigger or smaller. For a potential update to my enhancements, I will create an instance where if the object is in orthogonal view, the zoom feature will be disabled.

With my rotating implementation, the most difficult part is understanding the glm math variables and how quickly/slowly the math conversions are relative to the OpenGL environment. In the first few tries at the implementation; my rotation was at weird angles and also rotating at too fast of a pace. Over tweaking and trial and errors, I was able to dial in on a happy medium of a slower rotation and rotating only on the z axis, which makes the most sense for a cylindrically shaped object.

Overall, I believe my implementations displays software development skills of modular coding, proper commenting and indentations, and the adequate understanding and usage of external plugins and libraries. Additionally, I showcased proper software development decision making for my implementations of different features and obtaining fluid design based off of the given task.