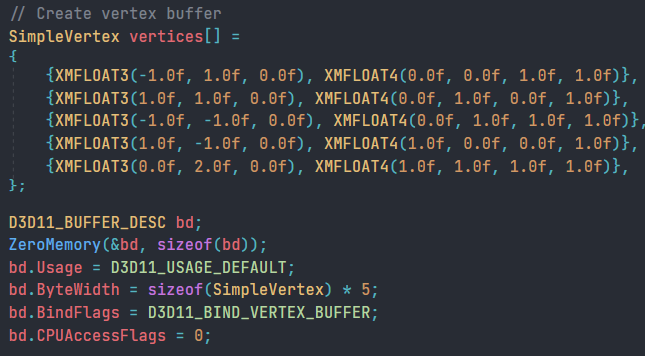
**FGCC Reflective Log**

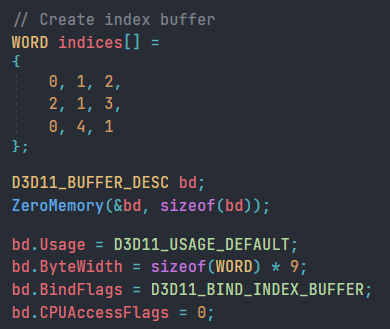
**Week 1**

Task 1: Add a third triangle and change the colors

* To add another triangle, the vertex buffer needs to be modified. Another vertex needs to be added and buffer’s byte width needs to be changed. The colors were also modified here:

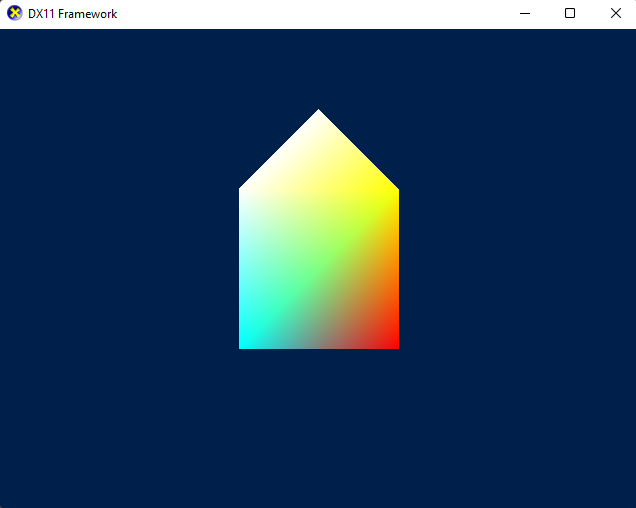


* Next, the index buffer needs to be changed to include the third triangle



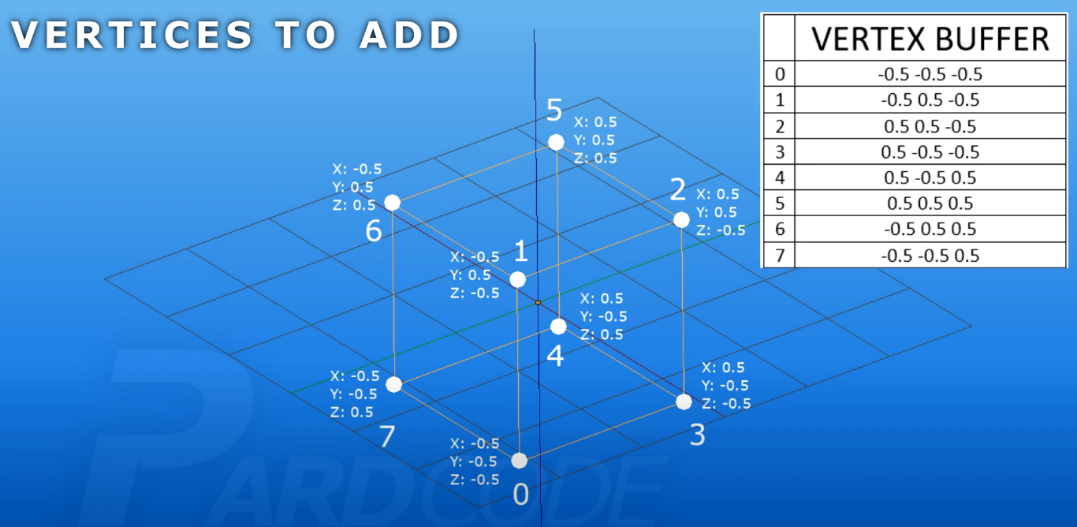
* Finally, the draw index function needs to be modified to include 9 indices (3 vertices per triangle)

**RESULT:**

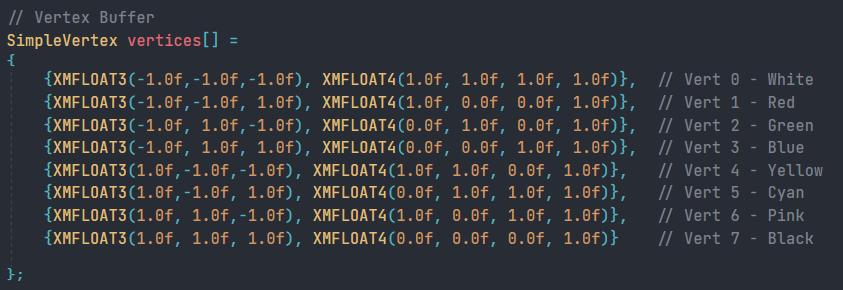


Task 2: Draw a spinning cube

* To draw a cube, the coordinates of all 8 vertices were required



* The vertex buffer was modified to include all 8 vertices with specific colors:



* The index buffer was then updated to draw the cube:



**Week 2**

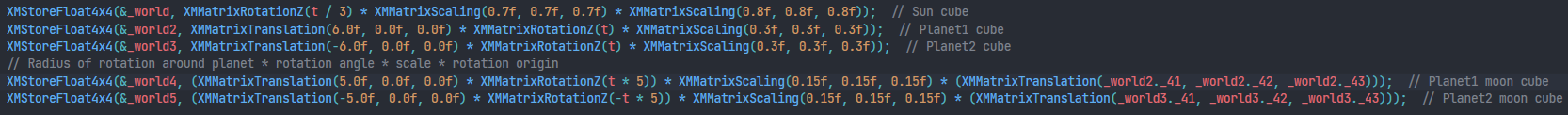
Task 1: Creating a Solar System

After creating the depth and stencil buffers, the next task was to make a solar system, where there is a central cube (sun) and 2 cubes revolving around the centre cube (planets) and each of those cubes having 1 cube rotating around it (moons). The objective of this task was to make sure that we understand matrix transformations.

* First create multiple world matrices (one for each cube)



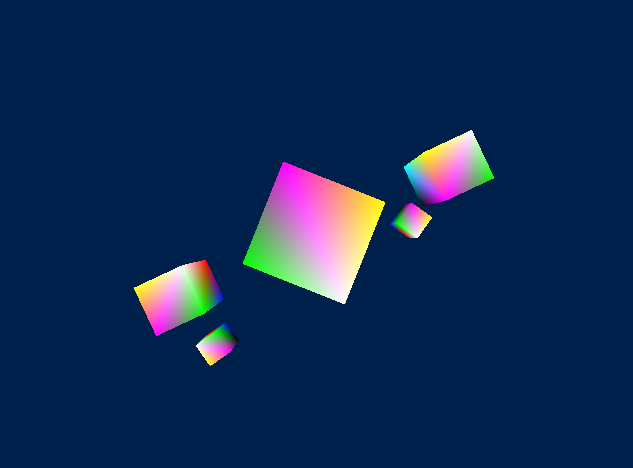
* Alter the transformation matrices for different positions, speed and direction of rotations and scales



* Update constant buffer and render all the cubes

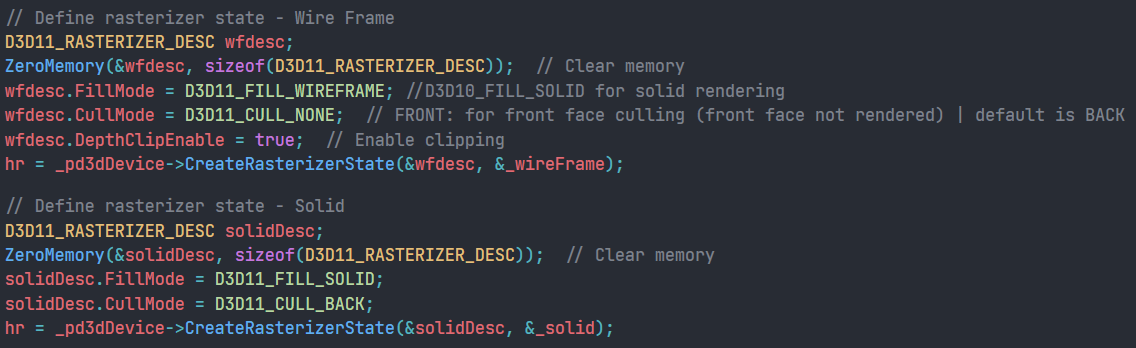


**RESULT:**

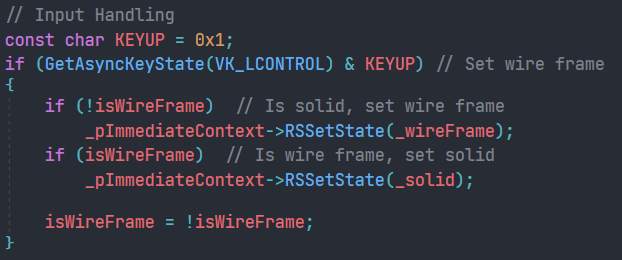


Task 1: Add a hotkey to switch from wireframe to solid view

* For this task we had to create 2 rasterizer state which can help for drawing in wire frame and solid



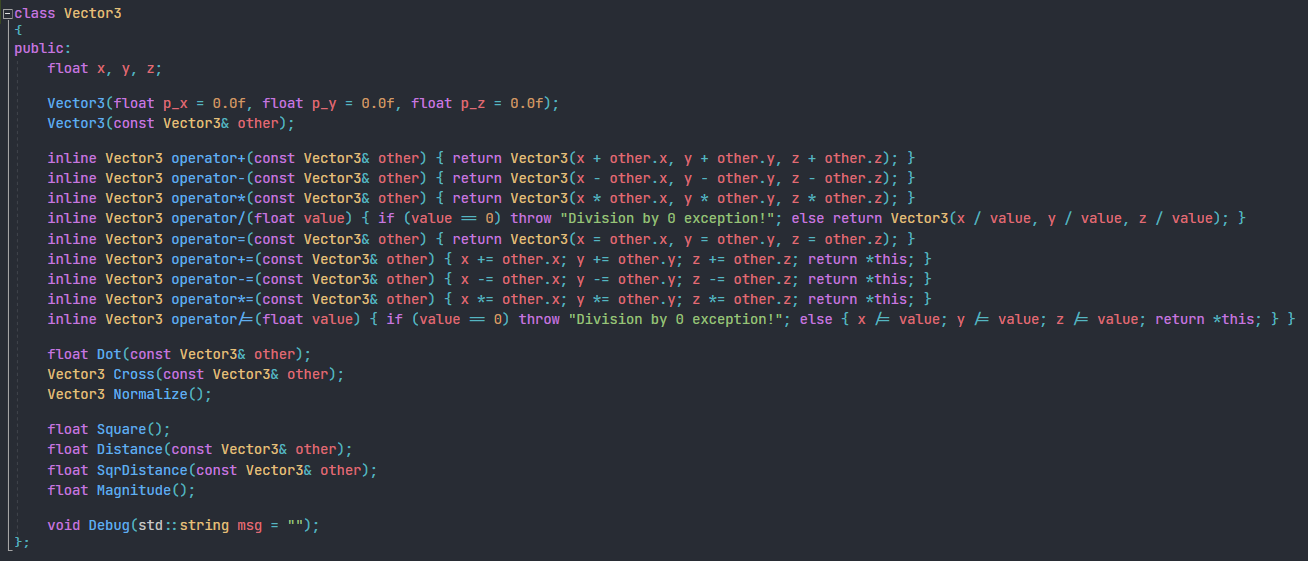
* Inside update add code to handle switching of rasterizer states



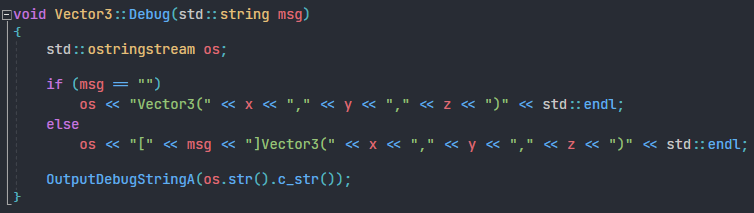
Task 2: Create the Vector class

For this task we had to create a vector class that that holds 3 floats and outputs them to the console when debugging

* The header class for Vector3 class was made where all the operation override functions were made inline



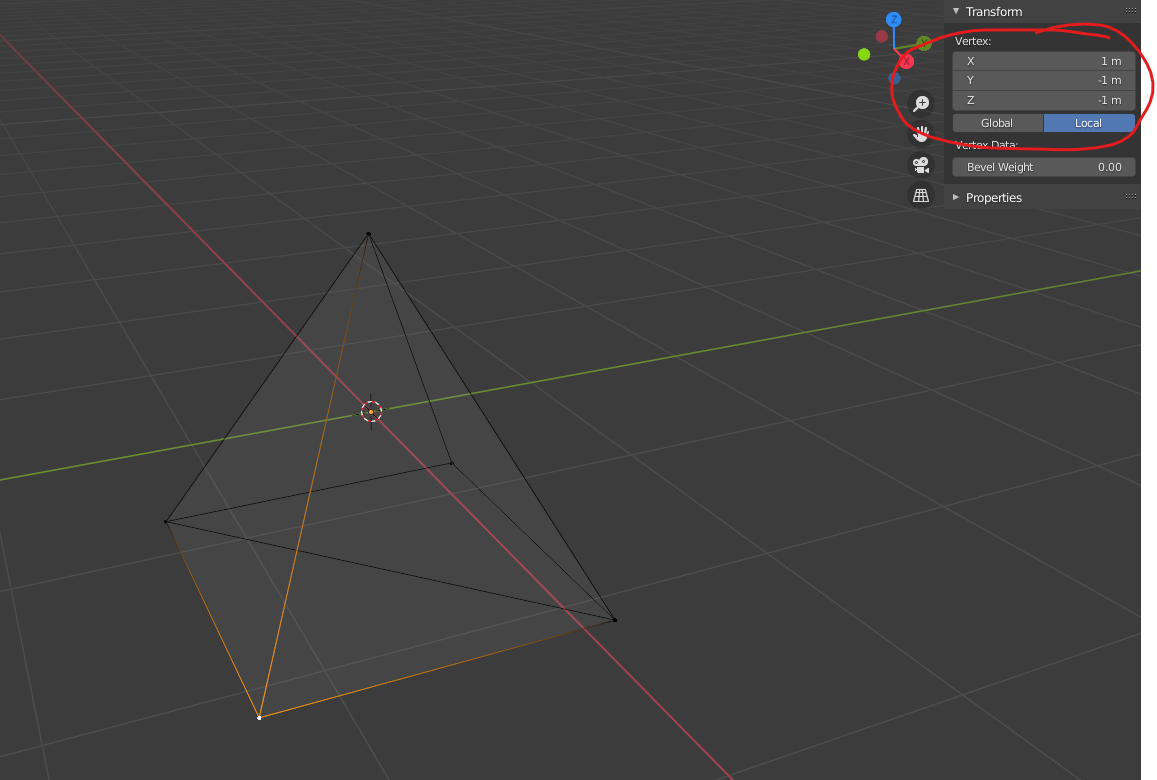
* A special function Debug() was added to print the vector information out to Visual Studio’s ‘Output Window’. This was done by creating a output string stream that was used to customize the output string and the OutputDebugStringA() function from Windows.h;



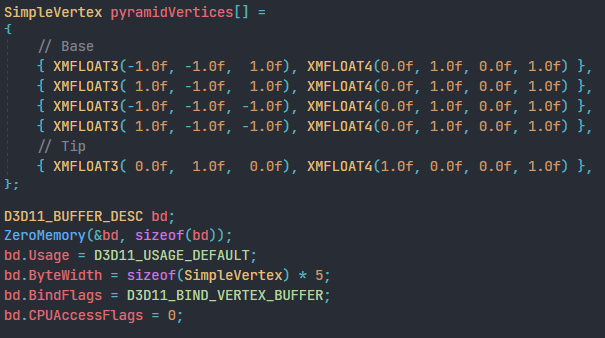
**Week 3**

Task 1: Creating a pyramid

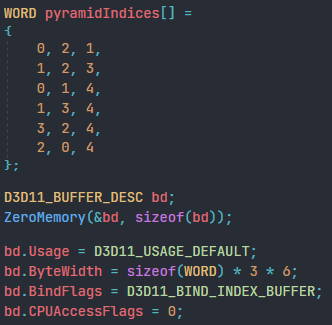
* To create a pyramid, I needed the coordinates of its vertices. I used Blender (a 3D modelling application) to get the vertex positions)



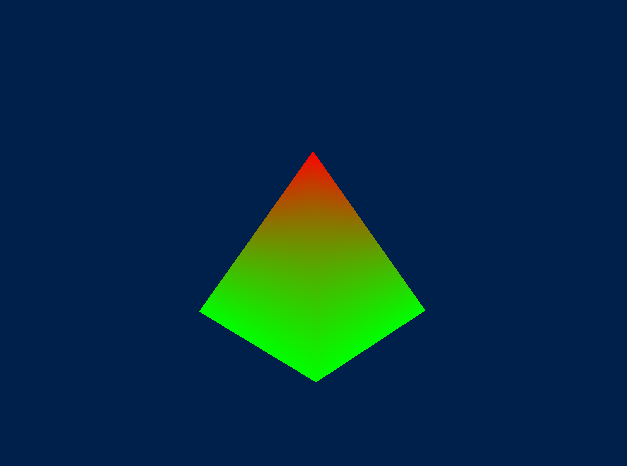
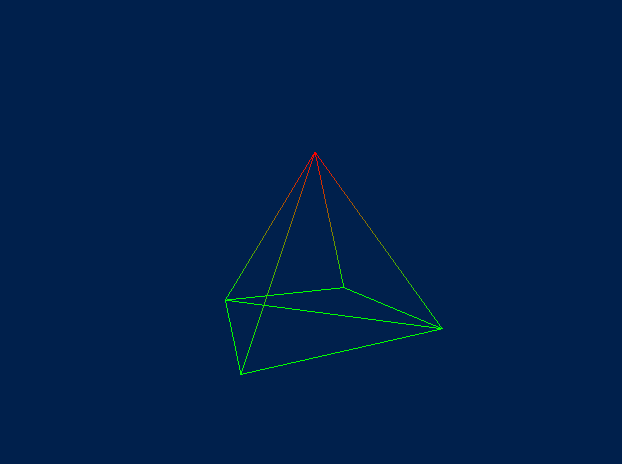
* I realized that the axis was flipped in Blender and DirectX Up in blender is the Z-Axis, whereas it is Y-Axis in DirectX). So, I had to switch the Y and Z axis in the vertex buffer



* The index buffer for the pyramid was then created



**RESULT:**

**** ****