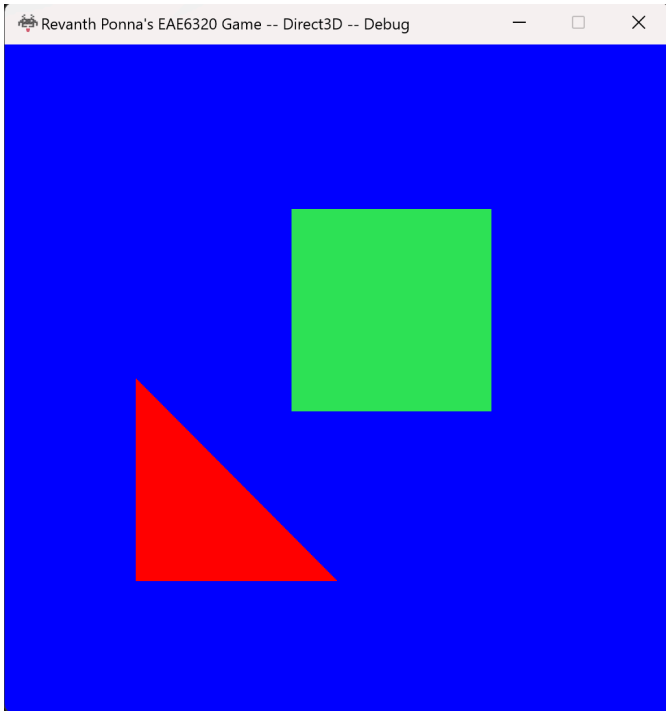
Write Up

Assignment 06 - GAMES6320

[Download Game]

Controls: WASD - Move Mesh, Arrow Keys - Move Camera, Space Bar - Change Mesh

During Assignment 06, I gained deeper knowledge about human readable file formats and why they are crucial to implement in game engines. Additionally, it was a great insight into the usage of Lua and how to integrate Lua into the loading of files in a game, something that I have been curious to learn for a long time.



Default State Of Game

Human-Readable Asset Files

One of the main steps in this assignment was to move the vertex and index data of the meshes from the MyGame project to a clean, human readable file format. This will now allow the game programmers to simply pass in the file path of the mesh they want to create, without specifying the mesh data itself.

There are several key advantages to converting the mesh data to a human readable file format. First of all, the data is now easy to read and understand, even for someone who is not as well versed with graphics. This is important because in game teams, we would want our mesh data to be readable by as many people as possible, even if they don't know about the underlying logic of the engine.

Having a human readable asset file is also much easier to debug whenever a problem arises. It is especially helpful for game programmers to efficiently find out the root cause of issues without having to worry about the Graphics engine. Since the asset files would typically be created by a content creation tool, it is essential to be able to understand the data that is being created, to then fix problems down the line.

Another advantage of having an effective file format is that it reduces any unnecessary data that need not be specified. For example, in my case, I did not need the asset file to define the number of vertices and indices in a mesh, as I left this data to be calculated by the engine itself. This helps with saving memory, but also avoiding the risk of mistakes when generating data.

At the end of the day, I believe a human readable asset file essentially follows the same concept of interfaces which we have discussed in class. As an engine programmer, we want to create an interface which is as easy as possible for other programmers to read, understand, and use based on the games they're making. Having a custom mesh file is a great way to ensure that vertex and index data can be effectively passed in without having to worry about the implementation under the hood.

Custom Mesh Files

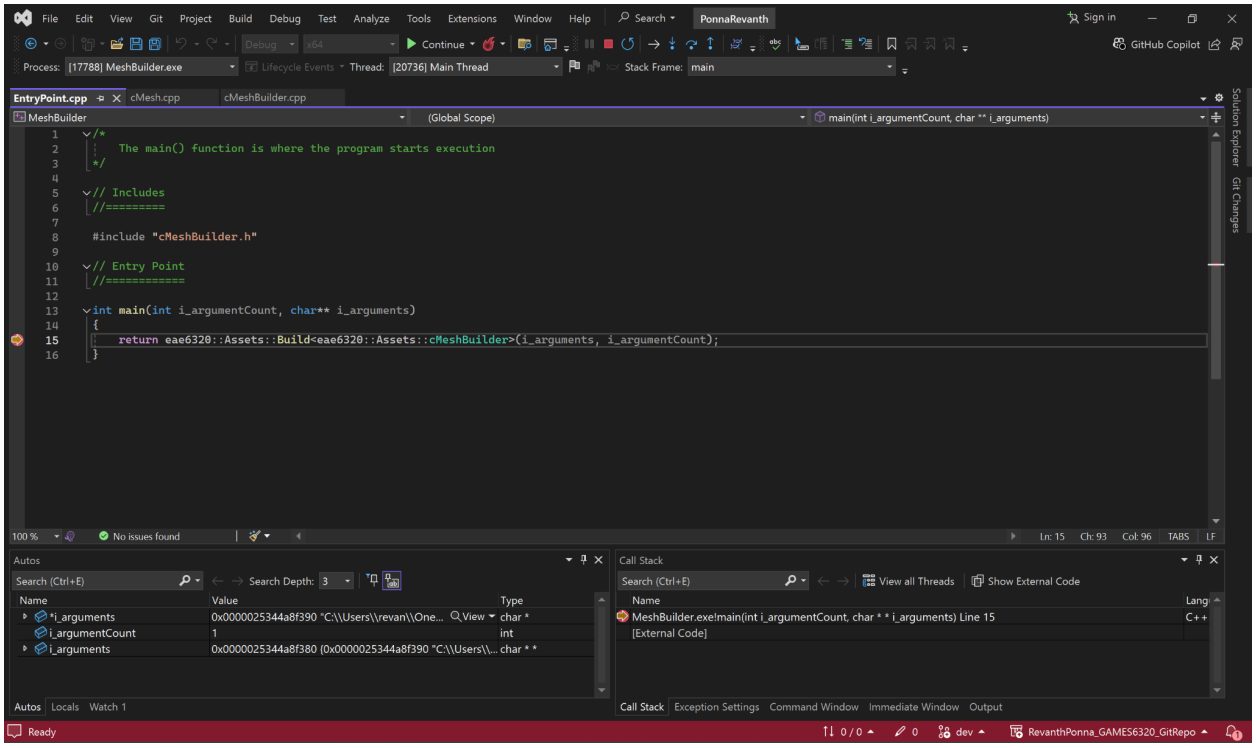
In my human-readable mesh file, there are simply two arrays defined. One of them returns the vertices of the mesh, and the other returns the indices in a right-handed winding order. At first, I also wanted the mesh file to pass in the number of vertices and indices. But after I started implementing the logic to create the mesh data using Lua, I realized that the count of vertices and indices is not necessary, since Lua takes care of calculating those values.

```
return
{
        vertices =
        {
                {0.0, 0.0, 0.0},
                {0.5, 0.5, 0.0},
                {0.0, 0.5, 0.0},
                {0.5, 0.0, 0.0},
        },

        indices = { 0, 3, 1, 0, 1, 2 },
}
```

Additionally, I made sure that each vertex in the vertices array is defined on a separate line, just to make the file more readable at first glance. One problem I came across during this assignment was that the meshes were not rendering on screen although the mesh file was being loaded correctly. After extensive debugging and frustration, I realized that it was because of me specifying each vertex as {0.0f, 0.5f, 0.0f} instead of {0.0, 0.5, 0.0}. Since the previous mesh data was in the MyGame project, I used ‘f’ to specify the float values, and copied over those lines of code into my custom mesh file. However, I realized that Lua does not need the ‘f’ to be specified and only expects numeric values when reading the data.

Debugging MeshBuilder



[Download Game]