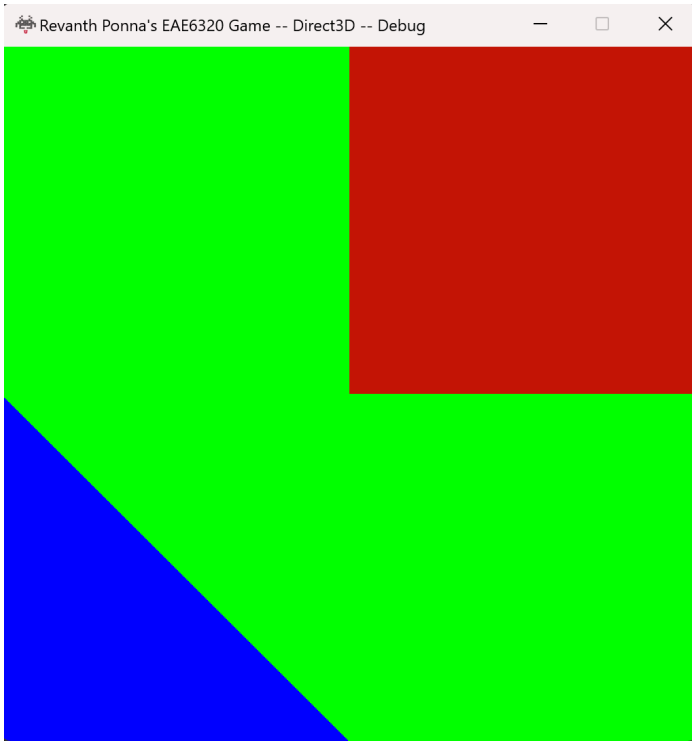
Write Up

Assignment 03 - GAMES6320

[Download Game]

In Assignment 03, I understood the importance of interfaces and how to ensure an easy and efficient approach for a user to call functions from the game engine. I also learned the core concepts used in a Graphics system, especially how a vertex buffer and an index buffer work together to draw geometry on a screen.



Platform-Independent Graphics.cpp

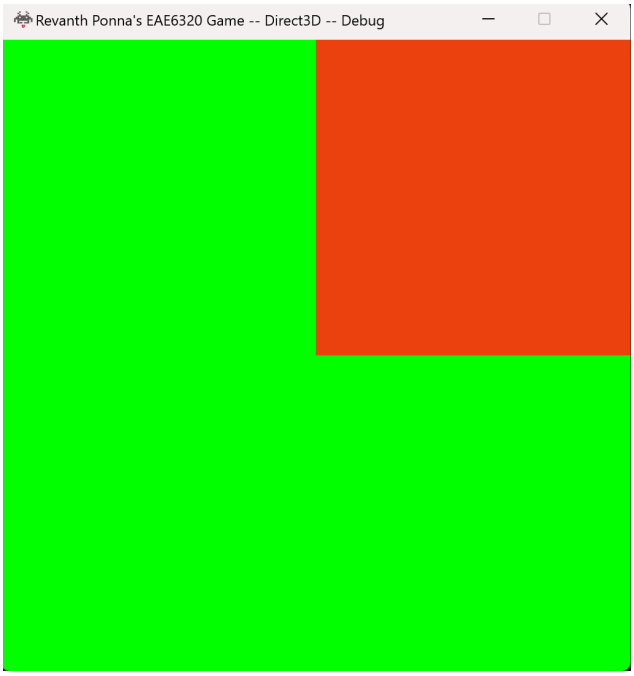
To combine the platform-specific Graphics.d3d.cpp and Graphics.gl.cpp files, I first had to read through both files and decide on how to separate the differences between them. I found that the main differences were in the clearing of buffers and the initialization of views in the Direct3D platform.

In order to provide a platform-independent implementation, I created an interface called RenderContext. This was defined as a class, just like the Mesh and Effect classes. I first created a header file cRenderContext.h, to declare the class and its methods, which included functions to clear buffers and initialize and clean up views.

Next up, I defined the platform-specific implementations for these functions in files called cRenderContext.d3d.cpp and cRenderContext.gl.cpp. These files were similar to what I did for the Mesh and Effect classes, and I ensured they only compile when I chose the appropriate configuration.

Creating the RenderContext interface seemed like an effective solution to abstract the differences between the platform-specific Graphics files. Since there were only a few key differences between the two, this interface made sure that I could call the functions in a clean way.

```
147     // Clearing buffers from the RenderContext class
148     // Setting the background color to green
149     newRenderContext.ClearBuffers(0.0f, 1.0f, 0.0f);
150
```



Effect Initialization Interface

The only piece of data that really needed to be included for the initialization of an effect was the name of the shader files. Since we want to use different shaders for each mesh, I designed an effective way to initialize the effect and pass in the name of the shader file as a parameter.

```
311         // Initialize Shading Data from Effect Class
312         result = effect1.LoadShaderFiles("animatedColor.shader");
313         result = effect1.InitializeShadingData(result);
314
315         result = effect2.LoadShaderFiles("constantColor.shader");
316         result = effect2.InitializeShadingData(result);
```

On a 64-bit system, a single effect takes up 17 bytes on a Direct3D platform, and 21 bytes on OpenGL. The memory size is slightly larger on OpenGL due to the presence of the GLuint m_programId variable. There is no significant way to reduce the size further without compromising functionality, since most of the memory usage comes from necessary pointers and platform-specific data.

```
25          private:
26              // Platform-specific private data will be declared here but defined in platform-specific files
27              struct PlatformData
28              {
29  #if defined( EAE6320_PLATFORM_D3D )
30                  eae6320::Graphics::cShader* m_vertexShader = nullptr;
31                  eae6320::Graphics::cShader* m_fragmentShader = nullptr;
32                  eae6320::Graphics::cRenderState m_renderState;
33  #elif defined( EAE6320_PLATFORM_GL )
34                  eae6320::Graphics::cShader* m_vertexShader = nullptr;
35                  eae6320::Graphics::cShader* m_fragmentShader = nullptr;
36                  eae6320::Graphics::cRenderState m_renderState;
37                  GLuint m_programId = 0;
38  #endif
39              };
```

Mesh Initialization Interface

To initialize a mesh, the user needs to specify four things:

- vertexData[]: The coordinates of each vertex of the mesh needed.
- indexData[]: The order of indices for the indexBuffer to draw the mesh on screen. The engine expects a default left-handed (clockwise) winding order. So, the indexData[] must always be specified in this way.
- vertexCount: The total number of vertices in the mesh.
- indexCount: The total number of indices.

```
56      // Defining the vertices of the geometry needed
57      // The vertices MUST be specified in LEFT-HANDED (CLOCKWISE) winding order
58      eae6320::Graphics::VertexFormats::sVertex_mesh mesh1Vertices[] = {
59              {0.0f, 0.0f, 0.0f},
60              {1.0f, 1.0f, 0.0f},
61              {1.0f, 0.0f, 0.0f},
62              {0.0f, 1.0f, 0.0f}
63      };
64
65      uint16_t indexDataMesh1[] = { 0, 1, 2, 1, 0, 3};
66
67      eae6320::Graphics::VertexFormats::sVertex_mesh mesh2Vertices[] = {
68              {-1.0f, -1.0f, 0.0f},
69              {-1.0f, 0.0f, 0.0f},
70              {0.0f, -1.0f, 0.0f}
71      };
72
73      uint16_t indexDataMesh2[] = { 0, 1, 2};
```

```
299         auto result = eae6320::Results::Success;
300
301         // Initialize Geometry from Mesh Class
302         result = mesh1.InitializeGeometry(mesh1Vertices, indexDataMesh1, 4, 6);
303         result = mesh2.InitializeGeometry(mesh2Vertices, indexDataMesh2, 3, 3);
304         return result;
```

On a 64-bit system, a single mesh takes up 32 bytes on a Direct3D platform and 12 bytes on OpenGL. This is because in Direct3D, the PlatformData struct stores pointers to ID3D11Buffer and cVertexFormat objects, which results in large memory usage due to the size of the pointers. There is not really an effective way to reduce the size because the pointers are necessary for the vertex and index buffers as well as the vertex format.

```
26      private:
27          // Platform-specific private data will be declared here but defined in platform-specific files
28          struct PlatformData;
29          PlatformData* m_platformData = nullptr;
30      };
```

However, I believe my approach to initialize a mesh can be improved going forward. Instead of requiring the user to input the vertexCount and indexCount manually, I believe those values can be calculated from the vertexData[] and indexData[] arrays. However, sizeof() can only be used when dealing with statically sized arrays, so I could not figure out a way to calculate those values in the given time. But, I will look into it going forward.

[Download Game]