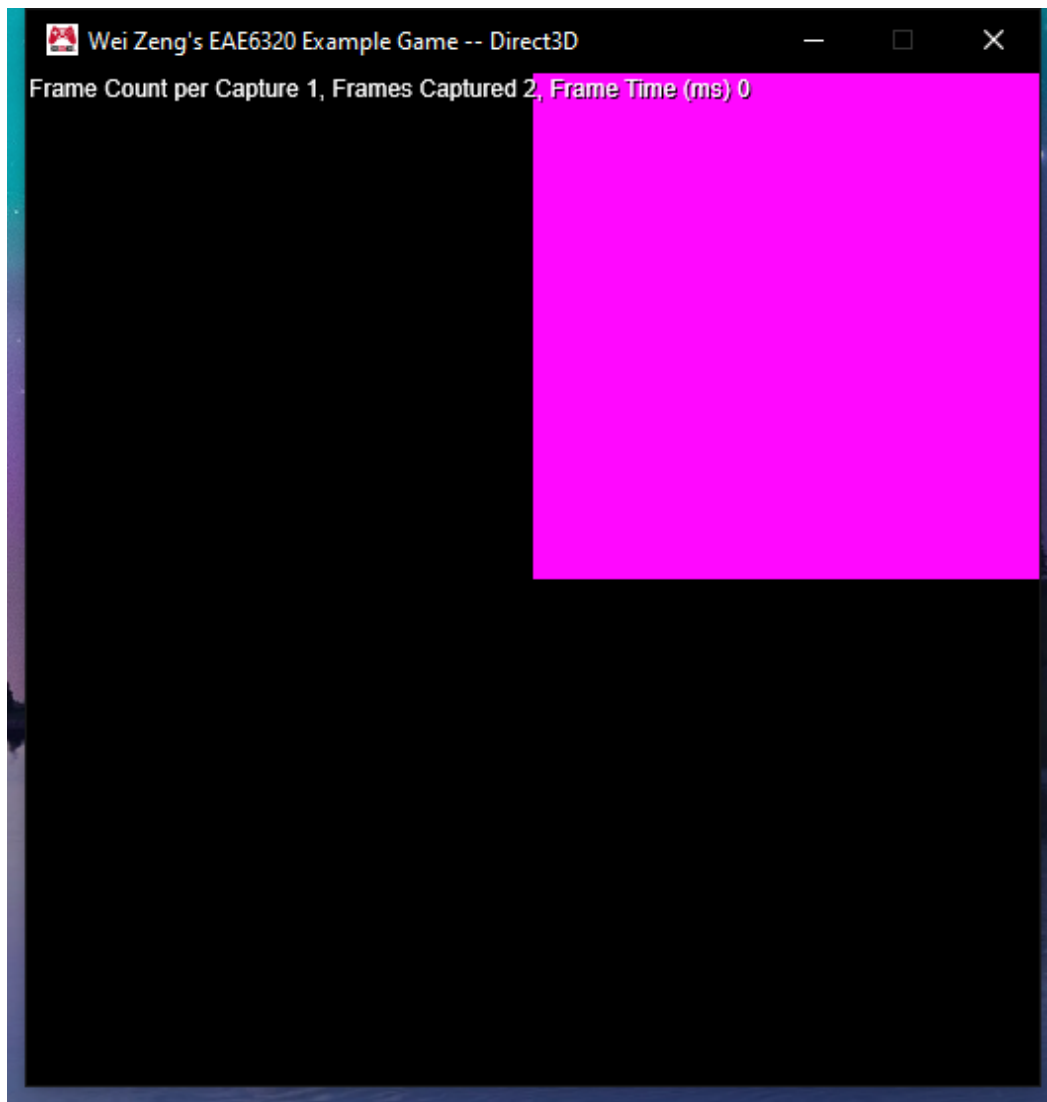


Assignment 2 Writeup

Game Executable:

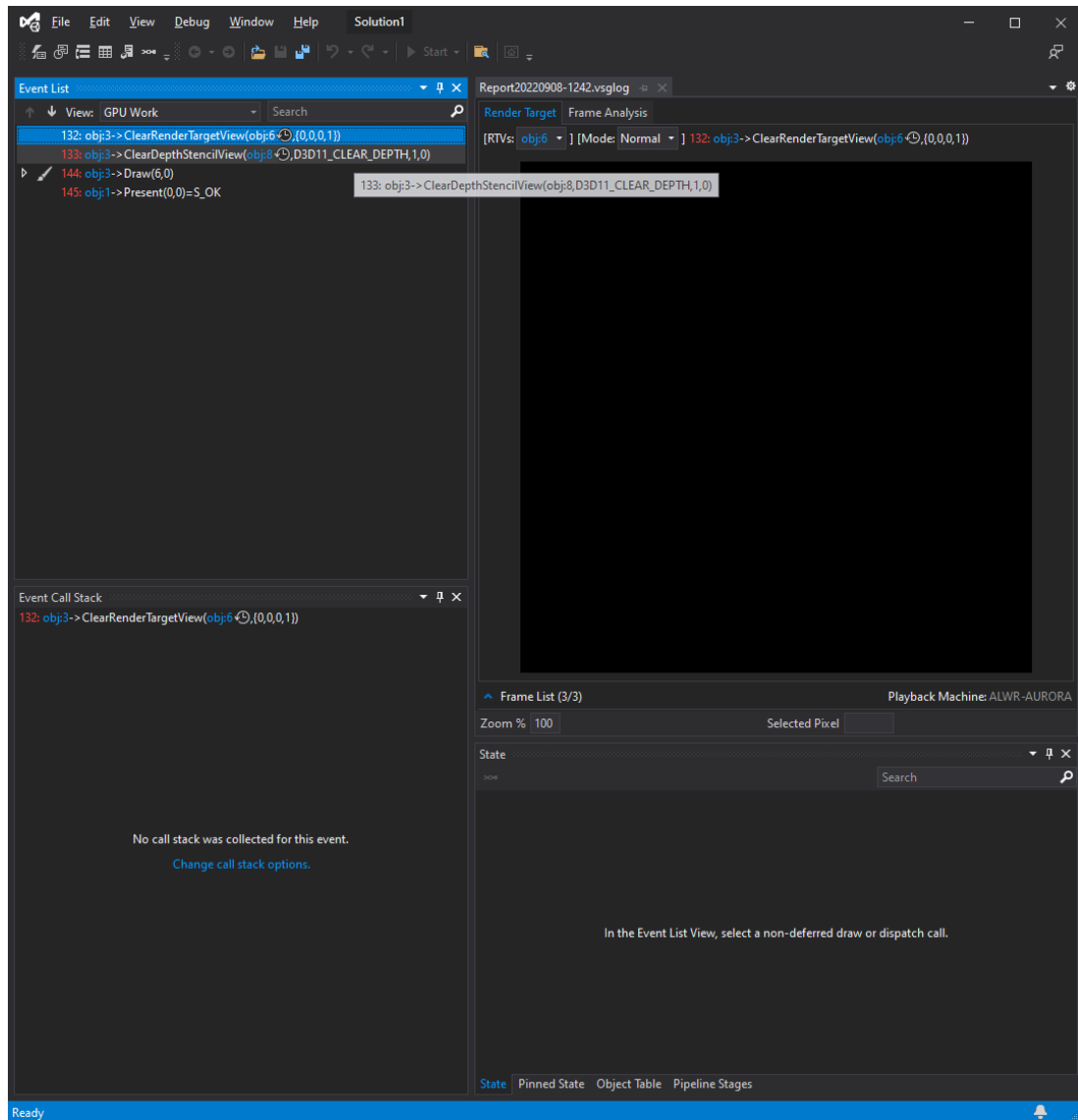
https://github.com/WayGold/EAE6320_Assignments/blob/main/MyGame_.zip

Game running:

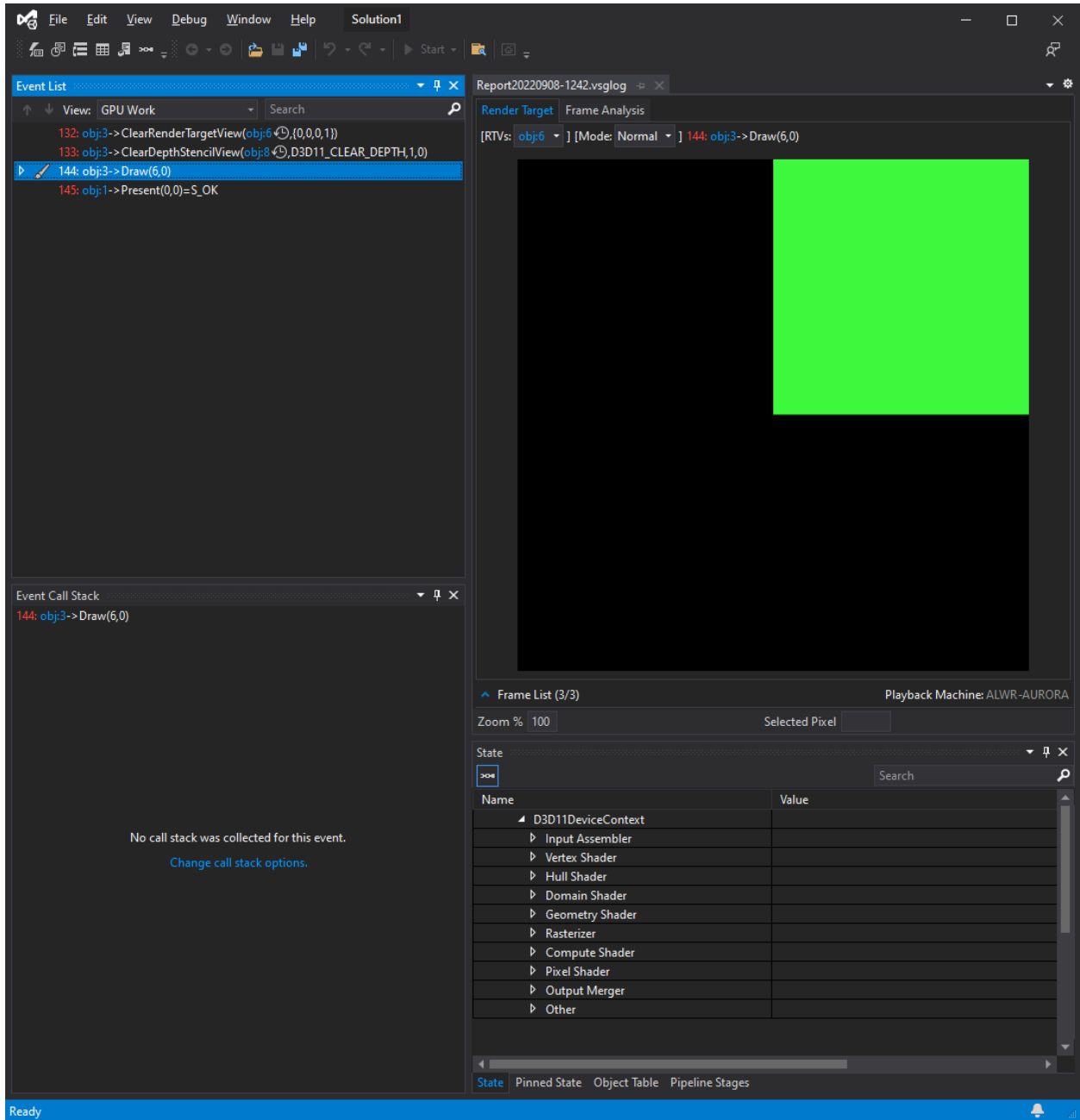


(Figure 1: two triangles animation)

Direct3D GPU capture:

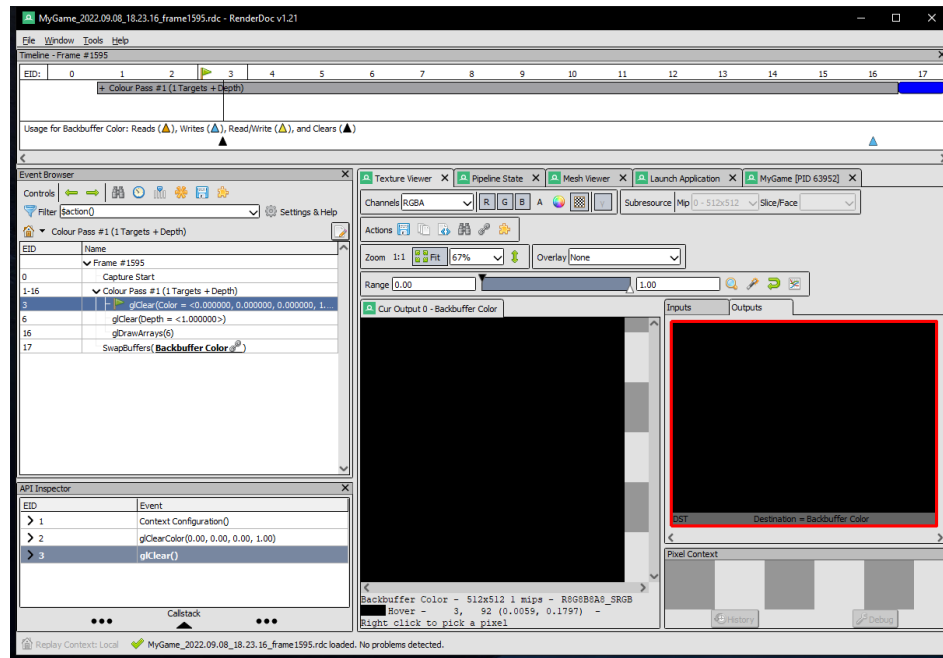


(First Capture: render target when it is all black)

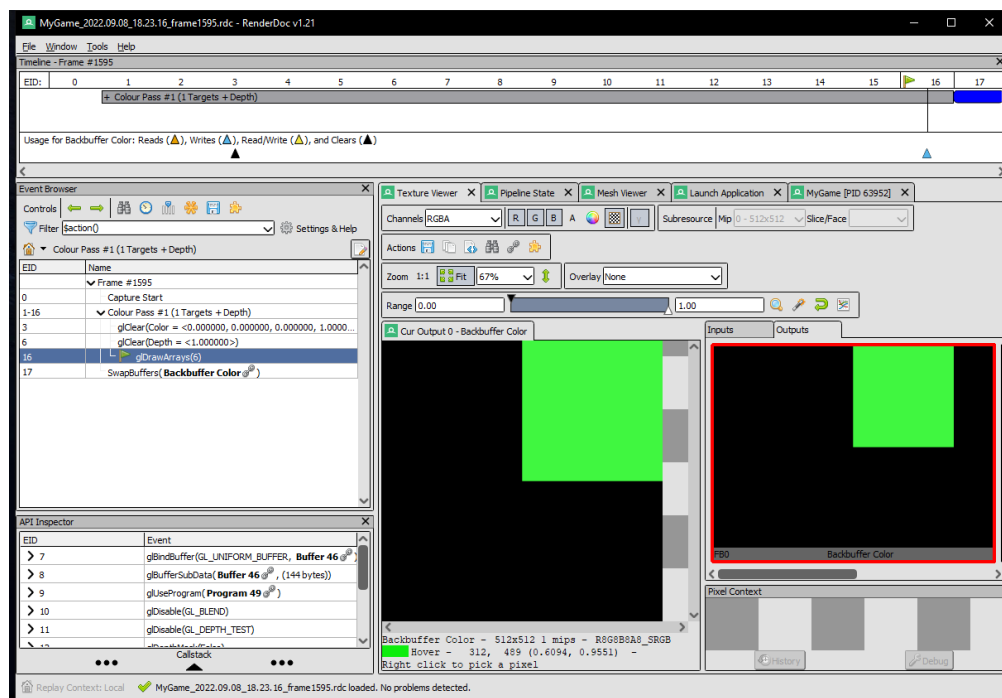


(Second Capture: render target with the mesh)

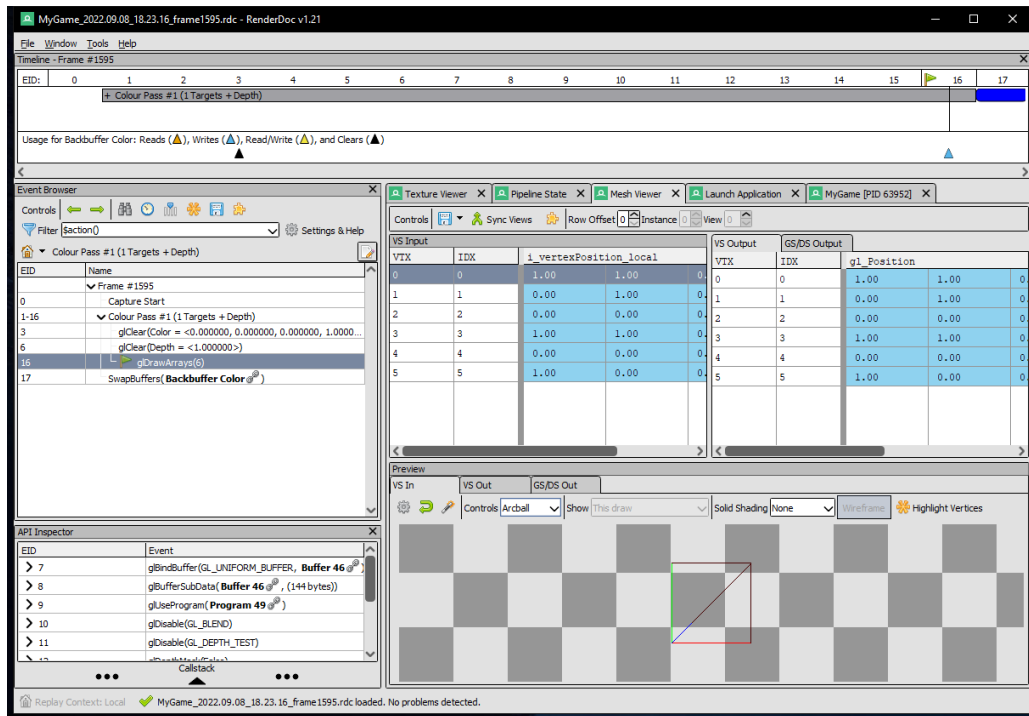
OpenGL GPU capture in RenderDoc:



(First Capture: render target when it is all black)



(Second Capture: render target with the mesh)



(Third Capture: show the mesh's two triangles)

Code that binds the effect and draws the mesh in Graphics.d3d.cpp/Graphics.gl.cpp:

```

185 // Bind the shading data
186 {
187     s_effect_1->Bind();
188 }
189 // Draw the geometry
190 {
191     s_mesh_1->Draw();
192 }

```

(Graphics.d3d.cpp)

```

188 // Bind the shading data
189 s_effect_1->Bind();
190
191 // Draw the geometry
192 s_mesh_1->Draw();

```

(Graphics.gl.cpp)

With the addition of the mesh and effect class, the difference between the two platform dependent Graphics.cpp files is greatly reduced. However, there are still places in each file that depend on platform specific instructions. For example, both files have platform specific instructions for clearing the color and depth buffer. The d3d graphics file also has a InitializeViews function with lots of d3d instructions. Like how we manage geometry and shaders, in order to eliminate the platform specific codes, we could have new interfaces for clearing color and depth buffers and refactor the platform specific codes into the newly created platform specific files, similarly for initializing view.