

Write Up

Assignment 01 – GAMES6320

[Download Game](#)

The first assignment of this class was a great opportunity for me to learn how to set up new projects in Visual Studio, as well as build them correctly. Before this assignment, I had never worked with a Visual Studio solution that contained so many projects, so it was really interesting to dive into how they work with each other.

To begin the assignment, I first went through all the example solutions that were provided (Solution Setup, Static Library, and Platform Specific Code). Reading through these pages was a great learning experience even before starting the assignment itself. For example, I learnt how to set up project dependencies, the difference between static and dynamic libraries, and also how and when to use the preprocessor for platform-specific code. Just playing around and exploring these examples gave me a good idea of what I should do going forward.

After reading through the example programs, I downloaded the assignment solution to get started. At first, I was a little overwhelmed seeing all the various projects and code. However, I just followed along carefully with each step listed in the assignment description.

References

Adding the Graphics project and property sheets was not a problem at all, as I had already practiced this in the Solution Setup example. However, when I was adding the Project Dependencies, I learnt a few interesting things.

First, I went through all the files in the Graphics project to see what other projects it's using functions from. For the Graphics project, I added references to the following projects:

- Asserts

- Assets
- Concurrency
- Logging
- Math
- OpenGLExtensions
- Platform
- Results
- ScopeGuard
- Time
- UserOutput
- Windows

For most of these projects, I could figure it out by looking at the source files. However, I did not add OpenGLExtensions as a reference at first. After I got a linker error when debugging in x86, I realized that OpenGLExtensions is a project that only the OpenGL files depend on. After that, I went through all the Platform-specific files once again to make sure I did not miss anything.

Even after adding all the references that I thought were correct, I still got one linker error regarding the D3D11 Library. After trying to debug the issue, I thought it had something to do with Windows and the Include files. However, I later realized that the program was failing to link an external library. So, I went back to the example readings to figure out the problem. I had forgotten to change the 'Forced Include File' field under the properties section, which means the program was not getting access to the library file. Once I changed that, everything worked fine.

The projects that needed a reference to the new Graphics project were:

- Application

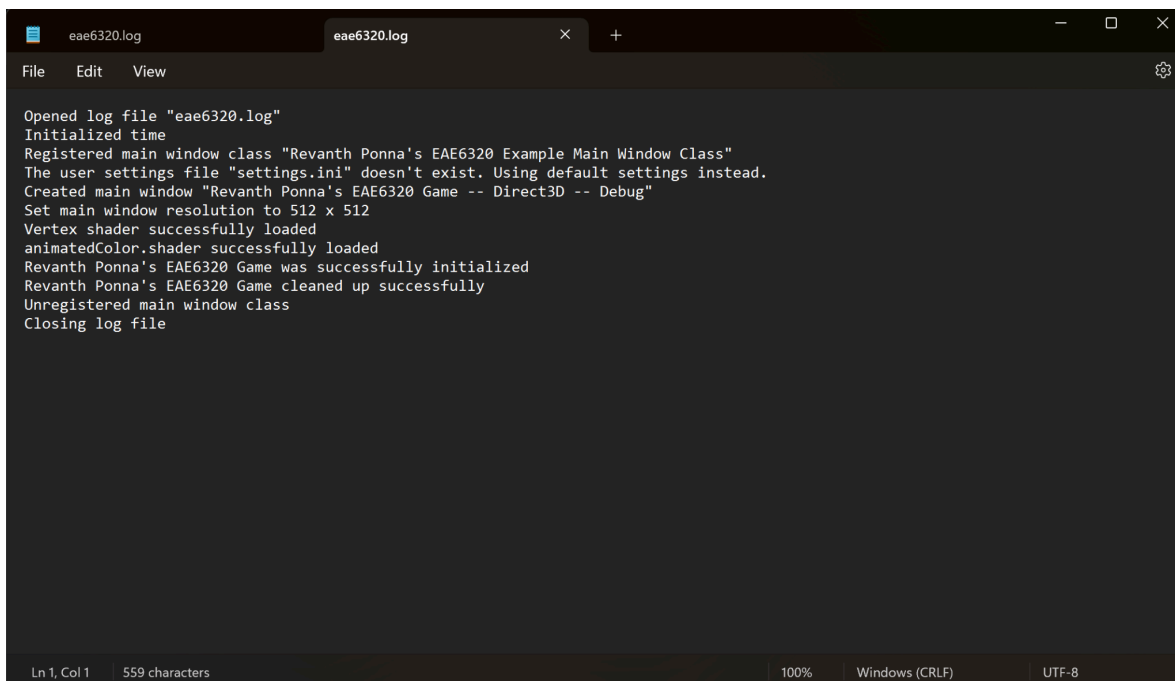
I also found that the ShaderBuilder project used the Graphics namespace, but it was only dealing with enumerations. A reference is only needed when a project calls a function from a CPP file, because it means that the project depends on the result of the function called. Since ShaderBuilder was not calling any function but only working with enums, it technically does not 'depend' on Graphics. Hence, it does not need a reference to the project.

After getting all my references and project dependencies working, I moved on to create the personal game project.

MyGame Project

Creating the MyGame project and copying over the files from ExampleGame was a step where I had to be really careful about the project settings. Tweaking the solution and project files from the text editor was an interesting way to look under the hood and see how they are structured. Although I saved the value of the ProjectGuid and copied it over, I have not yet fully learnt the purpose of this step. I would like to learn more about what ProjectGuid really represents and what its significance is.

Next up, I changed the game's window name, icon and tried to incorporate logging messages into my program. For now, I have logging messages for successful initialization, clean-up, and loading of the shaders. One thing I realized very late was that I added logging messages for only the Direct3D platforms at first. So, I had to go into the OpenGL-specific files and do the same later on. I wonder if I could write these messages only once for both platforms, instead of copying over the code. It is something I will look to work on as I build this project, as well as adding more logging messages at key intervals of the program.



```
Opened log file "eae6320.log"
Initialized time
Registered main window class "Revanth Ponna's EAE6320 Example Main Window Class"
The user settings file "settings.ini" doesn't exist. Using default settings instead.
Created main window "Revanth Ponna's EAE6320 Game -- Direct3D -- Debug"
Set main window resolution to 512 x 512
Vertex shader successfully loaded
animatedColor.shader successfully loaded
Revanth Ponna's EAE6320 Game was successfully initialized
Revanth Ponna's EAE6320 Game cleaned up successfully
Unregistered main window class
Closing log file
```

Ln 1, Col 1 | 559 characters | 100% | Windows (CRLF) | UTF-8

For the color animation, I calculated the red, green, and blue values independently with different sine and cosine functions (and a scaling factor for blue). $\sin(g_elapsedSecondCount_simulationTime)$ and $\cos(g_elapsedSecondCount_simulationTime)$ are used to create oscillating values between $[-1, 1]$. I then scaled and shifted these values to fit within $[0, 1]$ by using the formula: $0.5 + 0.5 * \sin$ or \cos value.



[Download Game](#)

Final Thoughts

All in all, I believe this assignment was an amazing way to learn how to handle Visual Studio solutions with multiple projects. The importance of project dependencies, and platform-specific code are key aspects that I will take away from this assignment. Although I am pretty satisfied with my current implementation, I would like to improve and refactor some aspects of the engine code to avoid duplication, as we discussed in class. However, I believe the base solution provided was clear and well-organized, with good readability and commenting.

For this class, I expect to dive deep and learn more about how game systems work on the engine level. Since the first week, I feel like I have already gained knowledge about graphics, shaders, and game engine architecture. Although being an engine programmer isn't my career goal, I believe that learning about how gameplay systems work integrally can be a massive boost to my knowledge. It will also help me understand my own code when I am implementing gameplay or UI on the surface level as a developer.