**Project Brief**

Project: Specialised 2D Game Engine

Many 2D game engines have the very basics when it comes to providing an environment for making games: such as being able to render 2D geometry, sprites, being able to do vector and matrix orientated maths and 2D physics. Although as an “environment for making games” many game engines do not have the facilities for making the sprites, music and other tasks that make use of other applications/software. My objective and aim of the project will be to make a 2D game engine that includes the facilities to make a game from scratch using only the engine and its interface.

Language: C++

I feel like this is the language that you should learn if you want to do anything with advanced computer graphics, But the main reason that I chose this language is that it provides the means to control the program more than any other language (except for Assembly Language) - this is very important when it comes to graphics programming.

Concepts to understand:

* Vertex Buffers and Arrays
* Index Buffers
* Textures
* Shaders
* SIMD for Maths
* 2D Physics such as objects in motion
* Entity Component System (ECS)
* UI for in-game and engine interface
* Synthesising Sounds from Mathematical equations
* Pixel Art Generation

All the concepts are needed to build a high performance 2D game engine with added functionality. As of now I understand most of these concepts, but I need to do more research on the pixel art generation and how I can implement that into the engine. I will be using "Aseprite" as an example to how to build such a thing simply because it is a specialised piece of software to draw pixel art and it is open source, so I can look at the source code.

[Aseprite](https://www.aseprite.org/)

I will be using OpenGL for this project because it is the API that I have the most experience with, but I will be doing abstractions to keep the engine as versatile as possible and, in the future – if I decide to integrate a new API then it won’t mean a huge refactor.

I also will try and keep the third-party library usage down to a minimum. This is because of the specialised nature of the engine. Many libraries are built to be generic as possible, which isn’t the best for a project like this. I will definitely use GLFW and GLAD for OpenGL but I will try to keep the engine less reliant on other third-party libraries.

[GLFW](https://www.glfw.org/)

[GLAD](https://glad.dav1d.de/)

Software to take inspiration and ideas from:

* Unity
* Godot
* GameMaker Studio
* RPG Maker

Things to code:

* Abstractions
* Vertex Buffers
* Index Buffers
* Shaders
* Vertex Arrays
* Textures
* Layer System
* Input System
* Batch rendering
* SIMD maths
* SIMD stands for “Single Instruction Multiple Data”, this is an optimisation for maths, because it means that maths can be up to 4x faster (or even faster)
* A CPU check needs to be done to see what level of SIMD the processor can handle
* Vector and Matrix Classes need to be made with the components that are public, so the user can edit them individually
* The Vector and Matrices will have float, double and integer versions which is useful for versatility of precision
* Physics system
* Basic Gravity simulation
* Objects interact with each other
* Other values such as restitution (bounciness) can be added
* Implementation of a rigidbody
* ECS (Entity Component System)
* Every entity can have different components added to them, this includes: “Sprite Renderers”, “Rigidbodies”, “Cameras”, “Tilemaps” and other components that play a role in what a specific entity will do
* Entities can be created, and components added to them at runtime
* UI and text rendering
* Synthesiser that can write to a wav file
* Drawing and writing to png files
* Scene runtime
* I want to have a play button, that will run the game from the current scene
* A scene is an area for a part of the game to take place (that might not be the best description), For example, a game’s main menu is one scene and the game's map can be another scene
* Scripting Language
* A scripting language is the language that the user will write code for the game
* A script is also going to be classed as a component, part of the ECS
* I am hoping to get a native scripting language meaning that the scripting language is going to be the same as the language the game engine is written in (C++)
* Other miscellaneous things
* A pseudo-random number generator
* DeltaTime
* Other helper functions such as a ScreenToWorldPoint() function, which will convert a screen point such as a mouse position to the point in the game world that corresponds to that point