JS based Game Engine

Synthesis document

Delgado Darren

February 13, 2025

1 Introduction

In light of our final project of the computer science bachelor, we wanted to inquire more about the possibility of using a JS based game engine for our card game. At first, the choice was between Pygame, Godot and Unity. Now we have narrowed down our choice to Godot or a JS based game engine.

2 JS language

Why should we use a JavaScript language for a video game? JavaScript is one of the most popular programming languages in the world, primarily used for responsive web development. JS' capabilities extend far beyond creating interactive websites and can even extend to video game development.

The obvious advantage lies in the fact that JS runs on all modern web browsers, making such games accessible to a global audience without requiring players to install additional software. It also has a certain versatility, because it can handle both 2D and 3D game development. And finally, JS interacts easily with other web technologies like HTML5 and CSS3.

2.1 Why we should use JS

The main advantage is a combination of how consistent it is across the most used systems and how rapidly you can develop it.

Going back to our previous choices of engines, JS does try to improve performance, which works better than python.

2.2 Negative poinst about JS as a game engine

It is not often used in the industry, because in most cases it is not the optimal choice. Most quality games are made by large teams with professional skills. This acts as an obvious reason for them not to use an underpowered language. Like python, being an interpreted language will result in it never being as effective as pure game-based engines.

2.3 The possible Frameworks

The possible engines are:

- Phaser: the most popular 2D game framework, known for its simplicity and extensive documentation. Support available for both Canvas and WebGL rendering.
- Babylon.js: Powerful 3D engine that supports WebGL, perfect for creating complex 3D games using JS.
- Three.js: Relatively lightweight 3D engine, great for creating games with nice visuals. Also has an easy-to-use API for working with WebGL.
- Pixi.js: A fast and flexible 2D rendering engine that can handle complex animations and effects. Pixi.js is often used in combination with other libraries to create interactive and visually appealing games.
- Kaboom.js: Easy-to-use game engine that thrives of its simplicity and rapid development. Supports both 2D and simple 3D games.

- Cocos2d-JS: It's an open-source game framework part of the Cocos2d family. Primarily designed for 2D game development. Supports both web and mobile platforms.

To not make the mistake of giving us too many choices, I have limited my detailled research to the first three frameworks.

3 Phaser

The Strengths of Phaser start of with the fact that it is free & open-Source. It is available at no cost with a strong community behind it. It is suited for cross-platform releases and compatibility. Phaser based games can run on web browsers, desktop, and mobile devices.

It is fairly beginner-friendly with a good documentation and a fair share of tutorials. Moreover, it supports both WebGL and Canvas for optimized performance.

The weaknesses of Phaser are that it is not suitable for 3D game development, but that will most likely not impact us. Secondly, there will be some performance limitation. While efficient for its niche, complex games might require additional optimizations, which would harm the easy to learn and use part.

Lastly, to have a desktop based version we need to use compilers that package this web-based game into something tangible for the desktop. This is mostly done with Electron, but we could also use others such as NW.js, Cordova or Game Launcher with Node.js.

4 Three.js

The main strength regarding Three.js is the fact that it boasts with its large number of libraries and extensions. This could potentially help our development team if we're able to find practical libraries that make our lives easier.

A downside is that many vital tools and extensions are not well-maintained, which often leads to compatibility issues. Furthermore, we need to also resort

to a compiler such as Electron.

5 Babylon.js

Babylon.js is a JavaScript library and 3D engine for displaying real time 3D graphics in a web browser using the architecture of HTML5.

This framework has many tools and extensions, which have been quickly integrated into the core framework. This ensure reliable compatibility. Lastly, babylon.js has solid features dedicated for multiplayer games.

A weakness of babylon.js is the fact that we also need to resort to Electron, just like Phaser, to compile something that can be run on desktop. This conversion might bring some problems with it.

6 Conclusion

In conclusion, I'd firstly remind the fact that the JS based languages would have us start with creating a WEB based game before trying to form it into a Desktop game. This simple fact would go against our main project directives, which state that our first task should be to have a working desktop app. This app should then be translated into a web or mobile version, which is one of the features that the non JS engine Godot supports.

If we would still want to take a leap into the world of JS based game engines, I'd believe that it should either be Phaser or Babylon.js due to their reliable and extensive libraries and features. Moreover, our game will most likely be fully 2D, which should not be a herculean task for these frameworks, besides not being a performance dedicated engine.

A Sources

- https://www.vanas.ca/en/blog/best-and-top-javascript-video-game-engines
- https://phaser.io/
- https://www.incredibuild.com/glossary/phaser-game-engine:˜:text=Phaser%20is%20a%20fast
- https://www.html5gamedevs.com/topic/37703-babylonjs-vs-threejs-choosing-a-webgl-framework-for-sony/
- https://www.html5gamedevs.com/topic/37703-babylonjs-vs-threejs-choosing-a-webgl-framework-for-sony/
- https://forum.babylonjs.com/t/babylonjs-to-desktop/1103