A logo on a black background

Description automatically generated with low confidence

Computer Games Development

Project Thesis

Year IV

C00246189

Andrew Greenslade

Date of Submission: 18/05/2023

A picture containing text, screenshot, document, font

Description automatically generated

**Contents**

[Acknowledgements 2](#_Toc54714373)

[Project Abstract 2](#_Toc54714374)

[Project Introduction and/or Research Question 2](#_Toc54714375)

[Evaluation and Discussion 3](#_Toc54714377)

[Conclusions 3](#_Toc54714378)

[References 4](#_Toc54714379)

[Appendices 4](#_Toc54714380)

# Acknowledgements

I would like to thank the following people who assisted in helping to break down the game and its features along with helping to complete this project including:

* Lei Shi, who helped me with coming up with the general concept for the game and helping to break down its gameplay for problems and issues that may arrive and coming up with possible solutions to them.

# Project Abstract

With the rise of blockchain technology’s presence in everyday life, from news reports of ‘crypto-crashes’ to individuals investing in these technologies e.g. cryptocurrencies like bitcoin/Ethereum, the games industry has become no stranger to these technologies. Over the last few years based on the rise in games featuring these technologies, it has become its own genre on storefronts that allow the use of them. E.g., <https://chaingames.io/>

Even with the variety of game types including different blockchain technologies, like racing games, card games, etc, they all are the same type of systematic approach where you are rewarded for doing a certain task, e.g., mining in a game for rewards, earning a card in a game as a NFT, etc. I aim to make my game different by having a randomised reward type of approach, like a loot box/fortune wheel.

# Project Introduction and/or Research Question

Over the last few years, blockchain technologies such as NFT’s (Non-fungible token’s) and Crypto currencies have surged in popularity and the topic of them have even hit mainstream media and news reports on multiple occasions, nearly on a daily basis on multiple Government run news outlets. With is gain in popularity, the games industry has also adopted these two technologies on a mainstream basis. One example of this is with the company Ubisoft, with Ubisoft Quartz, where you can earn, purchase and trade NFT’s that give in game items like unique weapons, armour, etc. A side effect of these technologies are that most of the players that play these types of games that focus around NFT’s or crypto-currencies aim to play with a purpose of making money.

An example of these economy base game’s that has this effect is CSGO (Counter-Strike: Global Offensive), where you can earn crates you can pay to unlock for sometimes unique weapons skins with or without stickers which increase the skins resale value. In this game it has also been applied to graffiti spray’s and sticker capsules you can put on your gun skin, as well as trade or sell individually.

With this project I aim to Investigate the process and technical challenges involved with making a blockchain-based multiplayer game. The purpose I chose for this project was to develop something that I could take into my future and possibly use in a games and software development role, along with expanding my knowledge in an area of games I haven’t had experience with to date.

Enjin’s API is broken into two main components:

**The App:** which is the main project component, which has its own schema separate from the player. This allows the app owner to manipulate and test the app before even stepping into making the game/project files and code.

* **Access tokens:** the project has an access token/secret key which is used for accessing the project schema and manipulating the overall project information that is hosted with Enjin.

**The Player:** the player once again has its own schema to play and work with, which is used for all player-related tasks like getting their wallet address, balances, etc.

* **Access tokens:** The player has their separate access tokens, which are different from the projects, for accessing the player’s relevant information as stated above.
* **Player object:** this is what is returned when using the access token for the player in code. This holds all of the relevant information in one object that is easily accessible and storable. This includes a QR code that new users can scan for accessing and linking their new account with a Enjin specific wallet

The other terminology included with this project is as follows:

**Enjin Wallet:** This is a wallet of sorts which holds all relevant crypto assets for a user, which can be either Cryptocurrencies or NFTs, both of which can be accessed exchanged and relevant to NFTs, minted *(a new NFT made for that user)* and in Enjin’s unity SDK.

**Assets:** These are virtual tokens that can either be fungible/non-fungible. The relevant actions in regard to these assets are:

* Creating tokens,
* Minting tokens
* Sending tokens
* And a few more actions are possible with the API.

**Major Technical Achievements**

A screenshot of a computer program

Description automatically generated with low confidence

Figure 1: The map generator script in unity editor

**Procedural terrain generation:** The procedural terrain, is produced using a Perlin noise function that is built into unity and features several customisable features which include:

* Randomised Seed to generate noise from
* Surface size width & height values
* Custom ore distribution objects (which include their stats like spawn threshold, worth/value, etc.).
* Scale of which to apply the noise.
* Indestructible border-radius
* Different layers
* Blocks to apply to the different aspects of the terrain.

**A screenshot of a computer

Description automatically generatedCustom ore Generation:** this is achieved through each ore being a scriptable object and being spawned in a tilemap using Perlin noise, which means the object can hold data neatly and be reused over and over again (Figure 1).This is achieved in unity from deriving a script from ScriptableObject instead of Monobehaveour (Figure 2)**,** and stating what you want the scriptable object to hold, in this case I also added a create asset menu attribute so I can create the in Unity easily from a dropdown menu like this(Figure 3):

Figure 1: the scriptable object in unity's editor

A screenshot of a computer

Description automatically generated with medium confidenceA screen shot of a computer program

Description automatically generated with low confidence

Figure 2: The drop down create asset menu in Unity

Figure 2: The scriptable object in Visual Studio 2022

**AI:** this AI features a destructible terrain navigation using a custom A\* solution I built in Unity. This allows the possible paths to include the destroyed terrain the player has blown up. The reason I created this was that I had issues using other solutions like A\* pathfinding project (<https://arongranberg.com/astar/>), and another package that I was recommended by another lecturer, which was NavmeshPlus (<https://github.com/h8man/NavMeshPlus>). They both had the same problem where when I went to rebuild the Navmesh (what the AI would use to navigate the map to see what is traversable and not), it would freeze the hosts game, which in turn froze the clients for a second, and this would have to happen either when a player destroyed terrain or a timed interval, which both where not feasible because of the face both actions would have to happen very frequently in the gameplay loop.

**Project Review**

# Conclusions

# 

# References

# Appendices