Athens University of Economics and Business

School of Information Sciences and Technology
Department of Informatics
Athens, Greece

Undergraduate Thesis

MMLAB BLOCKCHAIN GAME

Odyssefs Diamantopoulos-Pantaleon 3180049

Supervisor: Professor George C. Polyzos

January 2022

Odyssefs Diamantopoulos-Pantaleon 3180049
MMLAB BLOCKCHAIN GAME
3180049

January 2022
Supervisor: Prof. George C. Polyzos

Athens University of Economics and Business
School of Information Sciences and Technology
Department of Informatics
Mobile and Multimedia Laboratory
Athens, Greece

Abstract

In this paper we are going to showcase the Distributed Ledger Technologies and emphasize on the Ethereum Blockchain technology. Furthermore, we will analyze the possible fusion between the Ethereum platform and the gaming industry which can drastically improve gaming applications and create a whole new genre. Moreover, we will analyze a Blockchain - Card game application that was made for the sake of this thesis and will present all of its components and then evaluate its strengths and its weaknesses. Finally, we will discuss possible solutions that would improve both the game the application.

Περίληψη

Σε αυτή την πτυχιακή θα παρουσιάσουμε τα Distributed Ledger Technologies και θα δώσουμε ιδιαίτερη έμφαση στην τεχνολογία του Ethereum Blockchain. Πιο συγκεκριμένα, θα αναλύσουμε τον πιθανό συνδιασμό μεταξύ της πλατφόρμας του Ethereum και της βιομηχανίας παιχνιδιών που υπό προύποθέσεις μπορεί να βελτιώσει δραματικά τις εφαρμογές παιχνιδιών και να δημιουργήσει ένα εντελώς καινούριο είδος. Επίσης, θα αναλύσουμε ένα Blockchain παιχνίδι καρτών που φτιάχτηκε στα πλαίσια αυτής της πτυχιακής, θα παρουσιάσουμε όλα τα κομμάτια του και θα αξιολογήσουμε τα δυνατά και τα αδύνατα σημεία του. Τέλος, θα αναφέρουμε τρόπους βελτίωσης τοσο του παιχνιδιού όσο και της εφαρμογής.

Acknowledgement

First of all, I want to express my sincere gratitude to my supervisor, Professor George C. Polyzos, for his continuous and valuable guidance and support during the preparation of this thesis and in my studies in general. I am also very grateful to PhD candidate Iakovos Pittaras for the technical guidance and knowledge he provided me, helping me to complete this thesis successfully and all the members of the Mobile and Multimedia Laboratory (MMlab) in Athens University of Economics and Business, as their research activities pave the way for newcomers, like myself, to study in this amazing field. Last but not least, I would like to thank my thesis partner Ilias Marios Stogiannidis, my family, my parents, and all my friends as well, for their wholehearted support during my studies.

Contents

Abstract iv

Acknowledgements v

1 Introduction 1
- 1.1 Motivation and Problem Statement . . . 1
- 1.2 Thesis Structure . . . 2

2 Background and Related Work 3
- 2.1 Background . . . 3
  - 2.1.1 Distributed Ledger Technologies . . . 3
  - 2.1.2 Ethereum Blockchain . . . 4
  - 2.1.3 Unity Game Engine . . . 4
- 2.2 Related Work . . . 5

3 Game Rule Book 7
- 3.1 Basic Card Information . . . 7
- 3.2 Game Rules . . . 10
- 3.3 Card Effects . . . 16

4 System Design and Implementation 17
- 4.1 Design . . . 17
- 4.2 Implementation . . . 18
  - 4.2.1 Card Game . . . 18
  - 4.2.2 Android Application . . . 24
  - 4.2.3 Servers . . . 26
  - 4.2.4 Ethereum Block chain . . . 26

5 Evaluation and Future Work 29
- 5.1 Evaluation . . . 29
  - 5.1.1 Performance and cost evaluation . . . 29
  - 5.1.2 Qualitative evaluation . . . 31
- 5.2 Future Work . . . 31
  - 5.2.1 Multiplayer . . . 32
  - 5.2.2 Adding New Cards . . . 32

5.2.3 Merging some of the application's components 32
5.2.4 User information Database 32
5.2.5 Improvements on the AI 33
5.2.6 Reward System 33

6 Conclusion **35**

Bibliography **37**

List of Acronyms **39**

List of Figures **41**

List of Algorithms **43**

1 Introduction

In this chapter I am going to introduce the topic of this paper, provide a statement for why I decided to devote time in this field and finally showcase the structure of the thesis.

1.1 Motivation and Problem Statement

In the twenty-first century there has been a burst of exciting new technologies that have influenced every aspect of human life. The Distributed Ledger Technologies(DLTs) have had a huge impact on the research and finance communities and have inspired many people to develop numerous applications that have redefined how we conceive decentralized systems. A form of DLTs that has been dominating the industry over the past 2 years is Blockchain. A Blockchain is essentially a shared database filled with entries that must be confirmed and encrypted([Net18]). Researchers have tried to blend it with many different industries and achieved results in securing the sharing of medical data, on applications focused on the Internet of Things([Dal21]), and have even managed to create entirely new currencies that are fundamentally different than anything that has existed so far, such as Ethereum([Eth22]). However, there is a huge industry that so far very few have ever thought that it could be benefited by the rise of the Blockchain. This is none other than the gaming industry which has a global market worth of 180 million dollars, almost double the size of the global filming industry([NAD21]). Advancements in technology, such as improvement in Machine Learning algorithms and development of Augmented Reality (AR) and Virtual Reality (VR), have always pushed game developers to create brand new and immersive content that continuously elevates gaming experience. The combination of the gaming industry and the Blockchain technology allowed the creation and the rise of the first gaming economies that have produced exciting results and built fascinating new communities([Dap22]). Is this just a trend that will gradually fade away and become forgotten, or does it have the potential to shake the foundation of one of the biggest industries worldwide? In this thesis, the aim is to showcase a Card Game that I developed on the Unity gaming engine that utilizes some of the benefits of the Ethereum Blockchain and critically evaluates its potential.

1.2 Thesis Structure

Chapter 2

In this section I am going to provide all the necessary background information that is needed for someone in order to understand the contents of the thesis. Additionally, I am going to mention some related work that is close to the contents of this research.

Chapter 4

In this section I am going to talk about the system structure and carefully analyze all of its components, as well as their connection and provide insight to some of the important algorithms that exist in this application.

Chapter 5

In this section I am going to evaluate the contents of my work and provide possible future upgrades or additional features that the application could get in order to improve.

Chapter 6

In this section I am going to provide the conclusion of the thesis.

2 Background and Related Work

2.1 Background

In this section I am going to give a brief explanation of the technologies used in this application. More specifically, I am going to explain what are DLTs, what are their advantages (2.1.1) and provide more details on the Ethereum Blockchain (2.1.2). Lastly, I am going to provide an introduction to the Unity Game engine, showcase its strengths (2.1.3).

2.1.1 Distributed Ledger Technologies

A ledger is an often used term in accounting and it is simply a collection of financial accounts([Ins22]). Almost all company ledgers in the past were managed and checked by a third party that had full control over them. That is what accountants were hired for and as expected this practise promoted a centralized system. The DLTs provided an alternative to this problem by setting a set of protocols and utilising cryptography in order to make sure that the function of a decentralised database is safe([FRA21a]). This technology eliminated the need for a central authority to keep check on possible,intentional or not, manipulation of the ledgers([FRA21a]). More specifically, distributed ledgers make sure that that the financial records are stored and managed by many different groups in distinct locations([Ins22]).Since these ledgers can be accessed by multiple people, if someone decides to add a transaction, then everyone must be notified and their databases must show the changes made([Ins22]). In order to make sure that the databases are accurate and they indeed show the correct information, they are synchronized([Ins22]).

The DLTs has a lot of advantages. First of all, the need for a centralized authority and all the bureaucracy associated with this is eliminated. This saves a lot of time and money that many times can make the difference([Ins22]). For example, in the supply chain field, sensors can do the work of the third party and thus save a lot of time and effort([Ins22]).Another important advantage is the safety associated with this technology. After the records are saved into distributed ledgers, no party can change them. In other words, once the information is distributed in the network it

becomes an immutable database([FRA21a]). Lastly, the high transparency that is available is surely one of the greatest advantages. Once information gets stored, it is extremely easy for everyone to access it and view it, something that many industries strongly desire([Ins22]).

2.1.2 Ethereum Blockchain

While the Blockchain technology is often thought of being the same with the DLTs, this is not true. In fact, the former is a type of the latter, but since it is the most popular of that group of technologies, many people confuse it as being the very same thing([Net18]). A Blockchain is a database that contains files that need to be safely encrypted, but the crucial difference is that each file has to have a logical relationship with the files before it([Net18]). In other words, there are "blocks" of transactions that are being added to an already existing chain of transactions([Net18]).

Ethereum is a huge platform that was launched on 2015, which is based on Blockchain technology([FRA21b]). This platform has its own type of coin, which is called Ether and has grown enormously over the last few years. As we mentioned, Ethereum uses Blockchain to establish a secure environment for all its user, as millions of computers worldwide maintain its vast network([FRA21b]).

The great thing about Ethereum, however, is that it supports the function of smart contracts, which give strong incentive to creators to develop stunning decentralized applications. A smart contract is a relative simple computer program that helps facilitate the interchange between two different parties([Sim21]). These contracts facilitate immutable transactions and their verification are carried out by various anonymous parties on the Ethereum platform, thus securing trust between both parties([Sim21]). Using these contracts, users can create any kind of decentralized applications that may involve trading. For example, games are a perfect example of an application that would benefit from this technology, since many times users wish to exchange assets with one another in a way that they are sure that they cannot be scammed.

2.1.3 Unity Game Engine

A game engine is a highly specialized piece of software developed specifically for helping people build games([MAR20]). It is essentially an IDE but for game development. When using one, the user can modify everything in the "game" world, from the placement of different objects, to the handling of their behavior and their

environment([MAR20]). It is also possible to develop levels and prepare the game for many different platforms.

Unity is one of the most famous and most used game engines worldwide. Many famous games have been developed using this engine, like City Skylines, Rust and the all time classic Cuphead([DRA20]). It also allows the developer to cater his application to almost all operating systems and platforms without needing to change anything in his code. It is one of the most common gaming engines to use and is definitely a good choice for 2d graphic games. Lastly, Unity utilizes the C Sharp programming language and promotes object oriented programming.

2.2 Related Work

As mentioned in the introduction, there are not many Blockchain applications that have focused on the gaming industry. With that being said, there are a few successful games that are really worth looking into. Perhaps the most popular one is Axie Infinity [1], which is essentially a replica of Pokemon. However, in order to play Axie Infinity, the user must first buy three little monsters, also known as Axies, that each cost from two hundred dollars to even more than a thousand depending on their rarity and their abilities[2]. Another game that has grown really fast and is probably the closest to what is going to be shown in this thesis, is Gods Unchained. It is a card game that is completely free to play and also utilizes the technology of Blockchain[3]. This game also has its own marketplace where you can buy and sell cards [4]. Both of these games use the Ethereum platform and utilize the advantages of the Solidity language.

[1] Axie Infinity, https://axieinfinity.com
[2] Axie Infinity Marketplace,https://marketplace.axieinfinity.com
[3] Gods Unchained, https://godsunchained.com
[4] Gods Unchained Marketplacehttps://market.x.immutable.com/assets?collection=0xacb3c6a43d15b907e8433077b6d38a

3 Game Rule Book

In this chapter I am going to showcase the components of the Card game and explain its rules.

3.1 Basic Card Information

The most important part of every card game is the structure of the cards. Each card has certain characteristics. Firstly, each card has a different name which is located on the top right corner of the image.

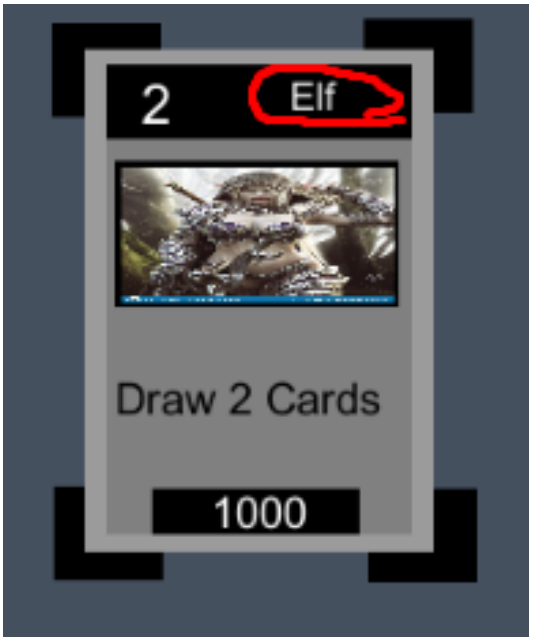

Fig. 3.1: Card Name

Additionally each card has a certain cost. In other words, cards cannot be played for free and depending on how powerful they are, they have a different cost.

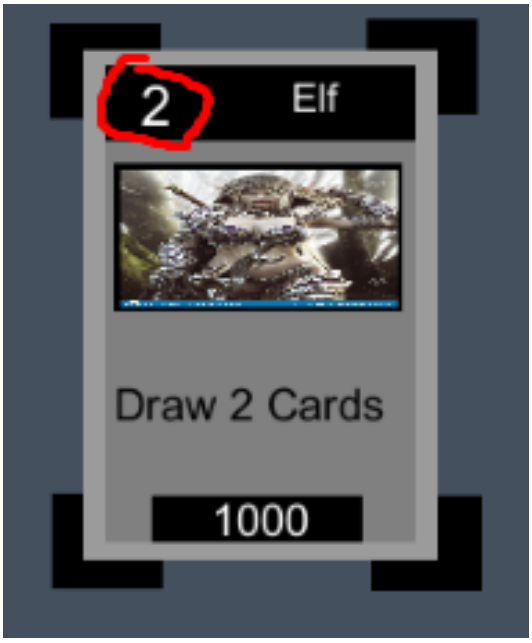

Fig. 3.2: Card Cost

Moreover, each card has a certain power. The power acts simultaneously as both the life of the card and its destructive ability. The more power a card has the more it may cost.

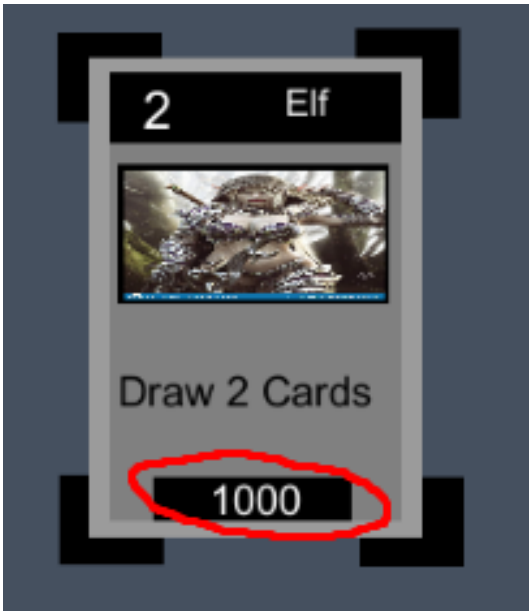

Fig. 3.3: Card Power

Lastly, every card has some kind of effect that is activated when the card is played. In this example, when the Elf with the cost of 2 is played, the user will draw 2 additional cards.

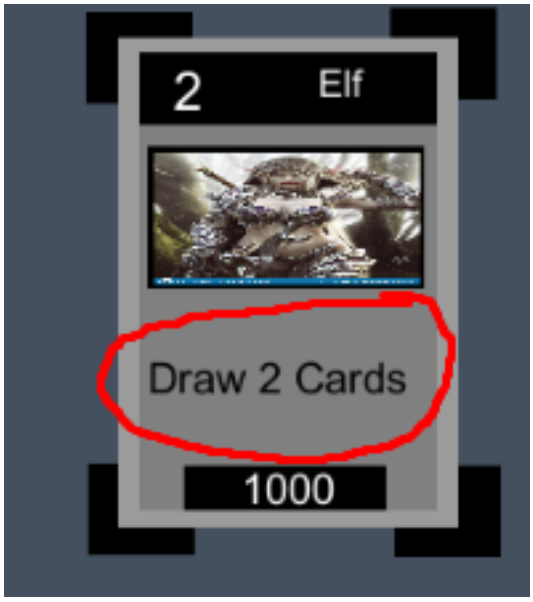

Fig. 3.4: Card Specific Power

There are also two types of cards. The first type is the one showcased above, which is cards that summon fighting characters. The second type is the spell card type that cannot be summoned but can only be played for its effect. For example, the fireball card does 3000 damage when played to anything that the player targets and then gets destroyed. We can see that it is a spell card because its power is 0.

3.2 Game Rules

Now I am going to explain some more rules regarding the game. More specifically, I am going to present the win conditions and all the things that every player should be aware of. First of all, every player has its own Life Points. If his Life points hit 0 the player loses the game. If his opponents Life Points hit 0 the player wins the game. The green arrow shows the player's hit-points, while the orange shows the enemy's hit-points. Each player starts the game with 15000 hit-points.

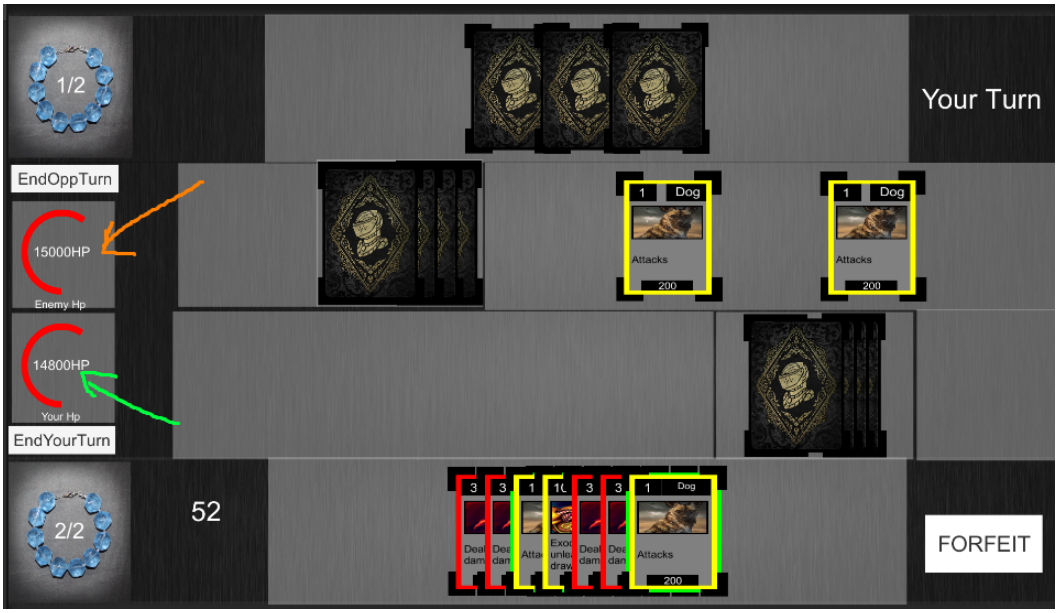

Fig. 3.5: Health Points

Each player also has an area where his cards are stored and its called "Hand". At the start of each game the player draws 5 cards from his deck. Additionally, at the start of each turn the player draws one more card. Each deck contains 40 cards. If someone's deck ever runs out of cards, then that player automatically loses the game.

Fig. 3.6: Player's Hand

Also, a player is able to summon his cards in his Zone.

Fig. 3.7: Player's Zone

Furthermore, each player has a mana pool.

Fig. 3.8: Player's Mana

The mana pool determines whether or not a player can play a specific card. For example, right now the mana pool is 1/1. That means the player can play a card with a cost of 1. The dog card costs 1 and that is why there is a green light around the border of the card.

Fig. 3.9: Cards that the player can play

If the players plays the card, then the cost of the card is subtracted from the player's mana pool and the card is summoned in the player's field.

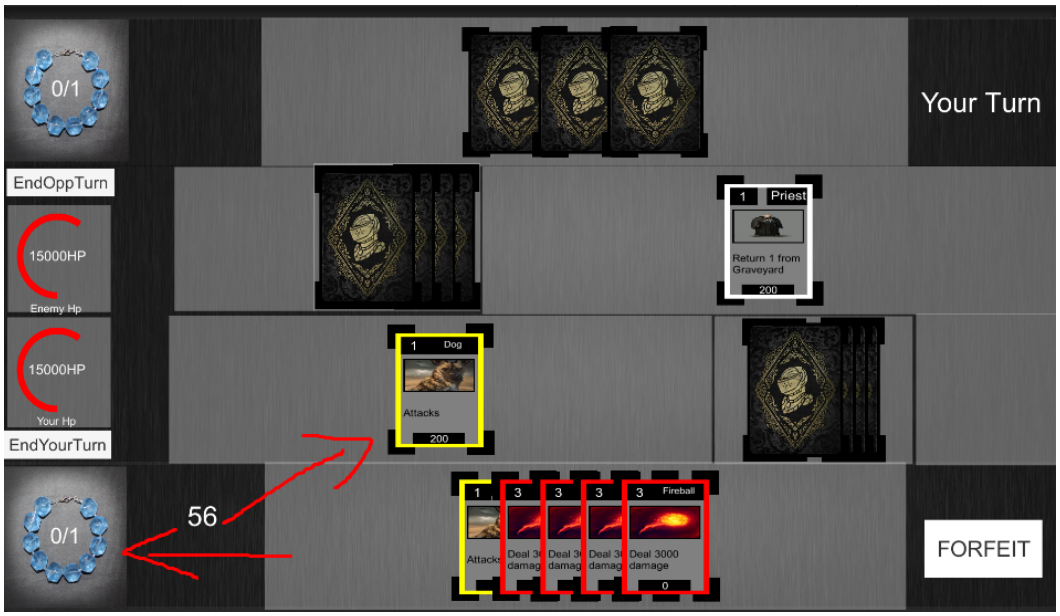

Fig. 3.10: When the player summoned the dog

A card that is summoned in the zone can attack after a turn has passed. When it is able to attack an orange color will show on the border.

Fig. 3.11: Dogs can attack

A card can choose to attack either a monster of the enemy, or the hit-points of the enemy. If the player decided to attack an enemy monster, two things need to be taken into consideration. The power of his own monster and the power of the other monster. When a card attacks another, its power is subtracted from the power of the other card and vice versa. If a card power reaches 0 then the card has died and needs to be transferred to the graveyard. In this example, one of the dogs attacked the green slime. However, the slime initially had 500 power, while the dog had 200. Consequently, the dog died and the slime power was reduced to 300.

Fig. 3.12: After dog attacks

The graveyard is the place where every card that dies goes and is located on the left corner of the screen.

Fig. 3.13: Player's Graveyard

Each player has a set amount of time to complete his turn. There is a timer of 60 seconds that resets every time a players starts its turn. There is also an indication on the top left corner of the application of whose turn is it.

Fig. 3.14: Timer and Turn system

Furthermore, there is a forfeit button that if pressed will allow the players to surrender and lose the game.

Fig. 3.15: Forfeit

3.3 Card Effects

Now I am going to talk about the card effects. The simplest effect is the "Draw X Cards" effect. This effect when activated makes the player draw X cards from his own deck. Another effect is the "Add X Max mana". This effect adds to your mana pool X mana that is replenished at the start of your next turn. There is also the "Heal for X Hp". This effect increases your hit-points by X amount. Moreover, there is the "Return X from graveyard", which allows the player to return X cards from the graveyard to his hand. Lastly, there is the "Deal X damage" effect that simply does X damage to whatever target is selected.

4 System Design and Implementation

4.1 Design

In this section I am going to briefly present all the different building blocks of the application and explain how everything connects. The following graph contains all of the components.

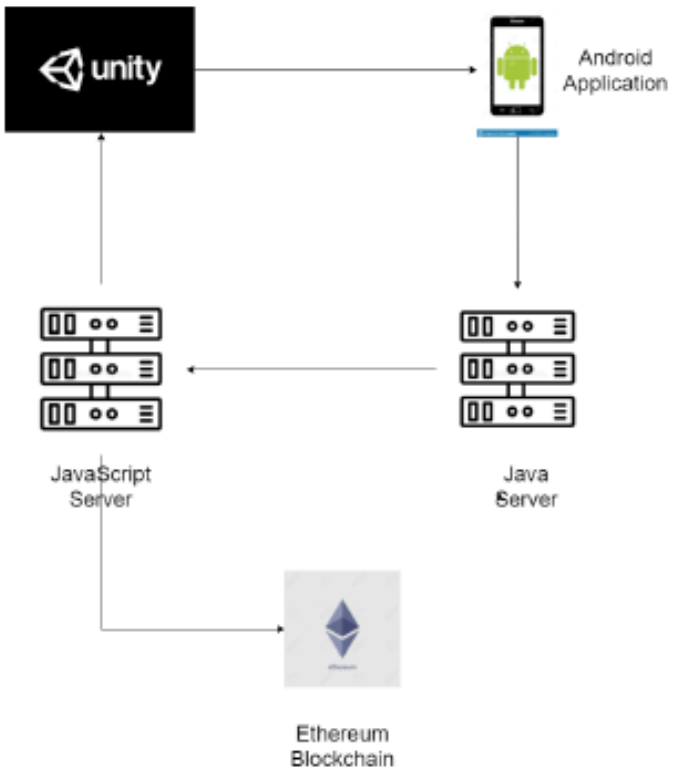

Fig. 4.1: System Graph

The core of this application is the Android app (4.2.2) which harbors the Card game (4.2.1) and collects and transmits user data in order to make sure the connection with the Ethereum Blockchain is possible. More specifically, the Android application

has embedded into it the unity game and receives user data that is later passed on to the Java server (4.2.3). The java server just makes sure to correctly process and pass the data to the JavaScript server (4.2.3) and lets it do the rest. The Java script server connects to the user's wallet and interacts with the tokens inside it by making use of the functions inside the contracts which are uploaded in the Ethereum Blockchain (4.2.4) in order to get all the necessary information about the user's cards.This information is then forwarded to the unity game which then presents the corresponding cards to the user and lets him/her have an exciting experience.

4.2 Implementation

In this section I am going to analyze each and every component of the application and showcase how everything is connected. There are also going to be examples of algorithms that play a crucial role in certain components.

4.2.1 Card Game

The most important and complex component of this application is the Card game developed by using the Unity engine. The game consists of six screens that the user traverses through. In order to keep this report coherent I am going to present each screen and then explain how everything works in it.

The first screen that the user encounters when he enters the game is the Menu Screen.

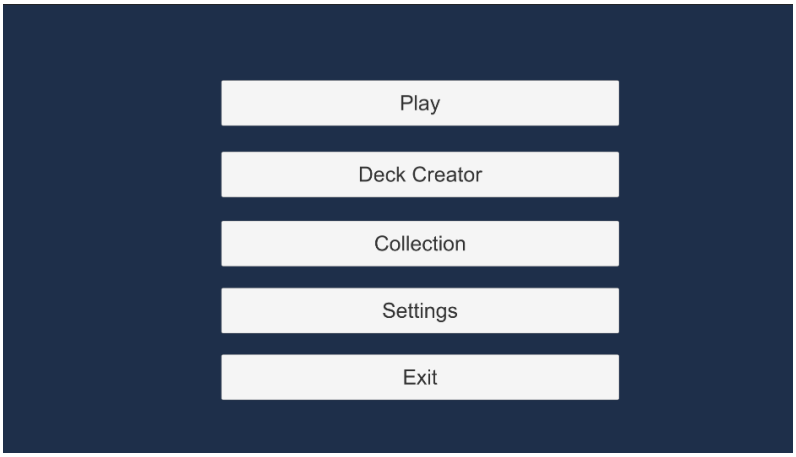

Fig. 4.2: Menu Scene

In this screen the user can see all of his options. If exit is pressed then the application closes immediately. The settings option is not yet developed so it does nothing. The collection option transfers the user to the collection scene (4.2.1), the deck creator option likewise transfers the user to the deck creator scene (4.2.1) and finally the play option lets the user play the game (4.2.1). However, the important part of the menu screen happens in the background. More precisely, the menu screen is responsible for taking the data that the JavaScript server is sending to the game, which contains information about the cards that the user has in its wallet. This connection between the game and the JavaScript server is established thanks to the SocketIO library([Ros20]). The server sends how many copies of a card a user has and then this data is saved to the local storage so it can be used again whenever the user is viewing his collection or when he is trying to create a new deck.

Algorithm 1 Receiving Data from Java script server

while true **do**
    Receive the data and place it in a temporal variable
    Add the card information correctly to the map responsible
    Increase the value of the length of the map

The second screen that I am going to talk about it is the collection.

Fig. 4.3: Collection Scene

Here the user can browse through all of the cards and see which cards he/she owns and in what quantity. To see which cards the player owns the game checks the local storage and gets all the information needed. In order to do that it uses the following algorithm:

If the player does not own a copy of a certain card, then the card's border is colored grey. In this example, the players has only one copy of the fireball card and that is

Algorithm 2 Set Card Values

while k < Length of the Map that contains the card information **do**
    while For each of the 16 existing cards **do**
        if if the card name that we get from the map matches the card name that we are traversing through **then**
            Copy the quantity information from the map to the local storage
        elseIf the value of the card we are traversing through is 0
            Put in the local storage that there are 0 copies of this card
    Modify a variable in order to signify that we have updated the values of the cards based on the local storage information

why only it is the only card with a color different than grey. Using the arrows the player can move to the next or the previous set of 4 cards and if the back button is pressed then the user is returned to the Menu screen.

The third screen that is going to be shown is the Deck creator scene.

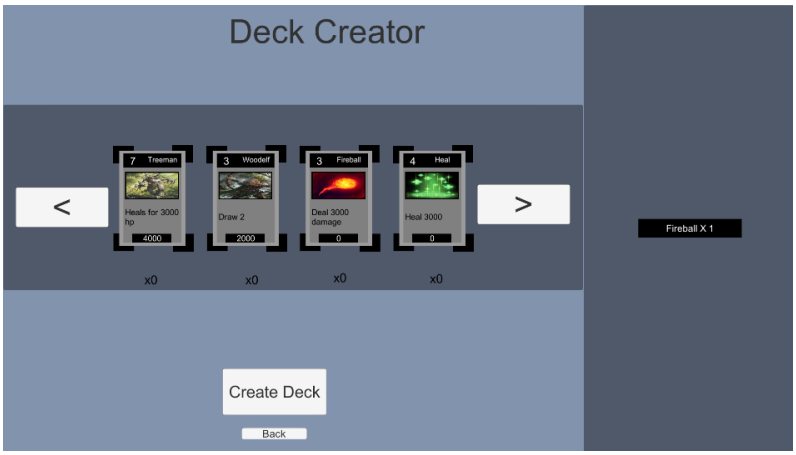

Fig. 4.4: Deck Creator Scene

In order for the user to play the game, he/she needs to create a deck consisting of 40 cards. This can be done in this scene. More specifically, in this screen the player can traverse through his cards, just like in the collection screen, but now he can drag the cards over to the right side and drop them in order to add them to his deck. In the picture the player has added the fireball card to his deck and because he had only one copy of the fireball card, now it is colored grey. The algorithm that stores the player's deck to the local storage is the following:

The fourth and the most important scene is the Game scene.

Algorithm 3 Create Deck

while For each of the 16 existing cards **do**
    Count the number of cards the user has selected for his deck
    if There are 40 cards in the deck **then**
        Save them in the local storage
        Then load them to the list that contains the currently selected deck

Fig. 4.5: Game Scene

Here the player is playing the game against an AI opponent and there are a lot of things happening in background. First of all there is an algorithm that loads the deck that the player has saved to the storage:

Algorithm 4 Load Deck

while For each of the 16 existing cards **do**
    if Copies of card > 0 **then**
        while As many times as the copy of the card **do**
            Add the card to the deck

Another algorithm that is interesting and should be showcased is the algorithm via which the AI opponent selects which cards to play each round:

There are many more algorithms that can be shown. For example, there is one control the timer of each player's turn. If the timer runs out then automatically the turn ends and the other player's turn starts. However, the last algorithm that I am

Algorithm 5 AI

while For each of the 40 deck cards **do**
    Mark the cards that are in the AI's hand
if It is the AI's turn to play **then**
    It is the AI's turn to draw
    while For each of the 40 deck cards **do**
        if If the card is in the AI's hand **then**
            if If the cost of the card is lower than the AI's current available mana **then**
                The card can be summoned
else
    No card can be summoned
    Every possible phase is set false
if If the draw phase is true but the summon phase and the attack phase are false **then**
    Turn the summon phase true
if If the summon phase is true **then**
    Put in a different list the cards that can be played
while For each of the 40 deck cards **do**
    if If the card can be summoned and its id is bigger than the last card selected to be summoned **then**
        Put the id of this specific card to a variable
Summon the card that is saved in the variable

going to show is the algorithm that handles the card attacking which is essential to how the game is played.

Algorithm 6 Card Attack

if If the card is summoned and it can attack **then**
    if If the target is the enemy player's hit points **then**
        Decrease the hit points based on this card's power
        Stop this card from attacking again this round
    elseIf the target is an enemy card
        Decrease the enemy card power based on this card's power
        Decrease this card's power based on the enemy card's power
        Stop this card from attacking again this round

There are three ways for the player to exit the game. The first one is to forfeit it. Then the player loses and is transferred to the defeat screen. The second way is if the player's hit points hit zero or if his deck cards end. Then that will also result to the same defeat screen.

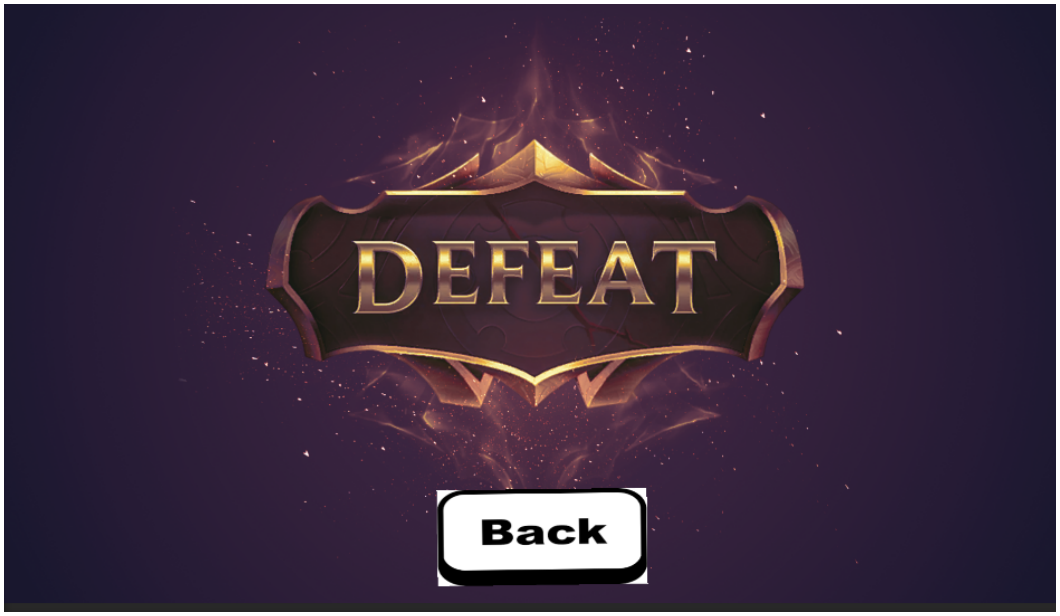

Fig. 4.6: Defeat Screen

The final way is for the player to win. This happens if the player manages to zero the enemy hit points or the enemy's deck ends. In that case the player is transferred to the victory scene.

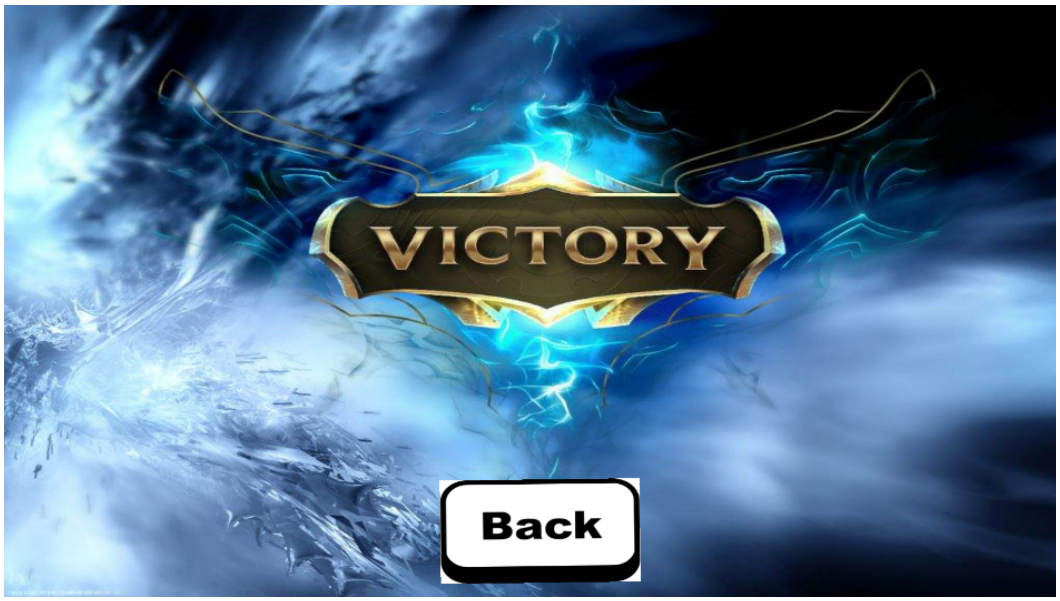

Fig. 4.7: Victory Screen

The back button in both screens allows the player to return to the Menu screen.

4.2.2 Android Application

The android application acts as the body of this work. More specifically, it is responsible for collecting the user's wallet information and also opens the unity game which is embedded into it. When the user first opens the application he is shown the screen that asks for his wallet data.

Fig. 4.8: Android Menu Screen

Here the user writes down his/her name and his special 12 word key phrase. When he/she presses the next button, this data is sent to the Java server through an Async function. The algorithm for this function is the following:

Algorithm 7 Message Sender

while True **do**
    Connect with the Java server
    Send the data to the Java server

The screen that the next button leads to is the one that opens the game. In that screen there is another button that if pressed is going to start unity and load the application.

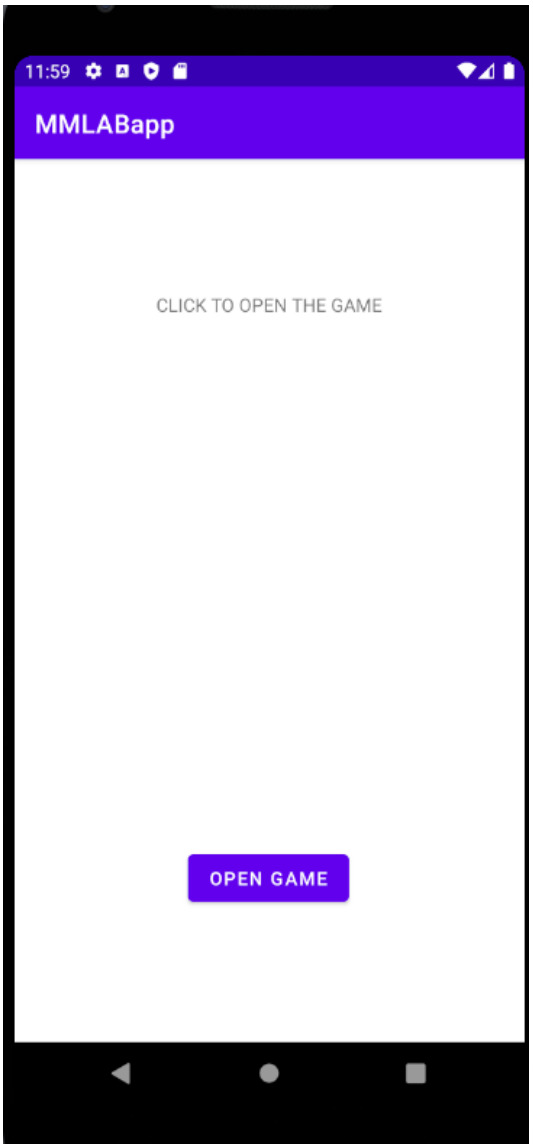

Fig. 4.9: Game Open Screen

4.2.3 Servers

There are two servers in this application. The first server is the Java server that receives data from the android app, processes it and then relays it to the JavaScript server. This is how the Java server functions:

Algorithm 8 Java Server

```
Open the server
while True do
    if If a connection is accepted then
        Gets the data from the application
        Modifies the data and sends it to the Java script server
    else
        Wait
```

The JavaScript server is a bit more complex. It gets the data from the Java server and then connects with the player's wallet. From that wallet it reads all the cards that he/she owns and then when it receives a connection from the unity game it passes this information. The following shows how this server works:

Algorithm 9 Java Script Server

```
Open the server that waits for the Java server connection
while True do
    if If a connection is accepted then
        Gets the data from the java server
        Uses the data to connect with the user's wallet
        Executes a contract for each type of card which connects with the Ethereum
block chain and receives the number of cards of this type that the user owns
    else
        Wait
Open the server that waits for the Unity game connection
while True do
    if If a connection is accepted then
        while For each card that the user owns do
            Send the card information to the unity game
    else
        if If a connection is stopped then
            Say that something disconnected
        else
            Wait
```

4.2.4 Ethereum Block chain

In this thesis everything is going to be conducted in the Rinkeby test network since we are only doing testing and we do not want to involve real money. In order to connect to the Rinkeby test network the user has to have a wallet that supports the

Ethereum platform, like MetaMask [1], which will also help the JavaScript server to collect all the necessary information about the cards.The cards are represented in the Blockchain by tokens. Tokens are entities that the user can own and simply signify the owning of an item, an asset. There are many types of tokens and in our case the type of the token depends on the rarity of the card. For example, common cards use the ERC-20 token([**WEB:erc-20**]) interface which allows multiple copies of the same card to exist. On the other hand, rare cards use the ERC-721([**WEB:erc-721**]) interface which permits the existence of only one copy. These assets are also known as Non Fungible Tokens(NFTs). When the user enters his wallet he should be able to see his token and the number of the tokens he has.

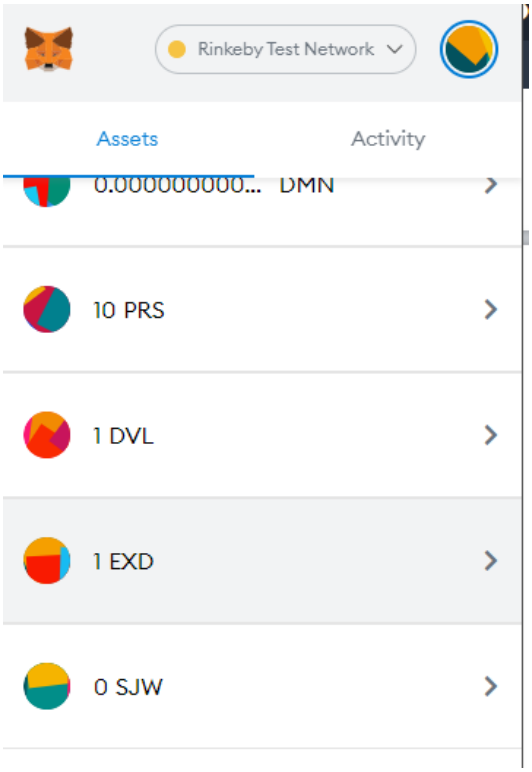

Fig. 4.10: Tokens

In this example, the player has 10 priests, 1 devil and 1 exodia. When the JavaScript server connects with the wallet, it is going to get this exact information.

Each card corresponds to a specific contract. Since there are two types of cards, there are also two types of contracts, that are associated with two types of tokens. The first kind of contract is for the Ethereum Request for Comments-20(ERC-20) token type cards and the second one is the ERC-721 token type cards. The difference between them is the amount of cards that each contract support. The ERC-20 that is for

[1]Metamask,https://metamask.io

common cards supports up to 100 copies, while the ERC-721 that is for rare cards, supports only one copy. Other than that, the contracts support standard functions like the balanceOf that when executed returns the amount of cards of a specific type that the player has and the function transferFrom that allows the transfer of the tokens from one account to another. There is also the constructor, which creates the token, and two other functions that check whether or not the user is allowed to make a transaction or not based on the number of cards he/she has.

5 Evaluation and Future Work

This chapter is going to be divided into two parts. The first part is going to evaluate the application in terms of cost and of time efficiency, while the second part is going to present possible ideas that would evolve the game.

5.1 Evaluation

In this section I am going to evaluate the application. More specifically, I am going to present current gas costs and measure the time needed for different application actions to complete. Also I am going to discuss the qualities that this application offers to the user.

5.1.1 Performance and cost evaluation

In this subsection I am going to evaluate the application using two different metrics. The first one is going to be the gas fees which the account that holds all the game cards has to pay in order to give other users game tokens. As previously mentioned there are two types of tokens. There is the ERC-20, that represents common cards, and the ERC-721 that represents NFT cards. The gas cost for transferring five copies of the common card "Demon" costs us 0.000132 Ethereum which translates to 0.45 american dollars.

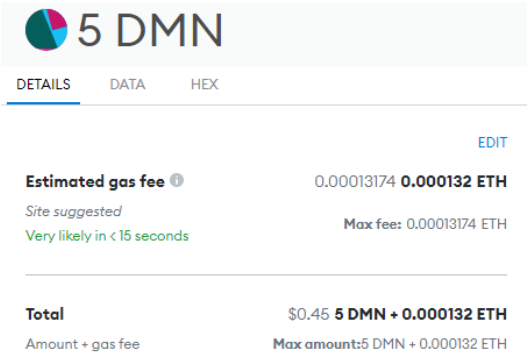

Fig. 5.1: Common Card Gas

However, the gas cost for transferring one copy of the NFT card "Hygeia" is 0.000159 Ethereum which translates to 0.48 american dollars.

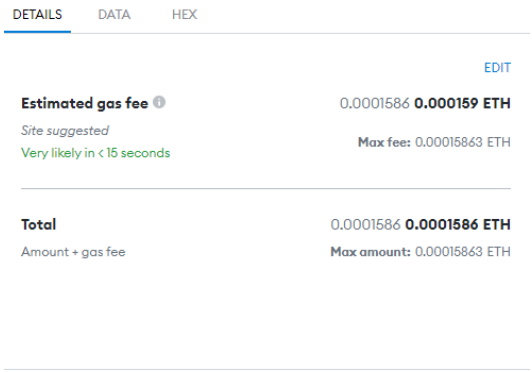

Fig. 5.2: NFT Card Gas

The second metric is the time consumed to perform key actions. The actions that we are going to measure time for are the time that the application needs to send the information and open the game, the time that the java server needs to send the data to the Java script server and the time needed for the Java script server to get the card information from the user's wallet and send it to the Unity game. Across ten runs the average time for the android to open the game is 15.7623 seconds. In order to measure this I used the performance tab of the android studio that provides many different details about the application. Below there is a screenshot of the time the Android application did in the last run.

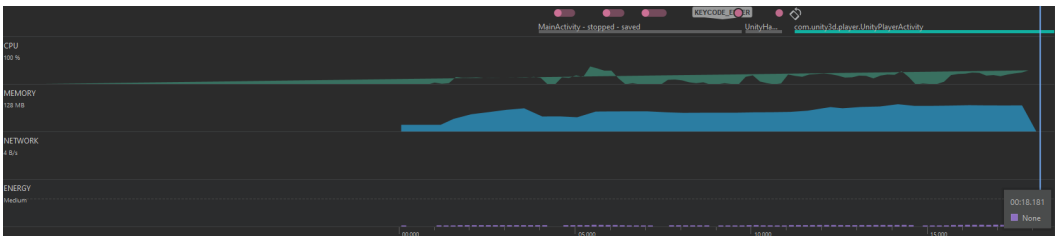

Fig. 5.3: Android Time to Open the Game

By using the System.currentTimeMillis() function I found out that the average time for the java server to send the data is 6.5 seconds. Below there is a screenshot of the time the Java server did in the last run.

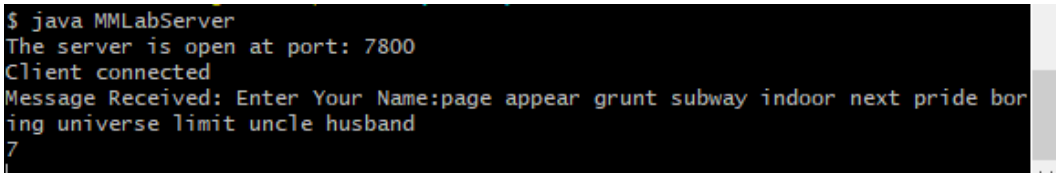

Fig. 5.4: Java time to relay data

Finally by using the performance.now() function I concluded that the average time for the Java script server to relay the card information to the game is 42.6616 seconds. Below there is a screenshot of the time the Java script server did in the last run.

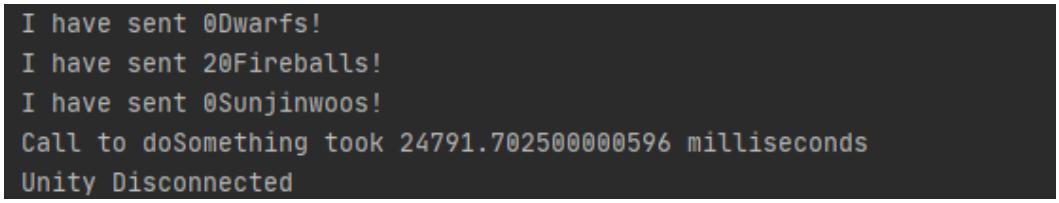

Fig. 5.5: Java script time to relay data

5.1.2 Qualitative evaluation

In this subsection I am going to discuss the qualities of this application. More specifically, this application offers a lot of safety to the player's assets, since in order to access the game account the user needs to input the mnemonic phrase of his wallet, something which is a lot more complex than a password. Furthermore, the user's transactions are immutable, which means that they cannot change under any circumstances once they are finalized and there is a lot of transparency available since anyone can view the content of the transactions and the ones behind them. Finally, the use of unity provides a lot of flexibility when it comes to choosing the platform of the game. The game can be built to be played in almost every operating system including Windows, Android, OS, iOS etc.

5.2 Future Work

In this section I am going to present some ideas that could be implemented in the future and that could drastically improve the gaming experience.

5.2.1 Multiplayer

The first and the most obvious improvement that the game could use is the Multiplayer aspect. Playing against the AI is fun for the first few times, but challenging another random player or a friend of yours would be really exciting. It could be created by setting up the multiplayer on the already existing JavaScript server and transferring game data back and forth from the two applications. This can work on both Internet multiplayer and LAN multiplayer and would be a really beneficial for the game.

5.2.2 Adding New Cards

Another idea that would be easy to implement and would add depth to the game is adding more cards and perhaps even more type of cards. Having 16 cards while the deck consists of 40 is not ideal, because it does not provide much diversity.

5.2.3 Merging some of the application's components

It is evident that the Java server has very little use to the application and is used only to relay a message to the Java script server. It would be possible to merge the Java server with the JavaScript server and perhaps even merge the android application with the unity game. More specifically, I could create the whole application in unity and just put a few screens before the game starts that will get the user information just like it did in the android. Then the unity game would communicate through the SocketIO([Ros20]) protocol with the JavaScript server and give the information that the Java server would otherwise give, and then establish a connection again in order to get the card data of the user just like it did before. This would decrease significantly the amount of the components involved and would make debugging and understanding the application a lot easier.

5.2.4 User information Database

Another step towards the improvement of the game would be to create a database that would save the user information. More specifically, it would be relative simple to implement register and login screens that would make good use of this database and that would no longer require the user to put every time his 12 word mnemonic phrase.

5.2.5 Improvements on the AI

It is evident that the AI algorithm is very basic and could certainly improve and make games more interesting and challenging for the user. There are a number of machine learning algorithms that it would be possible to implement, but I think that the Min-Max algorithm([Epp19]) would make the most sense. That way the AI would search for the most efficient way to defeat its opponent.

5.2.6 Reward System

In order for the player to keep playing, it is extremely urgent to add a reward system. More specifically, when a player wins a battle, he/she can be rewarded with a copy of a card that would be added to his/her wallet. This would encourage players to keep playing to enrich their collection and create even more interesting combinations of decks.

6 Conclusion

All in all, in this thesis we showcased the strengths of the DLTs and more specifically the perks of the Ethereum platform. Furthermore, based on the game that was created and showcased in this thesis we concluded that the fusion of the gaming industry and the Blockchain technology would result into a groundbreaking new genre of games that are going to offer incredible advantages when it comes to security. Therefore, it would be wise to assume that Blockchain games are going to become increasingly popular over the next few year and are certainly the next big thing.

Bibliography

[Dal21] Sam Daley. *34 Blockchain Applications and Real-World Use Cases Disrupting the Status Quo*. 2021. URL: https://builtin.com/blockchain/blockchain-applications.

[Dap22] DappRadar. *Top Blockchain Games*. 2022. URL: https://dappradar.com/rankings/category/games.

[DRA20] JEFF DRAKE. *10 Great Games That Use The Unity Game Engine*. 2020. URL: https://www.thegamer.com/unity-game-engine-great-games/.

[Epp19] Marissa Eppes. *Game Theory — The Minimax Algorithm Explained*. 2019. URL: https://towardsdatascience.com/how-a-chess-playing-computer-thinks-about-its-next-move-8f028bd0e7b1.

[Eth22] Ethereum. *Ethereum Platform*. 2022. URL: https://ethereum.org/en/.

[FRA21a] JAKE FRANKENFIELD. *Distributed Ledger Technology (DLT)*. 2021. URL: https://www.investopedia.com/terms/d/distributed-ledger-technology-dlt.asp.

[FRA21b] JAKE FRANKENFIELD. *Ethereum*. 2021. URL: https://www.investopedia.com/terms/e/ethereum.asp.

[Ins22] Corporate Finance Institute. *Distributed Ledgers*. 2022. URL: https://corporatefinanceinstitute.com/resources/knowledge/other/distributed-ledgers/.

[MAR20] JENNIFER MARTIN. *WHAT IS A GAME ENGINE?* 2020. URL: https://usv.edu/blog/what-is-a-game-engine/.

[NAD21] MICHAEL NADEAU. *5 reasons why blockchain-based gaming economies are the future*. 2021. URL: https://cointelegraph.com/news/5-reasons-why-blockchain-based-gaming-economies-are-the-future.

[Net18] Marco Polo Network. *Difference Blockchain and DLT*. 2018. URL: https://marcopolonetwork.com/articles/distributed-ledger-technology/.

[Ros20] Michael Rosario. *Using SocketIO and Unity 3D To Build Multi-User Experiences*. 2020. URL: http://innovativeteams.net/using-socketio-build-multi-user-experiences/.

[Sim21] Simplilearn. *What is Ethereum: Understanding Its Features and Applications*. 2021. URL: https://www.simplilearn.com/tutorials/blockchain-tutorial/what-is-ethereum.

List of Acronyms

DLTs Distributed Ledger Technologies

AR Augmented Reality

VR Virtual Reality

NFTs Non Fungible Tokens

ERC Ethereum Request for Comments

List of Figures

3.1 Card Name 7
3.2 Card Cost 8
3.3 Card Power 8
3.4 Card Specific Power 9
3.5 Health Points 10
3.6 Player's Hand 11
3.7 Player's Zone 11
3.8 Player's Mana 12
3.9 Cards that the player can play 12
3.10 When the player summoned the dog 13
3.11 Dogs can attack 13
3.12 After dog attacks 14
3.13 Player's Graveyard 15
3.14 Timer and Turn system 15
3.15 Forfeit 16

4.1 System Graph 17
4.2 Menu Scene 18
4.3 Collection Scene 19
4.4 Deck Creator Scene 20
4.5 Game Scene 21
4.6 Defeat Screen 23
4.7 Victory Screen 23
4.8 Android Menu Screen 24
4.9 Game Open Screen 25
4.10 Tokens 27

5.1 Common Card Gas 29
5.2 NFT Card Gas 30
5.3 Android Time to Open the Game 30
5.4 Java time to relay data 31
5.5 Java script time to relay data 31

List of Algorithms

1 Receiving Data from Java script server 19
2 Set Card Values 20
3 Create Deck 21
4 Load Deck 21
5 AI 22
6 Card Attack 22
7 Message Sender 25
8 Java Server 26
9 Java Script Server 26