Abstract

The age of digital disruption provides opportunities for game industry to integrate with new elements and technologies. Blockchain is one of the emerging technologies that integrates with games in recent years. However, high transaction latency for public blockchain becomes a barrier in designing interactive games. Layer 2 solutions are introduced to mitigate latency. They also allow extensive scaling of the blockchain applications without worrying about the security issues.

By incorporating Ethereum and Loom network architecture with a turn-based battling game, the transaction latency will potentially be reduced significantly. The game maintains high interactivity, while still keeping the blockchain network safe and secure. Ethereum also features smart contracts, allowing for flexible game design. This project values the realization of layer 2 solutions in blockchain games and explores the feasibility of scaling blockchain applications with those solutions. Currently, some user interface prototypes are constructed and testing of smart contract is completed. Implementation of ERC-721 smart contract and character randomization are also completed according to the project schedule. Finally, a blockchain turn-based battling game prototype will be constructed in the final stage utilizing continuous design and modular approach.
Acknowledgement

I would like to express my deepest appreciation to my final year project supervisor, Dr. John Yuen, for his professional guidance and continuous support in each consultation throughout the first semester. He also gave invaluable advice on blockchain security which contributes to a robust game design.

In addition, I would also like to extend my sincere thanks to Mr. Cezar Cazan for his constructive advice in writing project plan and progress report. He provided a one-on-one consultation on writing support, assisting me in structuring the documentations for my project.
# Table of Contents

Abstract ........................................................................................................................................... i  
Acknowledgement .................................................................................................................. ii  
List of Figures ........................................................................................................................ v  
List of Tables ........................................................................................................................ vi  
Abbreviations ........................................................................................................................ vii  
1 Introduction .......................................................................................................................... 1  
  1.1 Background .................................................................................................................. 1  
  1.2 Outline ......................................................................................................................... 3  
2 Objectives ........................................................................................................................... 4  
3 Related Works ..................................................................................................................... 5  
  3.1 Introduction of blockchain games .................................................................................. 5  
  3.1.1 CryptoKitties ........................................................................................................... 5  
  3.1.2 Fishbank .................................................................................................................. 6  
  3.1.3 Total Poker ............................................................................................................. 6  
4 Justification on Framework Choices .................................................................................. 6  
  4.1 Overview of Root Blockchain Layer .............................................................................. 6  
  4.2 Root Blockchain Layer Selection .................................................................................. 6  
  4.3 Overview of Layer 2 Solutions ..................................................................................... 7  
  4.3.1 Raiden Network ....................................................................................................... 7  
  4.3.2 Loom Network ......................................................................................................... 8  
4.4 Layer 2 Solution Selection ............................................................................................. 8  
5 Methodologies ...................................................................................................................... 8  
  5.1 Introduction .................................................................................................................... 8  
  5.2 Game Setting ............................................................................................................... 9  
  5.3 Game Mechanism ......................................................................................................... 9  
  5.4 Design .......................................................................................................................... 10  
  5.4.1 User Interface Design ............................................................................................. 10  
  5.4.2 Security Design ....................................................................................................... 12  
  5.4.2.1 Overview ........................................................................................................... 12  
  5.4.2.2 Arithmetic Overflow Avoidance ......................................................................... 13  
  5.4.3 Architectural Design ............................................................................................... 14  
  5.5 Implementation ............................................................................................................. 14  
  5.6 Testing .......................................................................................................................... 15
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Current Progress</td>
<td>16</td>
</tr>
<tr>
<td>6.1</td>
<td>Design</td>
<td>16</td>
</tr>
<tr>
<td>6.2</td>
<td>Implementation</td>
<td>18</td>
</tr>
<tr>
<td>6.3</td>
<td>Testing</td>
<td>19</td>
</tr>
<tr>
<td>6.4</td>
<td>Conclusion</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>Limitations and Difficulties</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>Future Plan</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>Project Schedule and Milestones</td>
<td>22</td>
</tr>
<tr>
<td>9.1</td>
<td>Overview</td>
<td>22</td>
</tr>
<tr>
<td>10</td>
<td>Conclusion</td>
<td>24</td>
</tr>
<tr>
<td>References</td>
<td></td>
<td>25</td>
</tr>
</tbody>
</table>
List of Figures

Fig. 1 Simplified Bitcoin Block Chain by Blockgeeks [2] ......................................................2
Fig. 2 The CryptoKitties marketplace by M. Greg[4] ..............................................................5
Fig. 3 Selected section of blocks in battle .............................................................................9
Fig. 4 The battle system ........................................................................................................10
Fig. 5 The inventory system ..................................................................................................11
Fig. 6 The marketplace system .............................................................................................12
Fig. 7 Architecture of the battling game .............................................................................14
Fig. 8 Character modelling with texture paint ......................................................................16
Fig. 9 The hat model .............................................................................................................17
Fig. 10 The rendered result of a character in Unity ..............................................................18
List of Tables

Table 1 Project Schedule and Milestones .................................................................22
**Abbreviations**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABIs</td>
<td>Application Binary Interfaces</td>
</tr>
<tr>
<td>ERC-721</td>
<td>Ethereum Requests for Comments 721</td>
</tr>
<tr>
<td>HP</td>
<td>Health Points</td>
</tr>
<tr>
<td>PoS</td>
<td>Proof-of-Stake</td>
</tr>
<tr>
<td>PoW</td>
<td>Proof-of-Work</td>
</tr>
<tr>
<td>SDK</td>
<td>Standard Development Kit</td>
</tr>
</tbody>
</table>
1 Introduction

Due to the rise of new technologies, the gaming world has been revolutionizing in recent years, giving players a brand new experience throughout the gameplay. Some game companies research different new technologies to unblock the bottleneck of the market, blockchain is no exception. In fact, there is an existing pioneer in the blockchain game industry called CryptoKitties. It has been a great success with total transaction amount of 32 million USD by April, 2018 [1]. It indicates that blockchain technology has great potential for game development.

However, public blockchain suffers from latency. Each transaction takes a long time for validation which is undesirable for gaming experience. In light of this issue, some layer 2 solutions have been developed on top of the root blockchain layer to support swift transaction and scalable network. This project focuses on the realization of layer 2 solutions by building a turn-based battling game using smart contracts.

1.1 Background

Blockchain is a distributed ledger technology that allows stakeholders to hold a transaction history in the network. In public blockchain, each transaction is validated via a consensus algorithm. The most famous one is called Proof-of-Work, the use of enormous computational power to validate trust-less transactions. Each block in the blockchain corresponds to a transaction which is cryptographically hashed with a mathematical puzzle. People who try to solve the puzzle are called miners. The miner gets some rewards like cryptocurrencies which are newly generated once the puzzle is solved. Then, the transaction is said to be validated and the block is attached to the current
As shown in Fig. 1, each block contains the hash of the previous block to avoid block violation. The transaction is then immutable and the entire network is tamper resistant. In general, the more the computation power, the higher the possibility of getting the rewards.

Despite greater transparency of the blockchain network, one of the famous applications is tokenization. Blockchain technology has been gaining currency these days due to the invention of Bitcoin, which is the first cryptocurrency launched in the world. The founder of Bitcoin, Satoshi Nakamoto proposed a peer-to-peer electronic cash system, promoting anonymous value transfers without third party authorities. Apart from Bitcoin, another blockchain platform called Ethereum focus on decentralized applications that run smart contracts. Smart contracts are sets of well-defined functions and instructions deployed in the blockchain. Every participant in the blockchain network can perform some tasks like initiating a transaction via making calls to smart contracts. Ether, which is the cryptocurrency in Ethereum is used in those transactions. Each transaction costs gas which is the price for running the smart contracts. They are self-executing once deployed in the network, thus a high quality smart contract design is of paramount importance in blockchain applications, preventing vulnerabilities and attacks.
The beauty of blockchain and cryptocurrency realizes transparent value transfer and asset tokenization, facilitating a new era of game development. Blockchain game is not a new concept. The first blockchain game called HunterCoin was launched in 2014, where players competed with each other and collected coins back to their campsites [3]. It turns a traditional mining concept into gaming experience, players can move around, combat with other players, as well as collecting resources. Many blockchain games are currently being developed, facilitating the integration of blockchain game and game economy.

1.2 Outline

This report first explains the concept of blockchain and its applications. Then, it highlights the current progress and justifications on framework choices. Next, overall design of the game is introduced. Finally, recommendations for future work are made to facilitate effective project delivery in the final stage.
2 Objectives

The goal of this project is to apply blockchain technology in game design, and evaluate existing layer 2 solutions to solve the latency problem of public blockchain. Moreover, security is also a main focus in blockchain game. A secure trading system prevents fraudulent transactions and exploitations. It is an essential element especially in games that involve rare item collection. The secure trading atmosphere is one of the key features to help build a sustainable game economy.

From the traditional view of public blockchain, transactions are validated via a consensus algorithm called PoW, requiring huge power consumption. In order to improve the transaction speed of peer-to-peer network, layer 2 solutions are introduced to provide a secure and scalable infrastructure on top of the root blockchain layer. The project aims to utilize suitable layer 2 solutions for game development to reduce transaction latency.

Apart from the latency issues mentioned above, the project aims to safeguard in-game transaction during the gameplay. During the gameplay, each move is recorded in a block. The smart contract helps to identify the player who makes a move before committing it to the blockchain. It is important to avoid perpetrators to take advantage of the gameplay, safeguarding the players’ rights.

Finally, a good game design caters for all audiences. Striking a balance between veteran players and new players is imperative, especially for games that involve scarce and powerful items. A sustainable game should be free to play without superfluous advantages offered to the paid players. The project aims to ensure fairness of the gameplay, regardless of nature of the players.
3 Related Works

3.1 Introduction of blockchain games

There are different types of blockchain games in the current market, focusing on different features and values. Some games feature crypto-collectibles where players collect rare items and trade with other players. Some games build traditional game using blockchain architecture to safeguard the in-game transaction, like online casino. These existing blockchain games offer insights into the design of my project.

3.1.1 CryptoKitties

![Fig. 2 The CryptoKitties marketplace by M. Greg][4]

CryptoKitties is a crypto-collectible game where people can buy virtual cats using cryptocurrencies. There are infinite combinations of cats since they are generated by non-fungible tokens. Each cat is unique and is tied to a token with specialized appearance, pattern and accessories. Players can trade with the others using cryptocurrencies and breed a new cat using two cats they owned.
3.1.2 Fishbank

Fishbank is a multiplayer fish battling game. Players can collect different fishes with various abilities. In spite of collecting fishes, they can battle with other players and hunt other fishes to growth their weights. There are also items to protect the fishes from attacks for certain period of time.

3.1.3 Total Poker

Total Poker is a mobile poker game based in blockchain architecture, featuring secure and fair gameplay. Deck operations are decentralized and cards are randomly distributed. Players can create or join a game room anonymously. The latest update of the game features free gameplay to attract more players to join the poker community.

4 Justification on Framework Choices

4.1 Overview of Root Blockchain Layer

There are two mainstream root blockchain layers in the marketplace: Bitcoin and Ethereum. Bitcoin is an electronic cash system, emphasizing on peer-to-peer anonymous value transfer. The Bitcoin network is highly secured via PoW. Another platform called Ethereum focuses on building decentralized applications via smart contracts. Its underlying security is safeguarded by PoS. Validators are chosen in a deterministic way depending on the amount of cryptocurrencies they own. A portion of cryptocurrencies, that is the stake, is taken from the validators. When the validators complete their works, they can gain transaction fees from it. If they approve a fraudulent transaction, the stake will be added to the transaction fees. It ensures true validation when the stake is higher than the transaction fees.

4.2 Root Blockchain Layer Selection

Ethereum is used as the root blockchain layer for this project. Different from Ethereum, Bitcoin network focuses on lightweight transaction like value transfer. As game transaction often contains complex structures, Bitcoin network is not suitable for transferring high volume of data which overloads the block size in the blockchain.
Instead, Ethereum can model those structures in smart contracts. It also allows customized consensus algorithms and contracts, providing greater flexibility in designing the game. Moreover, the average transaction speed of Ethereum is 13 times faster than that of Bitcoin [5]. In case there are many transactions being sent to the base blockchain layer, Ethereum has a greater performance as compared to the Bitcoin network.

4.3 Overview of Layer 2 Solutions

Layer 2 solutions are used to scale a public blockchain network. The base of the blockchain is a layer 1 protocol that relies on itself for security via proof-of-work. With large hash power generated by miners, it prevents fraudulent transactions and double-spending attacks. Layer 2 solutions sacrifice security in exchange for faster transaction throughput, and rely on layer 1 for security. Participants in the network need to periodically check relevant transactions on layer 2. If there are suspicious transactions, participants can request for auditing by giving evidences to the smart contracts residing in layer 1[6]. After processing the request, the system can then punish the cheaters in layer 2 by returning their funds back to the owner in layer 1. Therefore, layer 2 solutions allow faster transaction throughput with a mechanism to safeguard participants from exploitation. The layer 2 solutions listed below are the most popular frameworks in Ethereum.

4.3.1 Raiden Network

Raiden network is a scaling solution for Ethereum applications. It supports off-chain instant payments which is secured by blockchain smart contracts. By creating a state channel between two participants, they can transact many times until timeout is reached. The final result after subsequent transactions will be sent to the blockchain for validation, freeing up some blockchain bandwidth. It can accommodate large amount of transactions instantly with small transaction fees, facilitating the scalability of Ethereum transactions.
4.3.2 Loom Network

Loom network aims to provide a scalable infrastructure for Ethereum blockchain applications. The Loom SDK produces DAppChains which are layer 2 blockchains based in Ethereum. The Loom SDK has a large spectrum of support to various environments such as JavaScript, Phaser, Go and Unity. Different from proof-of-work in the root blockchain, pluggable consensus and smart contracts can be applied to DAppChains to optimize network scalability. It specifically supports a consensus called delegated proof-of-stake (DPoS), enabling flexible scaling of online games and social applications. The DAppChains utilizes Plasma-based relays for secure value transfer in the underlying Ethereum platform.

4.4 Layer 2 Solution Selection

Loom network is used for scaling the Ethereum network in this project. It is tailor-made for blockchain game development as it supports different game development platforms like Unity and Phaser. Also, DAppChains in Loom network is highly scalable using pluggable consensus and smart contracts, allowing for flexibility implementation. Apart from SDK support, the underlying transaction mechanism is also a determining factor. Unlike Loom network, Raiden network requires a small amount of transaction fees. It is undesirable for players to pay a certain amount of fees whenever they make a move in the gameplay. Therefore, Loom network is best suited for blockchain game development due to its cost-free transactions.

5 Methodologies

5.1 Introduction

This project follows agile software development methodology. The agile approach focuses on adaptive development, allowing for change management throughout the system development lifecycle. The following sections first introduce the game setting and mechanism, and then outline the design, implementation and testing of this project.
5.2 Game Setting
The style of the landscapes, backgrounds and characters is cartoonish. The proposed theme is CryptoFluffies Fight. Each character is unique, as there are different combinations for its body parts. The goal for the players is to collect as many characters as possible, and upgrade them for battling. The more the characters they owned, achievements will be unlocked with ability enhancement. When a player wins a match, the winner earns some money. The game is therefore a mixture of crypto-collectibles and battling elements.

5.3 Game Mechanism
The game is a turn-based game where players strategically attack the opponents or avoid the characters from being attacked. Each player chooses three characters for a battle with the opponent. Each character has a speed attribute, indicating the order of battling. The player can either attack opponent’s characters or defend opponent’s attack. The player plays a puzzle game during the battle by choosing blocks with the same color. The player defenses when choosing the two vertical consecutive blocks with the same color. The player attacks when choosing the two horizontal consecutive blocks with the same color (see orange blocks in Fig.3). The player can also deal attacks to all the opponents by choosing 2x2 blocks with the same color. If there is no matching block, the player can choose a single block, like the red block at top-right corner in Fig.3, dealing only a small amount of damage.

Fig. 3 Selected section of blocks in battle
Apart from the battling logic, there are different combinations of characters. Each character is generated using the ERC-721 smart contract. The generated ERC-721 tokens are non-fungible, each of them is tied to a unique character. Players can buy characters in the marketplace, and use in-game currency to strengthen the character’s ability.

5.4 Design

The design phase starts in early October. It primarily includes user interface design, security design and architectural design. Due to time limit constraint, some design elements and resources are found in Unity online assets store.

5.4.1 User Interface Design

The following part demonstrates the layouts of the three systems: the battle system, the inventory system and the marketplace system.

Fig. 4 The battle system
The game is designed in landscape mode because it provides more space for displaying the two camps. As shown in Fig. 4, the battle system is divided into three regions, the upper part shows the player information, including username, level and battling information. The middle part shows players’ characters, the red square indicates the arbitrary places for the characters. The lower part shows character status, for example, skills and species.

Fig. 5 shows the high level structure of an inventory system. It is divided into three parts. The lower part indicates a list of characters the player owned. Each grid shows the image of that character. The upper right part shows position and structure of the battle team. Player can drag and drop character to that area to form a battle team. The upper left part lists some character information, such as HP, ability and skills, with an image of the character.
The marketplace allows players to purchase characters used in battling. The circle represents the characters currently available in the marketplace. The purchase function is also available in this system. Once the purchase is successful, the character will appear in players’ inventory system (see Fig.5). Character information is shown when clicking a character. It includes character abilities and skills, as well as the owner of characters. Players are also allowed to sell their characters in the marketplace in exchange for in-game currency.

5.4.2 Security Design

5.4.2.1 Overview

The Loom network is secured by the underlying Ethereum network. The game logic is deployed in the DAppChains to support fast and interactive gameplay. The idea of Loom network is that attackers are disincentivized, since they cannot benefit at all by hacking in the DAppChains.
Moreover, the Unity SDK developed by Loom network provides identity management. It validates players’ identity in the gameplay. Since the game logic is running on DAppChains, the player identity is checked using digital signatures in each turn which enhances security.

Apart from security guaranteed by Loom network, smart contract design is vital to prevent fraudulent transactions. Simple methods like input field checking, are used to prevent unexpected behaviors from happening.

### 5.4.2.2 Arithmetic Overflow Avoidance

In Solidity, “uint” is a typical data type used for unsigned integers. Attackers may make use of arithmetic overflow or underflow to assign unexpected value to functions in smart contracts. For example, adding 257 to an uint8 value of 0 will result in 1, leading to an overflow problem. In order to tackle this problem, a safe math library called “SafeMath.sol” is used. When there is transaction regarding important information or data, like value and token transfer, the library is used for arithmetic calculation.
### 5.4.3 Architectural Design

The architecture of this project consists of three components, the Unity application, Loom network and Ethereum network.

![Architecture of the battling game](image)

The Unity application contains the user interface for players and interacts with the DAppChains directly in the Loom network. The Loom network has transfer gateways, connecting to the smart contracts and Ethereum network. The battling logic is implemented in smart contracts. Whenever a player makes a move in the gameplay, it is validated using smart contracts via DAppChains. The result is send back to Unity application for updating the user interface, for example, initiating an attack. Most transactions are going through DAppChains for efficiency. However, transactions such as transferring tokens back to Ethereum wallets are required for greater security. The overall gaming experience is not affected since most of the data communication of the game resides in DAppChains.

### 5.5 Implementation

The implementation phase starts in mid-October. The project is divided into 6 sub-tasks, each individual part is self-complete for easier integration. These divisions are subject to changes.
• Battling logic and in-game transaction design
• Character models, game assets and user interface design
• ERC-721 smart contract implementation
• Inventory management system implementation
• Marketplace system implementation

During the implementation phase of the project, each sub-task may be further divided into smaller sub-tasks for clarity and change management. Finally, a fully functional blockchain battling game prototype will be implemented.

5.6 Testing

The testing phase starts right after the implementation phase starts. The project is divided into several modules to simplify the testing procedures. Unit testing is crucial to avoid unexpected situation from happening. It is easier to track the problems and bugs by testing each individual module separately.

Apart from unit testing, testing of smart contracts is vital to avoid unnecessary behavior. Once the smart contracts are deployed, it is difficult to make changes and is subject to loss and failure. A tool called Truffle is used for smart contract development, testing and deployment. Truffle compiles all smart contracts and deploys to the local DAppChain.

Moreover, user acceptance test will be carried out. During the initial release of the game, it is important to know whether the user interfaces and the gameplay are acceptable for the players. There are different metrics for assessment, for example, user interface design, gameplay organization, graphics and responsiveness. After collecting some feedbacks from beta testers, improvements can be made during the final stage of the project.
6 Current Progress

6.1 Design

During the early stage of the development cycle, the layouts for battle system, inventory system and marketplace system were drawn (see Fig.4 to Fig.6). They provide a baseline for high level prototypes in the future. The system architecture of the game was also defined (see Fig.7). It provides an overview of the system, demonstrating the data communication between different components.

Moreover, the character modelling is completed with armature bones for rigging (see Fig.8). It is subject to modification in later stages. Those armature bones indicate the position of different body components being generated in Unity game engine. The character is designed to be fluffy, therefore texture paint is applied to determine the hair growth density. The red region indicates higher density whereas the blue region indicates lower density. As a result, the legs of the character do not have hair on it.

Fig. 8 Character modelling with texture paint
Apart from character modelling, some body components are completed to form different combinations of characters, like a hat (see Fig. 9). The hat is anchored to the top bone of the character. This mechanism is applied throughout different parts of the body, forming different combinations of characters.

The rendered result of the character in Unity (see Fig.10) is used in marketplace. In the marketplace, there are different characters for players to purchase. This is an important step since the player must possess some characters before starting a battle.
6.2 Implementation

Apart from the design aspect, getting familiar with the syntax and logic of the smart contracts is important. A platform called CryptoZombies provides tutorials for blockchain beginners to learn the basics of smart contracts. It also provides advanced topics on how to access, manage and structure the data in blockchain. After finishing the tutorials, coding skills of smart contracts were developed.

Currently, an ERC-721 smart contract regarding non-fungible tokens is implemented. Each token is tied to a character called CryptoFluffy, containing information about the appearance, abilities and skills of the character. It can be deployed in the local Loom network for interacting with Unity SDK. The current stage focuses on how the Unity application interacts with this smart contract, and displays some information on screen.
Apart from ERC-721 smart contract implementation, some JavaScript files are implemented for smart contract migration used in Truffle framework, a tool for smart contract deployment. These files generate the ABIs which are used for communicating with the smart contracts from Unity.

6.3 Testing

Despite learning concepts of smart contracts, some example games available on the Loom network website were tested. When testing an example game called Etherboy, an error occurred. After trials of debugging, the problem persisted. A bug report was sent to an online forum for blockchain community called DelegateCall. After several weeks, the response suggested testing the game on mobile devices. A testing on Etherboy under iOS platform was carried out and it was launched successfully. Several functions were tested in order to understand the data flow between the game and the DAppChains.

Another example game called Tiles was also tested. The testing result was satisfactory. Each player can color a tile and its information is stored as a block in the blockchain. Everyone can view the whole picture formed by colored tiles.

Moreover, the interaction between the DAppChain and Unity was tested. The data was successfully fetched from the DAppChain to Unity using a static call. In Unity, data was also successfully sent to the DAppChain via CallAsync() function.

6.4 Conclusion

Currently, there is some progress in user interface design and architectural design. ERC-721 smart contract is implemented and testing on example games is carried out. Moreover, determining framework choices is of paramount importance since they affect the generic design of the game.
7 Limitations and Difficulties

Although Loom network is used to scale the blockchain game, the transaction speed to the Ethereum network is still slow. It is limited by the nature of public blockchain network. In order to ensure a high level of security, mining is required which results in transaction latency. Therefore, the transactions to the Ethereum network are reduced as many as possible for optimization. Most of the transactions are accommodated in DAppChains for higher transaction speed.

Moreover, the Loom SDK was released on June, 2018. It is a newly launched development tool for developing applications under Loom network. There are only a few online tutorials and implementations. It is difficult to understand how to communicate with different components in Loom network at first. In order to cope with this problem, testing of smart contracts and thorough understanding of the documentations are crucial in developing this project.

On the other hand, there was a problem encountered when testing a DAppChain example with Unity and Truffle frameworks in GitHub. There are missing files and the ABIs were not successfully exported. Therefore, I try to set up the development environment from the beginning by viewing the Truffle framework documentation on my own. After the initial set up, the example is compiled successfully without any error.
8 Future Plan

In the coming months, modelling of character components will be completed in February. It focuses on forming more combinations of characters. Each character is uniquely formed, and is associated with an ERC-721 token.

After finishing character modelling, marketplace system is being implemented. Since players must possess some characters before starting a battle, marketplace system is coming major milestone. The marketplace system will be completed in mid-February.

Moreover, the inventory system will be completed in late February. It allows players to store and upgrade their characters. They can also form different teams for battling.

Finally, the battling logic will be completed in March. It includes different types of attacks, puzzle algorithms and turn-taking mechanisms. Some character motions are associated with battling logic, therefore it will be implemented after character modeling.
9 Project Schedule and Milestones

9.1 Overview

The following table shows different milestones with corresponding tasks according to the timeline. The project is divided into 4 phases. In the first phase, the project focuses on layer 2 solutions research and function testing on Loom network. In the second phase, the project focuses on the implementation of ERC-721 smart contract and character randomization. In the third phase, the main battling logic and marketplace system are the major milestones. In the final phase, the most important part is game component integration and testing, as well as preparing for final presentation and exhibition.

<table>
<thead>
<tr>
<th>Timeline</th>
<th>Milestones</th>
</tr>
</thead>
<tbody>
<tr>
<td>September – Early October</td>
<td>1. Preliminary research on blockchain game and layer 2 solutions</td>
</tr>
<tr>
<td></td>
<td>2. Design user interface prototypes</td>
</tr>
<tr>
<td>Early October – Mid-October</td>
<td>1. Test demo game from Loom network to familiarize with the architecture and data flow</td>
</tr>
<tr>
<td></td>
<td>2. Implement ERC-721 smart contract</td>
</tr>
<tr>
<td>Mid-October – Mid-November</td>
<td>1. Test ERC-721 smart contract</td>
</tr>
<tr>
<td></td>
<td>2. Refine artifacts of previous design</td>
</tr>
<tr>
<td>Mid-November – Late December</td>
<td>1. Integrate ERC-721 smart contract with assets</td>
</tr>
<tr>
<td></td>
<td>2. Design main game logic and battling turns</td>
</tr>
<tr>
<td></td>
<td>3. Design graphics and motions of characters, assets and user interfaces</td>
</tr>
<tr>
<td>Late December – Mid-January</td>
<td>1. Prepare some demonstrations for presentation</td>
</tr>
<tr>
<td></td>
<td>2. Design inventory management system for characters</td>
</tr>
<tr>
<td></td>
<td>3. Refine main game logic</td>
</tr>
<tr>
<td>Period</td>
<td>Tasks</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Mid-January – Mid-February | 1. Implement inventory management system for characters  
|                         | 2. Design marketplace system                                         |
|                         | 3. Integrate inventory management system with characters and assets   |
|                         | 4. Refine main game logic                                            |
| Mid-February – Mid-March | 1. Implement marketplace system                                     |
|                         | 2. Test battling components                                          |
| Mid-March – Mid-April   | 1. Perform final testing on all components                           |
|                         | 2. Integrate all components and streamline game flow                 |
|                         | 3. Prepare demo sections for final presentation                      |
|                         | 4. Design exhibition poster                                          |
| Mid-April – Late May    | 1. Prepare for project exhibition                                    |
|                         | 2. Prepare demo sections for final presentation                      |
10 Conclusion

By and large, game design in public blockchain suffers from latency. Various layer 2 solutions are being developed to support high scalability of the blockchain network. They also help reduce the transaction latency in the public blockchain via sidechains.

Having discussed the rationale behind framework choices, Ethereum and Loom network are used for implementing the turn-based battling game. Ethereum supports pluggable consensus and smart contracts, assisting in structuring the game. Loom network provides direct support to the game development platforms, saving time for testing and deployment. The Loom network also provides example games for developers. Testing of some example games is carried out and the corresponding problem is properly acknowledged. Moreover, preliminary design of the game is partially completed with some illustrations (see Fig. 4 to Fig. 6). The system architecture of the game is also defined (see Fig. 7). Implementation of ERC-721 smart contract and character randomization are also completed according to the project schedule.

In the coming months, a simplified smart contract regarding the main battling logic will be implemented. Moreover, some of the assets used in characters will be created for interacting with the smart contract. A simple transaction simulating a battling scenario will also be made based on the aforementioned components of the game. Together with the battling scenario, more detailed user interface prototypes will be constructed to help visualizing the entire system.
References


[4]. M. Greg, “Do you Really Own Your CryptoKitties?”. [Figure]. Available: https://medium.com/@gmcmullen/do-you-really-own-your-cryptokitties-d2731d3491a9 [Accessed Nov. 21, 2018]

