BLOCKCHAIN BASED ONLINE 3D GAME

by
Hürkan Uğur

Engineering Project Report

Yeditepe University
Faculty of Engineering
Department of Computer Engineering
2021

BLOCKCHAIN BASED ONLINE 3D GAME

APPROVED BY:

Assist. Prof. Dr. Onur Demir (Supervisor)	. . . . .
Assist. Prof. Dr. Tacha Serif	. . . . .
Assist. Prof. Dr. Esin Onbaşıoğlu	. . . . .

DATE OF APPROVAL: .../.../2021

ACKNOWLEDGEMENTS

I would like to thank my advisor Assist. Prof. Dr. Onur Demir wholeheartedly for his guidance throughout the years.

Furthermore, I am profoundly obligated to friends like Elvan Türk, Aslıhan Nur Pehlivan, Blin Qipa, Uygar Kovancı, and Ece Buse Demiray for their emotional supports and their impact on my life.

Finally, I wish to thank and express my gratitude to my family; Mühibe Uğur, Serdar Uğur, and Melis Uğur, for their moral and financial supports.

ABSTRACT

BLOCKCHAIN BASED ONLINE 3D GAME

With the invention of blockchain technology, mankind has been facing a new era that has been ruled by cryptocurrency. By using the technology behind the cryptocurrency, the security of the players can be enhanced in the game platform. This paper proposes the overall design and implementation of Hükocraft Online, blockchain based online 3D game, with the help of the Ropsten Ethereum Blockchain Test Network. Since the game is blockchain-based, no authority can manage the players, change their data or store their information. In other words, it is a decentralized game. The players are managed by themselves. Since the player and item information is stored in Smart Contracts of Ropsten Ethereum Blockchain Test Network, thanks to the transparency of the blockchain, anyone can access any information without any restrictions. In this project, Unity3D is used to create the game environment, Nethereum is used to connect Unity3D to the Ethereum Blockchain Network, and Solidity Programming Language is used to create the Ethereum Smart Contract of the game. As a result, the players are provided with a highly secure open-source, blockchain based online 3D game, where they can play with their friends together.

ÖZET

BLOKZİNCİR TABANLI 3D ÇEVRİM İÇİ OYUN

Blokzincir teknolojisinin icadıyla birlikte insanoğlu, kripto para birimi ile yönetilen yeni bir çağ ile yüz yüze geldi. Kripto para biriminin arkasındaki teknolojiyi kullanarak, oyun platformlarında, oyuncuların güvenliği yüksek derecede artırılabilir. Bu makale, Ropsten Ethereum Blockzincir Test Ağı yardımıyla, blokzincir tabanlı 3D çevrim içi bir oyun olan Hükocraft Online'ın genel tasarımını ve uygulamasını anlatmaktadır. Oyun blokzincir tabanlı olduğu için, hiçbir otorite oyuncuları yönetemez, verilerini değiştiremez veya bilgilerini saklayamaz. Başka bir deyişle bu oyun, merkezi olmayan bir oyundur. Oyuncular kendi kendilerini yönetirler. Oyuncu ve eşya bilgileri Ropsten Ethereum Blokzincir Test Ağı'nın Akıllı Sözleşmelerinde saklandığından, blokzincirin sağladığı şeffaflık sayesinde, herkes herhangi bir kısıtlama olmaksızın, herhangi bir bilgiye erişebilir. Bu projede, oyun ortamını oluşturmak için Unity3D, Unity3D'yi Ethereum Blokzincir Ağı'na bağlamak için Nethereum ve oyunun Ethereum Akıllı Sözleşmesini oluşturmak için Solidity Programlama Dili kullanılmıştır. Sonuç olarak oyunculara, arkadaşlarıyla birlikte oynayabilecekleri son derece güvenli ve açık kaynaklı, blokzincir tabanlı 3D çevrim içi oyun hazırlanmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

MMO	Massively Multiplayer Online Game
MMORPG	Massively Multiplayer Online Role-Playing Game
FPS	First-Person Shooter
FPS	Frame Per Second
UI	User Interface
UX	User Experience
CPU	Central Processing Unit
GPU	Graphics Processing Unit
IPFS	InterPlanetary File System
ETH	Ether
GWEI	Gigawei
HTTPS	Hypertext Transfer Protocol Secure
URL	Uniform Resource Locator
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
MS	Milliseconds

1. INTRODUCTION

With the invention of Blockchain [1] technology, mankind is facing a new era that is ruled by cryptocurrency. By using the technology behind the cryptocurrency, the security of the players can be enhanced in the game platform. Nowadays, most MMO [2] and MMORPG [3] are authority dependant games. In other words, they are centralized games. The user information is stored in server-based external storage. When something happens to centralized games, such as servers being caught on fire or being censored and closed by authorities, it yields a loss of players' data and efforts. For instance, Florensia, an MMORPG, was closed for one month on March 10th in 2021 [4], due to Europe's largest cloud services firm OVHcloud severs being caught on fire [5].

In the perspective of item and money transactions, in the case of unpredicted bugs occurring in-game and out-game trades, that makes people defenseless against the system. This may yield the exposure of sensitive personal data as well as hurt the reputation of the game company. Moreover, various online games do not give an open-source data search to ensure their data security.

In my project, the overall design and implementation of Hukocraft, a decentralized [6] online 3D blockchain game is proposed. There is no authority that regulates the users or stores their information with the help of the Ethereum Blockchain Network [7]; in other words, it is a decentralized game. The players are in charge of their management. Since player and item information is stored in Smart Contracts [8] of Ethereum Blockchain Network, there is no restriction to access any information, including whether an item exists in blockchain or not, if it does, whose item it is, how much experience the owner has, what rank they are, etc. As technologies, Unity3D is used to create the game environment, Nethereum [9] is used to connect Unity3D [10] to the blockchain, and Solidity Programming Language is used to create the Ethereum Smart Contract of the game. Thanks to the transparency of the blockchain, the players are provided an open-source, highly secure, online, and interactive 3D blockchain game.

1.1. Motivation

1.1.1. Decentralization

Thanks to the advanced characteristics and assistance of the blockchain, no authority manages the players, changes their data or stores their information. In other words, a decentralized game is provided.

1.1.2. Peer to Peer Interaction

The players are managed by themselves. Since the player and item information is stored in Smart Contracts of Ropsten Ethereum Blockchain Test Network [11], thanks to the transparency of the blockchain, anyone can access any information without any restrictions. Furthermore, players are able to make transactions among the other players without requiring any third party.

1.1.3. Secure Data

Since the data stored in the blockchain can't be modified and interfered with by anyone. As a result, the players are provided with a highly secure open-source online 3D blockchain game where they can play with their friends together.

1.2. Problem Definition

1.2.1. Centralization

Nowadays, the majority of online MMO and MMORPG are authority-dependent. They are, in other words, centralized games. The owners of the game can modify and censor any data of the players. Furthermore, the user data is stored and protected in server-based external storage such as databases.

1.2.2. The Concern of Data Safety

When something wrong happens to centralized services, such as physical damage to the servers or the application being blocked or terminated by authorities, it results in the loss of players' data and their efforts.

1.2.3. Data Access Restrictions

Most of the game companies do not provide fully transparent data access to ensure the data security of the games. Therefore, the players are not allowed to see what happens on the background.

1.2.4. Limitations in Data Storage and Data Search

When a search operation is a matter in the database such as item search, and user information search, the entire database is scanned one by one to find the sought information. This process might cause a serious slowness when the number of users reacher at millions in the database. Moreover, on the centralized MMORPG, there is a limitation in the character levels in general. The leveling up process of the characters is stopped after a while. In other words, it is not limitless.

1.3. Aims

1.3.1. Decentralization

Thanks to the advanced characteristics and assistance of the blockchain, there shall be no authority that can govern, modify or maintain the information of the players such as, the total experience and level of the players, the items that the players have, the records of the existing and destroyed items in the game, and so on.

1.3.2. Fully Secure Data Environment

As a matter of fact that since players' information and items' information are mined in Smart Contracts of Ethereum Blockchain Network, no matter what happens, the data shall remain in the blockchain forever. Therefore, the blockchain provides the players a relief about data safety.

1.3.3. Open-Source Data Access

Since the smart contract of the game is open source, anyone can access any information and examine how the smart contract works in the blockchain. By doing so, the players are shown the transparency and the safety of the online 3D blockchain game.

1.3.4. Data Operation Advantages of the Ethereum Blockchain

The map data structure of the Ethereum blockchain has a special property that enables any type of information can be accessed with O(1) complexity. This data structure grants the blockchain game an extreme speed in data access operations compared to centralized applications. Thanks to this feature of the blockchain, the slowness problem of the material search process can be solved. Furthermore, since the blockchain has no storage problem, the players can level up almost limitless. In other words, almost a never-ending game is provided to the players.

1.4. Requirements

Windows 7 (SP1+), Windows 10, and Windows 11, 64-bit versions only. X64 architecture with SSE2 instruction set support, DX10, DX11, and DX12-capable GPUs.

Ubuntu 20.04, Ubuntu 18.04, and CentOS 7, X64 architecture with SSE2 instruction set support, OpenGL 3.2+ or Vulkan-capable, Nvidia and AMD GPUs [12].

1.4.1. Functional Requirements

Launching the Game

When the icon of the game is clicked on, the user encounters a menu which has contains several buttons such as, Create A New Account Button, Store Existing Account Button, Ethereum Login Menu Button, Get Daily Ether Button, Show My Accounts Button, and Exit Button. It is required for the user to have an Ethereum Ropsten Test Network Wallet to be able to log in to the game.

Login with Ethereum Blockchain Wallet

Creating a New Ethereum Account: When the user is a newcomer without an Ethereum Ropsten Test Network Wallet, they must create a new wallet by clicking on Create New Account Button and then Generate An Ethereum Wallet Button. After the new wallet is created, they can go back to the main menu.

Saving an Existing Ethereum Account: When the user is a newcomer who has already an existing Ethereum Ropsten Test Network Wallet, they can choose Store Existing Account Button and save their Ethereum Wallet in the localhost. Furthermore, it is enabled to examine all existing personal Ethereum account via Show My Accounts Button, which calls a text file named Hukocraft.txt where the account addresses are listed.

Claiming Daily Ether: When the user has an Ethereum Ropsten Test Network Wallet, since all blockchain smart contract transactions consume some amount of gas, the user must have some amount of Ether in their wallet before they log in. Therefore they should claim some Ether via Daily Ether Button.

Connecting to the Game: Having completed all these processes, the user is finally ready for login with their Ethereum Wallet via Ethereum Login Menu Button. Having selected the electronic wallet they want to log in with, Login Button must be clicked. At this stage, Game Login Screen is shown. The player is asked to enter their name on the condition that the name can have a maximum of 8 characters in length. Finally, the player is ready to press the Login Button and connect to the game server.

Game Mechanics

Player Movement: The player can move with W, A, S, D, Space, and Arrow keys.

Camera Angle: The player can zoom the camera to play in First-Person Shooter (FPS) by pressing on the key Z.

Shooting The Enemies: The player can perform shooting by pressing on the Left Mouse Button.

Communication with Others: The player can communicate with other players by using the in-game chat section. Pressing on the Enter key activates the Text Field of the game chat.

Wallet Details: The player is provided a Details Button, so that they can go to their Ropsten Blockchain Network and observe the blocks and the transaction status.

Ether Transfer: There is an Ether Transfer section where the players can transfer Ether to any other online players.

Observing Transaction Status: Since the game offers the players maximum transparency, any blockchain transaction process is shown via Ethereum Notification Chat, which is placed right below the screen. In this notification chat, every blockchain transaction takes place such as level experience transactions, Ether transfer transactions, character initialization transactions, item mining transactions, item removing transactions, in-game item trade transactions, and so on.

Item Transactions: The users are provided in-game item exchange by typing the "/trade Username" command on the game chat. Whenever the person who is sent the trade request accepts the request, a Trade Menu UI is shown to the user.

Collecting Items: The in-game items pop up in a variety of fields where it is called item spawners. When the user collides with the item of the corresponding item spawner, the item is stored in the player's inventory if it is validated by the blockchain.

Using Tokens: When an item is acquired by the player, it can be used via pressing the slot number where the item is stored in the inventory.

Game Music and FPS Settings: The user is enabled to mute the game music and change the Frame Per Second (FPS) in-game which highly affects GPU performance.

Interaction with the Smart Contract of the Ethereum Blockchain: There are some fields where the users can check the game ranking, query, and search game tokens as well as player inventories, which are stored in the blockchain.

Exit the Game: The players can exit the game by jumping in the Exit Game field.

1.4.2. Non-Functional Requirements

Availability: As long as the game server is online, the players must be able to log in to the game whenever they wish to. Furthermore, the user must be able to access any item and user information in the blockchain, whenever they want.

Reliability: The game shall minimize and handle the failures, which might occur on runtime.

Usability: The UI (User Interface) must be straightforward and user-friendly. Furthermore, the users are given a How To Play text file to explain to them how the game mechanics work.

Performance: Since it is an online game, the server must be able to handle multiple users at the same time. The users must be provided Frame Per Second (FPS) calibration options to optimize the GPU usage and provide the optimal game experience. The game camera must not render the monsters, item spawners, and skill effects, which are far from the user to optimize the CPU usage.

Security: The user and item information must be stored in the blockchain rather than in centralized external storage such as databases.

2. BACKGROUND

Blockchain is a distributed ledger system, which provides tamper resistance and traceability at the very heart of Bitcoin [13] technology. Ethereum, also known as a blockchain 2.0 platform, adds support for smart contracts, in other words, the programs that can be stored and executed on a blockchain system. Developers can use smart contracts to build a variety of decentralized applications, primarily games. Blockchain games are considered to have the advantage of being differentiated from existing online games in the point of game data and logic being stored and executed transparently on the blockchain. These benefits are especially true for games with in-game payouts and randomness mechanisms, such as gambling, which often suffer from trust issues in existing online game environments. In conclusion, most blockchain gaming projects are about the ownership, production, and trading of digital assets.

2.1. Previous works

An article established by Xin-Jian Jiang and Xiao Fan Liu [14] is about the analysis of CryptoKitties [15], as known as the first blockchain game, which was released on Ethereum in November 2017. The image can be seen in Figure 2.1.

Figure 2.1. CryptoKitties – The First Blockchain Game

In CryptoKitties, the players are capable of owning, creating, and trading crypto kitties with the help of ERC-721 [16] non-fungible tokens. All the transaction information such as kitty birth, buying from the SalesAuction contract, transferring through the core contract, trading through the offers contract, are stored in the Ethereum blockchain. The contracts can be seen in Table 2.1.

Transaction type	From	To	Amount
Kitty birth	0x	Owner	1,923,901
Listing on the SalesAuction contract	Seller	SalesAuction	1,126,964
Cancel listing on the SalesAuction contract	SalesAuction	Seller	241,614
Buying from the SalesAuction contract	SalesAuction	Buyer	668,981
Transferring through the core contract	Sender	Reviver	633,208
Trading through the offers contract	Seller	Buyer	2,336
Listing on the SiringAuction contract	Lender	SiringAuction	326,553
Cancel listing on the SiringAuction contract	SiringAuction	Lender	92,507
Rental finished	SiringAuction	Lender	157,457

Table 2.1. Nine Types of Transactions Related to the Movements of Kitties.

When it is needed to change the ownership of the crypto kitties, the players are required to use SalesAuction contract, Offers contract, or Core contract. The usage rate of the contracts for the ownership modification over time is shown in Figure 2.2. In the first years of the game, the SalesAuction contract had the highest rate of use. On the other hand, the Core contract took SalesAuction's place over time.

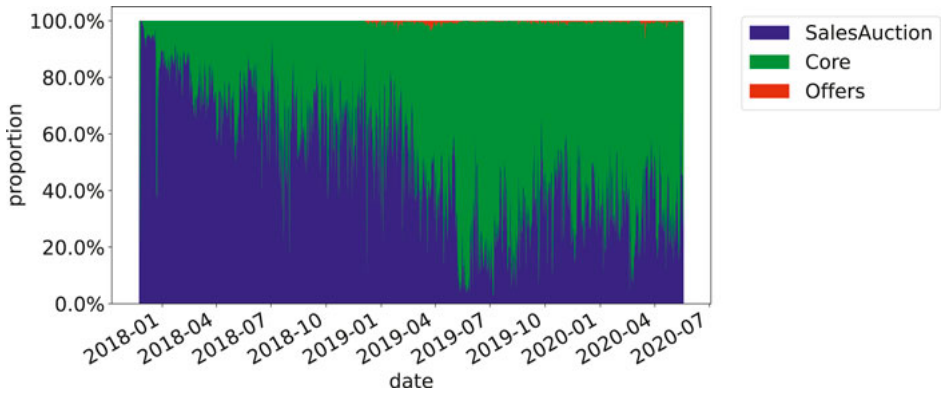

Figure 2.2. The Change in Proportion of Three Methods to Transfer the Kitties.

The cost of the transactions has a special role in this change. The players were in favor of transferring crypto kitties at a lower cost. By using SalesAuction and Offers contracts to buy or sell crypto kitties, players are required to pay a transaction fee, 3.75% of the transaction amount, to the game publisher. On the other hand, to use the transfer method in the Core contract, only the gas fee is paid.

As a result, they found that the thing that made crypto kitties so popular was the idea of crypto kitties being sold at an extremely high price. The reasons why Cryptokitties lost their popularity are; the loss of profits from trading in-game items due to the rapidly changing value of cryptocurrencies, the growing gap between the distribution of wealth among players, and the limitations of the blockchain.

A study, carried out by Yunifa Miftachul Arif, Muhammad Naufal Firdaus, and Hani Nurhayati [17] are studied for a scoring system for the multiplayer game, Widow Waterfall, based on blockchain technology. The game UI (User Interface) is shown in Figure 2.3.

Figure 2.3. Widow Waterfall – Ethereum-Based Multiplayer Game

They created an Ethereum blockchain network-based scoring system for an online multiplayer game which is designed by using Unity Game Engine. They used Visual Studio to create Ethereum blockchain transactions and used MetaMask [18] to connect to the digital wallet. The structure of the system can be seen in Figure 2.4. They allowed the players to access the users' scores via the score table which can be accessed by any device connected to the network.

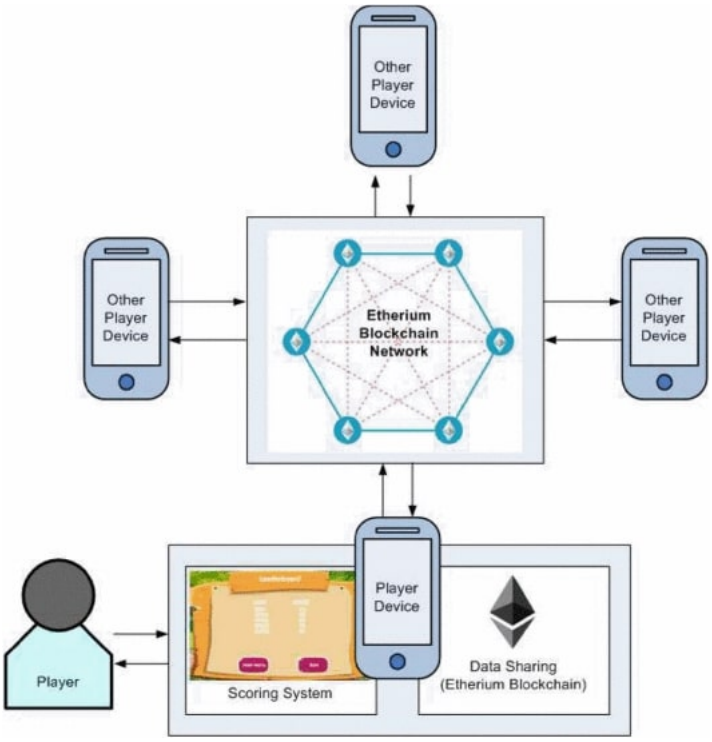

Figure 2.4. Blockchain-Based Data Sharing For Scoring Among Players

When a user is on the main menu screen, the user is required to connect the device to the Ethereum Blockchain Network in the Blockchain Connection stage. Moreover, the user can also retrieve the balance of the wallet address. Having connected to the game, whenever the player acquires score points, the score information is transmitted to the Ethereum blockchain network through blockchain transactions. The game stages can be seen in Figure 2.5.

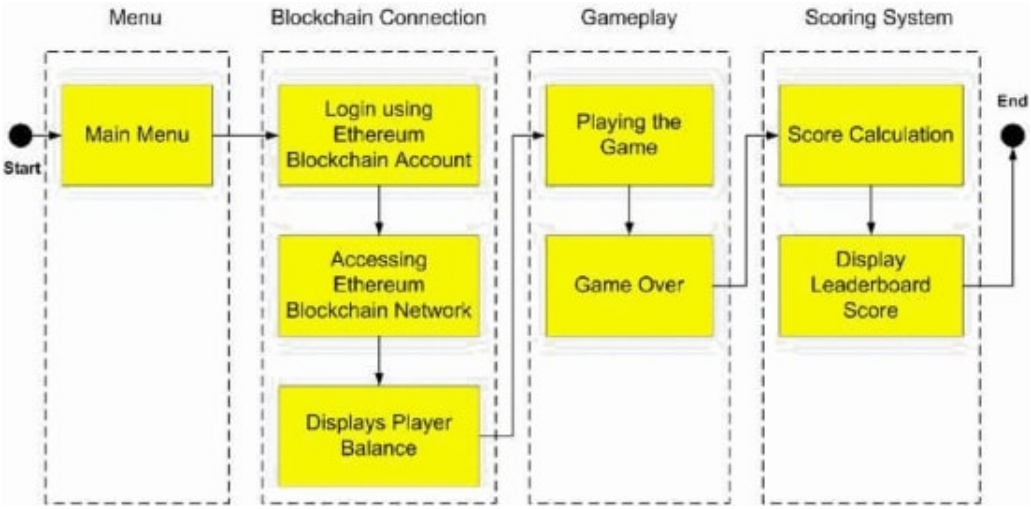

Figure 2.5. The Flow of Widow Waterfall

They connected the game to the smart contracts by using ABI and bytecode, which are acquired from the compilation result of the smart contract, which is written in Solidity Programming Language.

When it comes to the transaction speed and the transaction cost, they modified the gas limit value, which was used in advance. Then, they calculated the average time required for the transaction validation by performing these steps several times, and then they used the (Equation 2.1) to calculate the average transaction speed. They used the gas limit values 500000, 600000, 700000, 800000, 900000, and 1000000 to examine the speed of the transactions.

The average transaction speed can be calculated as:

$$T = \frac{1}{n} \sum_{i}^{n} T_i \tag{2.1}$$

where n is the number of the transactions, T_i is the latency of the i^{th} transaction, and T represents the average speed of the transactions.

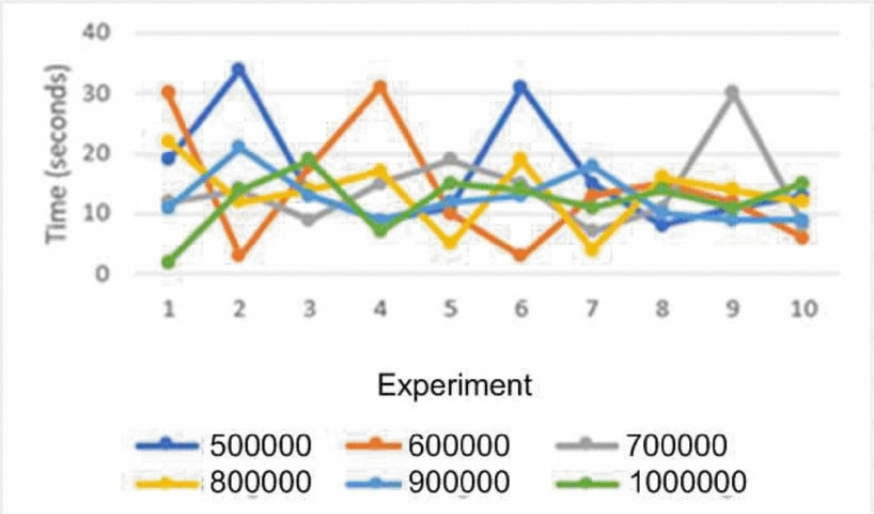

Figure 2.6. Transaction Speed Test

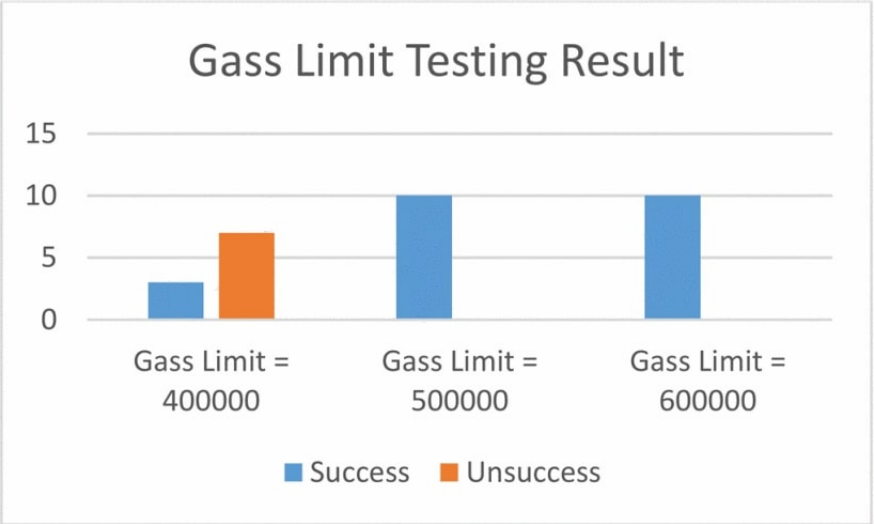

Figure 2.7. Gas Limit Test

In conclusion, they examined the more gas limit value they used, the faster transaction validations as you can in Figure 2.6. Furthermore, when they used the gas limit lower than the optimal gas limit, which is 500000, they observed that some transactions could not have been validated, as is shown in Figure 2.7.

Another work, An Architecture for Game to Game Data Transfer Using Blockchain [19] was carried out by Chanon Yaklai and Vishnu Kotrajaras. They presented a game model, which allows players to receive rewards for their efforts in the game. The outcome of these efforts can be transferred to the other games, thanks to the features of ERC20 [20] tokens. These ERC20 tokens can be redeemed in the other games because they share the same Ethereum Blockchain Network.

In the paper, Blockchain for Gaming, Applications of Blockchain Technology in Business: Challenges and Opportunities [21], conducted by Mohsen Attaran, and Angappa Gunasekaran, the authors approached the cons and pros of online blockchain games. Currently, game corporations typically keep in-game data and items on centralized servers to guard them against being duplicated. However, blockchain technology permits complete transparency and decentralized manipulation of digital property, by allowing players to have complete possession of their digital properties. Additionally, the blockchain allows the players to earn tokens for playing, reviewing, or sharing online games on social media platforms. Since the statistics are publicly available, it encourages the players to act accordingly. Furthermore, blockchain improves protection against fraud, as well as presents transparent and safe transactions by using Smart contracts.

In conclusion, they discovered that there were three main limitations in Blockchain gaming. The first was the scalability problems, which affect the progress of online blockchain games dramatically. The second was the sustainability of the game, which is linked to the loss of interest of the players in blockchain online games, decentralized properties, or item exchanges. The third was the domination of the centralized games over the decentralized games, as known as blockchain-based games.

3. ANALYSIS AND DESIGN

In my research, a 3D online blockchain game is proposed where the user and in-game data are stored in Ethereum Blockchain Network. In this section, the problems which are mentioned in Problem Definition section are discussed and a design pattern is provided about the structure of Hükocraft 3D Online Blockchain Game.

3.1. Login with Ropsten Ethereum Blockchain Wallet

The flow of the login process is demonstrated in Figure 3.1. The login stages shall be elaborated section by section.

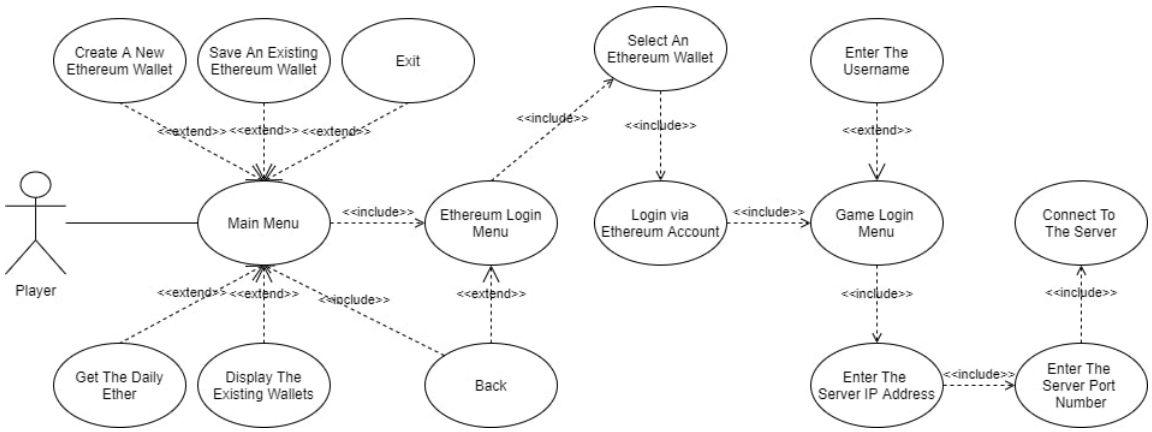

Figure 3.1. The Use-Case Diagram shows the flow of the game login stages.

3.1.1. A User without an Ethereum Blockchain Account

In case of the user does not have a Ropsten Ethereum Wallet, they are provided to create one through the Create A New Ethereum Wallet button.

3.1.2. A User with an Ethereum Blockchain Account

In case of the user already has a Ropsten Ethereum Wallet, they must click on the Save An Existing Ethereum Wallet button to save their Ethereum Wallet in their localhost. Furthermore, they are provided to review their all existing Ethereum accounts by clicking on the Show My Accounts button. This is going to lead the Hukocraft.txt file pop up, where the users can access their account information.

3.1.3. Demand Your Daily Ether

Since all Ethereum Blockchain Smart Contracts require some amount of gas to execute their functions, the users are expected to have some amount of Ether in their wallets before they connect to the game. Because of the fact that the user registration process also requires some amount of ether, the user with 0 Ether in their wallet cannot be stored on the blockchain. To put it another way, they cannot be mined on the blockchain. Thus, before they log in, they must claim some Ether via the Daily Ether button. The user shall be directed to Ropsten Ethereum Faucet [22].

3.1.4. Login to the Hükocraft Game Server

After the user has completed all these steps, they are ready to connect to the Hükocraft game server through the Ethereum Login Menu button. Firstly, they must choose the Ethereum account they wish to log in with, and then they should click on the Login button which directs the user to Game Login Menu. The player is expected to enter their name. The given name can have a maximum of 8 characters in length. After the player enters the Server IP address and Server Port number, they are ready to connect to the Game by pressing the Connect button. The flow of the login process has been shown in Figure 3.1 in advance.

3.2. Gameplay

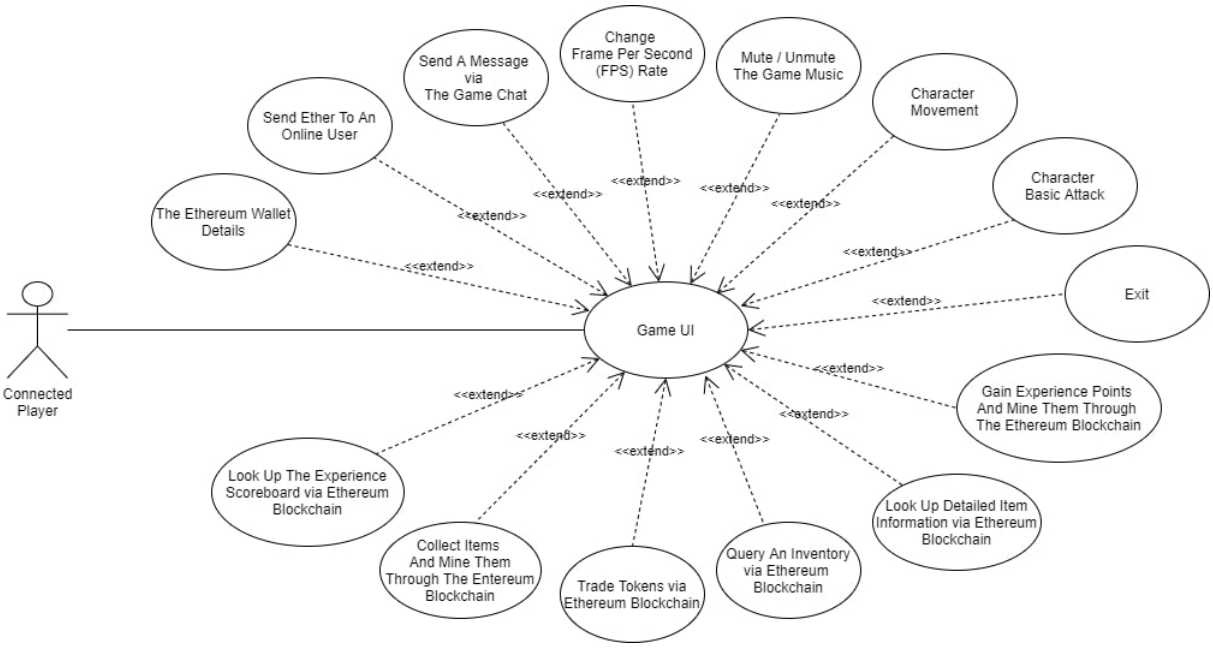

Figure 3.2. The Use-Case Diagram, which represents the features of the Hükocraft Blockchain Game.

Having the game is started, there are several actions, which can be performed by the players (Figure 3.2). The player can move with W, A, S, D, Space, and Arrow keys. The player can zoom the camera to play in First-Person Shooter (FPS) by pressing on Z key and performing shooting by pressing on the Left Mouse Button. The player can communicate with other players by using the in-game chat section. Pressing on the Enter key activates the Text Field of the game chat. Furthermore, the players are provided a Details button, which allows them to take a deep look into their Ethereum Blockchain transactions. Moreover, there is an Ethereum Transfer section where the players can transfer Ether to other online players. Since the game offers the players maximum transparency, any blockchain transaction process is shown via Ethereum Notification Chat, which is placed right below the screen. In this notification chat, every blockchain transaction takes place such as level experience transactions, Ether transfer transactions, character initialization transactions, item mining transactions, item removing transactions, in-game item trade transactions, etc. The users are provided in-game item exchange by typing the "/trade Username" command on the game chat. When an item is acquired by a player, it can be used by pressing on the inventory slot number, where the item is kept. The user is enabled to mute the game music and change the Frame Per Second (FPS) in-game, which highly affects GPU performance. There are some fields where the users can check the ranking, query, and search an item, and also they check someone's inventory which is mined in the blockchain. Finally, they can exit the game by jumping in the Exit Game field.

3.3. The Communication Between the Players and the Game Server

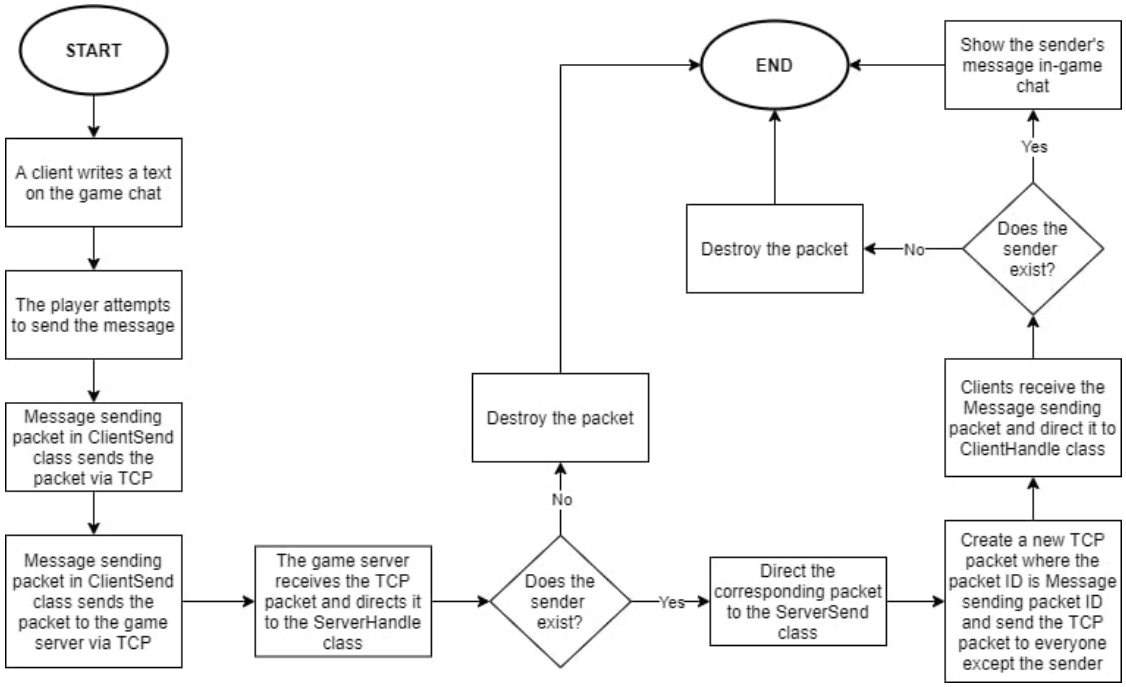

Figure 3.3. Flowchart of the packet transfer between the players and the server

Since it is necessary to connect players and interact with them, a game server is created to provide this feature. However, the server only transmits instant actions such as in-game character movements, activated skill information, the current location of the in-game monsters, the state of players and monsters, etc. In other words, the game server does not store the game content in third-party external storage such as a database and the server does not know anything about player information such as the total experience, player properties, and blockchain information of the players. While transmitting instant information, the game server uses both TCP and UDP packets. When the data to be transmitted is essential to be delivered such as skill and attack instantiations, sending a game chat message, spawning items in-game fields, spawning new monsters and players, TCP packets are used as they guarantee precise transmission. On the other hand, when the transmission speed is more important than the definite transmission of packages such as the position and rotation of characters, monsters, skills and shots, and the spin of spawned items, UDP packets are used due to their fast transmission speed. The flowchart of the packet transfer pattern can be seen in Figure 3.3.

3.4. The Infinite Level and Experience Calculation

With the help of the data type uint256 of Ethereum Blockchain, almost a never-ending game is provided to the players. Because, the maximum amount of level experience, which can be stored in uint256 is $2^{256}-1$, which is 1.1579209 x 10^{77} (Figure 3.4). Thus, this amount of experience points can be considered infinite for a game platform.

```
struct Player
{
    string username;
    uint256 experience;
    Item[INV_LENGTH] inventory;
}
```

Figure 3.4. Smart Contract representation of game experience by using uint256

The smart contract stores only the total amount of experience. Since the level calculation process contains loop operations, and the gas cost for the loops is higher than regular functions, this process is conducted by the client-side. Once the player logins the game, the total experience of the player is extracted from the Smart Contract by using HukocraftShowCharacterExperience method (Algorithm 1).

Algorithm 1 ShowCharacterExperience Query Request Algorithm **(Unity Side)**

Ensure: Your character is registered on the Hükocraft Smart Contract

Require: Ethereum Wallet Address

$queryHandler \leftarrow$ Get $ShowExperienceQueryHandler$ in the Smart Contract

$funcParameters \leftarrow$ Create a $ShowExperience$ instance with $WalletAddress$ params

$transaction \leftarrow$ Send the query and wait for the blockchain response

$experience \leftarrow$ Extract the experience info in $transaction$

return $experience$

After the total experience of the corresponding player is acquired by the HukocraftShowExperience method, ExperienceAndLevelCalculation method is called to calculate the player level, required experience to level up and current level experience (Algorithm 2).

Algorithm 2 Experience and Level Calculation Algorithm **(Unity Side)**

Require: Total amount of experience of the character "$characterTotalExp$"

$prevLevelRequiredExp \leftarrow 0$

$nextLevelRequiredExp \leftarrow 5$

$level \leftarrow 1$

while nextLevelRequiredExp $\leq$ characterTotalExp **do**

$prevLevelRequiredExp \leftarrow nextLevelRequiredExp$

$level \leftarrow level + 1$

$nextLevelRequiredExp \leftarrow nextLevelRequiredExp$ + (5 * $level$)

end while

return $level$

The level arrangement of Hükocraft can be seen in Table 3.1.

Level	Lv1	Lv2	Lv3	Lv4	Lv5	Lv6	Lv7	Lv8	Lv9	Lv10	Lv100	Lv1000	Lv10000	∞
Required exp for the previous level	0	5	15	30	50	75	105	140	180	225	24750	2497500	249975000	∞
Required exp for the next level	5	15	30	50	75	105	140	180	225	275	25250	2502500	250025000	Limit of uint256
	+10	+15	+20	+25	+30	+35	+40	+45	+50	.....				

Table 3.1. Level arrangement of Hükocraft

3.5. The Scoreboard with the Smart Contract

The players are provided to check their level and experience ranking by using in-game Scoreboard fields. Whenever the players get closer to those fields, the Scoreboard interface is going to be triggered and activated. After the Scoreboard game object is activated and the related smart contract method, HukocraftShowRanking (Algorithm 3) is called to retrieve the ranking information from the smart contract on the blockchain (Algorithm 4).

Algorithm 3 ShowRanking Query Request Algorithm **(Unity Side)**

Require: Ethereum Wallet Address

$queryHandler \leftarrow$ Get $ShowRankingQueryHandler$ in the Smart Contract
$funcParameters \leftarrow$ Create a $ShowRanking$ instance with $WalletAddress$ params
$transaction \leftarrow$ Send the query and wait for the blockchain response
$rankingList \leftarrow$ Extract the ranking list in the $transaction$ result
return Sort $rankingList$ by descending order

Algorithm 4 ShowRanking Algorithm in the Smart Contract **(Ethereum Blockchain Side)**

$names \leftarrow []$
$experiences \leftarrow []$
for $i \leftarrow 0$ to $playerAddressesLength$ **do**
  $names[i] \leftarrow$ the username of the i^{th} player
  $experiences[i] \leftarrow$ the total experience of the i^{th} player
end for
return $(names, experiences)$

3.6. Token Management with the Smart Contract

3.6.1. Token Collection Operation

The players can collect in-game tokens such as bombs and shields, which are spawned on the Item Spawners. When an item on an item spawner is taken, the game server sends a TCP packet to all the users to inform them about the item of the corresponding item spawner is taken. Then, HukocraftAddInventoryAddItem method (Algorithm 5) is called in Unity to send a mining request to the Hükocraft Smart Contract on the Ethereum Blockchain (Algorithm 6).

Algorithm 5 InventoryAddItem Transaction Request Algorithm **(Unity Side)**

Ensure: You need to have enough amount of Ether to fulfill the transaction request

Require: Item Code, Item Index in the Inventory, and Ethereum Wallet Address

$queryHandler \leftarrow$ Get $AddItemTransactionHandler$ in the Smart Contract

$funcParameters \leftarrow$ Create an $AddItem$ instance with $WalletAddress$, $itemCode$, and $itemIndex$ parameters

$transaction \leftarrow$ Send the transaction request and wait for the blockchain response

return The success status of $transaction$

Algorithm 6 AddItem Algorithm in the Smart Contract **(Ethereum Blockchain Side)**

Ensure: The sender has enough amount of Ether to fulfill the transaction request

Require: Item Code, Item Index in the Inventory, and Ethereum Wallet

if $(\neg(0 \leq itemIndex \leq 9)) \vee (characterItemIndex \neq EMPTY)$ **then**

return $TransactionRevertedInfo$

end if

$currentPlayer[myAddress].Inventory[itemIndex].isSlotFull \leftarrow true$

$currentPlayer[myAddress].Inventory[itemIndex].itemType \leftarrow itemCode$

$currentPlayer[myAddress].Inventory[itemIndex].itemID \leftarrow itemIDCounter$

$currentPlayer[myAddress].Inventory[itemIndex].ownerName \leftarrow myUsername$

$currentPlayer[myAddress].Inventory[itemIndex].ownerAddress \leftarrow myAddress$

3.6.2. Token Consumption Operation

When the player wants to use a token, a token delete request is sent by using Hukocraft-InventoryRemoveItem (Algorithm 7) method in Unity. The function requires the inventory slot index which contains the item the player wishes to use.

Algorithm 7 InventoryRemoveItem Transaction Request Algorithm **(Unity Side)**

Ensure: You need to have enough amount of Ether to fulfill the transaction request

Require: Item Index in the Inventory, and Ethereum Wallet Address

$queryHandler \leftarrow$ Get $RemoveItemTransactionHandler$ in the Smart Contract

$funcParameters \leftarrow$ Create an $RemoveItem$ instance with $WalletAddress$, and $itemIndex$ parameters

$transaction \leftarrow$ Send the transaction request and wait for the blockchain response

return The success status of $transaction$

After the delete transaction is fulfilled by the method InventoryRemoveItem (Algorithm 8) on the blockchain, the player is allowed to use the item in their inventory.

Algorithm 8 RemoveItem Algorithm in the Smart Contract **(Ethereum Blockchain Side)**

Ensure: The sender has enough amount of Ether to fulfill the transaction request

Require: Item Index in the Inventory, and Ethereum Wallet

if $(\neg(0 \leq itemIndex \leq 9)) \vee (characterItemIndex \neq FULL)$ **then**

return $TransactionRevertedInfo$

end if

$currentPlayer[myAddress].Inventory[itemIndex].isSlotFull \leftarrow false$

3.7. Query Operations with the Smart Contract

3.7.1. Inventory Query Operation

There are some fields in the corners of each world. When the players get closer to those fields. One of those fields is the ObserveInventory field which allows the players to look up the inventories of other players. When the ObserveInventory menu is activated, the players are required to select a wallet address whose inventory wants to be observed. Then, the HukocraftObserveInventory method is executed in Unity (Algorithm 9). This method requires a wallet address, whose inventory is going to be retrieved, as a parameter.

Algorithm 9 ObserveInventory Query Request Algorithm **(Unity Side)**

Require: Ethereum Wallet Address, and Target Wallet Address

$queryHandler \leftarrow$ Get $ObserveInventoryQueryHandler$ in the Smart Contract

$funcParameters \leftarrow$ Create a $ObserveInventory$ instance with $WalletAddress$ and $ObserveAddress$ parameters

$transaction \leftarrow$ Send the query and wait for the blockchain response

$inventorySlotList \leftarrow$ Extract the inventory slot list in the $transaction$ result

return $inventorySlotList$

Thanks to the Map data structure of Solidity (Figure 3.5), the corresponding player's inventory is accessed with O(1) complexity.

```
mapping (address => Player) private currentPlayer;
mapping (address => uint256) private playerIndex;
mapping (uint256 => Item) private currentItem;
address[] private playerAddresses;
```

Figure 3.5. The map data structure, which allows accessing the data with O(1) complexity on Hükocraft Smart Contract on the Blockchain side

Then, the corresponding inventory info is gathered on the Blockchain side by using the ObserveInventory method (Algorithm 10).

Algorithm 10 ObserveInventory in the Smart Contract **(Ethereum Blockchain Side)**

Require: Target Ethereum Wallet

$inventorySlots \leftarrow []$
for $i \leftarrow 0$ to $inventoryLength$ **do**
  if $currentPlayer[targetAddress].inventory[i].isSlotFull = TRUE$ **then**
    $inventorySlots[i] \leftarrow currentPlayer[targetAddress].inventory[i].itemType$
  end if
end for
return $inventorySlots$

3.7.2. Item Query Operation

The other field is called SearchItemBoard. The field allows the players to query the game tokens by item id. When the players trigger the field, the players are required to enter the item id. Then, the HukocraftFindItem method is executed (Algorithm 11). The method requires the item id to be queried.

Algorithm 11 FindItem Query Request Algorithm **(Unity Side)**

Require: Ethereum Wallet Address, and Item ID

$queryHandler \leftarrow$ Get $FindItemQueryHandler$ in the Smart Contract
$funcParameters \leftarrow$ Create a $FindItem$ instance with $WalletAddress$ and $itemID$ parameters
$transaction \leftarrow$ Send the query and wait for the blockchain response
$item \leftarrow$ Extract the Item Object in the $transaction$ result
return $item$

Thanks to the Map data structure, the corresponding item is accessed with O(1) complexity on the blockchain side, and it returns the corresponding item object immediately.

3.8. Token and Ether Transactions with The Smart Contract

Hükocraft allows the players to trade their tokens in exchange for other tokens and/or Ether. Initially, a player should send a trade request via the game chat by typing the trade command "/trade USERNAME". Analyzing the trade command is handled in GameChat class. As a result of the analysis of the text, the TCP packet for the trade operation is prepared and sent to the second player through the game server. When the packet arrives at the second player, a TradeRequest menu pops out. The trade request is accepted or denied by the second player. This operation is conducted in TradeRequestMenu class. The result of the second player's action is delivered to the first player by using the corresponding TCP packet. If the received response is "Accepted", the transaction session begins. The transaction session is held by TradeMenu class. Both users can put a token and enter an amount of Ether into the transaction field. When the inputs are given, both users are excepted to click on the Lock button. This button ensures that none of the given inputs can be changed. In other words, it seals the given inputs and makes them immutable. Having done this process, the users are excepted to press on Confirm button. As soon as both users click on the Confirm button, the transaction process begins and is conducted on the Ethereum blockchain.

The Unity side class interactions in Trade process is shown in Figure 3.6.

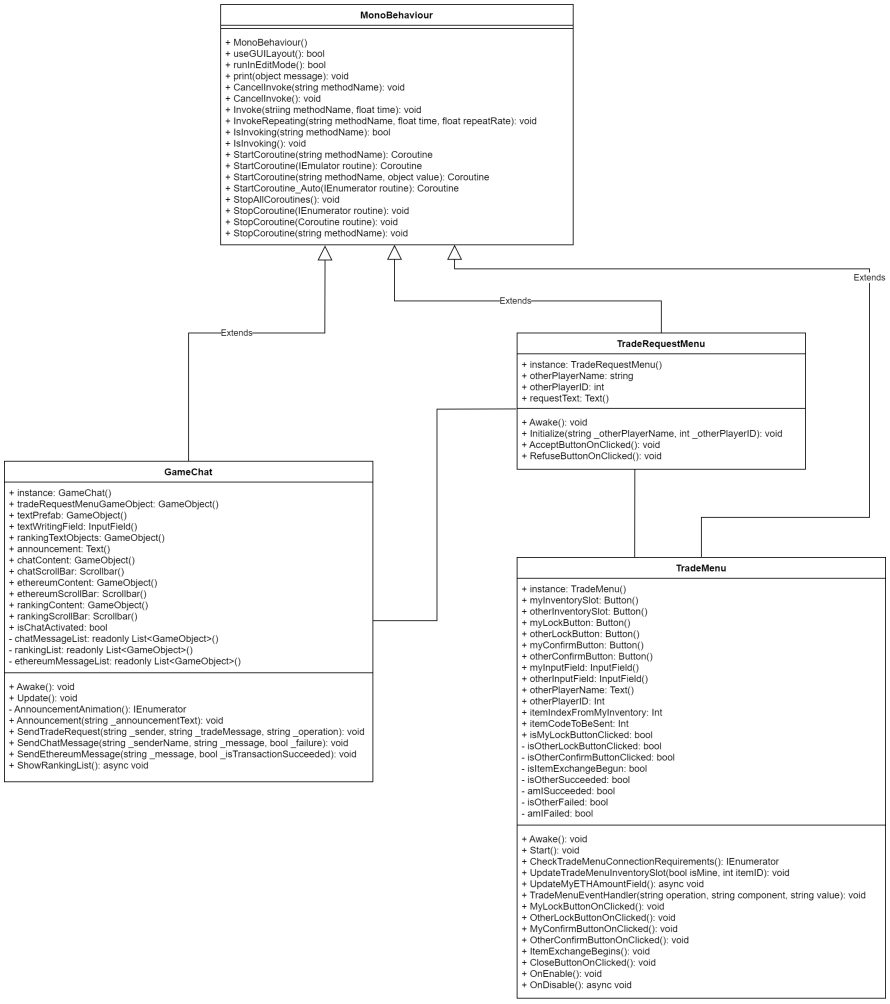

Figure 3.6. The diagram shows the class interactions in the trade operations.

The transaction process is delivered to the blockchain by using the HukocraftItemExchange method on the Unity side (Algorithm 12).Then, the ItemExchange method on the blockchain side fulfills the transaction process (Algorithm 13).

Algorithm 12 HukocraftItemExchange Transaction Request Algorithm **(Unity Side)**

Ensure: You need to have enough amount of Ether to fulfill the transaction request

Require: MyWallet, TargetWallet, ReceiveEther, MyItemIndex, and MyItemCode

$queryHandler \leftarrow$ Get $ItemExchangeTransactionHandler$ in the Smart Contract

$funcParameters \leftarrow$ Create an $ItemExchange$ instance with $MyWalletAddress$, $TargetWalletAddress$, $ReceiveEther$, $MyItemIndex$, and $MyItemCode$ prms

$transaction \leftarrow$ Send the transaction request and wait for the blockchain response

return The success status of $transaction$

Algorithm 13 ItemExchange Algorithm in the Smart Contract **(Ethereum Blockchain Side)**

Ensure: The sender has enough amount of Ether to fulfill the transaction request

Require: MyWallet, TargetWallet, ReceiveEther, MyItemIndex, and MyItemCode

if $TargetWallet.balance \not> ReceiveEther$ **then**

return $Revert\ the\ transaction$ "$Target\ not\ have\ enough\ ether$"

end if

if I send an item to the other player **then**

$isSucceeded \leftarrow FALSE$

for $i \leftarrow 0$ to $inventoryLength$ **do**

if The i^{th} inventory slot of the other player = $EMPTY$ **then**

$currentPlayer[myWallet].inventory[i].isSlotFull \leftarrow FALSE$

$currentPlayer[targetWallet].inventory[i].isSlotFull \leftarrow TRUE$

$currentPlayer[targetWallet].inventory[i].itemCode \leftarrow MyItemCode$

$currentPlayer[targetWallet].inventory[i].itemID \leftarrow MyItemID$

$currentPlayer[targetWallet].inventory[i].ownerName \leftarrow TargetName$

$currentPlayer[targetWallet].inventory[i].ownerAddress \leftarrow TargetWallet$

$isSucceeded \leftarrow TRUE$

break

end if

end for

end if

4. IMPLEMENTATION

The game consists of 5 major parts: Implementation of the game client and the game server in Unity, port forwarding for the game server, finding Public IP address for the game server, implementation of Nethereum which helps to interact with Ethereum blockchain smart contracts, and finally, Infura.io calibration which helps the transmission of the transactions between Unity and Ethereum blockchain.

4.1. Implementation of the Game Client and the Game Server in Unity

Initially, it is required to create two different projects. One for the game client, one for the game server (Figure 4.1).

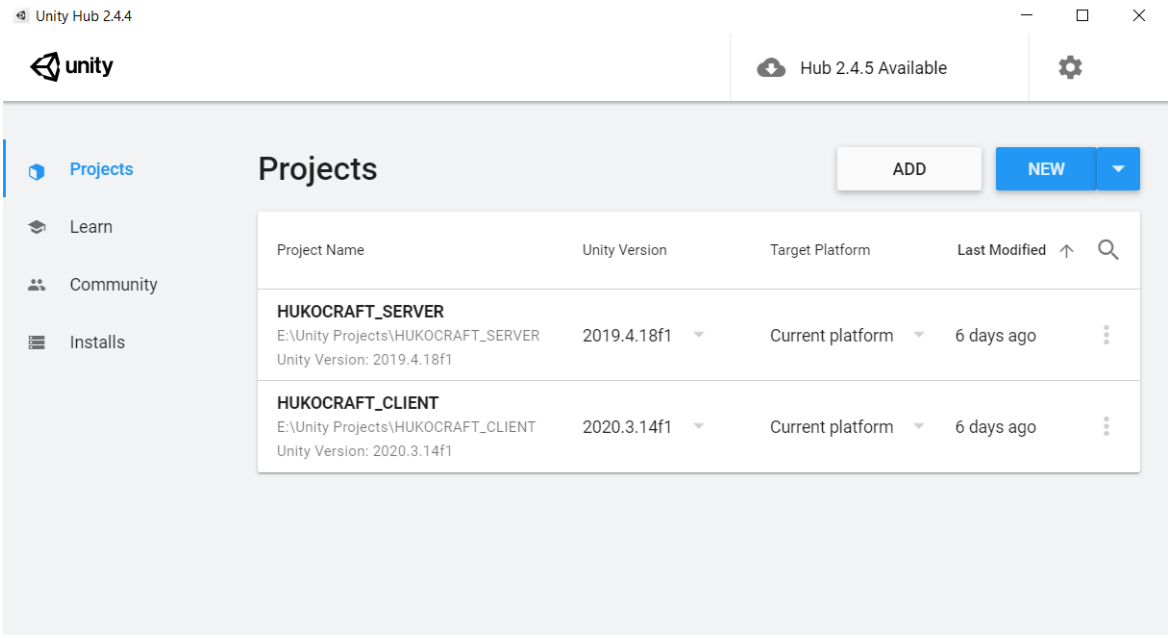

Figure 4.1. Hükocraft game projects for the client and the server in Unity.

The game server (Figure 4.2) should only contain the obstacles in the game. Because, the game physic calculations such as colliding an obstacle, gravity, momentum are server-side operations. Therefore, the positions of the game objects are more crucial than the texture of the objects. Thus, the obstacles are placed with no material texture. This enables the server to consume less GPU.

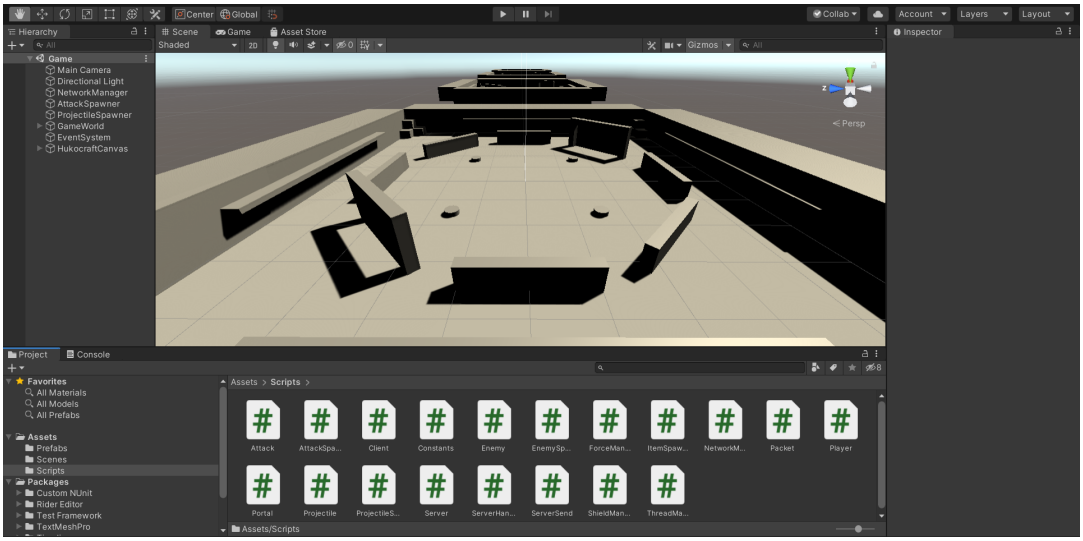

Figure 4.2. Hükocraft server-side appearance in Unity.

On the other hand, the game client (Figure 4.3) should contain the obstacles with the exact position and shape where they are put on the server-side. However, this time, the obstacles should be given the material textures, which are going to be seen by the players.

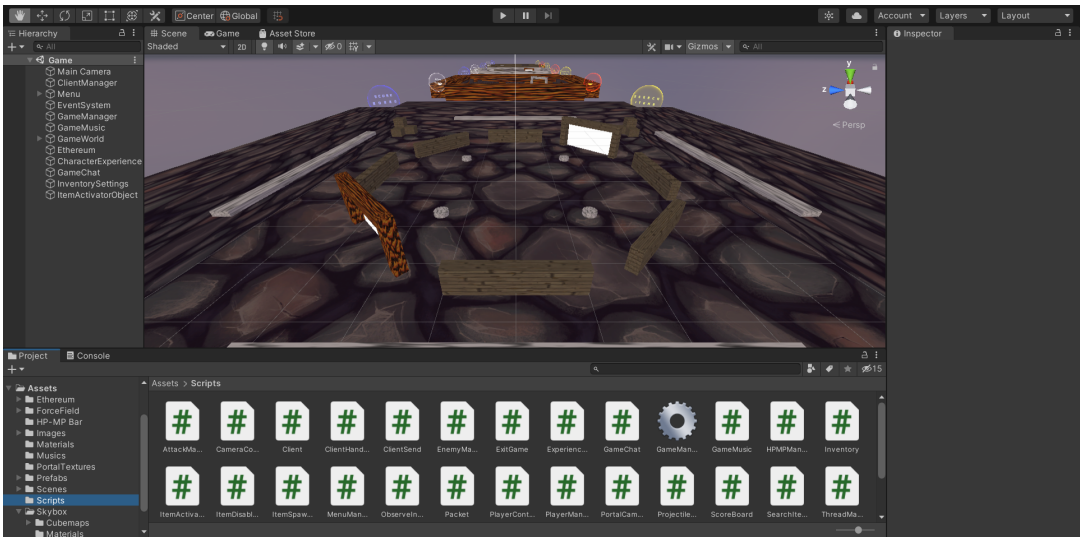

Figure 4.3. Hükocraft client-side appearance in Unity.

4.2. Sharing the Game Server Port with the Public (Port Forwarding)

Since the game server must be able to send and receive all the client requests and response through the TCP and UDP packets, the server port must be publicly available. Otherwise, the players cannot connect to the game server. This process is called port forwarding. Firstly, open the command prompt (as known as CMD) and type “ipconfig” (Figure 4.4).

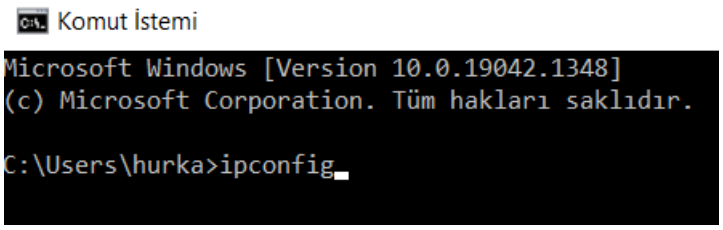

Figure 4.4. The command ipconfig in CMD.

Secondly, find “Default Gateway” (Figure 4.5). It could be below the title of ”Wireless LAN adapter Wi-Fi” or ”Ethernet adapter Ethernet 4” which depends on your internet connection. The IPv4 address is going to be required in the future. Therefore, do not close this page.

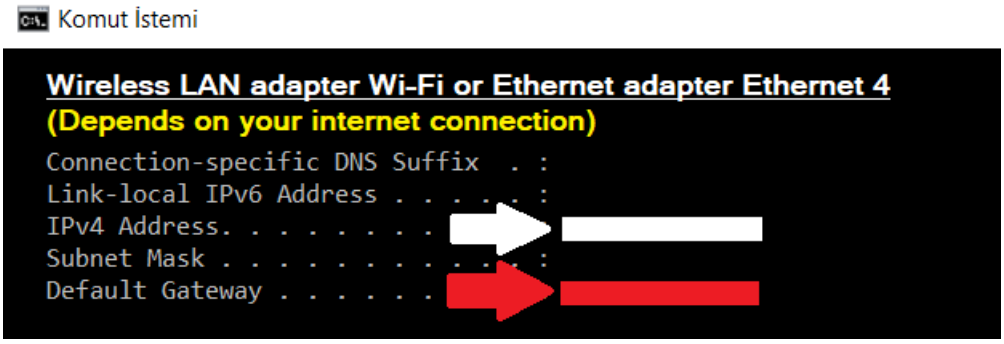

Figure 4.5. Find and copy “Default Gateway” address.

Thirdly, copy the “Default Gateway” and paste it into your web browser. Then, type “admin” in both of the text fields and press the log in to your internet router. Finally, find and press on the Port Forwarding section, and forward your ports by using the IPv4 address and the server port number.

Lastly, the public IP address is the address which the players use to connect to the server. Therefore, the players should be given the public IP address of the server. To find the public IP address, go to WhatIsMyIP website [23] and you shall find your public IP address (Figure 4.6). Then, you should send the public IPv4 address to the players.

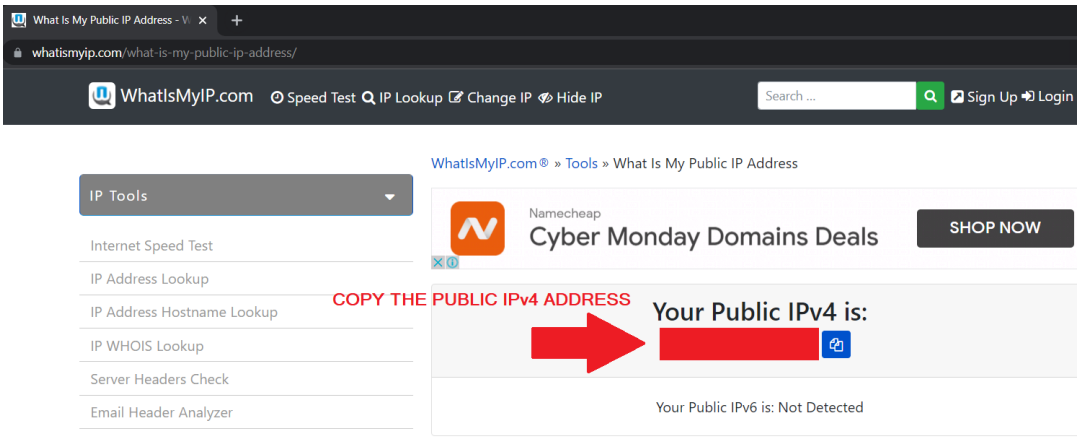

Figure 4.6. Find the Public IPv4 address of the game server and send it to the players.

4.3. Nethereum for the Ethereum Blockchain Interactions in Unity

Nethereum is the .NET library for Ethereum which enables the users to interact with smart contracts and Ethereum blockchain nodes. Go to Github/Nethereum [24] to download the GitHub libraries of Nethereum (Figure 4.7).

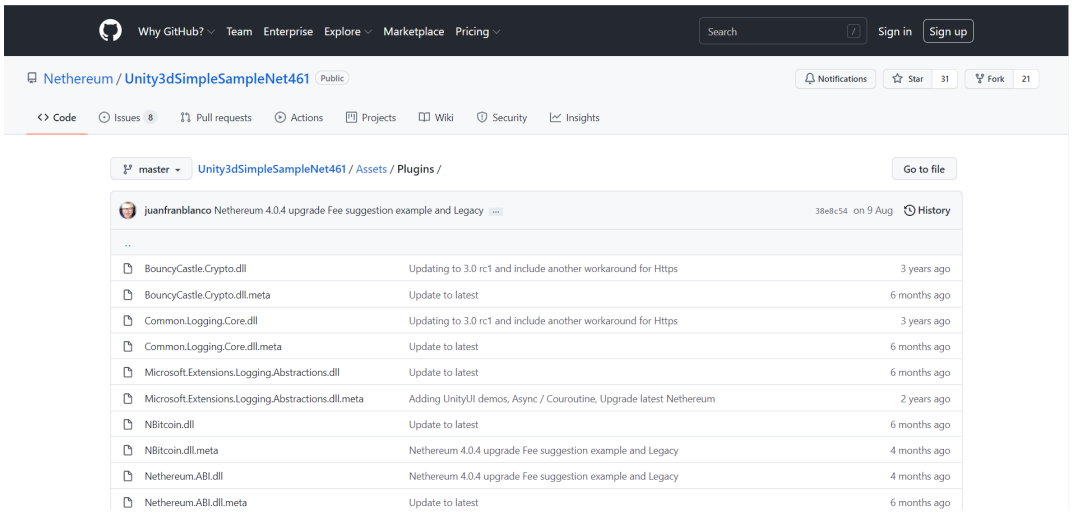

Figure 4.7. Download the Nethereum libraries from GitHub.

Then go to the Unity project for the client, under the Assets folder, create an “Ethereum” folder, then inside of the “Ethereum” folder, create a “Plugins” folder, then import the downloaded Nethereum libraries here (Figure 4.8).

Figure 4.8. Nethereum files, which require to be imported in Unity.

Having completed the installation of Nethereum, all the classes, which are necessary for communication with the Ethereum blockchain, can be created (Figure 4.9).

Figure 4.9. The relation of the classes, which have been created for Ethereum Blockchain communication.

4.4. Implementation of Hükocraft Smart Contract

The smart contracts of the Ethereum blockchain are the way to run programs on the blockchain. In the project, the Hükocraft Smart Contract is used to store player information, and perform peer-to-peer transactions such as Ether Transfer Transaction, Level Experience Transactions, Token Management Transactions, Token Exchange Transactions, Token Query Transactions, Inventory Query Transactions, and Scoreboard Operations.

Hükocraft Smart Contract has been created at Remix Ethereum IDE [25] and the contract is written in Solidity programming language which is developed to run codes on Ethereum smart contracts. After the smart contract is written, VS Code [26] is used to convert Solidity code to C# code to be able to run on the Unity platform 4.10.

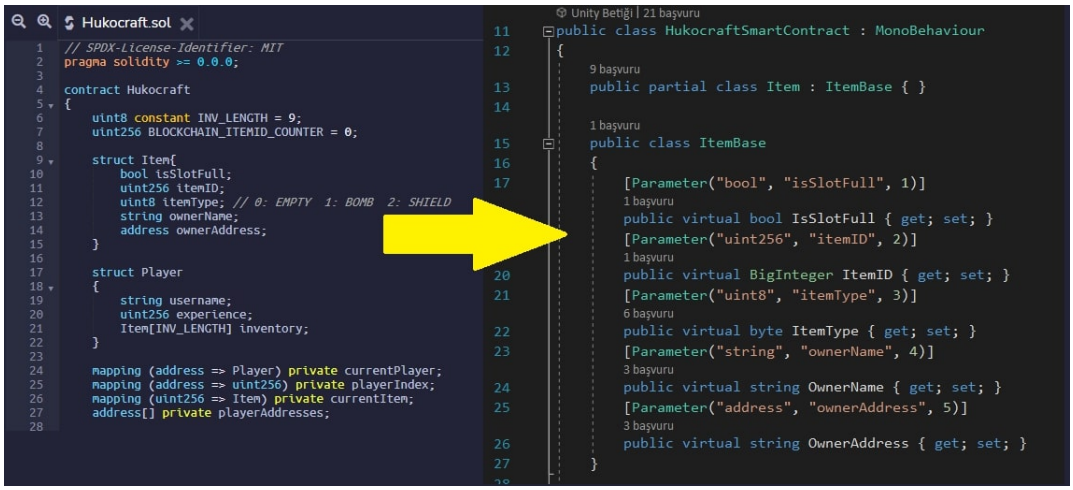

Figure 4.10. Converting Solidity code to Unity C# code

Finally, Infura [27] is used to establish the communication between C# smart contract methods and Ethereum blockchain. In other words, the players call the convenient smart contract method in Unity C# code, the request is sent to Infura.io, and then Infura.io transmits the request to the corresponding Ethereum blockchain to make the transactions.

4.5. Infura for the Communication between Ethereum Blockchain and Unity

Before a blockchain transaction is performed, it is necessary for a computer node to synchronize the blockchain for itself. This process can take hours or days. Thanks to Infura, the developers do not have to synchronize the blockchain by themselves. Because Infura already does this for them.

Infura helps the developers to connect to the Ethereum blockchain and IPFS [28] in a quick, cost-effective, and easy way. It allows the developers to access the synchronized blockchain networks by eliminating time-consuming synchronization steps. No complex setups are needed.

Firstly, go to Infura's website [27] and complete the registration process, and press the Create a New Project button.

Then, select Ropsten as the endpoint and copy the https link, which is going to be used in Unity to send the transactions to the Hükocraft Smart Contract on the Ethereum Blockchain (Figure 4.11). The URL is used as the parameter of the Web3 object.

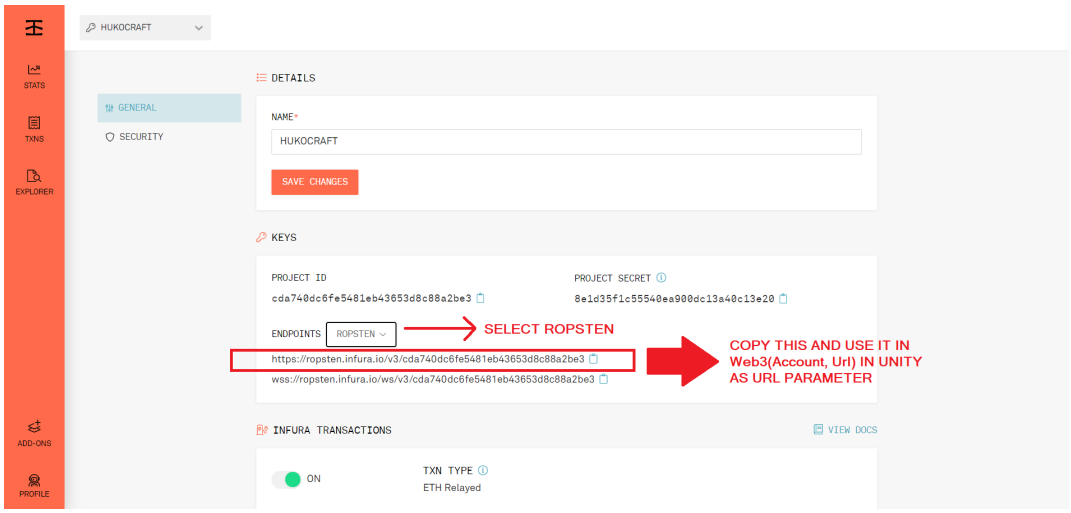

Figure 4.11. Copy the Infura link of the project, and use it in Unity.

4.6. Implementation Diagrams

The communication of Unity with synchronized Ethereum Blockchain Network is established by the Infura (Figure 4.12). Firstly, the player sends the transaction request to the Infura to be delivered to Hükocraft Smart Contract. Then, Infura deliveries the incoming request to the synchronized Ethereum Blockchain. The response of the invoked function on the Hükocraft Smart Contract is transmitted from the Ethereum blockchain to the Infura. Lastly, Infura deliveries the transaction results to the player.

Figure 4.12. Interaction of Unity with Ethereum Blockchain.

The game Server is only responsible for packet transmission related to the game chat and the game physics. The overall interactions are given in Figure 4.13.

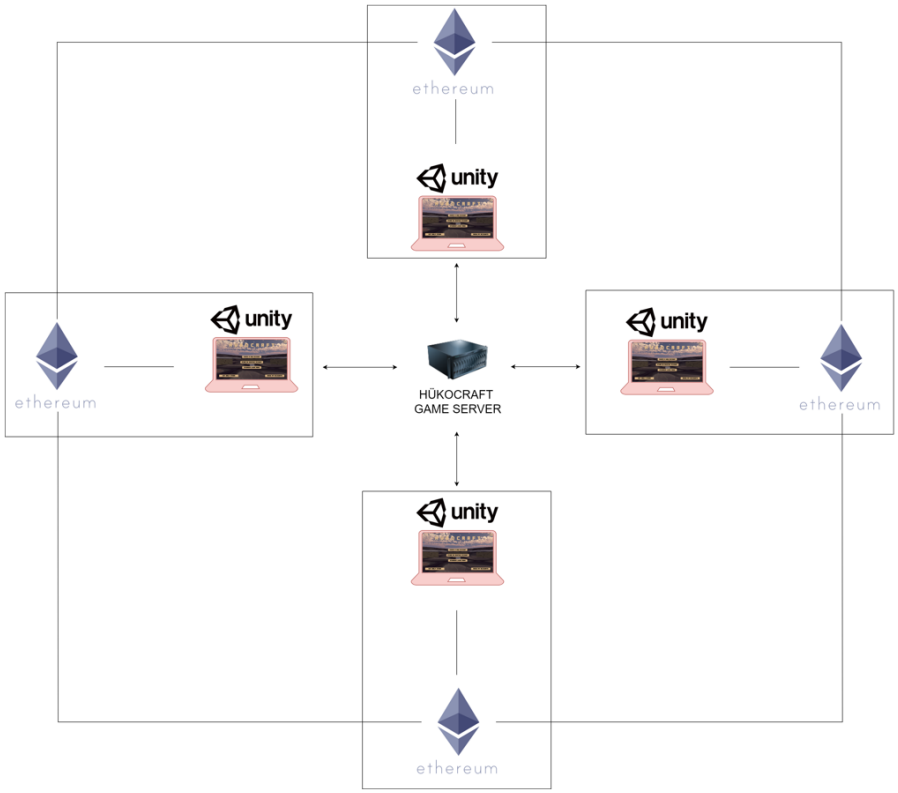

Figure 4.13. Interactions of the players and the game server

4.7. Application Walkthrough and User Interface

Unity Game Engine is used to build the interface, and the game UI is created by using C# Programming language. When the user successfully logins with their Ethereum wallet, they need to enter the username, server IP address, and the server port number. After connecting to the server successfully, there are some areas that enable to the users interact with the Ethereum Blockchain Smart Contract. To interact with the Ethereum blockchain, Nethereum is used, which enables the Unity Game Engine, interact with the Ethereum Blockchain.

The item query menu is shown in Figure 4.14. The existing tokens are listed on the right side of the screen. Moreover, the players are provided to search existing and destroyed tokens by their ID on the left side of the screen. The token owner and the owner's wallet address also can be retrieved by using this method.

Figure 4.14. Ethereum Blockchain Based Token Query Menu

The Scoreboard of Hükocraft Online is retrieved in Figure 4.15. The players are sorted by their total amount of experience. The list also contains the player names and the player levels.

Figure 4.15. Ethereum Blockchain Based Scoreboard Menu

The Inventory Query operation is displayed in Figure 4.16. The players are provided to look up any inventory of the existing players by using the ethereum wallets.

Figure 4.16. Ethereum Blockchain Based Inventory Query Menu

Furthermore, token exchange among the players is provided. Firstly the user must type ”/trade USERNAME” on the game chat. Then the text message is sent to the game server by using TCP packets. The server converts the ”USERNAME” to the ”User ID”, then finds the corresponding player, and delivers the trade request. When the other player accepts the trade request, the trade menu is displayed (4.17). The users can put some amount of Ether or tokens to be exchanged. Once the items to be exchanged are determined, both sides press the Lock button. This ensures that none of the items and ETH values can be changed during the trade. Then, Confirm buttons are displayed. When both of the users click on the Confirm buttons, the item exchange function through the blockchain begins.

Figure 4.17. Ethereum Blockchain Based Trade Menu (Real Time Transactions)

The item exchange function is carried, out of the Unity Game Engine, on the Hükocraft Smart Contract. That means, even if one of the players is disconnected, the transaction is not disrupted and continues.

5. TEST AND RESULTS

The evaluation of the project is held in this section. Having completed the test stage, the results of the evaluation stage shall be discussed.

5.1. Transaction Speed Test

The test is done to measure the latency of the read and write operations on Ropsten Ethereum Blockchain Network. During the test, 30 transactions are made to measure the latency of the write operations, and 30 transactions are made to measure the latency of the read operations. The gas limit [29] is set to 1.000.000 and the gas price is set to 5 Gwei [30] (0,000000005 ETH).

The transaction speed test is carried out according to the transaction types which are shown Table 5.1.

	Write Operations	Read Operations
1	Ether Transfer Transactions	Token Query Operations
2	Level Experience Transactions	Inventory Query Operations
3	Token Management Transactions	Scoreboard Query Operations
4	Token Exchange Transactions	Get Balance Operations

Table 5.1. The read and the write operations in the Hükocraft Smart Contract.

The latency of the transactions is recorded in Table 5.2. The complete list of the results can be found in the appendix section of this report in Table A.1.

	Write Operation Latency	Read Operation Latency
1	10117 ms	164 ms
2	18498 ms	167 ms
3	7471 ms	311 ms
4	12022 ms	635 ms
5	7955 ms	223 ms

Table 5.2. Some of the test results of the read and the write operation latencies.

Based on the test results, the standard deviation, minimum transaction time, maximum transaction time, and average transaction times are calculated and the results are demonstrated in Figure 5.1.

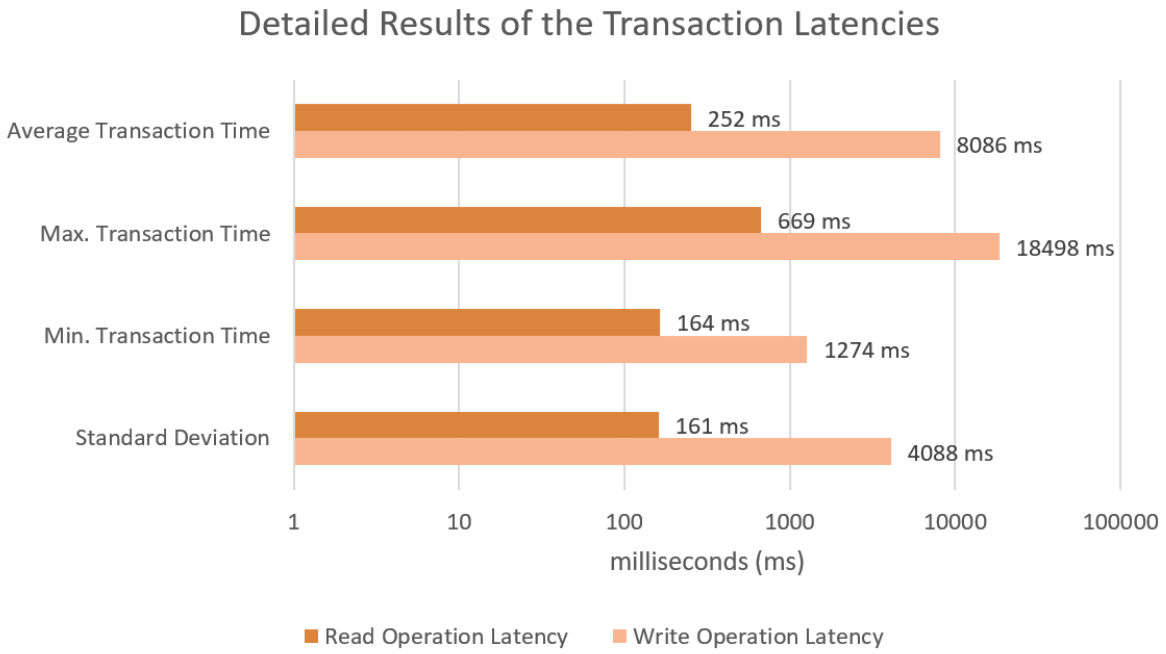

Figure 5.1. The bar chart shows the summary of test results.

The test results show that while read operations are executed almost instantly, write operations take time to be fulfilled. The highest transaction latency for the write operations has been measured 18498 milliseconds, the lowest has been measured 1274 milliseconds. Furthermore, the highest transaction latency for the read operations has been measured 669 milliseconds, the lowest has been measured 164 milliseconds.

5.2. Connection Test

The game server can handle up to 5 players due to the speed limit of the home network. Since the game server is connected to the home network, a better internet connection can help the server to handle more than 5 players. When the number of players increases, received and delivered TCP and UDP packets increase as well. Also, the latency of the TCP and UDP packet delivery depends on the user distance from the game server. When a player is connected from abroad, the latency increases dramatically. Therefore, lagging and packet losses are observed on the client-side. Furthermore, the speed of the Ethereum transactions decreased significantly because of the network traffic.

5.3. Transaction Cost Test

The test is carried out to examine the amount of gas used in each transaction type that is shown in Table 5.3.

	Transaction Types (Write Operations)
1	Hükocraft Smart Contract Deployment Transaction
2	Level Experience Transaction
3	Token Add Transaction
4	Token Delete Transaction
5	Token Exchange Transaction
6	Ether Transfer Transaction

Table 5.3. The transaction types which take place in the gas cost test.

The gas cost results of the transactions are shown in the Figure 5.2.

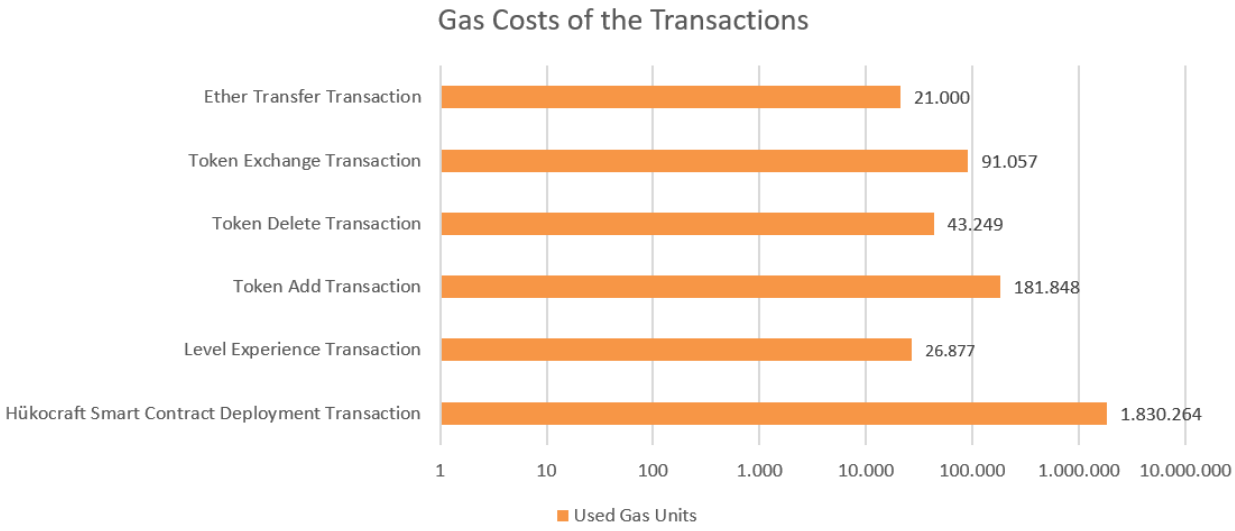

Figure 5.2. The bar chart shows the summary of test results.

The results show that the gas cost of the transactions varies, according to the complexity of the invoked function. The functions with high complexity have higher gas costs such as Token Add Transaction using 181.848 gas. On the other hand, functions with low complexity have lower gas costs, such as Ether Transfer Transaction using 21.000 gas. Furthermore, it is observed that only the write operation consumes gas. The read operations do not consume any gas.

6. CONCLUSION AND FUTURE WORK

Through this project, the people are provided a blockchain-based online 3D game, where their game data is managed by themselves without needing a 3rd party. The data management is conducted on the Ropsten Ethereum Blockchain Test Network. The players can perform Token Search Operations, Inventory Query Operations, Retrieving Game Ranking Scoreboard Operation, Real-Time Item Exchange, and Ether Transfer Operations.

The results of the transaction speed test have shown that although the read operations are fulfilled almost instantaneously, the write operations have some latency, which depends on the complexity of the invoked function on the Ethereum blockchain side. Furthermore, when the number of players increases, the increment of the latency of the Ethereum transactions is observed. Moreover, it is observed that the speed of the home network affects the game flow and Ethereum transactions significantly. Also, it is seen that the used gas amount in the transactions varies according to the complexity of the invoked function. Functions with less complexity have fewer gas costs. Functions with high complexity have higher gas costs. In addition, it is observed that the read operations do not consume any gas, only the write operation consumes gas.

For future reference, due to the fact that the Gas prices are very expensive on the Ethereum blockchain currently, the smart contract of the game is kept on Ropsten Ethereum Test Network. Currently, the players are intimidated by the Gas prices on the Ethereum blockchain. When the Gas prices are optimized in the future, the smart contract of the game can be moved into the Ethereum blockchain. Furthermore, the current game system holds the public address and private keys in a local file named Hukocraft.txt. Therefore, MetaMask can be used to provide more transparency on private key security. Lastly, the game server can be moved into the cloud so that, it can handle more and more people in the future.

APPENDIX A: Test and Results

	Write Operation Latency	Read Operation Latency
1	10117 ms	164 ms
2	18498 ms	167 ms
3	7471 ms	311 ms
4	12022 ms	635 ms
5	7955 ms	223 ms
6	10661 ms	167 ms
7	9755 ms	199 ms
8	13435 ms	168 ms
9	9115 ms	167 ms
10	12974 ms	173 ms
11	6861 ms	667 ms
12	2238 ms	167 ms
13	4515 ms	223 ms
14	1274 ms	167 ms
15	3486 ms	167 ms
16	2615 ms	168 ms
17	8909 ms	268 ms
18	7239 ms	635 ms
19	3252 ms	166 ms
20	14776 ms	167 ms
21	8250 ms	166 ms
22	12697 ms	267 ms
23	4762 ms	166 ms
24	10218 ms	669 ms
25	8074 ms	167 ms
26	10566 ms	168 ms
27	6677 ms	201 ms
28	2951 ms	166 ms
29	3285 ms	165 ms
30	7917 ms	256 ms

Table A.1. The results of the read and the write operation latencies.

Bibliography

[1] IBM, *"What is Blockchain Technology?"*, https://www.ibm.com/topics/what-is-blockchain, Online; Accessed October 25, 2021.

[2] WIKIPEDIA, *"Massively Multiplayer Online Game"*, https://en.wikipedia.org/wiki/Massively_multiplayer_online_game, Online; Accessed October 25, 2021.

[3] L. Achterbosch, R. Pierce, and G. Simmons, "Massively multiplayer online role-playing games: The past, present, and future," *Computers in Entertainment (CIE)*, vol. 5, no. 4, pp. 1–33, 2008, Online; Accessed October 25, 2021. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/1324198.1324207.

[4] FLORENSIA TEAM, *"Florensia - [UPDATE] Important News!"*, https://store.steampowered.com/news/app/384030/view/3035961725860477171, Online; Accessed October 26, 2021, STEAM.

[5] Judge, P., *"OVH fire destroys Rust game data, takes other sites offline."*, https://www.datacenterdynamics.com/en/news/ovh-fire-destroys-rust-game-data-takes-other-sites-offline/, Online; Accessed October 26, 2021, DCD, March 10, 2021.

[6] ETHEREUM, *"Definition of a DAPPs"*, https://ethereum.org/en/developers/docs/dapps/, Online; Accessed October 27, 2021.

[7] ETHEREUM.ORG, *"What is Ethereum?"*, https://ethereum.org/en/what-is-ethereum/, Online; Accessed October 27, 2021.

[8] ETHEREUM, *"What are smart contracts?"*, https://ethereum.org/en/smart-contracts/, Online; Accessed October 27, 2021.

[9] Nethereum Community, *"What is Nethereum?"*, https://ethereum.org/en/smart-contracts/, Online; Accessed October 28, 2021.

[10] Unity Technologies, *"Unity Platform"*, https://unity.com/products/unity-platform, Online; Accessed October 28, 2021.

[11] Ropsten Testnet, *"TESTNET Ropsten (ETH) Blockchain Explorer"*, https://ropsten.etherscan.io/, Online; Accessed November 13, 2021.

[12] UNITY, *"Unity - Manual: System requirements for Unity 2020 LTS"*, https://docs.unity3d.com/Manual/system-requirements.html, Online; Accessed October 28, 2021.

[13] BITCOIN, *"Open Source P2P Money"*, https://bitcoin.org/en/, Online; Accessed October 28, 2021.

[14] X.-J. Jiang and X. F. Liu, “Cryptokitties transaction network analysis: The rise and fall of the first blockchain game mania,” *Frontiers in Physics*, vol. 9, p. 57, 2021, Online; Accessed October 31, 2021. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fphy.2021.631665/full.

[15] CRYPTOKITTIES, *”CryptoKitties | Collect and breed digital cats!”*, https://www.cryptokitties.co/, Online; Accessed October 31, 2021.

[16] ERC721, *”What is ERC-721?”*, http://erc721.org/, Online; Accessed October 31, 2021.

[17] Y. M. Arif, M. N. Firdaus, and H. Nurhayati, “A scoring system for multiplayer game base on blockchain technology,” pp. 200–205, 2021, Online; Accessed November 1, 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9435249.

[18] METAMASK, *”A crypto wallet & gateway to blockchain apps”*, https://metamask.io/, Online; Accessed November 1, 2021.

[19] C. Yaklai and V. Kotrajaras, “An architecture for game to game data transfer using blockchain,” pp. 75–79, 2020, Online; Accessed November 1, 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9268250.

[20] Support.Blockchain Community, *”What is an ERC20 token?”*, https://support.blockchain.com/hc/en-us/articles/360027491872-What-is-an-ERC20-token-, Online; Accessed November 3, 2021.

[21] M. Attaran and A. Gunasekaran, *Applications of blockchain technology in business: challenges and opportunities*. Springer Nature, 2019, Online; Accessed November 3, 2021. [Online]. Available: https://books.google.com.tr/books?id=9Q-yDwAAQBAJ&lpg=PP6&ots=siwNc-rgbb&dq=Applications.

[22] Faucet.Ropsten, *”Ropsten Ethereum Faucet”*, https://faucet.ropsten.be/, Online; Accessed January 13, 2022.

[23] Whatismyip.com, *”What Is My Public IP Address?”*, https://www.whatismyip.com/what-is-my-public-ip-address/, Online; Accessed January 13, 2022.

[24] Github/Nethereum, *”Nethereum/Unity3dSimpleSampleNet461”*, https://github.com/Nethereum/Unity3dSimpleSampleNet461/tree/master/Assets/Plugins, Online; Accessed January 13, 2022.

[25] Remix.Ethereum, *”Remix - Ethereum IDE”*, https://remix.ethereum.org/, Online; Accessed January 13, 2022.

[26] Visual Studio Code, *”Code Editing. Redefined”*, https://code.visualstudio.com/, Online; Accessed November 13, 2021, November 3, 2021.

[27] INFURA.IO, *"Infura - Ethereum API | IPFS API"*, https://infura.io/, Online; Accessed January 13, 2022.

[28] IPFS.ORG, *"IPFS powers the Distributed Web"*, https://ipfs.io/, Online; Accessed December 8, 2021.

[29] ETHEREUM, *"Gas and fees"*, https://ethereum.org/en/developers/docs/gas/, Online; Accessed December 8, 2021.

[30] ETHEREUM.ORG, *"Ethereum Units — Ethereum Homestead 0.1 Documentation"*, https://ethdocs.org/en/latest/ether.html, Online; Accessed December 8, 2021.