Bachelor’s thesis

Information and Communications Technology

2023

Eric Ng

DESIGNING AND DEVELOPING A MULTIPLAYER VIRTUAL REALITY ENVIRONMENT

TURKU AMK

TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR’S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and Communications Technology

2023 | 46

Eric Ng

DESIGNING AND DEVELOPING A MULTIPLAYER VIRTUAL REALITY ENVIRONMENT

The goal of this bachelor thesis was to investigate the design and development principles for the implementation of a networked VR environment and produce a networked VR application with a client that has more authority when used to connect to an external server. An additional goal was to assess the need for such functionality inside a VR environment and the possible improvements.

The commissioner for this thesis was Turku Game Lab at Turku University of Applied Sciences and the case study was implemented with the requirements introduced by the commissioner.

A case study was developed for the test environment, the test environment was built with Mirror. The testers were random, and the same devices were used to keep consistency, the location and network were arbitrary. The testers were instructed to perform certain tasks on a canvas, these tasks could be completed if they pressed certain buttons. After the testers had gone through all the functionalities, they were interviewed about the functionality of the canvas.

The result of this thesis was a networked VR application with the use of Mirror which is connected to an external server, research data with regards to the canvas, and this study report. According to the data, out the of five testers, all five of the testers felt positive towards their experiences with the canvas, and three out of five of them expressed an interest to see more development with regards to the canvas. This confirms the need for these functionalities within a training environment.

KEYWORDS:

Virtual Reality, Mirror

CONTENTS

PICTURES

TABLES

LIST OF ABBREVIATIONS

HMD	Head-Mounted Display
MLAPI	Mid-Level API
PUN	Photon Unity Networking
RPC	Remote Procedure Call
SDK	Software Development Kit
UI	User Interface
VR	Virtual Reality

1 INTRODUCTION

In recent years Virtual Reality (VR) has been on the rise. This popularity has enabled other sectors, such as virtual training. Throughout the coronavirus lockdown, several educational institutions were forced to look for an alternative to contact teaching. One of the many alternative methods was VR training. The teachers who opted for this type of teaching received positive feedback. The problem with this method was that the teachers lacked control over the environment. The teachers could not efficiently control the environment as needed. This combined with the difficulty to monitor all the students in VR made it difficult for teachers to efficiently teach and guide the students.

This challenge created a demand for such functionality in a VR environment and the development and design process for it.

This thesis will analyse the need for such an environment through qualitative research, the possible improvements and problems the current environment has, and the various ways to develop such an environment.

This thesis can serve as a foundation for an environment to train in VR, an environment that allows the teacher to monitor students and control them to a certain degree.

The reason for this thesis was the lack of documentation for VR with Mirror, a networking library for Unity. On top of that, there is an increased interest in VR training. Various network technologies have been researched and compared against Mirror. Therefore, this thesis has two purposes. The first one is to gather information about the development process for a training VR environment and produce an environment that simulated functionality that gave the teacher control over the testers. The second purpose is to determine whether these functionalities were needed and what was successful or did not work well. Turku Game Lab at Turku University of Applied Sciences was the commissioner of this thesis project.

The research was conducted in a primary qualitative manner and a case study to test with. There was a requirement to communicate the needs of the commissioner for this environment.

This thesis is divided into nine chapters. Chapter 1 starts with the introduction to this thesis and then moves on towards Virtual Reality. Chapter 2 provides an overview of VR, the history of VR, the use of cases and its potential problems. Chapter 3 consists of theory, and relevant technologies such as game engines and network libraries are analysed and examined. In Chapter 4, the design and development for the environment with Mirror are researched. Chapter 5 focuses on the case study Master UI Simulation. The details about the test, its environment, and the manner it was conducted are discussed in Chapter 6. The results of the data from Chapter 6 are analysed and discussed in Chapter 7. Chapter 8 reflects on the work carried out in this thesis and Chapter 9 concludes this thesis.

2 VIRTUAL REALITY

This chapter provides a general overview of Virtual Reality (VR) development and its use cases in training throughout time. Additionally, the concept and the problems of virtual training. The VR system components and the future of the VR as well as the VR training market are explored. This chapter is expected to provide the reader with a general overview of the history, the use of cases, and technologies for VR.

2.1 Virtual Reality Defined

VR is a simulated experience. VR aims to give the user a sensory immersive experience, to include sight, touch, smell, or even taste. This experience is mostly simulated with the use of an HMD (head-mounted display). (Alsop, 2018)

2.2 History of Virtual Reality

2.2.1 Sensorama Simulator - 1956

Morton Heilling created the Sensorama in 1950 as he wanted to build a cinema of the future. The Sensorama was a fully functional 3D video machine. The machine was large and bulky and required the person to sit in. The purpose of Sensorama was to immerse the user more in the use of 3D films, but throughout those times they were very difficult to obtain. Thus, he produced his films for it. Heilling made 5 movies, in these movies it was possible to hear, smell, feel and touch the environment. This creation was innovative. Unfortunately, the project was put on hold because of a lack of financing. (Brockwell, 2016)

2.2.2 Sword of Damocles - 1968

The Sword of Damocles was invented by a team led by Ivan Sutherland in 1968. It was one of the first HMDs to connect to a computer instead of a camera. This change made user interaction possible. Although the experience for the view was very primitive, the display consisted of wire-frame models with basic shapes. The device was a contraption that was suspended from the ceiling. (Lewis, 2018)

2.2.3 The Super Cockpit - 1986

The Super cockpit was a military training program led by Thomas Furness. The program was essentially a flight simulator. The super cockpit was designed to train pilots with the use of an interactive system, this system would simulate 3-D virtual space. In this virtual space, the pilot can view and hear in real time. As opposed to its predecessor, the helmet tracking system and sensors allowed the pilot to operate the aircraft with the use of gestures, speech, and eye motions. Which removed the need for a physical aircraft. (Järvinen, 2021)

2.2.4 Project VIEW

NASA launched their VR HMD prototype in 1986, called the Virtual Interface Environment Workspace (VIEW). This VR HMD included gloves for hand-tracking purposes. This meant that the user had control over the hands and would input data in the virtual world. The primary reason for the development of this prototype was to train astronauts. The intended use was for astronauts to control the extender arms, cameras, and other objects to accomplish dangerous tasks and exploration without putting the user in danger. (Turi, 2013)

2.2.5 Oculus Rift - 2010

By the early 2000s, most commercial VR products failed to withstand the passage of time and were discontinued. However, Palmer Luckey, a VR enthusiast, collected an arsenal of either discontinued or failed HMD prototypes over the years. He was dissatisfied with the quality and usage of the previous models, which encouraged him to build his own. The HMD was put together with some random components and combined with duct tape. This HMD was superior to all previous iterations. He announced his projects on a forum and was discovered by John Carmack. John Carmack was a famous game developer, he worked on a lot of big games which earned him recognition in the gaming community. John Camack displayed the early prototype of Oculus Rift at one of the largest game conventions, E3 USA. This event reignited global interest in VR. (Rubin, 2014)

2.2.6 Oculus Go – 2010

The Oculus Go is the first commercially available VR HMD that was wireless. This model was affordable to the public and allowed the user to view their environment. Unfortunately, this did not include positional tracking. The lack of positional tracking limited the users' movements, as they were only allowed to stay in one position. The cheap price tag with the ease of use opened the testing ground for several organisations. Most organisations interested in VR began to embrace VR HMD more after the release of this device. (Wong, 2018)

Walmart participated in a VR training program. The trainees saw an increased performance by 10 to 15% while also increasing confidence and retention. (Incao, 2018)

2.2.7 Oculus Quest

Oculus Quest is an improved iteration of the Oculus Go. One of the problems that the Oculus Go had, was that the Oculus Go is a stationary device. This implies that with the use of the HMD, the user can only gaze about. Oculus Quest users, on the other hand, have more freedom of movement and may digitally wander around with the HMD on. (Betker, 2019)

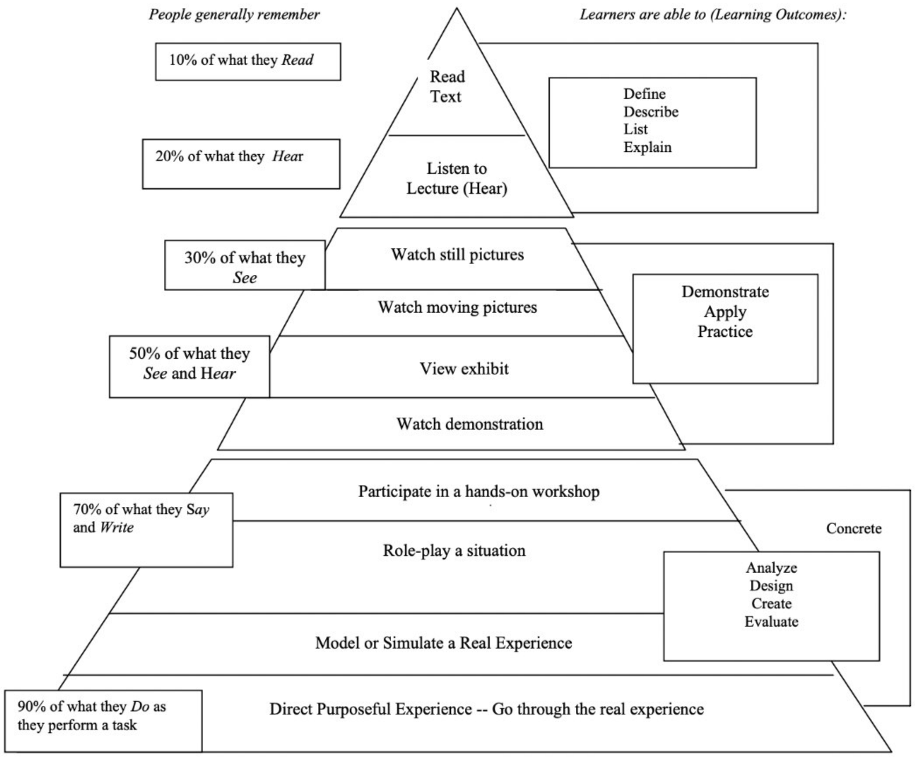
The U.S. Space Force is training with the use of the oculus quest. The training activities consist of situations that occur in space such as repairing a satellite, which was otherwise impossible without sending the pilot to space. (Brown, 2021)

2.3 Virtual Reality Training

Edgar Dale proposed the idea that learning occurs more effectively when students "do" rather than "hear," "read," or "observe" to remember information. The Cone of

Experience is a result of his research (Table 1). These days, "learning by doing" is referred to as "action learning" or "experiential learning." (Edgar, 1969)

Table 1. Benefits of doing. (Edgar, 1969)


People generally remember
Learners are able to (Learning Outcomes):
10% of what they Read
20% of what they Hear
30% of what they See
50% of what they See and Hear
70% of what they Say and Write
90% of what they Do as they perform a task
Read Text
Listen to Lecture (Hear)
Watch still pictures
Watch moving pictures
View exhibit
Watch demonstration
Participate in a hands-on workshop
Role-play a situation
Model or Simulate a Real Experience
Direct Purposeful Experience -- Go through the real experience
Define Describe List Explain
Demonstrate Apply Practice
Analyze Design Create Evaluate
Concrete

VR training is an immersive experience. The purpose is to simulate workplace obstacles and recreates real-life surroundings. In the past, this was an expensive procedure, and in addition to that, the equipment was hard to use, which made it difficult to justify the high price tag. This started to change with the Oculus Quest, which was an affordable device and had ease of use. As the cost of the equipment goes down, VR training becomes more affordable for companies. Throughout the pandemic, educators were forced to look for an alternative way to teach. A teacher in Poland used VR to teach his class. The school recorded numerous classes in different subjects. The initial feedback seems positive, and this news has sparked a newfound interest in other bodies from the government. (Bretan, 2020)

2.3.1 Virtual Training Advantages

VR training is a relatively new concept, but much research has been conducted into this field, and the results show that VR training offers a substantiate number of benefits. A study by PricewaterhouseCoopers showed that students learned on average 4 times faster compared to physical training in a classroom. Virtual Reality gives students more confidence and it also provides more immersion. This allows the students to focus better and feel more connected to the class. (PricewaterhouseCoopers, 2021)

2.3.2 Virtual Reality training Difficulties

While there are advantages to VR training, it certainly brings its own set of disadvantages. A study by Alsop (Alsop, 2021) argued that certain issues prevented more widespread adoption of this technology.

3 RELEVANT TECHNOLOGIES

The advantages and disadvantages of different game engines and network technology are discussed and at the end of this section, they will be compared. The network architecture will be analysed and the implementation on top of the reason behind the choices that have been made will be discussed. Security or migration will not be considered.

3.1 Game Engines

To develop a game takes a lot of time. This is where game engines are useful. When the development with game engines shortens and eases development cycles by huge margins. This is no different for VR development. There are multiple different game engines for various purposes, so only the best game engines for VR are considered.

3.1.1 Unity

Unity is the preferred platform by most developers for the use of VR. The strong points of Unity are its accessibility and availability. Unity has great platform coverage; With Unity, there is an option to develop games and applications with a wide variety of VR tools and a Software Development Kit (SDK). Most VR development tools come in the form of SDK. The use of SDK eases the development process. While it is recommended to use these tools, there can be some sort of drawback to their usage. The problem they bring along is that the SDK is only available for its dedicated hardware which causes compatibility issues.

One of the ways to solve this problem is with the use of an (SDK) called XR Interaction toolkit. This SDK converts all the different hardware functions into a singular functionality which is much more modular and solves the compatibility issue with all the different hardware. With the use of SDK. The input will be reset, and the functionality needs to be built with the toolkit in mind. This could result in difficulties if one wishes to use this with big codebases. (Garbett, 2022)

Unity was chosen for the development framework because of the before-mentioned reasons, and this was the preferred platform by the commissioner.

3.1.2 Unreal Engine

Unreal Engine is one of the most popular frameworks to develop games with. It boasted the 2nd biggest market share on Steam with 24% of the total market. Unreal Engine has provided a lot of high-quality tools that deliver good results for developers, animators, and designers. Blueprints are one of their main core components, which is a low-code, drag-and-drop solution for beginners and advanced users alike.

Blueprints are very easy to use and require no prior coding skills although knowledge of the logic and usage is recommended. Unreal Engine provides tools that are tailored to provide good graphics. Unreal Engine supports up to 15 different platforms but is not capable of cross-platform which means there will be an additional cost to remake the games on another platform. The entry bar for beginners is considered intermediate since it is required to have intermediate or advanced experience. (George, 2022)

3.1.3 Comparison

While Unreal Engine is the preferred solution for big development studios. Unity has cross-platform and is very comprehensive for newcomers. These functionalities make Unity very attractive to newer developers.

Unity is used on 60% of the VR/AR market on Steam. Despite Unity being much more popular, as per research Unreal Engine is also a valid solution as both are very compatible with VR functionality.

The commissioner request was to develop the case in Unity and which is an ease-of-use framework with a big market share in VR/AR. These are the reasons why Unity was picked as the preferred framework for the case. (Gajsek, 2019)

3.2 Network Topology

Network topology describes how a network is set up. There are a lot of different types of networks available for the public, but this chapter will only look at the most relevant ones for the case. The client-server model and peer-to-peer. The listen server architecture and potential problems and solutions will be discussed.

3.2.1 Client-Server

The client-server model is the more popular model. In a client-server model, all the clients are connected to a single server and communicate through the server. Typically, this model has been used alongside an authoritative server. An authoritative server means that the server has the correct game state, and the client needs to request permission from the server to act. Hence, the name is authoritative. When the client request to act, the server needs to process this and allow or disallow this action. One of the most common problems is to find a proper way to traverse connections with other clients this means that connections to the server will either not be found or blocked.

There are various types of servers, but only the relevant ones will be covered. Which are listen and dedicated game servers.

Listen servers are active participants in the game itself, usually referred to as the ‘host’. The listen server is being hosted by a client, there was no need for another machine to act as a server, which greatly reduces the cost to rent or operate servers. The disadvantage of this method is that the listen server needs to be powerful enough to deal with this increase in traffic. Since the host is both a client and a server, it needs to be able to handle the additional connections and run the client simultaneously. In this model, the other clients are connected to the host and the host sends data to the other clients. This can be problematic because if the host disconnects or closes the game for any reason, the state of the game will be discontinued, hence the other clients will lose their progress. To connect with other clients is often another problem because often there will be security on the networks such as firewalls or routers. In these cases, the clients will be unable to connect to the clients.

dedicated game servers run the logic and communicate with the other clients. The logic runs in the background for the dedicated game servers and is separate from the client.

This means that a dedicated game server does not display graphics and is only in charge of the logic and communications between the clients. This is usually referred to as a 'headless server'. The state does not affect clients that join or leave the server because the host is the server itself. This method is preferred for games with a large capacity as it removes the heavy load process which renders graphics. Therefore, it can focus purely on communications with other clients. The downside of a dedicated game server is that it is required to set up a separate machine or environment to run it, which can be costly. The implementation for this type of architecture is also vastly more difficult than a listen server because the server will be the one to hold the authority and synchronization of the state over the network. However, dedicated game servers act as relay servers. Relay servers eliminate the communication problem listen servers face. This is the reason why the decision was to use a dedicated game server for the case. The commissioner request was to deal with a strong firewall and have high uptime or availability. (Glazer & Madhav, 2016)

3.2.2 Peer-to-peer

With the peer-to-peer model. It is often implemented without an authoritative server therefore the input or lag is not as much of a problem compared to the server-client model. In this model, every client is connected to every other client. This means that there will be more than one connection that the client will receive data, which can be problematic because the game state can be different and unsynchronized from the other clients. The game state needs to be consistent between all clients. The clients receive all the data from every other client and the load these clients will receive or send out is increased drastically with more players, therefore peer-to-peer is commonly used in smaller-scale projects. This method was not suitable and therefore was not considered. The commissioner requested a server-client model. (Glazer & Madhav, 2016)

3.2.3 Remote Procedure Calls

RPC stands for the remote procedure call. A program can utilize a Remote Procedure Call, a software communication protocol, to communicate with another program on a network for a service while it is not familiar with the specifics of the network. RPC is used is call procedures on other systems as if it called was on a local system. A procedure call is often referred to as a function call. The client-server concept is used in RPC. The service program that provides is the server, while the program that makes the request is the client. This is commonly used in other frameworks to replicate the state or communicate with other systems. (Glazer & Madhav, 2016)

3.2.4 Connecting to another computer

A connection can be established to another computer. This process is complex as there will be various securities implemented to protect the other computer from malicious activity. In this subsection, the possible solutions will be discussed and the reasons that are chosen for this method. (Coughlin, 2021)

3.2.4.1 Port forwarding

Port forwarding is when the ports on the router or modem are opened so that other machines can connect to this port. It is one of the more popular methods but also a more difficult one as the user is required to have some technical skills to follow this approach. On top of that this approach might not always be possible as the user might not have access to the router. This approach is not recommended due to unreliability and unreachability. (Coughlin, 2021)

3.2.4.2 Relay Server

The relay server has its port forwarded and therefore mitigates the problems of the user not being able to connect. As the port is publicly available, the connection to a relay server should always succeed. The way this works is that the data is sent to the relay server and the relay server responds to the data and sends it to the appropriate clients. The problem with this approach is that there might be an additional cost involved because there is a need to host the server on a device. This usually is an external device so the data needs to take a round trip for it to reach the right destination. As there is a need for stability and security this approach was considered for the case. (Coughlin, 2021)

3.2.4.3 NAT Punchthrough

NAT punchthrough happens when a direct connection is opened between different clients. The problem with NAT Punchthrough is that this method is that it is possible to fail, this depends on the type of security there is. This is a big problem because the game must be available to everyone. There are various ways to open a direct connection, but these options will not be explored due to time constraints. Therefore, this method was not considered. (Coughlin, 2021)

3.2.4.4 NAT Punchthrough and Relay Fallback

This method combines both Relay Server and Nat Punchthrough. When a direct connection is opened, it now becomes possible to check if the connection will fail or not. If the connection fails, the connection will be re-established to the relay server which reduces the amount of data that is sent to the server. This method is preferred because it saves the cost of relay servers. This option is valid for the case but due to time constraints, it will not be considered. (Coughlin, 2021)

3.3 Network Libraries

In this subsection, the different types of network libraries that would be relevant are discussed and compared. As there are a lot of different types of network libraries it will be difficult to cover them all so only a select few, will be discussed.

3.3.1 Mirror

Mirror is an open-source 3rd party network SDK. It gained traction after Unity discontinued its support for its net code called UNet. This decision caused other developers to look for alternatives and Mirror was one of the alternatives that came on top. Mirror was built on top of the discontinued net code which made it attractive to use, mainly because the functionality and implementation stayed the same. Mirror has also accumulated a big community over the years and now offers a large range of features like game managers and networked components. Mirror has withstood the passage of time and has evolved to this environment that changes. It has proven to be reliable and offers tons of features together with its ease of use. (Petrov, 2022)

3.3.2 MLAPI

MLAPI is the abbreviation for (Mid-level API). MLAPI was another open-source network library. It was started in 2018 and was acquired by Unity in 2020. It is the preferred solution to build networked games because of the support Unity will provide and the term open-source means that it is free. Since it is relatively new, it still misses some features and proper documentation and examples, but features have been added rapidly due to its endorsement by Unity and its community. MLAPI does not support multithreading which can be problematic for games with larger quantities but there have been instances where developers managed to scale a game up to 64 players (House, 2020)

3.3.3 Photon Unity Networking

Photon Unity Networking is the abbreviation for PUN. The blocks for quick server responses, lag-free rooms, and a solid network are produced. PUN uses a peer-to-peer instead of a client-server model. It uses a relay service to avoid the communication problem, often referred to as a ‘Master Server’. This is a misconception as the server is not doing any synchronization or communication between the clients. This may be an issue if it is required to prevent any attempts of cheats that are used.

It will be difficult to transition to peer-to-peer from a client-server model if there is a need to scale the game. (House, 2020)

3.3.4 Comparison of Network Libraries

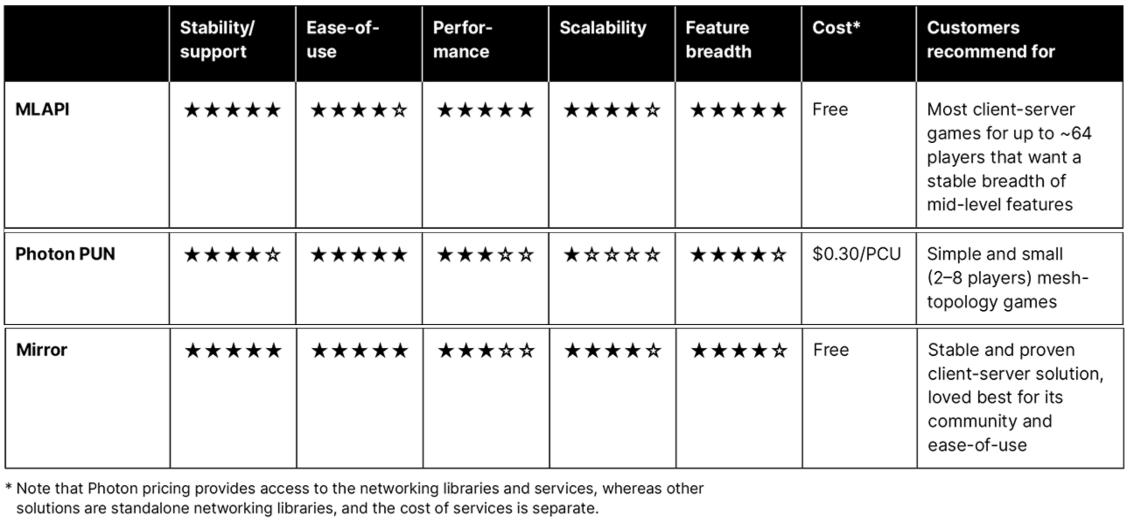
PUN is only recommended for smaller-sized games because it is difficult to transition from peer-to-peer to server-client. This architecture also limits the scale of the game. Peer-to-peer synchronizes every other connection on the client. This means there will be more data sent and received per client; hence, it limits player limits. PUN also has a hidden cost as it is free for up the 20 active users, afterwards, the cost increases up to 95$ for 100 active users per month while Mirror is free. (House, 2020)

Mirror has one of the biggest communities. As Mirror is open source, there are a lot of resources available for the public. This also means that Mirror is actively supported, and a lot of the problems are being fixed by either the team or the community members. This makes Mirror a very stable solution. Mirror is easier to scale than PUN. There have been complaints with larger-scaled games, but the feedback in regard to this issue has been mixed and for most games, Mirror is a good solution. Mirror is free and provides the most stable solution. (House, 2020)

MLAPI shares a lot of similarities with Mirror, but unlike Mirror it has not proven to be stable enough yet as it is relatively new to the market. However, due to the support of Unity, this might change eventually. The documentation and guides are also not as complete compared to Unity at the current time. Despite these reasons, MLAPI is a valid option. However, they will not be considered due to the somewhat lack of stability it provides compared to Mirror. (House, 2020)

The commissioner requested to make a case with Mirror. It was decided to use Mirror for the case due to the beforementioned reasons.

Table 2. Overview of network tools. (House, 2020)



	Stability/ support	Ease-of-use	Perfor-mance	Scalability	Feature breadth	Cost*	Customers recommend for
MLAPI	★★★★★	★★★★☆	★★★★★	★★★★☆	★★★★★	Free	Most client-server games for up to ~64 players that want a stable breadth of mid-level features
Photon PUN	★★★★☆	★★★★★	★★★☆☆	★☆☆☆☆	★★★★☆	$0.30/PCU	Simple and small (2–8 players) mesh-topology games
Mirror	★★★★★	★★★★★	★★★☆☆	★★★★☆	★★★★☆	Free	Stable and proven client-server solution, loved best for its community and ease-of-use

* Note that Photon pricing provides access to the networking libraries and services, whereas other solutions are standalone networking libraries, and the cost of services is separate.

4 NETWORKING

In this section, the Mirror network architecture, and the development technologies that are used in this case are explained. The reason for this choice is that Mirror is free, relatively stable compared to other technologies, and was requested by the commissioner.

4.1 Mirror Topology

Mirror uses a server authoritative system. The transport layer offers support for various types of network topology; however, it has a functionality that allows one of the participants to simultaneously serve as both the client and the server. Mirror has some strong points, such as its ease of use, and iterative development, as well as its various services for networked games. Mirror is composed of several functionally added layers. For engine integration, there is NetworkTransform, NetworkAnimator, and NetworkProximityChecker; for player control, there is NetworkLobbyManager; for networked game state control, there is NetworkManager; for object life-cycle, there is NetworkScene or ClientScene; for object state & actions, there is NetworkIdentity or NetworkBehaviour; for connection management, there is NetworkClient or NetworkServer; to send messages and serialization, there is connection or reader or writer; for low-level API, there is transport or configuration. (Mirror, 2021a)

4.1.1 Server and Host

Mirror multiplayer games include a server and multiple clients.

When multiple players want to play simultaneously, they connect to a server. A server frequently oversees the game and its many components, including scorekeeping and transmitting details to the client. There are two types of servers: dedicated game servers which only serve as the server for the games; and the host server, which is a client that imitates the role of the server. The host is a single instance that can be the server and the client at the same time. For local client communication, the host employs a particular type of internal client, whereas other clients are remote clients (who are connected to the local server on different servers). The local client can directly communicate with the server through function calls and message queues since it participates in the same process as the server. Additionally, the Scene is shared with the server. Remote clients, on the other hand, rely on a shared network connection to communicate with the server. The multiplayer system aims to ensure that local and remote clients have the same code. Mirror handles this automatically.(Mirror, 2021a)

4.1.2 Authority

Authority is a method to determine who oversees and has ownership of an object.

By default, a server holds control over an object. This implies that the networked objects other than the player, such as, NPCs, and collectables, would be managed, and controlled by the server.

Client authority refers to the authority a client has over an object. In this scenario, an object will be automatically destroyed when a client that has no authority issues commands and will be disconnected. Even though a client has control over an object, the server retains control over SyncVar and other characteristics of serialisation. A component sync with other clients when it updates the server state, this state is updated via Command.

A client can give authority to or remove authority from a client at any time. A message will be delivered to that client to let them know when authority is added to or taken away from an object. A client can check if they have authority over an object, while the server can check which client has authority over an object. (Mirror, 2020)

4.2 Mirror Components

4.2.1 NetworkManager

The NetworkManager component is responsible for overseeing the network components. Among its functionalities are game state management, spawn management, scene management, debug information, and customization. The game features three multiple states: dedicated server, host, and client modes. The host can function as both the client and the server. By relying on the player's selection, the

NetworkManager's state management determines which mode to initiate. NetworkManager can handle the state and manage the instantiation of a networked game object from Prefabs. The NetworkManager automatically handles the scene state and scene transitions.The NetworkManager component makes it easier to build, run, and debug multiplayer games which consolidates a lot of valuable features into a single location. (Mirror, 2021c)

4.2.2 NetworkIdentity

The Unity network core element is NetworkIdentity. It manages a game object's network identity and makes use of that identity to inform the network system of the game object.

Game objects that are saved as part of a scene, can also be networked and assigned a separate identity. Unity disables all scene-based game objects so that they will not be in the wrong locations when the clients start to play. It also ensures that Unity does not spawn and immediately destroys the game objects upon connection. (Mirror, 2017)

4.2.3 Match Interest Management

The Match Interest management should be added to the NetworkManager. This will isolate different scenes on the same server. This management system is not recommended for games that are based on physics. This component works best with non-physics games. (Mirror, 2021b)

4.2.3.1 NetworkMatch

The NetworkMatch should be added on all the networked objects that need to be connected to the Match. This component ensures will be used to communicate with Match Interest Management. (Mirror, 2021b)

4.2.3.2 MatchId

The MatchId should be assigned to all the objects that are supposed to be in the same match. This will allow the network to isolate different players between different Matches. (Mirror, 2021b)

4.2.4 NetworkTransform

The location, rotation, and scale from the networked game object are all synchronised by the NetworkTransform component.

A NetworkIdentity component is a need for any game object that needs a NetworkTransform component. If a game object does not already have a NetworkIdentity, Mirror would assign one when there is NetworkTransform component added to it.

NetworkTransform delivers movement updates at a set interval. These updates might arrive late, out of order, or be lost in transit since network conditions are never ideal.

One of the trickier issues in a game network is how to move smoothly over less-than-ideal network circumstances. Mirror found a solution when Snapshot Interpolation was utilised. (Mirror, 2016)

4.3 Mirror Synchronization

4.3.1 SyncVars

Classes that inherit from NetworkBehaviour have properties called SyncVars. These NetworkBehaviour can be synchronised from the server, and then to the clients. The most recent state of all SyncVars on networked objects is delivered whenever a game object is created, or a new player joins a game.

Any type of data that Mirror accepts can be used by SyncVars. A single NetworkBehaviour script can contain up to 64 SyncVars.

There is no need to keep track of SyncVar because the server automatically sends SyncVar updates when changes happen. (Mirror, 2021g)

4.3.2 SyncList and SyncDictionary

SyncLists are array-based lists. They work their contents are received from the server first, and then to sync them to the clients.

Any supported mirror type may be contained in a SyncList.

Once a SyncList field has been added to a NetworkBehaviour class, the developer can notice any changes in the client or server on the SyncList. This helps revive a character when the developer adds equipment or figures when a database needs to be updated. (Mirror, 2021f)

An unordered list of key and value pairs is contained in an associative array called a SyncDictionary. Any recognised mirror type can be used for keys and values. Similar to SyncLists, SyncDictionary sends changes made on the server to all clients and calls the callback when necessary. (Mirror, 2021e)

4.4 Mirror Communications

4.4.1 Commands

Player objects on the client and server receive commands from one another. The client cannot command the objects of other players since, by default, commands may only be delivered from objects that the client has authority over for security reasons. This method is used for the server to communicate with the clients. (Mirror, 2022)

4.4.2 ClientRpc Calls

Server objects make ClientRpc calls to their respective client objects. This is feasible for any server object equipped with a NetworkIdentity. Due to the server's authority, there is no need to worry about security regarding server objects that can execute such calls. Even if the local client can run in the same process as tshe server when a game is run as a host with a local client, ClientRpc calls will still be made to the local client. Therefore, the behaviours of local and remote clients are the same. (Mirror, 2022)

4.4.3 TargetRpc Calls

User code on the server invokes TargetRpc functions, which are subsequently carried out on the relevant client object on the client of the provided NetworkConnection. The RPC arguments are serialised over the network such that the client function is called with the same arguments as the server function. (Mirror, 2022)

4.4.4 Network Messages

The RPC calls and SyncVar are generally preferred. If the goal is for clients to provide messages that are not directly related to game objects, then this can be helpful.

To create serializable network message structs, one can extend the public NetworkMessage interface. (Mirror, 2021d)

5 CASE: MASTER SIMULATION UI

In this section, the implementation of authority simulated in a VR environment for Turku Game Lab at Turku University of Applied Sciences will be studied and explained. This section explains the requirements, and design choices that are used and are used to give insight into the development process.

5.1 Requirements

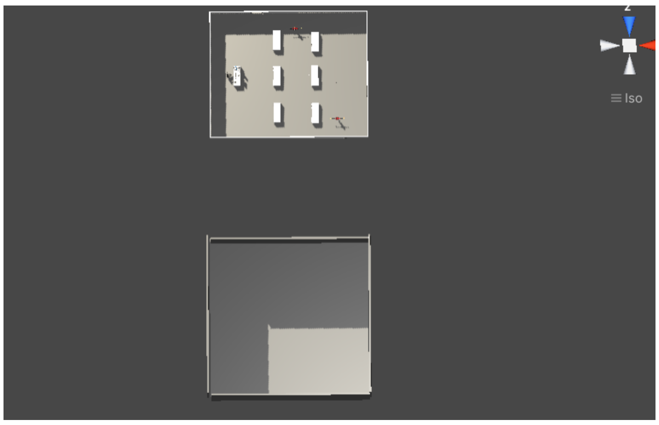
The requirements for this case study are discussed. A game scene with an external server. The Game scene had to be in VR and the controls had to be adaptable for different types of HMD. On top of that in the game scene, there was a need to distinguish a client with more authority than the other clients and simulate functionality only this client will have access. The framework that is used for the game scenes would be Unity and the network framework should be Mirror.

5.2 Game Scene

An overview regarded the details will be given first. The game must be playable with any controller and various HMDs. The objectives for the players are identical. Two player dummies move around on the field and the player is given a menu list with two different commands; one allows the user to take control of the camera. These controls are used to understand their perspective and prevent movements if needed.

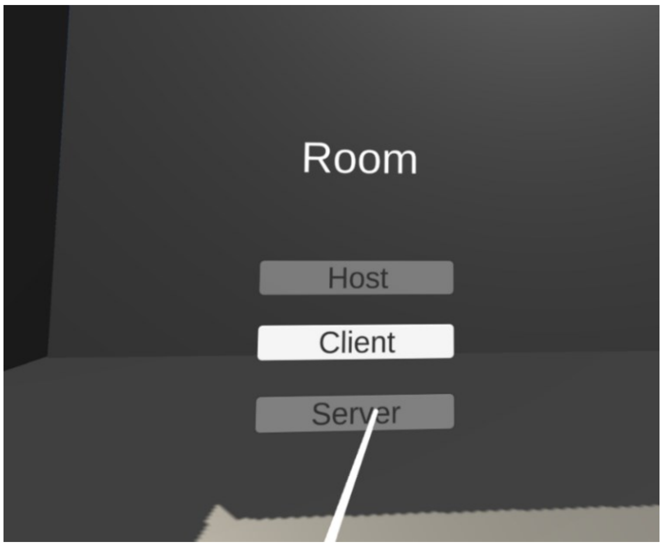
The game takes place in a classroom, with the player in a separate space and the player can choose to start a server, join a server or be a host. The Server and host options will be disabled for the player as these functionalities should be only available for the developer (Picture 1). Due to budget and time constraints, the test will be run on an external server instead of a dedicated game server.

Picture 1. Overview of the classroom and login area.



The player spawns in a room and has the option to press Join (The other options are greyed out) (Picture 2).

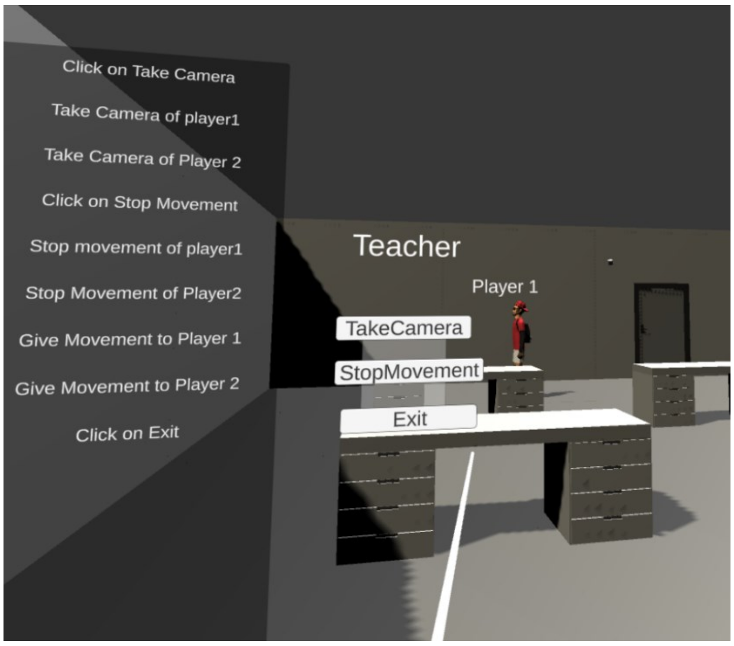
Picture 2. The login area with the host and server greyed out.



Once the player clicks on Join Game. The option to create a room will now be available and the first person who creates the room will be assigned as the host and given extra authority over that room. The extra authority comes in the form of a canvas that enables control over the dummies. This functionality is given by the server to the first one who creates the room.

When the room is created, the player starts near the door of the classroom. A set of instructions with a menu will appear (Picture 3).

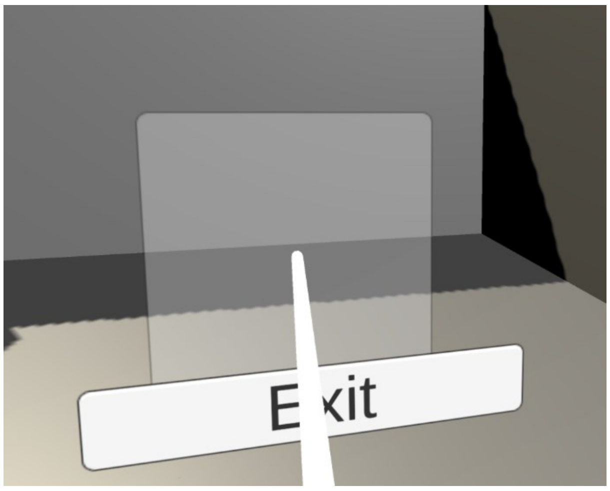
Picture 3. Inside the game room.



With a click on the commands, another menu will be opened. This menu will have the decision for which of the dummies the functionality will be applied to.

When the “Take Camera” functionality on “Player 1” is pressed, the player gets to see the camera view from the dummy with the tag player1, there will be an option to return to the previous state that floats in front of the player (Picture 4). These commands will apply to the correct target.

Picture 4. floating button to return.

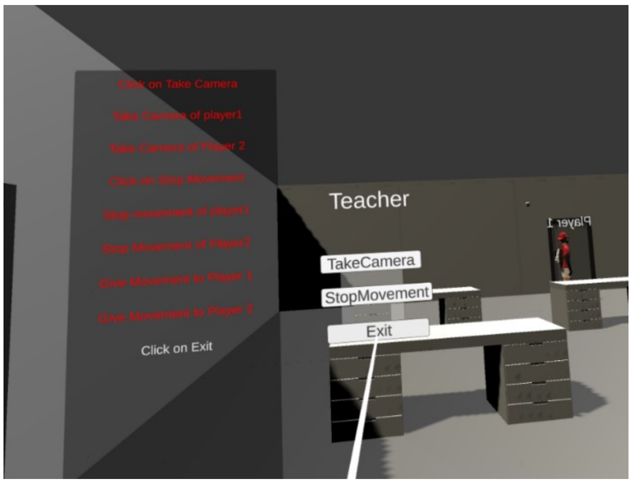


When the “Stop movement” functionality is pressed, the movement of the target that is chosen is stopped.

When pressed the button functionality will change to “Allow movement”, which enables the movement once again

And this is the result, where the player has completed all the tasks with exit left (Picture 5).

Picture 5. All tasks are completed.



5.3 NetCode

The network architecture is modified based on one of the examples scenes provided by Mirror. This is available on their GitHub under ‘Examples’, called ‘Matchnetworking’. (Mirror, n.d.)

5.3.1 Match

The case is built on top of this example because it provides the basic architecture for a network server with authority, and it is recommended to use any of their examples.

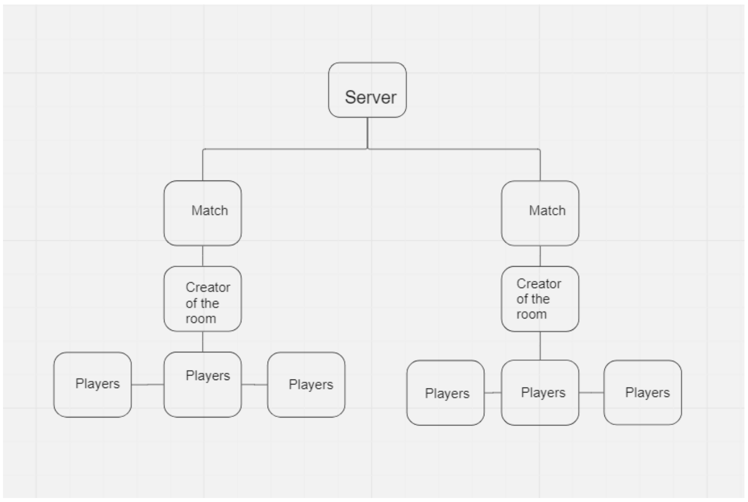
The scenes are separated from each other with the use of NetworkMatch. Each player can create their room and these rooms will be assigned a MatchId. Every object will then be spawned with the same MatchId.

The sample scene contains a CanvasController for the canvas and a MatchController which handles the connections of the Network. The MatchController oversees the state of the server while the CanvasController handles the interactions locally, these 2 will be the most crucial components for the case. When a player joins or disconnects, a network message is sent to every other client which informs them of their state (Joined, Disconnected, Started).

5.3.2 To distinguish the teacher

Another of the requirements is a method to distinguish the teacher from the other clients. Since the environment is hosted on a dedicated server, every person is a client thus there is no direct way to find the differences, every client is the same and there is a need to distinguish specific clients with more authority. For this reason, a matchmaking-based architecture is chosen. The client creates a match, and this will be sent to the server when the match is started. The server then relays this information to every other client in the form of SyncVar (Picture 6). Therefore, every other client will know locally who the teacher is.

Picture 6. Network architecture for the room.



5.4 User Interaction Canvas

5.4.1 VR Canvas

User Interaction (UI) is important to games as it enables the player to communicate with the game system and enable customization such as game settings. Unity handles this type of communication with an event system. These events can then be sent to a canvas. To display UI components, these need to be included inside a canvas. Canvas will render them; this renders depends on the mode. In Unity, there are three modes available to render canvases: screen space overlay, screen space camera, and world space. The topic will revolve around world space because the other two options are not very compatible with VR accessories. The screen space options mostly are rendered as an overlay on top of the rest. This was often too close to the eyes which would bring discomfort. Therefore, the world space option is different because it renders it as an object alongside the rest. Hence, there is a need to change the configuration to world space. A Tracked Device Graphic Raycaster will also be added because this helps the XR toolkit SDK recognize the inputs from different VR devices. (Feltham, 2021)

5.4.2 VR Interaction

For VR interaction, XR Interaction Toolkit will be used. The XR Origin(action-based) component will be the preferred object to use. The player will be built on top of this game object. The root object will be assigned a NetworkIdentity and NetworkTransform so that the position of the player can be tracked and synchronized for the other clients.

As XR Origin is action-based there is a need to set up the action-based input for Unity. This must be generated and placed in the XR Action Manager on the game scene.

5.4.3 Canvas communication

There will be a separate canvas that communicates with the Network for each client. This canvas is instantiated when the player spawns. This canvas also holds the functionality which controls the other objects. These functionalities are only available to the one who created the room. The server sends to every client the data for who is the master so that the canvas will only be visible to this person. This is done with a SyncVar so this will be the same person for every client as the data needs to be synced with the server.

With the use of the canvas from the matchmaking example, to start with, the canvas should be converted to the world space so that the canvas will be visible for the VR camera and add Tracked Device Graphics Raycaster to the canvas for the detection of the XR input system. Now there is a visible canvas for the VR environment.

Under the current architecture, the matches are being made so that messages are sent to the server and the client responds to

With the use of NetworkMatch, there is now a way to distinguish the one who started the game because only one client can start the game. This client will be assigned as the host.

This canvas will only be instantiated by the server and assigned to the local client so that they can communicate with the NetworkManager.

5.4.4 Simulating functionality

The local functionality to simulate the authority will be created. These functionalities will be linked and added as a child object of the canvas. Therefore, the networked player maintains a connection to the network and can create additional functionality with the server.

Two different functionalities will be stimulated. One of them enables the camera of another object so the user can see what the other party experiences. This can be done when the camera of another person is disabled or enabled. This will cause Unity to use the camera of the other object.

The other functionality is to take away the controls and give them back. This means that movement from the other object can be prevented. This will be done when we disable or enable their movement functionality when the buttons are pressed.

6 TESTING

In this chapter, the tester their interview answers are described and analysed. The total participation count was five.

The testers were anonymous. The location of the test was conducted at the testers' preferred location; therefore, the location was random, but the test was implemented with the given HMD with the server hosted on the same external device so there will not be much impact caused by the location or networks.

The testers were given an overview of what would happen and what the purpose of this test was. Afterwards, the controller and HMD device was handed out, and the tester entered the start scene. The testers were not able to move around the game scene, the scene only allowed the controllers to point and select. They could then create a new match and start it by themselves. Once the match started, they were be teleported to the game scene. A set of instructions popped up and the testers were left on their own devices to complete the instructions. The testers received assistance if they encountered any problems. Once they had finished the set of instructions, they were asked a set of questions which had to be answered.

7 RESULTS

This thesis project resulted in the creation of a VR application with Mirror for Turku University of Applied Sciences. The application was designed to give an insight into the development and design of a networked VR environment with some clients that have more rights than others and probe the thoughts and opinions of such functionality in a training environment. Those questions were asked in an interview while we conducted the test.

Most of the testers were between twenty and thirty years old and only two of the testers have experience with VR. As the testers were thrown into the game, three out of the five players asked for controls or more instructions on how to use the controls.

The initial lack of instructions could be improved and made clearer because of the goals and what the testers are supposed to do. Once they understood the basic controls, the rest of the steps went very smoothly. One user preferred a linear tutorial with more instructions.

Most of the problems and questions occurred at the start.

The testers seemed to like the functionality of the canvas. They expressed an interest in whether there were more various functionalities. The ability to take over the camera was the most positive function with neutral opinions on the being able to take and give controls. Albeit all testers think that the functionalities of the canvas could be a positive addition to any VR training environment. Some of the extra functionalities the testers wanted to see or add is the ability to see over the whole classroom instead of only one user at a time. The tester mentioned that this can help the teacher have a better insight into the occurrence. Another tester thought that the inclusion of eye tracking would be very beneficial and lastly there was a want for the ability to paint or write midair like a chalkboard.

The feedback about the functionality has been very positive. Four out of five testers expressed an interest in more functionalities being developed.

When asked about what could be improved, two of the testers expressed that the canvas was too far and would have liked it to be closer. Three of the testers wanted a more modern UI. These were problems more related to the game scenario as opposed to the functionality.

8 DISCUSSION

The purpose of this research was to investigate the possible implementations and probe into the opinions and confirm the need for the canvas functionality.

Based on the results of this thesis, it is evident that the objectives have been reached. A networked VR environment with functionality that simulated the host was produced and collected answers about the test environment. It would have been ideal if the functionality were more than a simulation but due to time constraints and the increased difficulty of such a project, it was not deemed plausible. This application was developed with agile methods via weekly sprints.

The environment is suitable for multiple users and the functionality can be applied used with other players. The next step would be investigating possible ways of improving the input, and the look and feel of the environment since the negative opinions were mainly regarded towards these.

Using MLAPI to build this project would be a nice alternative MLAPI, the VR support Unity provides for this framework is much better than Mirror. Mirror still consistently adds newer features and is consistently being developed, it has proven to be stable for years and so far, both would be possible to develop such an environment with. PUN would not be viable because it only supports up to 8 players unlike MLAPI and Mirror, which both support a lot of connections. MLAPI might not scale as well as Mirror with a large player count because it does not support multithreading. Which makes Mirror better for applications with a large number of connections.

Unity was a great platform to develop VR with. There were so many tutorials that covered VR for Unity. Which made it easy to set this environment up. This was not as difficult as the networked part. The process of the test proved to be slightly difficult as there would need to be both a client and a server. Both these changes need to be tracked and updated, so a great deal of time is spent to build the applications. Luckily, a great way that sped up this development was found. With the use of a tool called Parcel, Unity could be cloned and have two instances of the same environment, these two would be updated at any changes therefore, the changes made are synchronized over the cloned and the original environment.

Mirror was more difficult to work with in VR as the guidance and tutorials are slightly more limited. Most of the tutorials veer towards the older architecture, which is UNET, so some materials are outdated. In the end, VR is just a different set of input and synchronization, once that configuration is set up, the networked architecture is not very different from a regular networked game.

MLAPI is quite young since it started in 2018 and was acquired by Unity in 2020 but once the development has caught up to Mirror this could be a better alternative. As of right now, there are quite a lot of VR tutorials and guides for MLAPI but for now, it has yet to be a more stable solution than Mirror.

The sample size for the results may be a little small but the results can be used to build a better foundation, as it was intended to be a probe into the current functionalities and confirm the need for these functionalities. The results give a good insight into how these functionalities are perceived and how this project could move forward if it is to be implemented in a commercial application.

9 CONCLUSION

This chapter contains the summary, the results of the case study, the use of this application, and the potential improvement that could be made to the project.

As the technology for VR advances, the cost of HMD devices will become more affordable and commercially available. This will inevitably lead to the adaptation of virtual training. Some companies have already opted for VR training instead of physical training. As time goes by, VR training will only become more popular.

The objective of this thesis was to determine whether there is a need for canvas functionalities in a VR networked training environment. In addition to this, research for the development and design choices for implementing a VR networked environment was also performed. This environment gives the teacher extra authority.

After analysing the different network frameworks, and game engines that are currently available, the features they contain, and the differences between them, the intention was to focus more on the network architecture and less on the platform or game scene. This overview of different technologies showed that Unity is the more dominant one in the mobile VR market, but Unreal Engine has great tools for VR development also. The difference could be attributed to the entry difficulty that Unreal Engine has compared to Unity. As for network technologies, the competition is fierce with newer technologies that pop into existence as MLAPI rose in popularity after being acquired by Unity. As MLAPI is new, it has yet to prove its stability compared to the other solutions that exist. Mirror was the preferred choice for now.

The Master UI Simulation case study had many challenges due to the huge scope of the project. There were several discussions on the network architecture and the possible problems that could be encountered. Eventually, these problems were overcome, and this environment was created.

The lack of VR documentation on Mirror slowed down this process, but eventually, a VR networked environment was produced with the extra authority given to a client with a connection to an external server which acts as the host. This environment can serve to smoothen this process as the different development processes for the case study are being run through.

Despite the small sample size of testers, the opinion and feedback about this environment have been vastly positive. If one were to develop this project further, it is recommended to take notes with regard to the problems and possible features the testers have mentioned. One other comment made by the testers would be to improve the game scenes.

To reiterate, this thesis has given insight into the development and design of a VR networked environment with a case study. A VR networked environment with an external headless server with the use of Mirror was successfully developed. In addition to the insight into this process, primary qualitative interviews were also conducted with testers to probe their opinions on the functionality the canvas provides. The feedback confirmed the need for such functionality in a training environment.

A networked environment with a host that has extra authority was replicated with the use of the methodologies researched.

The app can be used as a building block to implement these functionalities, and more features could be added. It can be used as a foundation for a VR training environment.

This thesis shows the different design and development principles which could be used to find a suitable networked VR environment for the project, with a more in-depth use-case of Mirror. Mirror is a great and stable network framework.

The case study provides insight into the development of such an environment and into the potential need for functionality that enables the teacher more control on top of the possible improvements and problems this environment had. The initial feedback on this environment is very positive and confirms the need for such functionality. Many of the testers wanted to see more various functionality and improvement on the canvas. Some other testers preferred a more modern layout.

10 REFERENCES

Alsop, T., 2018. *Virtual reality (VR) - statistics & facts.* [Online] Available at: https://www.statista.com/topics/2532/virtual-reality-vr [Accessed 01 09 2021].

Alsop, T., 2021. *Issues that stop VR or AR/MR integration in a business according to XR professionals worldwide as of the 3rd quarter of 2019.* [Online] Available at: https://www.statista.com/statistics/1099157/issues-that-stop-vr-or-ar-mr-integration-in-a-business/
[Accessed 01 07 2022].

Betker, M., 2019. *Oculus Go vs Oculus Quest: an Honest Comparsion.* [Online] Available at: https://4experience.co/oculus-go-vs-quest/ [Accessed 29 11 2021].

Bonasio, A., 2019. *Throwback Thursday: The Sword of Damocles.* [Online] Available at: https://inside.com/campaigns/inside-vr-ar-2019-04-25-13690/sections/throwback-thursday-the-sword-of-damocles-104754
[Accessed 15 10 2021].

Bretan, J., 2020. *How Teachers In Poland Used Half-Life: Alyx And VR For Remote Teaching During A Global Pandemic.* [Online] Available at: https://uploadvr.com/teachers-poland-half-life-alyx-vr/ [Accessed 01 12 2021].

Brockwell, H., 2016. *Forgotten genius: the man who made a working VR machine in 1957.* [Online] Available at: https://www.techradar.com/news/wearables/forgotten-genius-the-man-who-made-a-working-vr-machine-in-1957-1318253
[Accessed 01 10 2021].

Brown, D., 2021. *Space Force is using virtual-reality headsets to train its Guardians to work on satellites.* [Online] Available at: https://www.washingtonpost.com/technology/2021/03/12/space-force-virtual-reality-training/
[Accessed 2021 03 29].

Coughlin, B., 2021. *Create a game with a listen server / host architecture.* [Online] Available at: https://docs-multiplayer.unity3d.com/netcode/current/learn/listen-server-host-architecture
[Accessed 21 10 2022].

Edgar, D., 1969. Cone of Learning. In: R. &. Winston, ed. *Audio Visual Methods.* New York: Holt, pp. 105-108.

Feltham, J., 2021. *Oculus Quest 2 Is Now The Most-Used VR Headset On Steam.* [Online]
Available at: https://uploadvr.com/oculus-quest-2-steam-most-used/
[Accessed 01 12 2021].

Gajsek, D., 2019. *Unity vs Unreal Engine - Which is Better for Virtual and Augmented Development?.* [Online]
Available at: https://circuitstream.com/blog/unity-vs-unreal/
[Accessed 23 09 2022].

Garbett, S. L., 2022. *Unreal vs. Unity for VR Game Development: Which Is Best?.* [Online]
Available at: https://www.makeuseof.com/unreal-vs-unity-vr-best-game-development/
[Accessed 30 08 2022].

George, A., 2022. *Why Unreal Engine is preferred for ...* [Online] Available at: https://timesofindia.indiatimes.com/business/india-business/why-unreal-engine-is-preferred-for-creating-aaa-games/articleshow/89163555.cms
[Accessed 19 09 2022].

Glazer, J. & Madhav, S., 2016. *Multiplayer Game Programming: Architecting Networked Games (2016).* [Online]
Available at: https://apprize.best/programming/game_3/7.html
[Accessed 25 09 2022].

House, B., 2020. *Choosing the right netcode for your game.* [Online] Available at: https://blog.unity.com/technology/choosing-the-right-netcode-for-your-game
[Accessed 15 09 2022].

Incao, J., 2018. *How VR is Transforming the Way We Train Associates.* [Online] Available at: https://corporate.walmart.com/newsroom/innovation/20180920/how-vr-is-transforming-the-way-we-train-associates
[Accessed 14 11 2021].

Järvinen, A., 2021. *The Reality Files #10.* [Online] Available at: https://medium.com/the-reality-files/the-reality-files-10-58325e3d891d
[Accessed 01 11 2021].

Lewis, R., 2018. *Father Ivan and his Children.* [Online] Available at: https://lewisartcafe.com/father-ivan-and-his-children/
[Accessed 01 10 2021].

Mirror, 2016. *Network Transform.* [Online] Available at: https://mirror-networking.gitbook.io/docs/components/network-transform
[Accessed 05 10 2022].

Mirror, 2017. *Network Identity.* [Online] Available at: https://mirror-networking.gitbook.io/docs/components/network-identity
[Accessed 04 10 2022].

Mirror, 2020. *Authority.* [Online] Available at: https://mirror-networking.gitbook.io/docs/guides/authority
[Accessed 02 10 2022].

Mirror, 2021a. *General.* [Online] Available at: https://mirror-networking.gitbook.io/docs/general
[Accessed 01 10 2022].

Mirror, 2021b. *Match.* [Online] Available at: https://mirror-networking.gitbook.io/docs/interest-management/match
[Accessed 17 10 2022].

Mirror, 2021c. *Network Manager.* [Online] Available at: https://mirror-networking.gitbook.io/docs/components/network-manager
[Accessed 03 10 2022].

Mirror, 2021d. *Network Messages.* [Online] Available at: https://mirror-networking.gitbook.io/docs/guides/communications/network-

messages
[Accessed 08 10 2022].

Mirror, 2021e. *SyncDictionary.* [Online]
Available at: https://mirror-networking.gitbook.io/docs/guides/synchronization/syncdictionary
[Accessed 07 10 2022].

Mirror, 2021f. *SyncList.* [Online]
Available at: https://mirror-networking.gitbook.io/docs/guides/synchronization/synclists
[Accessed 07 10 2022].

Mirror, 2021g. *SyncVars.* [Online]
Available at: https://mirror-networking.gitbook.io/docs/guides/synchronization/syncvars
[Accessed 06 10 2022].

Mirror, 2022. *Remote Actions.* [Online]
Available at: https://mirror-networking.gitbook.io/docs/guides/communications/remote-actions
[Accessed 08 10 2022].

Mirror, n.d. *Mirror Repo.* [Online]
Available at: https://github.com/vis2k/Mirror
[Accessed 20 10 2022].

Naszrai, A., 2018. *Virtual Reality UI with Unity.* [Online]
Available at: https://medium.com/@naszrai.andras/virtual-reality-ui-with-unity-6e3d02f671e4
[Accessed 15 10 2022].

Petrov, H., 2022. *Multiplayer in Unity with Mirror.* [Online]
Available at: https://www.prettyneat.io/welcome-to-mirror
[Accessed 27 09 2022].

PricewaterhouseCoopers, 2021. *How Virtual Reality Is Redefining Soft Skills Training.* [Online]
Available at: https://www.pwc.com/us/en/tech-effect/emerging-tech/virtual-reality-study.html
[Accessed 12 12 2021].

Rubin, P., 2014. *The Inside Story of Oculus Rift and How Virtual Reality Became Reality.* [Online]
Available at: https://www.wired.com/2014/05/oculus-rift-4/
[Accessed 29 11 2021].

Turi, J., 2013. *Time Machines: NASA goes virtual at CES.* [Online]
Available at: https://www.engadget.com/2013-12-15-time-machines.htmlachines.html
[Accessed 29 11 2021].

Wong, R., 2018. *Oculus Go review: This is the iPhone of VR headsets.* [Online]
Available at: https://mashable.com/article/oculus-go-review
[Accessed 2021 12 12].

XR Guru, 2021. *Top 5 Challenges in Implementing VR in Schools.* [Online]
Available at: https://www.xrguru.com/blog/2021/08/top-5-challenges-in-implementing-vr-in-schools
[Accessed 07 07 2022].