COMP4801 FINAL YEAR PROJECT

Interim Report

Topic: A multi-platform multiplayer game

Chan Ting Lok, Leon (3035574763)

Supervisor: Dr. Chim, T.W.

Submitted on 21/1/2022

Abstract

With the advancement of Virtual Reality (VR) technology, it is forecasted that the VR market size in 2024 will triple that of 2019. This project aims at developing a game that supports VR, mobile, and computer concurrently with a single codebase. With broader hardware coverage, not only more players can enjoy the game, but also the game revenue will increase drastically.

The final deliverable of this final year project will be a cross-platform multiplayer cooperative game that supports VR Devices (Oculus Quest 2), computers (Windows machines), and mobile phones (Android). **Unity** is chosen as the game engine for the project because it can deploy the game on more than 20 platforms and **Photon** is the chosen multiplayer engine because of its compatibility with **Unity**. Currently, 2 puzzle levels have been implemented and the boss fight level development is still in progress. The project will soon be moving into the VR control integration stage. With limited budget and time, it is impossible to support all kinds of gaming consoles/hardware and produce a standard-length, polished game. A demo-size game should be expected instead.

Acknowledgement

Firstly, I would like to express my deepest thanks to Project Supervisor, Dr. T.W. Chim, Lecturer, Department for Computer Science, The University of Hong Kong, for giving me opinion and comment when I am working on this final year project topic. When I have any inquiry, he always replies to my email swiftly and provides me with professional support. Secondly, I am grateful to Dr. Locky Law, Assistant Lecturer, Centre of Applied English Studies, The University of Hong Kong, for helping me in the aspect of technical English. He is always willing to lend a helping hand to me when I need help with the preparation of project documents/presentations.

Table of Content

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Meaning
AR	Augmented Reality
VR	Virtual Reality
MR	Mixed Reality
PC	Personal Computer
USD	United States Dollar
FYP	Final Year Project
PUN	Photon Unity Networking
CCU	Concurrent Users
3D	Three Dimensional
RPC	Remote Procedure Calls
UI	User Interface

1. Introduction

1.1. Overview

The gaming industry is advancing. The market will eventually be saturated with the traditional way of gaming. The game developers are, therefore, trying to integrate and develop new technology in their game; for instance, motion tracking, augmented reality (AR), virtual reality (VR), mixed reality (MR), etc. Consequently, there are more and more devices for gaming, for example, personal computer (PC), mobile, and VR headset.

However, there is a limited number of games that are compatible with PC, mobile, and VR headsets concurrently. For a game that can support more platforms, not only more players can enjoy it, but the revenue for the game will also be drastically increased (refer to Section 2). The goal of this project is to investigate whether it is possible to develop a game that supports multiplatform to provide players with a similar gaming experience. Therefore, a multiplatform, multiplayer game is the targeted deliverable of this project.

1.2. Motives

This section will cover the benefits for different parties concerned by this project. In section 1.2.1, the benefit for game developers will be explained. In section 1.2.2, the benefit for players will be discussed.

1.2.1. For Game Developer

From the game developer's perspective, the market size is positively related to the revenue. The broader the hardware coverage, the more profitable the game will be. But at the same time, covering more platforms by multiple separate codebases will drastically increase the developmental time and cost. Apart from that, if there are any additional updates for the game, the developers have to implement them in multiple codebases, it will be extremely hard to maintain. Supporting a cross-platform game with a single codebase will make the game more maintainable and economical.

1.2.2. For Player

Different players have different gaming preferences, for example, some prefer the convenience and choose the mobile platform, while others prefer immersive gaming experience and choose virtual reality. A cross-platform game can provide players with a similar gaming experience with different types of hardware.

In terms of game design, the game is designed in a way that one single player cannot do all the work by himself, the player must learn to rely on another partner, cooperation is the only way to ace the game. On the cooperative aspect, players will learn how to cooperate with their partners and communicate efficiently to exchange information; On the puzzle-solving aspect, the game can improve players' creativity, logical thinking (for example learning by observing patterns), and problem-solving skills.

1.3. Outline

This project plan is organised in the following way. In section 2, the background of the project will be introduced. Followed by section 3, the objective and scope will be brought out. After confirming the scope, in section 4, detailed methodology and relevant development tools will be laid down. In section 5, the current status of the project will be reported. Then, the implementation result will be discussed in section 6. After that, the future work will be covered in section 7. And finally, a summary of the main idea will be given in section 8).

2. Background

An overall picture of the traditional gaming market and VR gaming market will be shown in this section. The revenue data of the traditional gaming market (section 2.1) and virtual reality gaming (section 2.2) market will be analysed in the following section.

2.1. Traditional Gaming Market

Mobile, Console (e.g., PlayStations, Nintendo Switch, Xbox, etc), and PC are still the major mediums people use to play video games. It is observed that mobile platform is dominating the gaming market (generating more than 90 billion USD in 2021), followed by console and PC (generating 49.2 billion USD and 35.9 billion USD respectively in 2019) (see Figure 1) [2]. It is obvious that a game will generate much more revenue if it supports mobile, console, and PC concurrently. It gives grounds for why the game should be supporting both keyboard and mouse, game controller, and touchable monitor as input.

2.2. VR Gaming Market

Virtual Reality (VR) is defined as "the use of computer modeling and simulation that enables a person to interact with an artificial three-dimensional (3-D) visual or other sensory environments" [1]. It is a new trend for gaming. VR gaming is becoming more and more popular because of its intuitive control and immersive experience. It is forecasted that the VR market size (2.4 billion USD) in 2024 will triple that (0.8 billion USD) of 2019 (see Figure 2) [3]. The VR gaming market is expanding rapidly. In a not-too-distant future, VR will become a popular platform for gaming. This growing trend justified why it is important to support VR devices in this project.

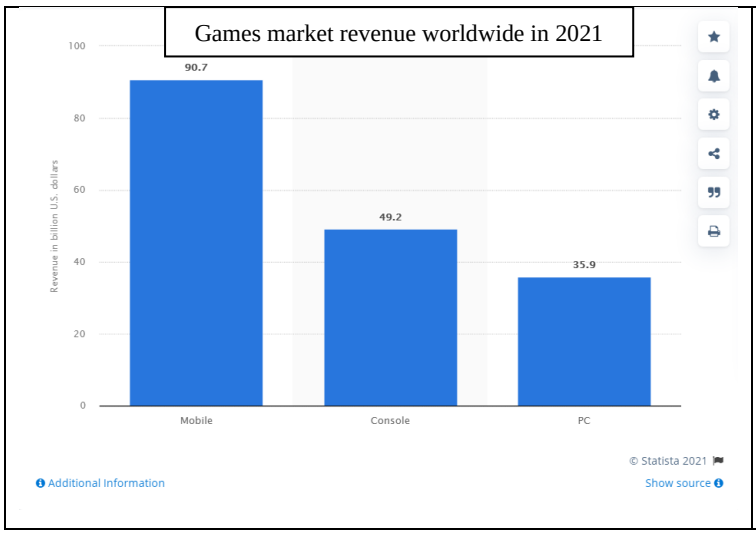

Figure 1: Games market revenue worldwide in 2021 [2]. It is concluded that mobile, console, and PC are still the major platforms for gamers to play games by comparing the revenue with Figure 2.	*Figure 2: Virtual reality (VR) gaming revenue worldwide from 2017 to 2024 [3]. It is observed that the VR gaming revenue is growing drastically over the years.*

3. Objective and Scope

In this section, the objective will be explained in section 3.1; Followed by a big picture of what will eventually be delivered in section 3.2.

3.1. Objective

This project aims at investigating the possibility to support multiple traditional gaming platforms and VR gaming with a single codebase. Using a single codebase instead of three sets of codebases to support three platforms can consequentially reduce the developmental cost and time.

3.2. Scope and Deliverable

A cross-platform, multiplayer game will be delivered at the end of the project. It is a compulsory 2-player cooperative game containing three levels - two puzzle-solving levels, and one boss fight level. Players can choose to play the game with various platforms. PC (Windows), Mobile (Android), and VR (Oculus Quest 2) will be supported. To solve the puzzle, players need to figure out how to solve the puzzle and perform certain actions at the right timing, communication is thus very important. In the boss fight level, players will be given different abilities. To beat the boss, both abilities are needed, cooperation thus plays a vital role in this level. To facilitate communication and cooperation, a built-in voice chat system will be available in the game.

4. Methodology

This section will be divided into three parts. The hardware setup and targeted platform will be detailed in section 4.1. The software aspect will be covered in section 4.2, justification will be provided for the development tools chosen. The programming design philosophy will be followed in section 4.3.

4.1. Hardware and targeted platform

Different VR headsets offer different functions, for example, some of them support hand tracking while others do not. Thus, a single set of code will not be compatible with all VR headsets. Oculus device is chosen for this project because it is the most popular VR headset player chosen. As of January 2021, around half of the gamers on Steam (A popular video game digital distributing platform) choose Oculus devices as their VR device [4]. I choose Oculus Quest 2 (a model of Oculus device) as a result of budget concerns. For the mobile platform, I decided to support Android because of its popularity and operating system openness. In 2021 more than 70% of mobile devices are Android devices [5]. As mentioned in the part of the motives (section 1.2.1), market size is one of my concerns, Android is still the most dominating the mobile market. In addition, building the game on an iOS device requires a Mac Machine, which will further complex the hardware requirement. Not supporting iOS devices will not make a negative impact on the FYP result, because the input of Android and iOS devices are similar (touchable monitor). For the PC platform, windows will be supported. macOS is seldom a choice for gaming, it is, therefore, removed from the scope of this project.

4.2. Software

The following part will be covering two essential development tools for a multiplayer game – game engine (section 4.2.1) and multiplayer engine (section 4.2.2). To explain analogically, if the final game is a dish, the game engine is the basic set of cooking utilities to cook the dish. Some dishes (games) may have some special features like baked (multiplayer). We need an oven to cook it (multiplayer engine).

4.2.1. Game Engine

Considering the project aims at supporting multi-platforms, ***Unity*** will be the best choice. With ***Unity***, a single code base can be deployed on more than 20 platforms [6]. Apart from that, ***Unity*** offers excellent integration with VR technology. In addition, the developer community is large, and its code is comprehensively documented; In case of facing any technical difficulties, there are more resources

available on the Internet. Windows is selected as the development platform for consistency and better optimization (see section 4.1 for justification). For programming language, C# is the only language that is supported by ***Unity***. It is an object-oriented scripting language.

4.2.2. Multiplayer and built-in voice chat system

Photon Engine, which includes ***Photon Unity Networking 2*** (for multiplayer) and ***Photon Voice 2*** (for the built-in voice chat system), is chosen. Its excellent integration with ***Unity*** can smoothen the development and prevent optimization problems [7]. Apart from that, Photon is free of charge for 20 concurrent users (CCU), which is good enough for testing a small-scale multiplayer game like this [7]. In addition, Photon will host the server for the game, extra time, and cost on the maintenance of the server computer can be saved. To explain it with an analogy, to offer services (multiplayer services) for some customers (players' computers), you will need a customer service officer (host computer). Normally, you will need to hire one by yourself; With ***Photon***, the customer service officer will be hired and managed by ***Photon***.

4.3. Programming Design

The main difficulty of the implementation comes from supporting different types of hardware. For one action, different types of input need to be handled and interpreted. Take picking up an object as an example, for VR players, they may directly use their hand to grab the object; for PC players, they may use their keyboard and press a certain key to act; for mobile players, maybe they will use their finger to tap a virtual button on the touchable screen. The main programming design philosophy is to reuse code as much as possible. I will try to achieve it by creating some generic functions, like in this example PickUpObject(Object) and this will be called by PC, VR, and mobile device. I will avoid creating three sets of code for three kinds of input.

5. Current Progress

The game development can be divided into three stages. Stage 1 focuses on the general game development in the PC platform (supporting both keyboard and mouse control and gamepad control); Stage 2 targets VR control Integration; Stage 3 puts emphasis on supporting the mobile platform. Please refer to Table 1 in the Appendix for a detailed timeline.

Currently, the development is almost at the end of Stage 1. As mentioned in Section 3.2, the game consists of three levels, two puzzle levels, and one boss fight level. Puzzle level 1 is covered in section 5.1, puzzle level 2 is covered in section 5.2. The boss fight level is almost finished, it will be covered in section 7.

5.1. Puzzle Level 1 - Toggle Plane

The game design of the toggle plane level is fairly simple. It will act as a tutorial level for players. The floor plan is attached below (see Figure 3). The goal of this level is to get to platform B from platform A (starting point). In the floor plan, red and blue rectangles are used to represent the corresponding toggleable planes. On each platform, there is a toggle button that can toggle the plane's state. If the planes are in the red state, only red planes will be shown and vice versa (see Figure 4 and Figure 5). Please go to https://wp.cs.hku.hk/2021/fyp21057/2021/10/04/progress/ for video demonstrations.

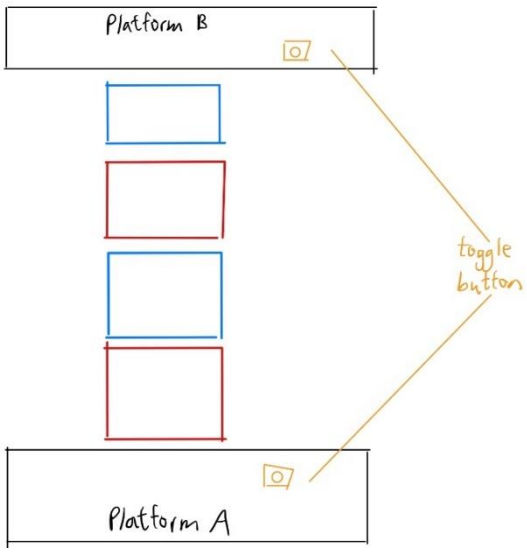

Figure 3: Floor plan of puzzle level 1 – toggle plane

Figure 4: Player screenshot showing the red state of the toggle planes.

Figure 5: Player screenshot showing the blue state of the toggle planes.

The solution of this level is that player A controls the toggle button, and toggles the planes at the right time (click the button when player B jumped and is in the mid-air between plane) to help player B to get to the platform B. And then player B uses the toggle button on platform B to do the same thing for player A.

5.2. Puzzle Level 2 – Conductor

The floor plan is attached below for reference (see Figure 6). The goal of this level is to transmit a stream of magical energy from the start point to the endpoint. The orange line is used to represent conductors and there are gaps between conductors (labelled by number in the figure), please refer to Figure 7 for the actual setting of the game. When the player steps on the pressure sensor, a stream of magical energy will be transmitted from the start point. And when the energy reaches any unhandled gap, the energy will vanish. Therefore, players need to grip both ends of the conductor to close the gap of the conductor and after the energy pass through the player's body, the player will be paralysed (i.e., they cannot move for a couple of seconds) (see Figure 8). Therefore, if a player closed gap 1, it is impossible for that player to reach gap 2 on time. The solution of this level is to handle gaps and pressure sensor alternately by two players.

Please refer to https://wp.cs.hku.hk/2021/fyp21057/2021/10/04/progress/ for video demonstrations.

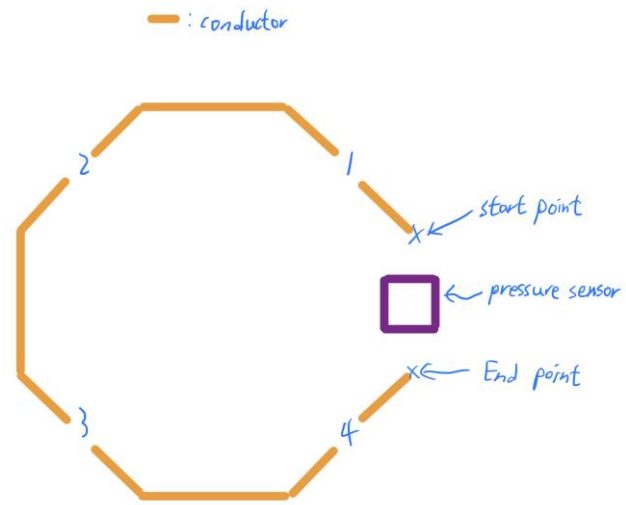

Figure 6: Floor plan of puzzle level 2 – conductor.

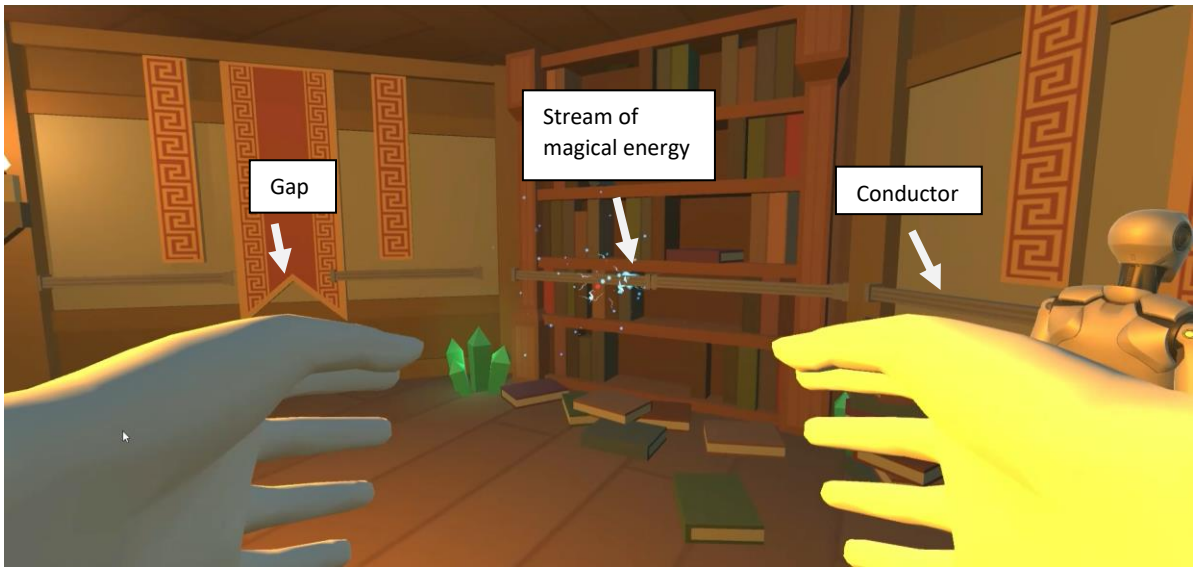

Figure 7: Screenshot demonstrating the setting of puzzle level 2 – conductor

Figure 8: Screenshot demonstrating player close the gap by gripping both ends of the conductors. The lightning particle in the centre represents the player is paralysed.

6. Discussion of Result

In section 6.1, some technical findings will be discussed. One limitation will be pointed out in section 6.2.

6.1. Finding and Discussion

This part covers the two most important findings of this project. Firstly, section 6.1.1 discusses how to support different types of input devices with unity. Secondly, section 6.1.2 covers how to synchronize the players.

6.1.1. Supporting Different Types of Input devices

In recent years, Unity introduced a new input system. In this section, a brief comparison between the two systems will be given and justifications of why the new input system is employed will be talked about.

In the old input system (see Figure 9), developers have to check the player's input in the update function (it is a function that Unity will call in each frame) if a certain key is pressed, perform a certain action. There are some problems with this approach. Firstly, unity will check player input every frame which is inefficient and wastes computing power. [8] Secondly, players usually have many input actions, making the update function massive. [8] As a result, it will be hard to maintain and debug. Thirdly, the device compatibility is low. You have to manually add more conditions to check for additional devices input. [8]

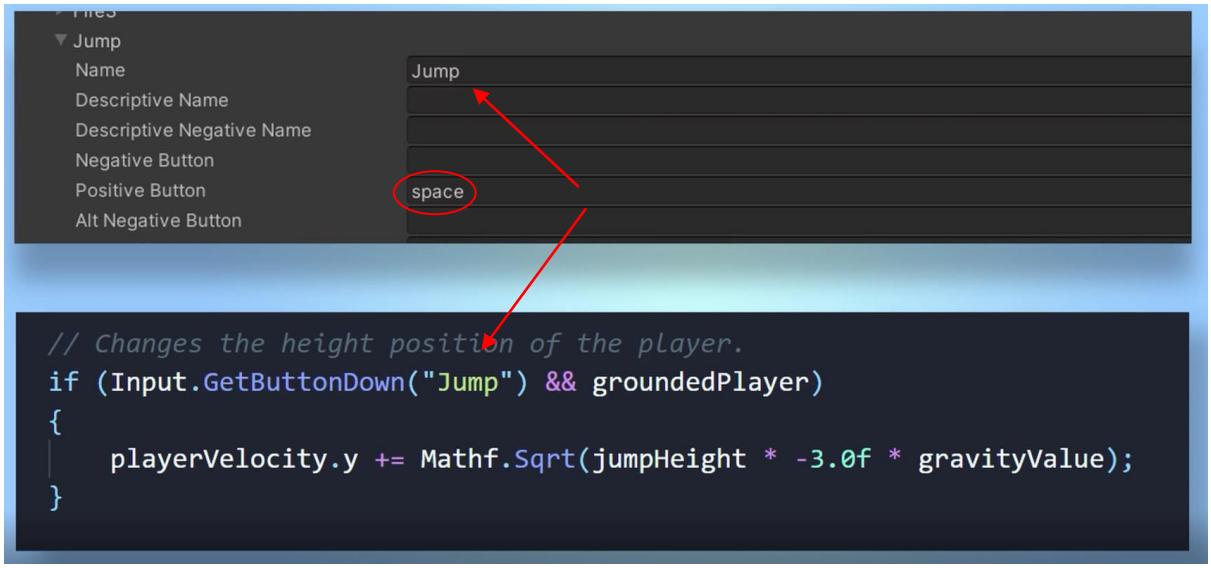

Figure 9: demonstration of unity old input system. [8]

The new input system fixed most of the problems mentioned above. Unity introduced a type of asset called input actions (see Figure 10). Action maps can be thought of as a container of actions. Different actions can be mapped to an infinite number of buttons/input keys. For example, the "Jump" action in Figure 10, can be triggered by the space bar on the keyboard or the button south on a gamepad.

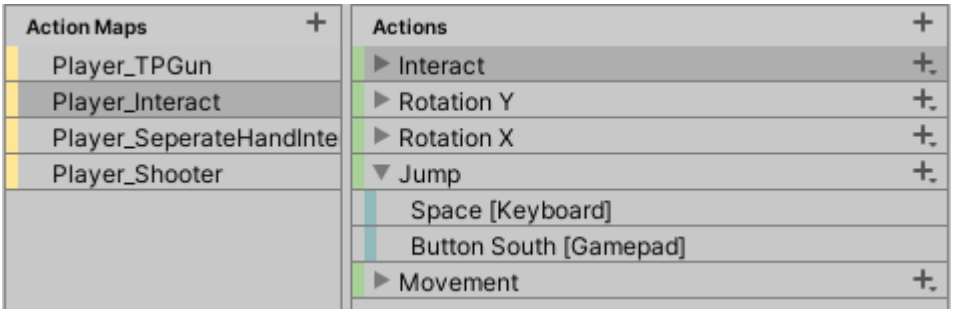

Figure 10: Input actions in Unity.

And when jump action is triggered, "Jump" function in "PC First Person Player (Base)" script will be fired (see Figure 11). The new input system increases the code maintainability by separating device and input. Also, it promotes device compatibility because developers can map an action with an infinite number of input keys coming from different input devices.

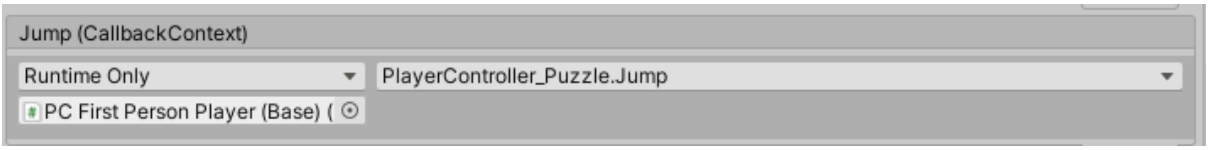

Figure 11: Player Input component in Unity.

6.1.2. Multiplayer Implementation – Synchronization

Photon provides two types of options for synchronization. In this section, both of them will be introduced, followed by how I implemented the synchronization in the puzzle levels.

The first option is photon view, it is a type of continuous synchronization. One of the examples is photon transform view (see Figure 12). With photon transform view put in the game object, photon will sync the checked parameters (position and rotation in Figure 12) continuously [9]. Photon transform view is just one of the examples, there are also other photon views that sync other types of parameters. Developers can also define their own parameters to sync. The advantage of this option is that it is easy to implement and maintain (all troubles are left to photon to handle for us). However, because of the continuous data transfer between peers and master, it will create a great burden to the network if it is used for everything.

Figure 12: Photon Transform View component in Unity.

The second option is Remote Procedure Call (RPC). RPC allows developers to fire functions in their peers [10]. Take how I synchronize the plane state in puzzle level 1 – toggle plane (refer to section 5.1) as an example (see Figure 13). "photonView.RPC()" takes three parameters. The first parameter (ChangePlaneState) is the name of the function to call in other peers; The second parameter is the target of the function call, RpcTarget.All means to call this function in all targets (including itself), there are also other options like RpcTarget.others (call the function in all targets except itself). The third parameter (true) is the parameter input for the function call [10]. In this case, I pass a true to all the targets in the ChangePlaneState() function call. This downside of this method is that relative to photon view it is harder to implement (because the logic is more complicated to manage). However, thanks to the on-demand usage of network resources, it is more network efficient.

```
public void Trigger_ChangePlaneState()
{
    photonView.RPC("ChangePlaneState", RpcTarget.All, true);
}

[PunRPC]
2 references
public void ChangePlaneState(bool toggle = true)...
```

Figure 13: code segment for synchronization of plane state in puzzle level 1 – toggle plane

To enhance network efficiency, my rationale is to use continuous synchronization as little as possible. I will use it only if I need to synchronize something continuously. In puzzle level implementation, I use photon transform view to synchronize players' positions. And for all other things in the scene (like plane state and conductor gap state), I use RPC to synchronise.

6.2. Limitation – Action Map Inheritance

For the action map, I want a feature to support action map inheritance. For a game containing different levels, in each level players may have different sets of actions, but maybe some actions are commonly used and shared in many action maps. For example in this project, in both puzzle level and boss fight level, players need to move around, jump, and look around (see Figure 14 and Figure 15). If

I can create a new action map that inherits a “base action map” containing these actions, I can simply add new specific action for that level (for example shoot action in the boss level is not needed for puzzle level).

Without inheritance, maintainability is reduced. For example, if I want to change the button map to the “Jump” action, I have to go through all action maps and change each of them. However, after the investigation in the documentation, action map inheritance is not supported by Unity now. As a result, I have no choice but to copy all basic actions to all the action maps.

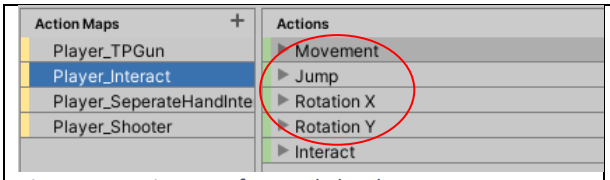

Figure 14: action map for puzzle level

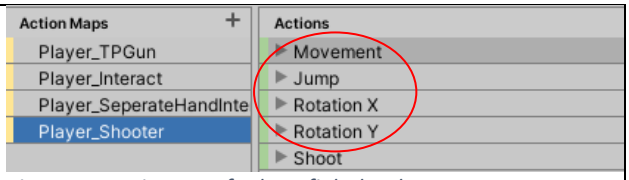

Figure 15: action map for boss fight level

7. Future Work – Boss Fight level

After implementing puzzle levels, the next step will be the implementation of the boss fight level. In fact, the boss fight level is almost finished by the time I am writing this report (see Figure 16 for sample user interface (UI)). Only synchronization is left for boss fight level.

The goal of the boss fight level is to beat a dragon by destroying all the weak spots of the dragon. However, the weak spots are hidden. Players will be given a different weapon.

One player will get a wand, which deals no damage, however, it can reveal the weak spots (see Figure 16).

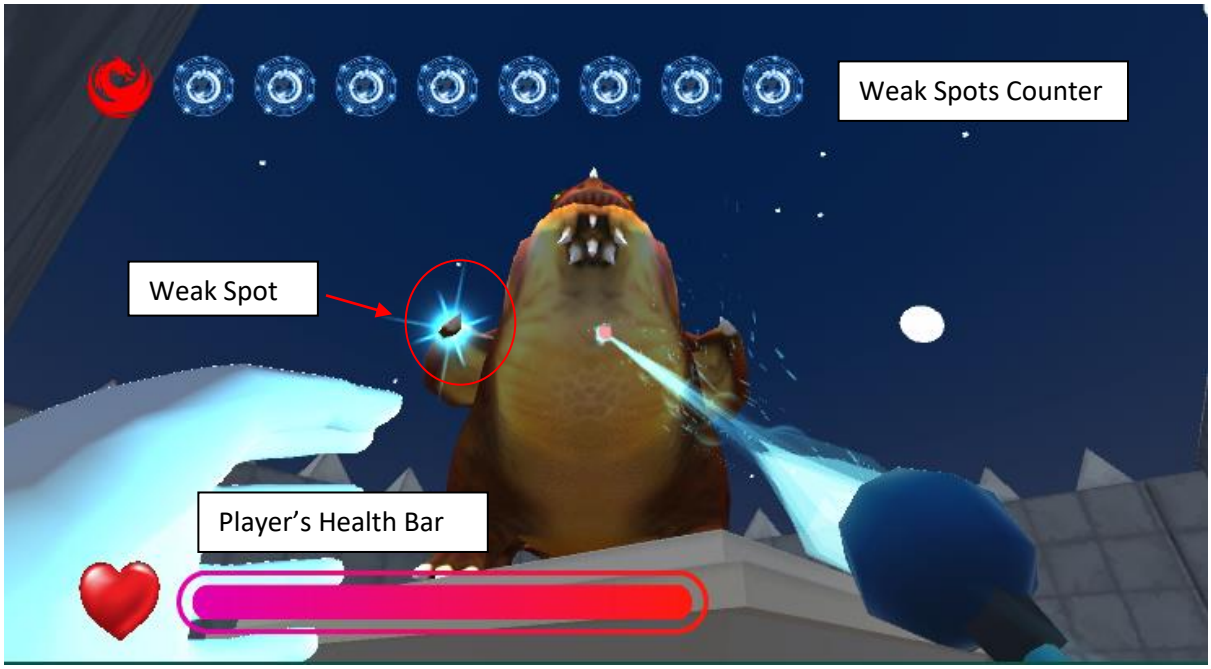

Figure 16: Screenshot of player using wand.

The other player will be given a magical gun, which deals damage to the dragon if and only if it hits the spots of the dragon (see Figure 17).

In terms of game design, both players rely on each other in order to beat the dragon. When all 8 weak spots are destroyed, the players win. Cooperation is the only way to defeat the dragon.

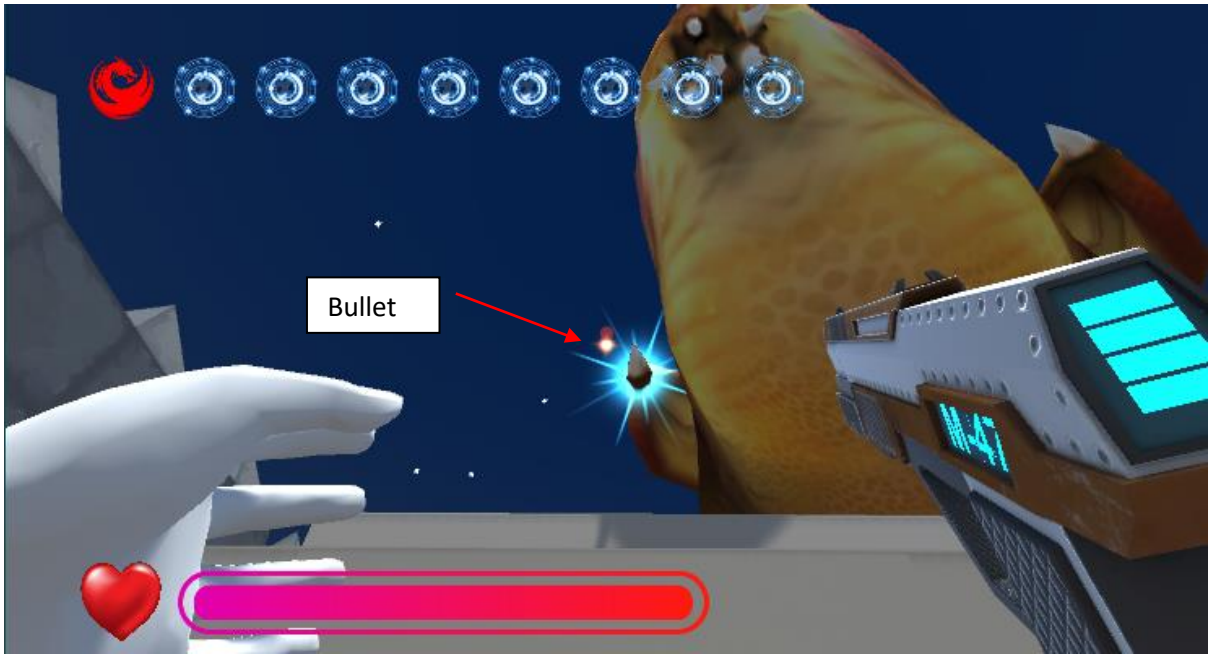

Figure 17: Screenshot of player using a magical gun

8. Conclusion

Given the rapid expansion of the VR gaming market, it is profitable to increase the hardware coverage and player base of a game. This project will be valuable to the gaming industry as a demonstration. Therefore, the goal of the project is to develop a game that supports cross-platform multiplayer. To achieve this without increasing too much developmental cost, this project will try to support multiple platforms by one generic code base that can handle different kinds of inputs from the touchable screen, keyboard and mouse, gamepad, and VR controller. For now, the general game implementation is almost finished. It is foreseeable that the project will be transited to the VR control integration stage soon.

However, some limitations prevent the project from achieving the goal perfectly. In the market, there are too many gaming platforms. It is impossible to support all these platforms for two reasons. Firstly, to test my game on all platforms, all kinds of hardware are needed, for example, Nintendo Switch, PlayStation Machine, Xbox Machine, Mac Machine, etc. It is impossible to purchase these gaming consoles and devices with $1000 (FYP budget). Secondly, the time is limited, different hardware has its interface (analogy: like different adaptors, use different ports), it is infeasible to support all of them within seven months. Although this project cannot support all gaming platforms, this project does support the gamepad which is commonly used for different gaming platforms, making the project sufficient to prove the feasibility to support all platforms if time is allowed.

In addition, a polished game should not be expected. In normal game development, there will be at least two roles – game programmer and artist. Without proper art support and sufficient time, a demo-sized unpolished game will be developed as a result.

Reference

[1] Britannica,“Virtual reality,” *Encyclopædia Britannica*. [Online]. Available: https://www.britannica.com/technology/virtual-reality. [Accessed: 26-Oct-2021].

[2] J. Clement, “Video game, gaming industry revenue,” *Statista*, 01-Jun-2021. [Online]. Available: https://www.statista.com/statistics/278181/global-gaming-market-revenue-device/. [Accessed: 24-Oct-2021].

[3] J. Clement, “Global VR Gaming Market Size 2024,” *Statista*, 29-Jan-2021. [Online]. Available: https://www.statista.com/statistics/499714/global-virtual-reality-gaming-sales-revenue/. [Accessed: 24-Oct-2021].

[4] N. Gilbert, “74 virtual reality statistics you must know in 2021/2022: Adoption, usage & market share,” *Financesonline.com*, 06-Apr-2021. [Online]. Available: https://financesonline.com/virtual-reality-statistics/. [Accessed: 24-Oct-2021].

[5] GlobalStats , “Mobile Operating System Market Share Worldwide,” *StatCounter Global Stats*. [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide. [Accessed: 24-Oct-2021].

[6] Unity Technologies, “Multiplatform,” *Unity*. [Online]. Available: https://unity.com/features/multiplatform. [Accessed: 24-Oct-2021].

[7] Photon, “Pun,” *Photon Unity 3D Networking Framework SDKs and Game Backend | Photon Engine*. [Online]. Available: https://www.photonengine.com/pun. [Accessed: 24-Oct-2021].

[8] samyam. "Why You Should Use The New Input System In Unity + Overview," *YouTube*, Feb 28, 2021. [Video file]. Available: https://www.youtube.com/watch?v=GyKBoDF_Oxo. [Accessed: Dec 15, 2021].

[9] Photon, “Synchronization and State,” *Synchronization and State | Photon Engine*. [Online]. Available: https://doc.photonengine.com/en-us/pun/current/gameplay/synchronization-and-state#object_synchronization. [Accessed: 21-Jan-2022].

[10] Photon, “RPCs and Raiseevent,” *RPCs and RaiseEvent | Photon Engine*. [Online]. Available: https://doc.photonengine.com/en-us/pun/current/gameplay/rpcsandraiseevent. [Accessed: 21-Jan-2022].

Appendix

Detail Timeline for this project

Date	Task
Summer 2021	Choose/Draft FYP Topic
15 Sept 2021	Technical Investigation • VR Development • Cross-platform Multiplayer • Built-in Voice Chat
3 October 2021	Deliverables of Phase 1 (Inception) • Detailed project plan • Project web page
15 October 2021	Finalize game design
31 December 2021	Puzzle Level 1 and 2 Implementations
10-14 January 2022	First Presentation
23 January 2022	Deliverables of Phase 2 (Elaboration) • Preliminary implementation • Detailed interim report
31 January 2022	Boss Fight Level Implementation
28 February 2022	VR Control Integration
31 March 2022	Touchable Screen (Mobile) Control Integration
15 April 2022	Final Test and Adjustment
18 April 2022	Deliverables of Phase 3 (Construction) • Finalized tested implementation • Final report
19-22 Aril 2022	Final presentation
4 May 2022	Project exhibition
31 May 2022	Project competition (for selected projects only)

Table 1: Timeline for this FYP