An FPS Game Based on Unity Game Engine

Xuemeng Li
2018963
supervisor: Paul Levy
word count: 8491

Abstract

The shooting games that currently exist in the game market now are mostly team-based. For players who occasionally can’t find someone to play games with and players who just want to relax a little by themselves, there are fewer options. Also, because some of them are online games, these games pay more attention to the fighting between players instead of implementing the AI part.

Based on the Unity game engine, this project has produced an FPS(first personal shooting) game with three levels of difficulty, which not only includes basic differently dynamic maps and built-in props but also includes the monster and boss, which are generated by some AI algorithms. So that they can automatically find paths and attack players. It also has a system that determines the player's passing judgement through time calculation, etc. This project hopes to make players feel the charm and fun of FPS games and fight against AI even in a short period. The game is running well.

Contents

Chapter 1

Introduction

In contemporary life, games have become a major choice for relaxation and recreation. Even in childhood, puzzle games are recommended, and as young people and adults, games occupy a large part of life. After a busy working time, everyone only needs to turn on the computer or mobile phone, enter the game interface, and then go into the virtual world of the game. In the game world, people do not need to worry about anything else, just need to devote all their senses to enjoy the excitement brought by the game. After winning the game, players will also feel great joy and relaxation. Now the various game devices on the market have also proved their popularity in the game market. PS series, switch series, Xbox series, as long as these products are launched, they will be sought after by everyone. Even more so with PC gaming, where you can do everything with just one computer.

The embodiment of the game is mainly divided into 2-Dimensions (2D) and 3-Dimensions (3D); these are two fields. The main difference between them is the way the game screen is presented. Compared with 2D, 3D is more impactful and shocking. The same is true for visual feelings. In terms of technology, to display images in 2D, the mainstream approach is to put pre-drawn images into files and call them in the game. In 3D games, each object is regarded as a three-dimensional object, which is composed of several aggregated polygons. Specific and detailed descriptive statements must be stored in the program file, and an object can be synthesised in real-time when displayed, and through several real-time calculations of solid geometry and plane formulas, which shows an arbitrary angle picture of the object. An experiment(*H. R. Khairuddin et al. 2013*) that used an EGG machine has proved that 3D has further increased signal complexity than 2D[1].

Game development is also part of computer science and developers can make it in any language of choice. However, after making a 2D game in Java in the second year, when people tried that game, there was not as much satisfactory feedback as expected. After summarising, it may be due to the choice of game types and the nature of the 2D game itself. The drawback, as mentioned above, doesn't make the visuals of the UI interesting, so there is not any immersive gaming experience. Of course, there are solutions to these, such as changing a character from walking in four directions to when walking in eight directions, the developers only need to add related image resources and related animation scenes for walking in the other four directions, but 2D game manuscripts are generally derived from hand-painted manuscripts, which undoubtedly greatly increases the workload and increase the art cost consumption.

For a shooting game, even with a huge amount of action graphics, its shortcomings are obvious. It is a degree of freedom. The player can only move in a limited direction to complete the mission. Popular shooting games currently (CSGO, Overwatch, Pubg, etc.) are all 3D games. Among them, it can be subdivided into the first perspective and the third perspective. Inside the game, the most obvious difference between the two is whether the player's body can be seen. The first perspective, as the name suggests, is the same as what

the human eye sees, it will not contain the body of players so that everyone can focus more on the enemy detected.

The games mentioned above are all online games, which are battles between players. However, if at the same time, no friends are available or the number of people online at the same time does not meet the requirement to start the game, the waiting time for players will become very long. At this time AI monsters will be a good substitute. They will have a different random route each time they start, and they will also automatically pursue and attack after being attacked or when the player approaches. And they will have a certain degree of difficulty, instead of "scarecrows" who will only be killed standing up. These algorithmic actions can satisfy the player's needs for a shooting game and generate the idea of multiple attempts to achieve the best score.

In addition, for large-scale games, if players want to experience the entire game well, there will be requirements for the computer graphics configuration. So how can you start a simple game in a short time, and at the same time, you can enjoy the sense of movement brought by 3D? What if no one else is online at the same time, and you want to play a game to relax or improve your shooting accuracy? How to enjoy the fun of games with only a notebook with ordinary graphics configuration? This project has designed a 3D fps game with three levels of difficulty, not with other players but with AI monsters, which can run on multiple platforms. Players can interact with the computer through the keyboard and mouse, and perform various physical attacks by changing the player's state, attack or being attacked by collision until one of the AI or the player dies.

This project has a complete game system, which is opened with a short story, has a login registration system, the main menu, various sound effects, etc., so that players can enjoy the joy and excitement of the fps shooting game even when they are alone.

1.1 Unity

The Unity game engine has perfect technology and personalised functions, not only can be used by professionals but is also friendly to novices and even amateurs. Unity uses C# or JS as the scripting language, focusing on the development of 3D and 2D games and interactive content. It has the advantages of being free, cross-platform [2], and also outstanding in terrain editor and physical effects. Additionally, the Unity engine provides high-quality visuals and audio. Visuals developed in Unity adapt to each device and screen without compromising image quality or any impact on image quality.

1.2 C#

C# is an interpretive language, it is an object-oriented programming language derived from C and C++. It inherits the power of C and C++ while stripping some of their complex features (no macros and disallowing multiple inheritances, etc.), it also has some similarities with Java, in which all these three languages were all covered in the past two years. C# is implemented using Visual Studio in this project.

Chapter 2

Literature Review

The hotness of the game market has proved that the game is a must for people and its potential commercial value. After the first computer came out, so did gaming. In the era of no Internet, stand-alone games were once popular, such as minesweeper and spider solitaire, which came with Windows computers, and can be regarded as classic games.

These games, it is not only simple and interesting but also require a certain amount of thinking and calculation. Take Sudoku, which has always been very popular, as an example. The Sudoku board has nine palaces, and each palace is divided into nine small grids. Give certain known numbers and problem-solving conditions in these eighty-one grids, and use logic and reasoning to fill in the numbers 1-9 in the other blanks. Make each number 1-9 appear only once in each row, column and house, so it is also called "Nine Palaces". Although the rules are very simple, the corresponding game logic and game algorithms have also been applied [3], and many advanced solutions have been derived from the basic solutions. Although the rules are simple, the corresponding game logic and game algorithms are also applied. The game logic was introduced by Rohit Parikh in the 1980s. S. Enqvist and his team, building on R.Parikh and Afshari & Leigh (*LICS 2017*), have proved that the PDL-style system is not only reasonable but also has Integrity [4]. The game logic organises the possible changes caused by all actions of the player/objects and can be extended and supplemented.

Although new games are being mass-produced all the time, game development is not a simple process, and it still has a long way to go from an established workflow. Through the analysis of [5], the problems of game development are now a little different than imagined traditional software choices or even social issues. But the traditional software engineering problem is still a big part that cannot be ignored in this field. For other projects, game development is irreplaceable, because it is not only the realisation of the game, but other majors such as music, art, etc. are also included [6]. The development of games also promotes the development of software engineering from this aspect.

Choosing a good game engine is a critical first step. By comparing and understanding the performance of game engines, developers can find new and popular directions more quickly [7]. For today's players, in addition to functionality, more exquisite visual effects and a better sense of interaction are also indispensable for games. As a game engine that supports multi-platform development, Unity is a good choice for 3D games. Its internal model building and rendering are easier to master than other platforms. And its Particle System can simulate natural phenomena such as rain and snow in reality. B.Zhang and W.Hu [8] have designed "lightning" and character skills that belong only to them. The introduction of the Particle System means that game developers can create visual effects that only belong to the game, making the game unique.

AI will make the whole game system more complete, even one person can start the game. Like the M.Geng team's game [9], bionic robots (AI) are used a lot in the game, but their settings for the AI system are somewhat vague, and they do not clarify the specific behaviour of AI in this system. P. Spronck and his team(2006) conducted special research on the application of AI in games and investigated an AI online learning system "dynamic script", which successfully gave AI autonomous adaptability [10], adjust the difficulty and reacts to the performance of the opponent, improve various behaviours of AI in the game, although it can be applied to commercial games, the cost could be high.

The automatic pathfinding algorithm is one of the focuses of fps games. Pathfinding needs to solve the problem of finding the shortest path from the starting point to the target point or even the optimal shortest path in a given walkable area. The two most basic algorithms are breadth-first and depth-first, they will traverse all possible paths, and the results are satisfactory and close to the truth [11], but the disadvantage is also obvious when the map is too large, the time consumption is too large. Another graph-based pathfinding algorithm is Dijkstra's algorithm. Although it will have negative weights, it will never appear with negative weights for practical purposes, so Dijkstra's algorithm is very suitable for pathfinding M.Noto and H. Sato(2000) [12] extend a new method on the basic algorithm to solve the problem that the original time is too long, which may solve the problem of car navigation in real life, but it will not have a good application in the game.

In games, a variant of the Dijkstra algorithm, the A* algorithm, is used more often. It has a separate heuristic formula and a structure that assigns weights to each node. in A. In the Candra team's game(2021), the A* algorithm played a key role, and they confirmed that when there are many obstacles in the grid, the A* Pathfinding time increases [13], but the added time is about 0.07 seconds, which is an acceptable result for most games with less complex terrain. Sometimes the choice of distance algorithm will affect the performance of the A* algorithm, for example in 2D space, the Manhattan distance will be better than the Euclidean distance. As the value of the heuristic increases, A* checks fewer nodes, but the optimal path is no longer guaranteed, but in games, this is sufficiently acceptable. In the game of E.Aydogan and Y. Atay(2021), they used the basic A* algorithm to solve the problem of finding the shortest 3D path when the helicopter landed [14], but in real life, because the helicopter's route is planned and will be submitted for review, it may only come in handy in a few cases such as emergency rescue. The pathfinding of this project is implemented by the optimised A* algorithm.

In addition to the content of the game itself, the memory and performance consumption of the game are also a focus of game development. Players often consider the memory size of the game and some hardware requirements when downloading or purchasing a game. J. .Oslin and the team(1998) introduced a GPAI - a tool for reporting game performance evaluation [15], but it is not necessary for small games. This game has further optimised part of the code to ensure the smoothness of the game.

Chapter 3

Software Design

3.1. Requirements

As a game, this project should satisfy the overall requirements, this was done to ensure a complete game was well-generated before the inspection.
All of the requirements below are divided into broad categories and given importance according to the MoSCoW prioritisation model. which can be explained as M-Must have, S-Should have, and C-Could have.

3.1.1 Functional Requirements

No.	Requirement	Priority
1.	**Game basics**	**M**
1.1	The game has a Login/Register system.	M
1.2	The game has the main menu.	M
1.3	The main menu has the following entities: 1. Continue the game. 2. Start a game. 3. Settings. 4. Exit.	M
1.4	The game has 3 levels of different difficulties.	M
1.5	The game has a time count for each level.	M
1.6	The game shows the remaining health of players.	M
1.7	The game shows the remaining bullet of the current weapon.	M
1.8	The game allows the player to play again after passing/death.	M
1.9	The game allows the player to change the weapon after unlocking it.	M
1.10	The game has correct maps for the corresponding level.	M
1.11	The objects on maps all have shadows.	M
1.12	The game allows the player to pick up the health refill props.	S
1.13	Each AI monster has a health value.	C
2.	**Game feedback**	**M**
2.1	The Login/Register system shows failure and success of incorrect/correct username and password.	M
2.2	The game shows the correct scenario of failure and success after each level.	M

2.3	The game has a 3-star system to determine the performance of the player.	M
2.4	The game shows the hint after unlocking a new weapon.	M
3.	**Control**	**M**
3.1	The game allows the player to control their movement using the keyboard WASD and Space.	M
3.2	The game allows the player to control their direction using the mouse.	M
3.3	The game allows the player to control their movement using the keyboard with arrows UP/BACK/DOWN/NEXT.	S
4.	**Audio**	**M**
4.1	The game has background music.	M
4.2	The game has different SFX sounds for each action.	M
4.3	The game allows the player to adjust the volume for BGM and SFX sound.	M
4.4	The game has different background music for the menu page.	C
5.	**Physics/Collision**	**M**
5.1	The game has gravity when the player jumps.	M
5.2	The player mimics real-life human movements.	M
5.3	The player is able to steer left and right, move upwards and backwards, also to jump.	M
5.4	The player is able to stop moving during gameplay.	M
5.5	The AI monster has an acceleration when doing follow and attack actions.	M
5.6	There is a little delay after collisions between AI monsters and the player.	M
5.7	The game is able to detect if game objects have reached the boundary of maps.	M
5.8	The weapons have different shooting speeds.	S
5.9	The weapons have to recoil when fired.	C
6.	**Artificial Intelligence**	**M**
6.1	The game has basic AI monsters for every level.	M

6.2	The game provides different types of basic AI monsters for every level.	M
6.3	The game has a boss AI (shown as a red dragon) for level3 only.	M
6.4	The AI system will let the closest AI monster detect the player.	M
6.5	The AI monsters are able to make use of pathfinding algorithms.	M
6.6	The AI monsters will abandon the player that they had detected but cannot see anymore.	M
6.7	The AI system allows all AI to attack the player.	M
6.8	The boss AI has more health and more damaging attacks.	M
6.9	All AIs have a self-destroyed system if they are at a negative height on the map.	M
6.10	The AI system provides more and stronger basic AI monsters as the player moves to another level.	S
6.11	All AIs are not easily beaten.	S

3.2.2 Non-functional Requirements

No.	Requirement	Priority
1.	**Performance**	**M**
1.1	The game has at least 10 AI basic monsters per map	M
1.2	The game runs smoothly on any device.	M
1.3	The game does not crash or freeze whilst the user is playing.	S
1.4	The game reacts to user responses within a second.	S
2.	**Installation**	**M**
2.1	The game can run on Windows/Linux/Mac OS.	M
2.2	The game is able to be distributed as a single .exe file.	S
3.	**Implementation**	**M**
3.1	The game is implemented in C#.	M
3.2	The game follows object-oriented programming principles.	M
4.	**Usability**	**M**
4.1	The UI adheres closely to HCI principles to improve overall quality.	M

4.2	The UI is all in a low poly style.	M
4.3	The UI is clean and attractive.	M
4.4	All the buttons open the correct screens.	M
4.5	The UI has a vibrant array of colours.	S
4.6	The game windows fill the screen of the player.	S
5.	**Copyright**	**M**
5.1	Game code/resources do not infringe any copyright laws.	M
6.	**Security**	**M**
6.1	The game does not compromise the security of the device of users.	M
6.2	The game does not disclose personal user information to a third party.	M

3.2 Game Component

3.2.1 Game Introduction and Play

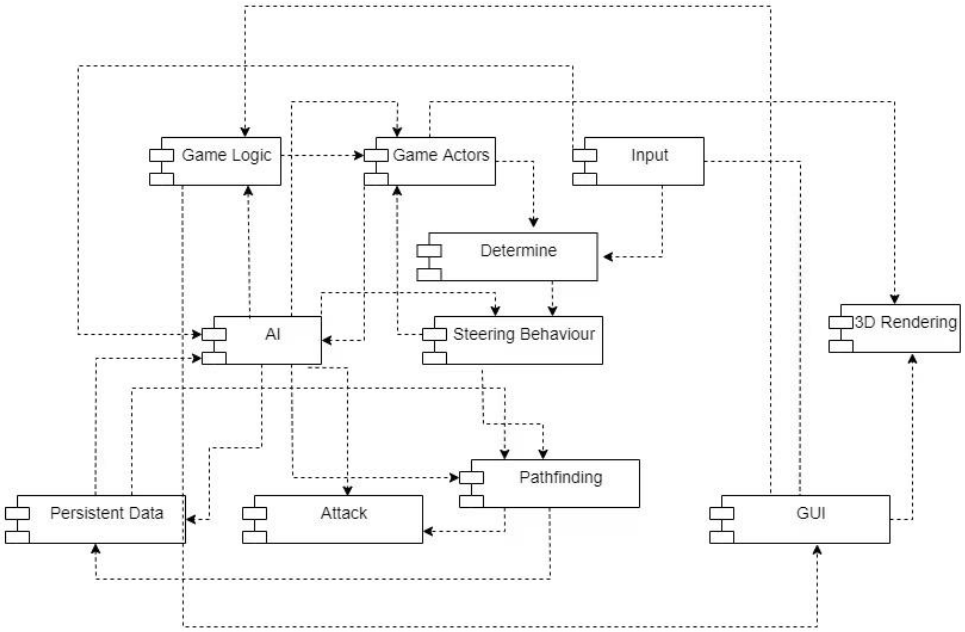


Figure 3.2.1. (a) shows the game(using the component module)

Two characters of the game are the player and the AI monsters. At the starting panel, the game has a story background: *"I'm in the middle of an unknown place! I have to find a way out of here!".* It tells the players what they are aiming to do and an introduction will let the players feel more immersive. Also inside the game screen, there is always a text hint shown on the left top of the screen, telling the players to shoot all monsters to become the winner.

After Register or Login, the players are allowed to enter the menu page. There are four buttons for choosing

1. Continue the game
2. Start a game
3. Setting
4. Exit

The first two of them give the player the choice for gaming. The Setting panel produces two sliders to control the volume of background music and SFX sound, they can be set from zero to 100. The Exit button will close the program and exit immediately.

The game includes 3 different levels with 3 different maps. The further the game progresses, the larger map and harder unbeatable AI monsters will come up. For Level Three, a special and unique Boss AI is shown. Every time a player fails any level, they always have the chance to try again. It is worth mentioning that a star marking system is covered at the end of each passing scene. The system gives the corresponding number of stars according to the clearance time. Getting one star is unconditional if the player passes it. For two stars, 90 seconds is set for levels 1 and 2, and the 3-star condition for these two levels is 60 seconds. Since the map for level 3 is bigger and there is a Boss AI, the time for the player is more relaxing, in 300 seconds will get 2 stars and 240 seconds will get 3 stars. Players can check the time past anytime on the right top of the screen. Compared to recording used bullets and then marking using limited bullets, the time-recording system did not have a limit on players' performance, and players will not stop trying to shoot because they are afraid of not getting a good score, which also defeats the purpose of the game. The marking system is set for the players who always want to get a full mark in a game.

The control of the player is set as in other normal games. WASD for four directions to move, any left or right can be combined with Go or Back, the player is allowed to move 360 degrees. Additionally, Up, down, left and right keys on the keypad can also be used to control the character. It depends on the personal habits of users. For Jumping, Space is used, it can also be used together with any of four directions. The mouse is used to control the degree of the view. The camera(view of the player) will automatically follow with the mouse. The left mouse button is the shooting event.

On the left bottom of the screen. The remaining health bar and remaining bullet bar can be seen. The player can choose attack distance by thinking about them. Less health or fewer bullets should shoot from a further distance in case of failure on passing. In Functional Requirements, one of them is ‘All AIs are not easily beaten’.So, A health refill prop is introduced by the game. Once the player has little health but is still far away from mission clear, the player can go to pick up the prop then the health itself will be full again. Replacing the solution of adding health to the player by adding props will increase the challenge of the game and make the player feel more excited about the game。

To enrich the content of the game, 2 weapons are introduced in this project. The second weapon will be unlocked after passing level2. Press Alpha1 and Alpha2 can change the existing weapon, the times of change are unlimited. The bullet storage and damage are also different. All game screenshots will be shown in *part 3.3.2*.

3.2.2 Audio

The AudioManager is in charge of all audio files. Sound can affect games a lot, it decides the type of game to some extent. This system provides a steady connection between the game and music files. All audio is in .mp3 format or .wav format. Maybe some occasions are not suitable for games with sound，then volume control can be found on the Settings page as mentioned, this allows the adjustment from 0 to 100. The AudioManager provides various sound effects(SFX sound) for actions, these 28 SFX sounds can increase the realism of the game. Nearly every event has an SFX sound, either from players, the AI monsters or the weapon.

3.2.3 VFX

VFX means visual effects. People often receive the greatest visual impact, so it is very important in any game. Dynamic maps and great strike effects often become a bonus for a game. The particle system is used for creating VFX in this game. The particle system represents some techniques in 3D graphics to simulate the blurring of visual abstract effects such as sparks, explosions, smoke, light trails, etc., which are often difficult to achieve with other rendering techniques, they have realistic game physics.

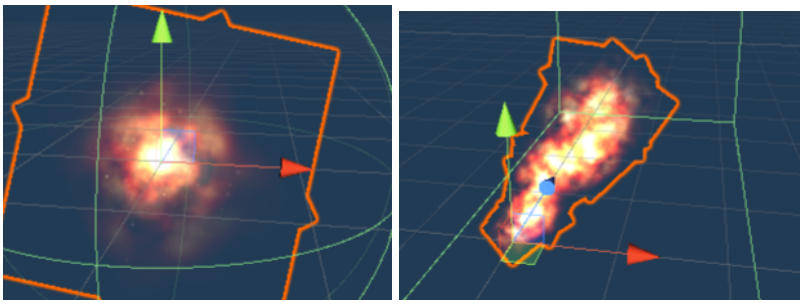


Figure 3.2.3.(a) shows the VFX of the 'ball' and 'fire' of Boss AI

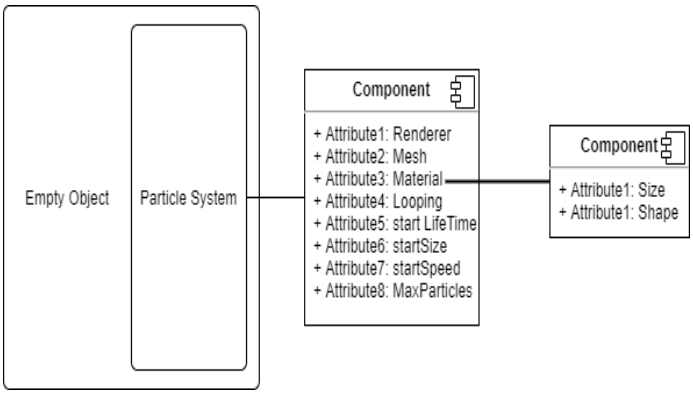


Figure 3.2.3. (b)shows the what attributes can be used to make a VFX

In Unity, particle systems are managed modularly. Take *Figure 3.2.3. (a)* as an example, to build them, the user needs to hang a particle System as a sub-object on an empty object, and then set the corresponding properties to achieve the effect, often Material, Mesh and MaxParticles will determine a rough model of this effect. For the two attack methods of Boss AI, the most different points are the interval time and the number of particles. Moreover, these two kinds of VFX appear in the form of crosses, and fire always has more lethality.

3.2.4 Game Recording Video link

A full recording of fast passing at all levels is provided(5x speed):
https://drive.google.com/drive/folders/1fFAdxCTmEfHNsR-ROmK1UB6ejecZqkQt?usp=sharing

3.3 User Interface

The style of the entire game uses a low poly style, which is not a common human war style. If the project used a war-style AI, then the attacks of AI will appear clumsy. When a human-AI faces a player who is also a human and has guns, it will still run up to attack with its arms. This is very unreasonable. But when changing the model to an AI that looks like a monster, it then makes more sense, because in their world there is no such thing as a weapon. Whether it is weapons, terrains, or AI monsters, the entire style is consistent throughout the game. All pages before entering the game interface maintain the same background image and the same design template: only buttons and text and some input fields, which makes the entire UI system look clean and tidy. There are a total of 7 UI Scenes in the whole game, and the control and jump of the game should be displayed in these seven Scenes.

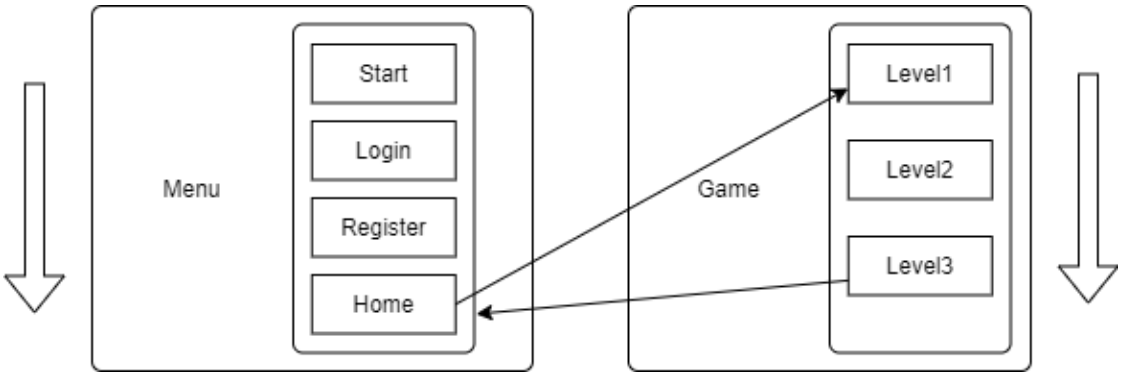


Figure3.3. (a) shows the order of scenes for the game, two arrows on two sides show the order of the scenes.

All other UI screens are also included in the game, including some prompt interfaces, such as the UI display of login/registration success or failure, and the volume control interface. Another more important UI is the feedback interface after each level, this gives the player a good idea of how they performed in the game just now. The feedback panels also maintain a simple and clean design style, consisting only of text and buttons. However, on the Pass UI, the number of stars assessed by the player's current round of clearance time will be displayed.

3.3.1 Maps and Enemies

As said in Game Introduction and Play, the UI of each level will be different, which is to add interest to the game and enrich the game content. It is not notified that it is mainly reflected in the difference between maps and AI monsters:

- Level 1 has the smallest map, the scene style is set to a western village, and has 13 AI monsters.
- Level 2 is based on Oasis village and has 21 AI monsters.

- The third level has the largest map, which shows the appearance of Alpine woodlands. There are a total of 17 AI monsters and a Boss AI.

CreatingMAP

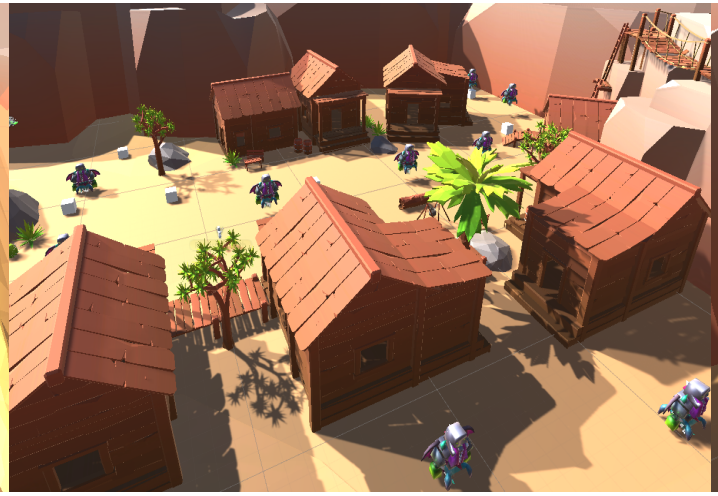
Figure3.3.1. (a)shows a part of the map 'Alpine woodlands' for level 3, called 'Fishing Village', it contains about 90 little components for making.

After determining the style of the game, I downloaded the relevant asset pack from the Unity Asset Store. The next step is to conceive and roughly draw what view I want the entire map to look like. The three maps in the game are surrounded by tall mountains and rocks, which can be more appealing to the story background in the introduction to tell the players they do not have any other ways to escape. Then filling the interior becomes the next step. Everything on the map needs to be added one by one. Take the Stone Path as an example, suppose it contains 120 stones, then these 120 individual stones need to be placed on the map one by one. The solution to this problem is to create a new prefab (which can be seen as a preset) so that the same structure can be reused, which saves a certain amount of time. The dynamic Particle System is also added in the same way. When creating a map, one thing to pay attention to is that the height of each object needs to match, for example, a top-down waterfall needs to be placed on a mountain of the same height. Also, the river should be created at a negative height so that the water level is the same as the ground.

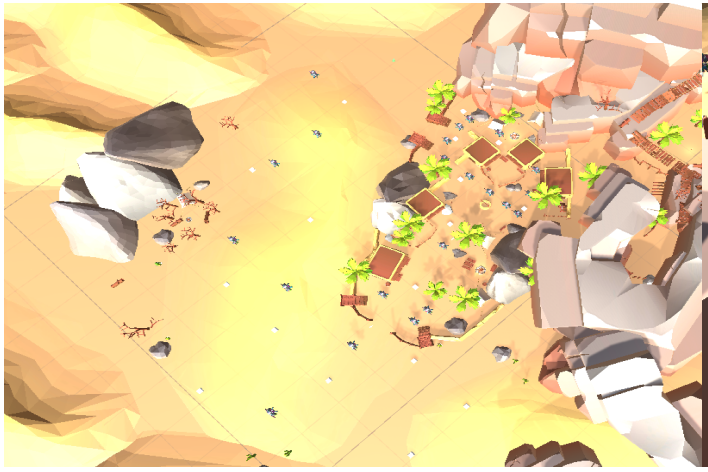
Full Maps UI

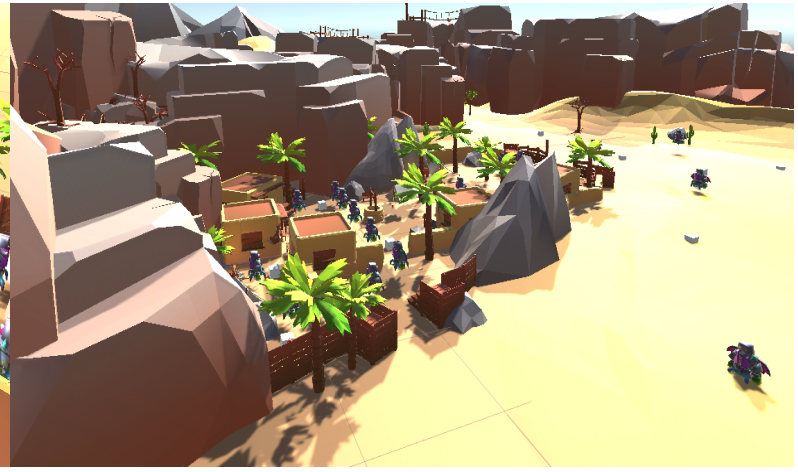
The pictures below show the top-down, left-side and right-side views of maps for different views(compressed size due to paper page).

Level 1

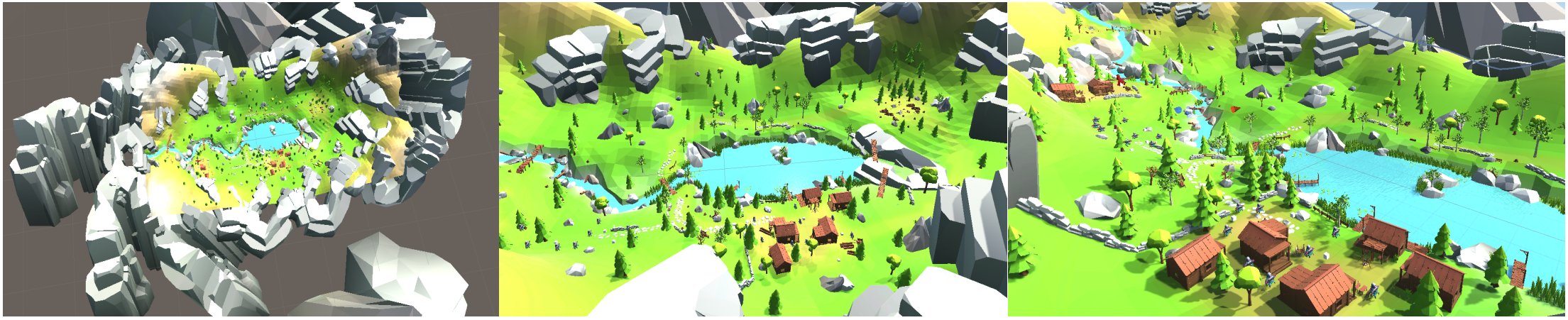




Level 2





Level 3



3.3.2 Main Game UI

The pictures show some main game UI, more UI can be checked in *Appendices A1*, and the different UI models of the AI monsters can also be seen.

One of the Functional requirements is that all objects on maps must have their shadows, which is also a must for 3D games. The source of the shadow is the light source, and the illumination of the light source from different angles will form different shadows, just like the direction of the shadow at noon and evening in real life is different. In the game screen, all objects within the UI page have shadows.

Figures 3.3.1. (b) shows shadows of the AI monsters and a tree

Chapter 4

Implementation

4.1 First Perspective with Camera

As the first part of this project, using the camera to simulate the first perspective view is the most basic part. For a human body, there is a distance between the two eyes, and different angles are used when observing objects, so even the same object, the observed appearance is different, resulting in a three-dimensional feeling. The principle of polarised 3D imaging is generated in the same way.

In Unity, a camera is a tool that simulates the human eye. We only need a camera, declare it in the fps controller, which is the script that controls the player, then set the camera to the position of people's eyes when holding a gun for aiming: the end part of the weapon (to ensure that the entire weapon model can be seen). At last, get the up/down direction of the mouse in the script, so that the mouse and the camera can be linked to simulate the human eye. And because the player needs to move in the YOZ plane, so the mouse is set to rotate based on the X-axis.

(a) (b)

Figure 4.1. (a) shows the position of the camera. Figure 4.1. (b) shows the movement in two directions of the mouse. X controls the rotation of left and right, and Y controls the up and down rotation. For Y, the angle should be limited to a range using the function Mathf.Clamp().

3.2 Architecture

This project requires the player to use the laptop as the client and also uses localhost as the server to connect to the registration and login database.

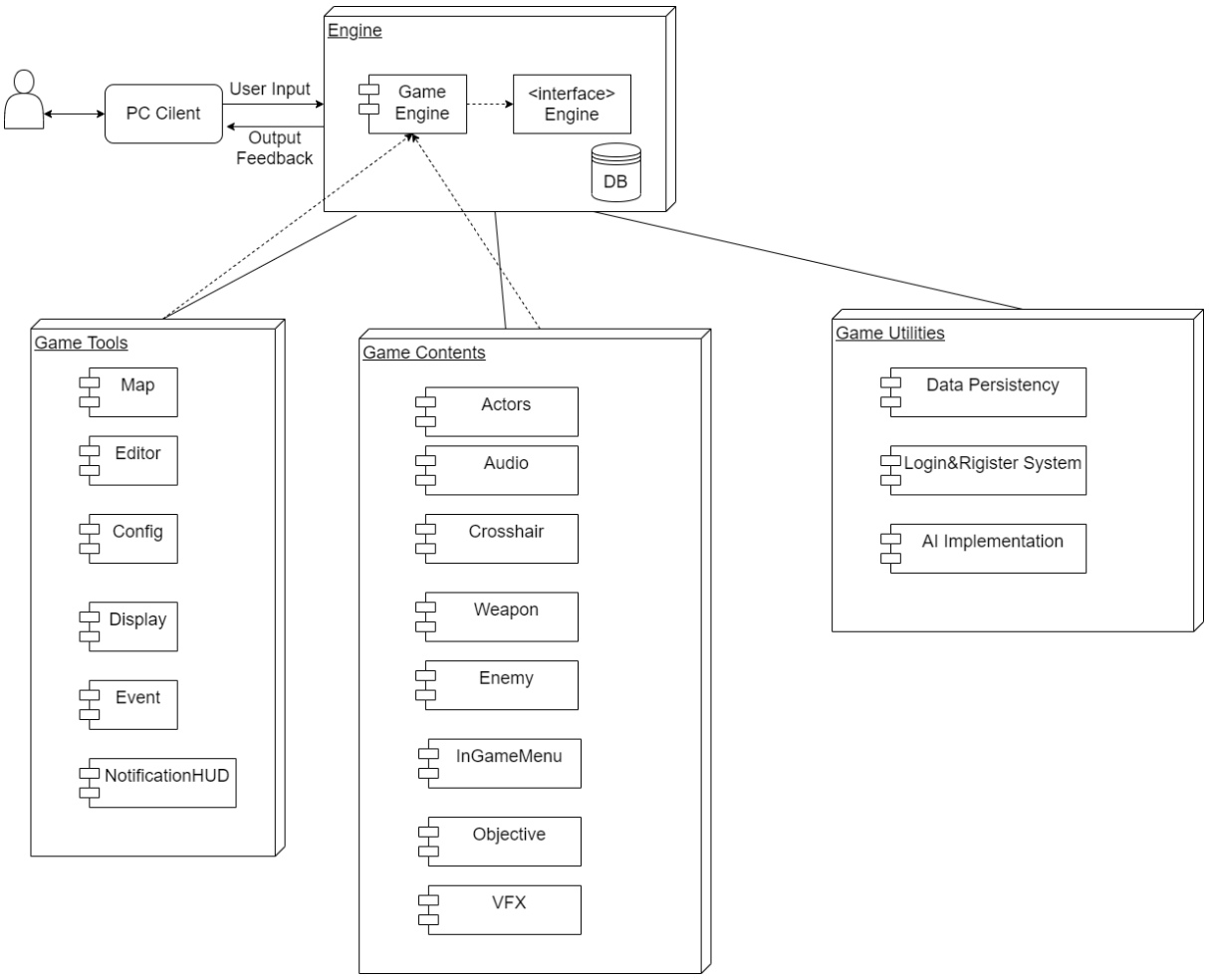


Figure 4.2. (a)shows the system architecture for the game.

In the system, the user first needs to open the execution file of the game, *user input* refers to the username and password entered by the user. When they start to enter the game page, the game engine starts running. Otherwise, when the user decides to quit the game, he or she will receive feedback in the form of UI, which can be the prompt information of Pass or Fail level, or the information of returning to the main menu after completing all levels. The Database stored here not only contains the database stored by the game running time but also contains the original Map, terrain, basic files and so on. The game engine is mainly composed of three departments: Tools, Content and Utility:

- Game Tools: Mainly composed of UI, Script, in-game system actions and config files. These are all necessary tools for the game.
- Game Contents: Contains all the content that appears on the game page, Actor here refers to the player. In this part, they are usually connected. For example, the VFX effect will only appear when the player hits the AI monster with a weapon attack.
- Game Utilities: After having these three utilities, the content of the game is more plentiful and interesting so that the whole game is more complete and attractive.

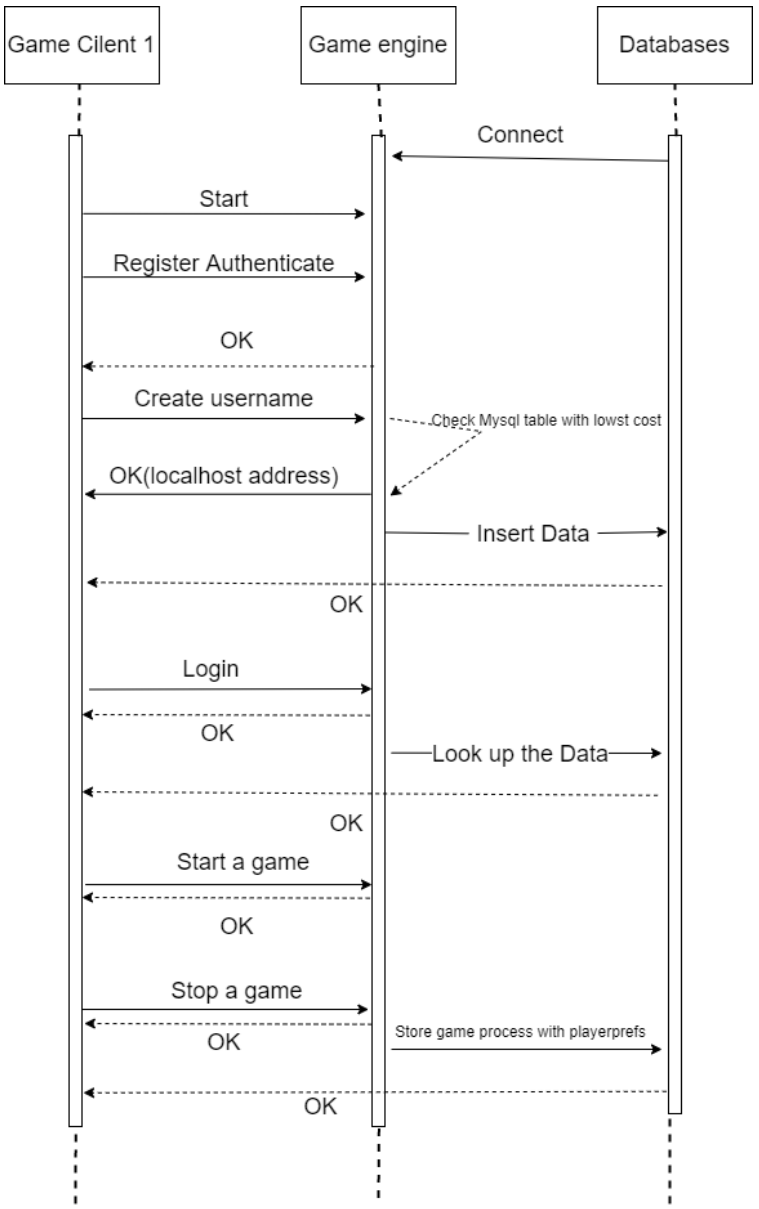


Figure 4.2. (b) shows a sequence diagram of how the three parts interact with each other

4.2.1 Databases

Mysql

For Registering and Logging a game, a truly existing database is needed. Usually, it has columns ‘username’ and ‘password’. Sometimes ‘email’ and ‘address’ are also included. In this project, a database for storing users which connects between Mysql and Unity is included. Compared with other databases, Mysql does not need to download and use other plugins and is free. As an open-source and cross-platform tool, Mysql provides an efficient but low-cost service. The database table for this game was created in *Mysql Workbench*, sometimes modified in the script directly which is written in Php.

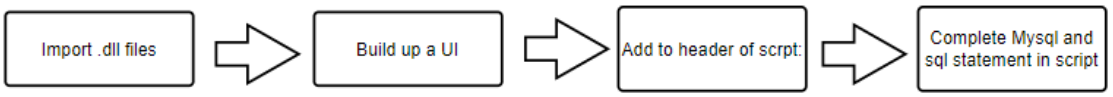


Figure 4.2.1. (a)shows an overview of basic steps for using Mysql in Unity

To create a connection. Firstly, some necessary .dll files should be imported into the project, these files are used as Plugins. Also, some UI screens should be prepared. Then, the scripts must have the following header:

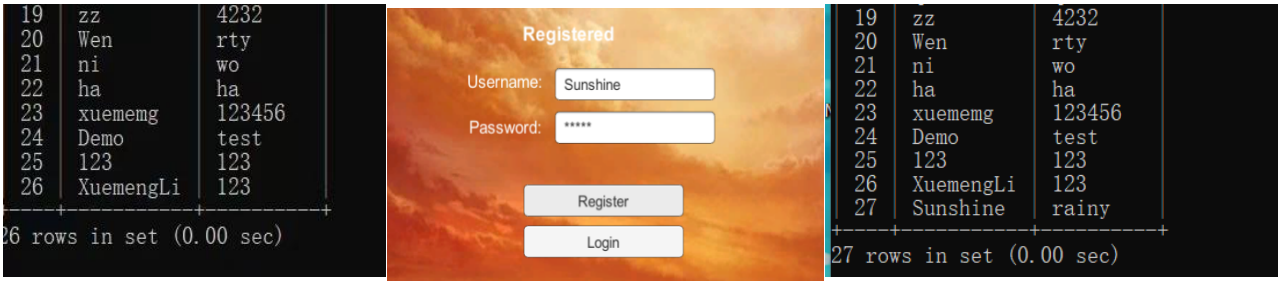
```
'using Mysql.Data.MysqlCilent'
'using Mysql.Data'
'using System.Data'
’using System.Collections‘
‘using System.Collections.Generic’
```

For the connection, the script contains the following code within Mysql:

```
public MySqlConnection GetConnection()
   {
        // get connection by matching the name of the Database
      MySqlConnection conn = new
MySqlConnection("Server=localhost;UserId=root;Password=root;Database=userinfo"
);
      conn.Open();
      return conn;
   }
```

The use of Mysql statements is quite important, any small syntax or even an extra space can result in a wrong connection to another database that has a similar name, even causing a null connection. At last, if the user completes the fields in Register game Screen, the insert commands will be used to find the corresponding column and insert the new contents. For the Login screen, instead of inserting, firstly go into the table in the database then check if there is a matching row for both username and password:

```
//For Register
"insert into userinfo(username,password) values('{0}','{1}');"
//For Login
"select * from userinfo where username='{0}'and password='{1}';"
```



Figures 4.2.1. (b) shows the action in the Register game screen and the before & after tables of the Database

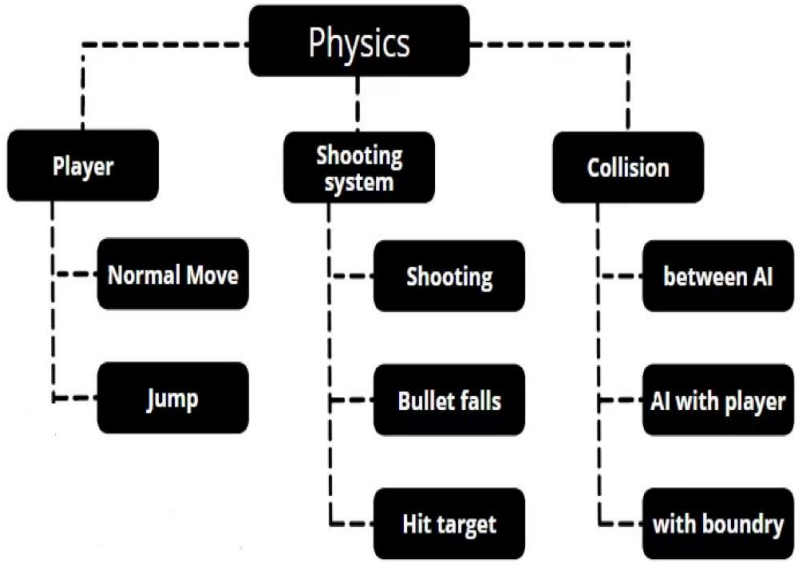
Playerprefs

In the game menu page, the player can choose to continue to rechallenge the game level that failed in the previous round or start from the first level. The task of storing levels is implemented by Playerprefs. Playerprefs is a persistent saving and reading class provided by Unity3d, which can be regarded as a dictionary. playerprefs is a sealed class and cannot be inherited. It obtains all data by using key values. In the game, the function of the "Continue the game" button can be completed by matching different players in the database first, then saving the player's level information and finally reading it.

```
1. void Start () {
2.         PlayerPrefs.SetString("username","Alice");
3.         if (PlayerPrefs.HasKey("Alice"))
4.         {
5.             Debug.Log(PlayerPrefs.GetString("username"));
6.         }
7.     }
```

Figure 4.3. (c) shows how PlayerPrefs can work with usernames from Mysql.

4.2.3 Physics

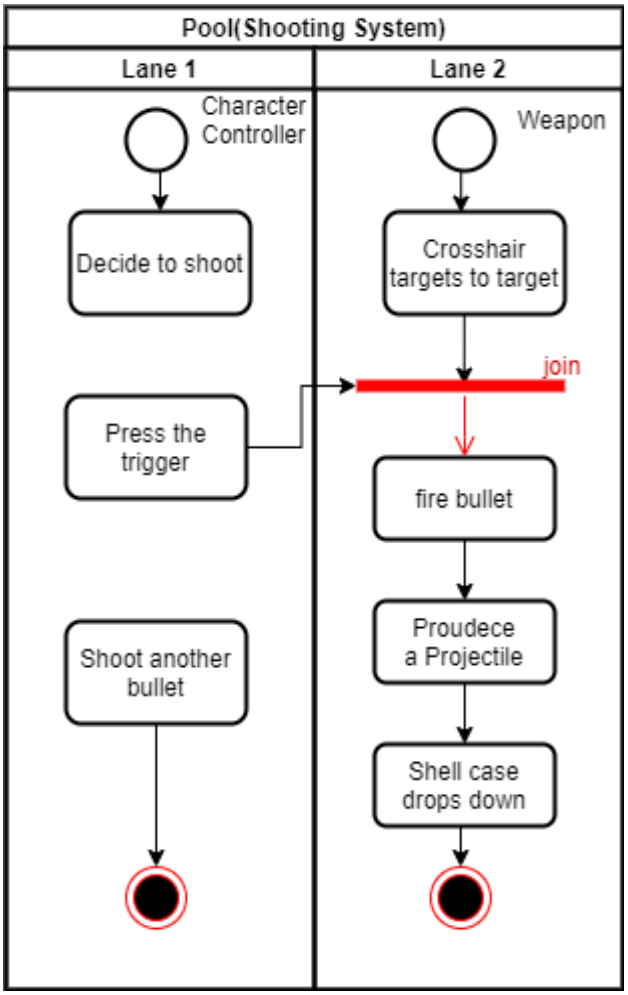


The physical system is an important part of 3D games, and the physical principles, in reality, are ubiquitous in the game world. In Unity, only objects which contain Rigidbody will be affected by the physics engine. So in this project, CharacterController is the basis for implementing various physical operations, which is composed of RgidBody representing Character and CapsuleCollider(a trigger) of the controller. For the

Figure 4.2.2. (a)shows possible actions which use physic theory

player jumping problem, the system needs to consider the gravity problem. This can be easily solved with the controller alone. In the game, the gravity was set to 9.8.

Shooting System



For FPS games, the shooting system is the focus of the physics mechanism. When the player shoots, the whole action is done by the controller along with the weapon. The weapon is composed of bullets, Shell Case, Crosshair and a trigger. Whenever the player presses the left mouse button, it means that a bullet is fired. Inside the weapon, a simple logic: press the trigger - the bullet is fired - the shell case is dropped and will be looped all the time. The projectile is the VFX(visual effect) of the parabola after the bullet is fired. When all the bullets in one magazine have been fired, the weapon will automatically refill the second magazine, and the number of bullets stored will vary depending on the weapon.

For weapons, there are two states of firing and not firing. Like the AI part of *3.3.1*, they are also distinguished by colours. Additionally, any activity in the shooting system is related to Audio and VFX (*parts 3.2.3 and 3.2.3*)

Figure 4.2.2. (b) shows an activity diagram with 2 swim lanes for the shooting system. Lane 1 represents the Character Controller, Lane 2 is for the Weapons

Collision

As mentioned, The CharacterController is used for physics mechanisms. Because it contains a Rigidbody and a Collider, it is possible to script forces and collisions that are fully influenced by the physics engine. It can also collide with GameObjects that only contain Colliders. From *Figure4.2.2. (a)* , three collisions are shown:

- When two AI monsters have collisions, they will separate immediately.
- If the player collides with the AI monster, the player will not be allowed to move forward anymore and will be attacked.
- Both the AI monsters and the player collide with the boundary of the map, they cannot move until they change direction.

4.2.3 User Interface

The completion of the UI consists of the deployment of a set of UI-related scripts written in C#, and the locations of elements were set by the Unity panel. The MVC architecture is used here, MVC stands for Model-View-Controller, which is a design framework that divides the functions of the code. The reason for choosing MVC is because it can start design in a short time and is easier to learn than other architectures like Struts which have more technical difficulties. The main principle and advantage are to separate the software user interface and business logic to enhance the flexibility of the code. It is more convenient to expand and reuse. Also, this structure helps to avoid complexity between components.

- Model - the part of an application that handles the logic of the application's data. Usually, Model objects are responsible for accessing data in the database. In this game, the model stores all game objects, components and data files.
- View - the part of the application that handles the display of data. Usually, Views are created from model data. Here, it also manages the rendering of the Unity3d engine, which requires access to the textures, materials and effects of the 3D models.
- Controller - the part of the application that handles user interaction. Usually, the Controller is responsible for reading data from the View, controlling user input, and sending data to the Model. In games, the user can influence his input attempts, such as login failures and successes.

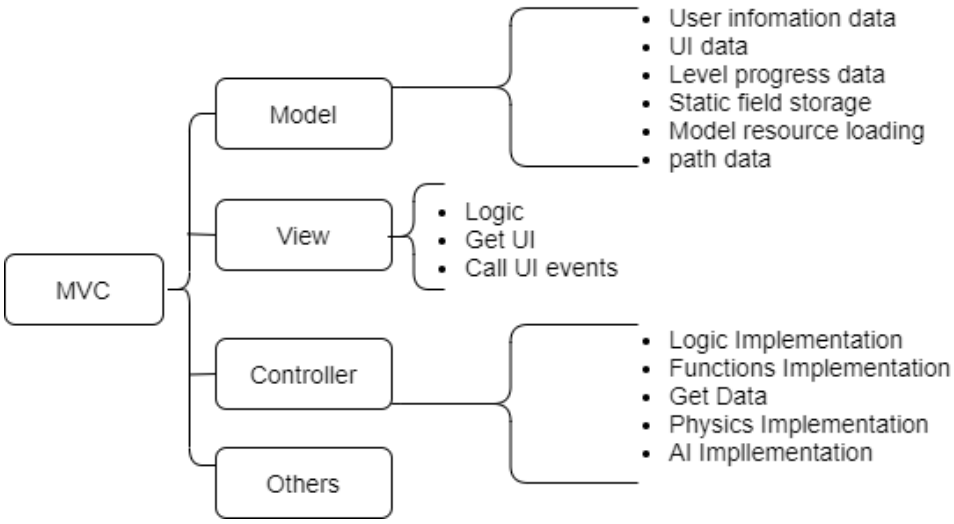


Figure 3.2.3. (a) shows The MVC architecture for this project

As can be seen from the above figure, each part of MVC is independent, and there is no need to consider the interaction between the two departments, which is convenient for adding more functions and better maintenance in the future.

4.3 Logic

For this project, two main logics are covered. The following flowcharts and explanations will show each of them in detail.

4.3.1 Artificial Intelligence with Algorlthm Implementation

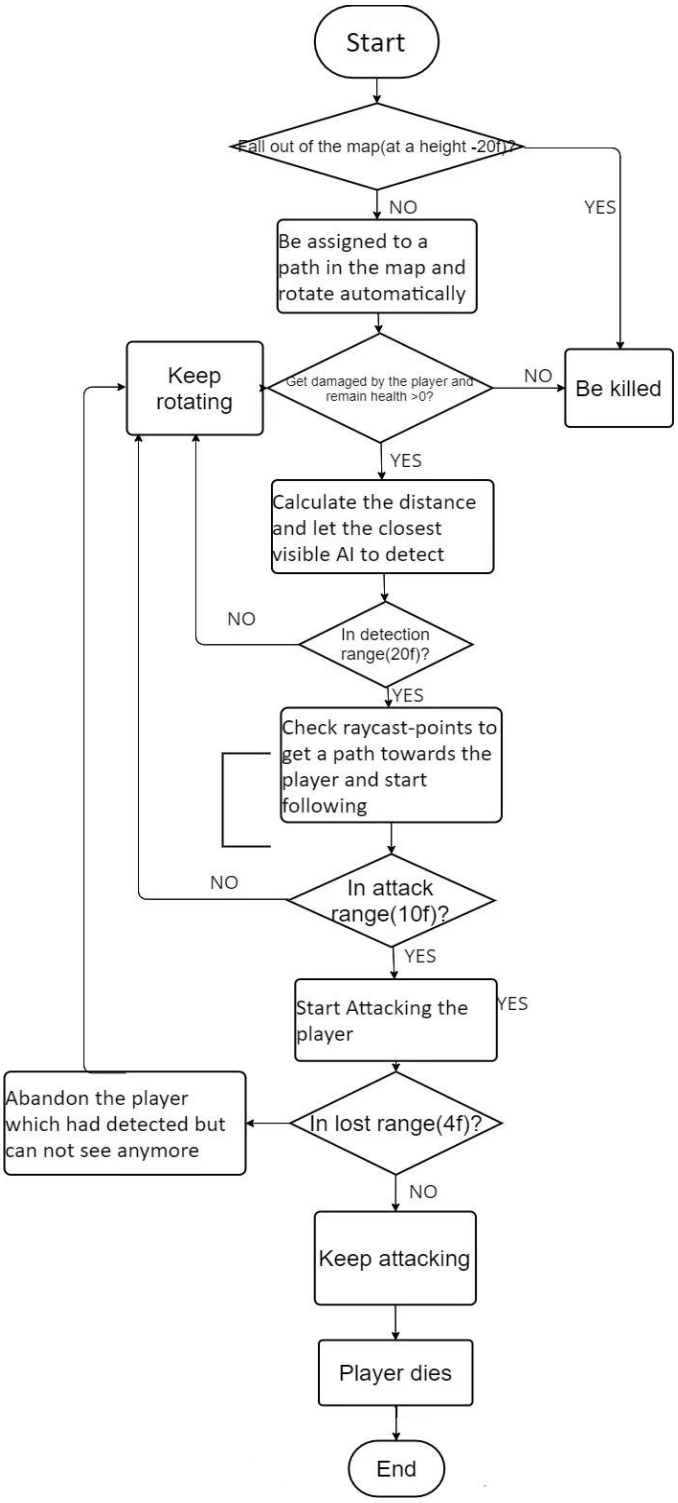


Figure4.3.1.(a) shows the flowchart for the logic of AI monsters

AI monsters are an important part of this project, they are the protagonists of the game besides the player. In this part, the principle of Finite StateMachine (FSM) is used. The AI monsters mainly have three States, namely patrol, follow and attack. As described in the flowchart, at the beginning of each level, which is the patrol state, the AI monsters will be automatically generated on the map. In this project, the conditional trigger is used for AI monsters. When the time starts to be counted, the bool value of the Controller will change, meaning AI starts to work. All monsters will be assigned to different paths, and they will have an initial movement speed. But Once AI monsters are attacked, they will start to calculate the player's position and own distance. During this period, once the player enters the AI's detection area, they will switch to the following state.

When following the player, the AI monsters will rush toward the player at a faster speed. At this state, the AI will set the position of the player as the destination. Due to the player is always moving, so the path will always be updated. The AI monsters have to always check the raycast-points to get a path.

Raycast

Raycast is used to detect a collider. If it collides with the collider, it returns true, otherwise, it returns false. This raycast is set by ourselves. For using RaycastHit, when true is received, ‘hit’ is the object collider that the raycast hits, incoming as ‘out hit’. Then the object information of the collided object can be found through the hit. Additionally, this raycast line can be drawn using the function *OnDrawGizmos()*.

Then the navigation will start working, which will always find a solution, that is the pathfindings. To achieve this goal, NavMesh[16] (*Figure 4.3.1(b)*) and the A*Search algorithms are applied.

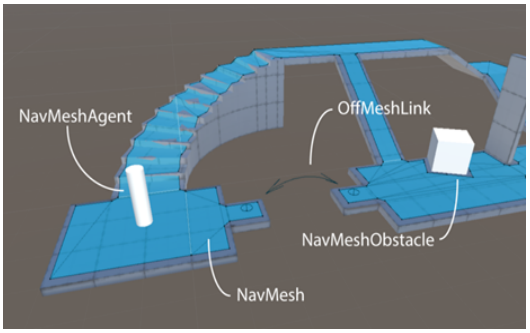


Figure 4.3.1. (b) shows the system of NavMesh, NavMeshAgent means can be seen as the AI monster here, NavMesh is the grid on the map. OffMeshLink shows the action of the NavMeshAgent, for here, is to jump over the gap. NavMeshObstacle is the object that the AI wants to avoid on the path.

Pathfinding

As mentioned, AI will be assigned to Node, and the node is also the basic unit of the A* algorithm. Also from Literature Review, it can be found that the A* is more suitable for game pathfinding. This game used an improved A*algorithm to complete this part.
The function of the A* algorithm is:

$$f(n) = g(n) + h(n)$$

where f(n) represents the comprehensive priority. In each calculation, the node with the smallest value (which means the least cost of the path) will be selected, g(n) is the distance from the nth node to the starting point, and h (n) is the cost function to the endpoint. For distance calculations, since the AI is allowed to move in any direction on the map, Euclidean distance is more appropriate:

$$\sqrt{(p2.x - p1.x)^2 + (p2.y - p1.y)^2}$$

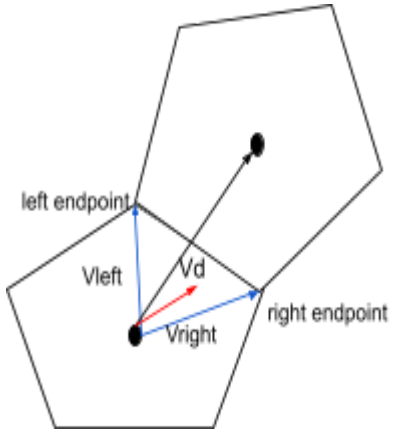
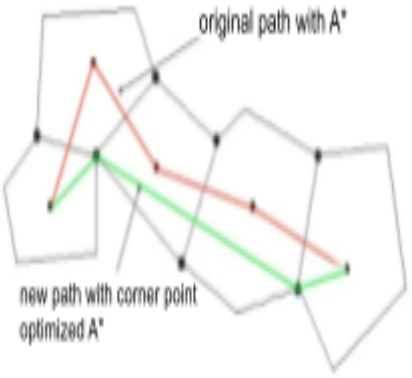


Figure 4.3.1. (c) inflexion point diagram

To better cover the entire 3D terrain and reduce the number of nodes, this project uses convex polygons as nodes. In NavMesh, the path is composed of adjacent polygons. Just need to pay more attention to the corners, the other straight walking areas are formed by polygons, which can pass directly. Since every two adjacent polygons have only one edge in common, only two points can be inflexion points. As shown in *Figure 3.2.1(c)*, Vleft and Vright can be calculated from the node of the two polygons and the coordinates of the left and right endpoints, moreover, Vd can be calculated according to the destination coordinates. The thus optimised A* algorithm can further shorten the path distance.

To obtain a complete path, an array can be set here to store all the inflexion points P, the first P is the centre point of the first polygon, and the second inflexion point P is the endpoint on

the vector which has a smaller angle between the two angles obtained above. Then repeat to the destination (if the vector is



zero, which tells it coincides with the endpoint, so count the endpoint instead). Thus, a path that is better than the original A* algorithm is generated, At last, this project sets the radius of the monster to be corrected as an offset, so that the AI monster will not be stuck with the map boundary.

Figure 4.3.1. (d)shows the difference between the two A algorithms*

Obstacle Avoidance

On the way to follow the player, it is inevitable to encounter obstacles. In the A* algorithm, avoiding obstacles is included, but in this project, these obstacles can be static buildings on the map, such as haystacks, Houses, etc., or other moving AI monsters. The way to deal with static obstacles and dynamic obstacles is different. For those items that originally exist on the terrain/map, a global solution is being used to solve the problem, the reason is that they block the way of all AI monsters. And for those items that are moving around the map, a local solution is taken, because they are not part of the grid and have no impact on the global, it is better to use the local obstacle avoidance processing, the local processing method only considers the upcoming collision, making the AI monsters more predictable so that It can effectively avoid dynamic obstacles.

Then, if there is a line of sight to the player and within attack distance, the AI monsters will switch to attack mode. In attack mode, they start trying to attack the player, when the health of AI is less than zero, it means they are killed and they will disappear from the map. On the other hand, if the game generates an error at runtime, causing AI monsters to spawn in a -20f or more negative height, they self-destroyed and also disappear. During follow and attack, if the player runs away, the AI monsters will abandon the player which had been detected but can not see anymore. But if they had seen the player once, it would be stored for the next time. If the running distance is greater than the attack distance but less than the detection distance, the AI monster's status will be converted to follow again. For every transition, the AI monsters have a distance of 2f to buffer in case of an action crash.

This project Uses eye colours to classify whether the AI is in each state:

- the path reaching range: yellow
- the attack range: red
- the detection range: blue

Although this idea will not be seen in-game because it is covered by the animation but is used in the logic.

Group Behaviour

Group behaviour means that all members in the same group will do the same behaviour. In this project, it means that a group of AI monsters start rotating on the map at the same time,

and they will maintain a certain distance between them. In this group behaviour, there are also individual behaviours. For example, two AI monsters change direction after giving up chasing the player and move toward one same node, which will lead to a collision but will separate immediately, but even so, the overall consistency is still maintained. Group behaviour consists of three manipulation forces, Separation, Alignment and Cohesion.

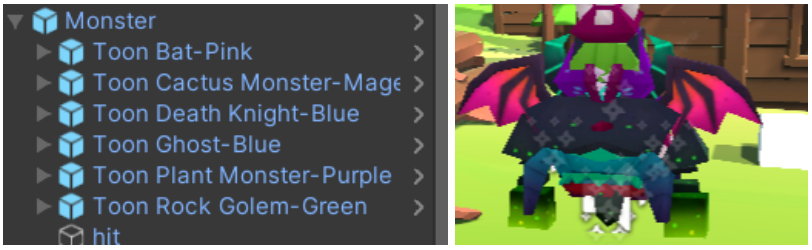
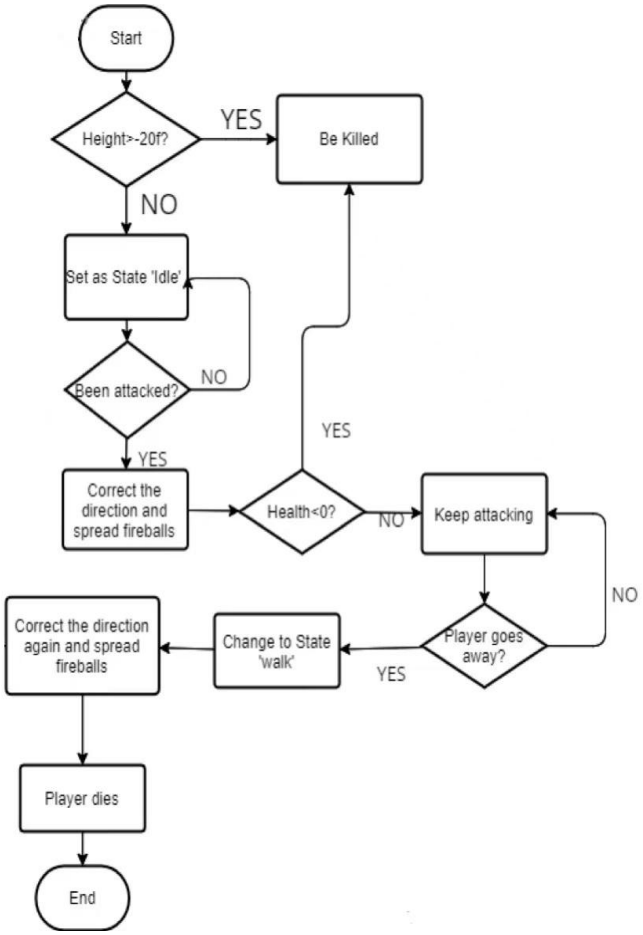


Figure 4.3.1. (e) shows one of the groups of AI monsters in Packages and the game screen

In each group, a leader will be selected, and other AI monsters will follow and move with the leader according to certain Flocking rules. Separation is the force to prevent individuals from overlapping locally, which means there should maintain a certain distance between any two AI monsters. Alignment is a force that allows all members of the group to follow the leader to move in the same direction. Cohesion is a force that lets the members move to the average position of nearby companions. This can make the overall group pathfinding effect more realistic.



Boss AI

The boss AI is shown as a red dragon in the game. The Boss has most similar actions but different states because it is for level 3 only. And once the player kills all the other AI monsters on the path, he or she then can see the Boss AI. The main states in FSM for Boss AI are Idle, walk and attack. Unlike the basic AI monsters, it does not need to rotate on the map, just needs to wait for the player to come(like a boss in real-life). Once it gets damaged by the player, the Boss AI will turn in the correct direction, which is the same principle as the weapon ‘turret’. Then it will start to attack the player using its unique way-fireballs, this is switching between balls and fire(*part 4.2.3*). If the player runs away in the fight, it will change to the state ‘walk’ to follow the player. The

Figure 4.3.1. (f)shows the flowchart for the logic of the Boss AI

Boss AI uses the same algorithms but is shown differently. Also, as a boss, it has more health, more damage and it is harder to beat.

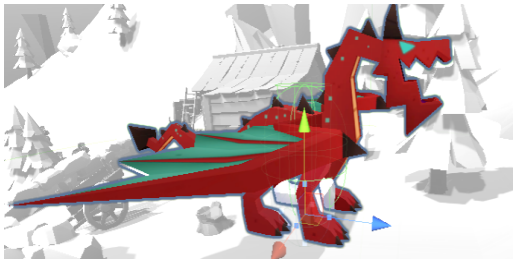


Figure 4.3.1. (g)shows the red dragon model of Boss AI

4.3.2 Player

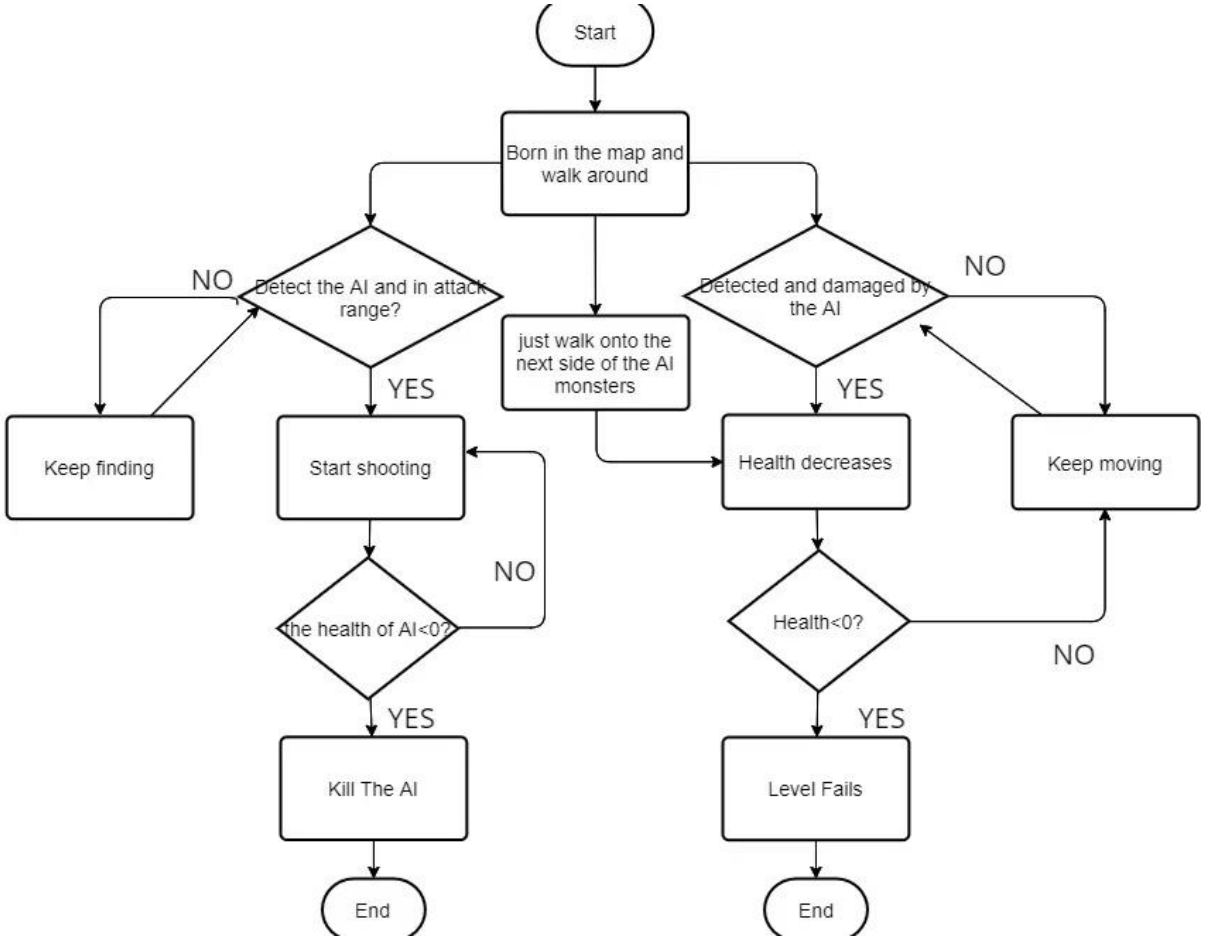


Figure 4.3.2. (a) shows the flowchart for the logic of the player

Compared to the artificial intelligence part, the player's logic is much simpler. Players have a total of two states, chasing and attacking the enemy or being chased and damaged. These two events can happen at the same time, depending on the health of those who are less than 0 first, and then ends in the corresponding scenario.

When chasing the enemy, just like the AI monsters, there is an attack range. Sometimes even if bullets are fired, those bullets have no damaging effect because they are outside the effective attack range. When the enemy is successfully attacked, it is only necessary to continuously shoot the AI monsters, just kill them.

In the state of being attacked, the player should try to run as far as possible to avoid further reduction of the health value. When the player's health is less than 0, it means that the level has failed. Additionally, if the player is a really small distance next to the AI monsters, the AI will also start to attack. This is set because the AI's purpose is to kill the player, since the player is all sent to their side, there is no reason to hesitate, this logic is also in line with the overall game logic.

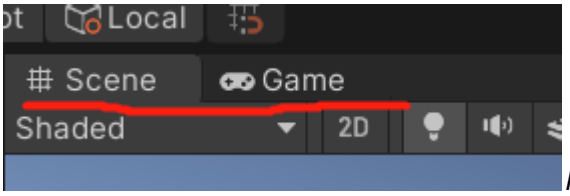
Chapter 5

Evaluation

To evaluate this project, this chapter will review and analyse it from both technical and gameplay perspectives.

5.1 Technical Evaluation

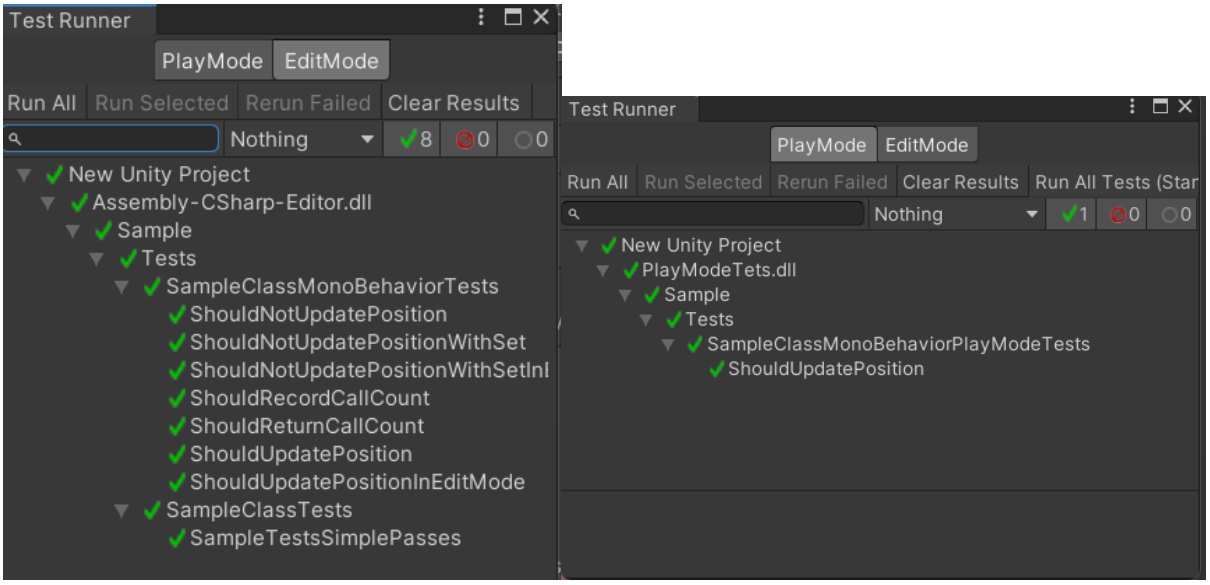
The technical evaluation is to ensure that the project can run without bugs and meet the requirements in *3.1*. The first test method is to intersperse the project with Debug files and Debug methods, which ensures that the code is improved as the project progresses and further ensures that the project is free of script errors.



Figures 5.1. (a) shows the screens for test method 2

The second test method is the Scene, Game screens in Unity editor. The Scene screen is editable, and the game screen is the final effect preview. What the game looks like here will be displayed on the final exported executable file. Once the script is edited or modified, it will be updated on both screens. This detection can timely feedback on invisible errors in the terminal. For example, when I finished writing part of the project and tried to export the .exe file, there is no response when changing guns, resulting in the disappearance of the entire gun, there was a sound when the left mouse button was clicked but no effect on the screen. Because there was no error displayed in the terminal, I could only find the relevant scripts and check them carefully and switch between the two screens after modification. In the end, it was found that the method was incorrectly used, resulting in an error in the execution order problem. The gun was actually under the camera. Then In the process of editing the game later, using the Game screen to preview the effect and testing while writing has also become an essential step.

The last test method is the NUnit test, just like the Junit Test in Java, which is a unit test. In unit testing, each test is only concerned with one point and only runs in its own small world (to prevent the tests from affecting each other). This improves code quality and stabilizes completed bug fixes. In Unity, Unit Test is also divided into Edit mode and play mode, which can improve the operation of the game. All tests that are explicit or involve *Update()* need to be placed in Play Mode, and others in Edit Mode.



Figures 5.1. (b) shows one of the test results.

5.2 Gameplay Evaluation

Gameplay is an important evaluation point in this chapter, and making a game fun is the main task. This includes gameplay, interface, character actions, reward mechanisms, and more. If a game just has a moving story, but it's not fun to play or has a difficult control system, then it cannot be considered as a good game. Overall look, graphics, music, mechanics and game balance are all part of evaluating a game.

As can be seen from the *Software Design* section, this project has taken into account the above in some places. First of all, in terms of UI, the choice of low poly style is in line with the story background of the game and is also more in line with the design of AI monsters. The gameplay is also simple to operate, and it is easy to get started. In the settings of some systems, it also increases the content and balance of the game to a certain extent, just like the setting of blood recovery props, which gives players more opportunities to pass, but there are also some other ways to replace them, such as giving players Set more attack methods, which can be completed in future work. In this project, the music part is also well filled, and the various SFX sounds can bring players a better game experience, but when two AI monsters or more (rarely) are killed at the same time, the repetitive sound can be a bit noisy. In terms of plot design, this project is given a plot, but it is somewhat simple and can be optimised into a complete storyline, but in a short period, it is difficult to create an interesting and complete story that matches the game.

For the most important point, fun, five players were invited to experience the game. Four of them are Year3 students from the School of Computer Science and one is from the School of

Engineering also in Year 3. Two of them don't usually play games, and the other three spend almost all their free time on games.The types of equipment used were two Mac, three Windows, and three Windows game computers. For three people with gaming experience, they were asked to pass all three levels during the experience, the clearance time was 26 minutes, 27 minutes and 31 minutes respectively, one of the participants said "*I underestimate this game, I always get beaten to death, which I hadn't thought of*.". For the other two participants who did not play the game very much, one of them gave up after trying the first level six times, and the other insisted on the second level. Of course, to give them a more comprehensive evaluation, I just opened the editor and let them try the third level 3, but both two of them tried 5 times without success. This all proves that this game has certain challenges. In terms of experience and experience, all five people gave ⅘. One of them said, "*The map is well built, especially the last level, and that Boss AI, it's really interesting*." Regarding the missing point, two of them felt that the whole game system was a bit simple, and some plots or props could be added to interact with the players. One felt that although it was fun to play, it was not particularly impressive after the end. The remaining two participants with no gaming experience thought it was fun, but they didn't like the fps mode and thought the third perspective would be better, and the other one felt it was a pity that they couldn't play with friends.

Discussion

Overall, this project has achieved the original goal of creating a complete 3D first-person shooter game against AI, with a certain degree of fun and difficulty. In terms of game content, more things can be added to the current basis to make players more impressed.In terms of operation and computer requirements, the original expectation has also been achieved, and the purpose of running smoothly even on non-game computers has been realised.

Further Work

If a game is to be successful, it needs a certain storyline and difficulty. In a limited time, making a game from scratch with a certain difficulty is not so simple. This project can add more props such as bombs and smoke bombs, which not only increases the fun of the game but also increases the interaction between users and the game. It can also increase his attack methods, such as fists and so on in melee combat. In terms of story, the background story introduction can be replaced by a complete story, and it is a good choice to set up dialogue and interaction with AI colour correction in the game. After experiencing this project, a player suggested that the game can add a multiplayer mode for users to choose 1v1 or kill AI monsters together so that the game can be transformed from single player to multiplayer.

In addition to the optimization of game content, further optimization can also be carried out on the script code. In Unity, the use of the *foreach* method can cause write issues on CPU usage. When the method is called in the *Update()* function frequently, it will cause the GC Alloc, which will generate Small pieces of garbage memory, leading to the early arrival of garbage collection operations, resulting in intermittent freezes in the game. Although replacing Part of the code, but also some scripts that use this method, these scripts can be further optimised to reduce CPU usage.

Chapter 6

Conclusion

the project aims to create a 3D version of a first-person shooter that plays against AI. It has a complete game system and does not require high computer hardware. By running the game many times and experiencing it, it has been proved that this project is a complete and smooth running game.

In this game, a Login & Register system is provided, so that the system can provide two choices of continuing the previous game or starting a new game according to the different progress of different players. On the game page, three levels of maps and difficulty are displayed, and almost every action of the player or the AI monster is accompanied by the corresponding VFX and SFX sounds. In the game, AI has its own system and can switch states at any time in patrol, follow and attack. Once it gets damaged or finds the player is nearby, it will find a path to the player automatically and start attacking. The appearance of BossAI means that the player has passed all the previous obstacles, just kill the Boss AI to get the victory of the game. The victory of the game can only belong to the AI or the player. This project provides a game for players to enjoy leisure time and experience the fun of the game even when they are alone.

References (should be managed using a citation manager)

1. paper references

[1] H. R. Khairuddin, A. S. Malik, W. Mumtaz, N. Kamel and L. Xia, "Analysis of EEG signals regularity in adults during video game play in 2D and 3D," *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2013, pp. 2064-2067, doi: 10.1109/EMBC.2013.6609938.

[2] P.E.Dickson,"Using Unity to Teach Game Development:When You've Never Wrotten a Game", *ITiCSE '15: Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*,June 2015 Pages 75–80https://doi.org/10.1145/2729094.2742591

[3] DELAHAYE, JEAN-PAUL. "The Science behind SUDOKU." *Scientific American*, vol. 294, no. 6, 2006, pp. 80–87, http://www.jstor.org/stable/26061494. Accessed 7 Apr. 2022.

[4] S. Enqvist, H. H. Hansen, C. Kupke, J. Marti and Y. Venema, "Completeness for Game Logic," 2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), 2019, pp. 1-13, doi: 10.1109/LICS.2019.8785676.

[5] F. Ptrillo, M. Pimenta, F. Trindade, C.Dietrich, "What went wrong? A survey of problems in game development", Computers in EntertainmentVolume 7Issue 1 February 2009 Article No.: 13pp 1–22https://doi.org/10.1145/1486508.1486521

[6] C. M. Kanode and H. M. Haddad, "Software Engineering Challenges in Game Development," 2009 Sixth International Conference on Information Technology: New Generations, 2009, pp. 260-265, doi: 10.1109/ITNG.2009.74.

[7] F. Messaoudi, G. Simon and A. Ksentini, "Dissecting games engines: The case of Unity3D," 2015 International Workshop on Network and Systems Support for Games (NetGames), 2015, pp. 1-6, doi: 10.1109/NetGames.2015.7382990.

[8] B. Zhang and W. Hu, "Game special effect simulation based on particle system of Unity3D," 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), 2017, pp. 595-598, doi: 10.1109/ICIS.2017.7960062.

[9] M. Geng and X. Huang, "The research and implementation of artificial intelligence in mobile applications," 2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2015, pp. 556-560, doi: 10.1109/ICSESS.2015.7339119.

[10] Spronck, P., Ponsen, M., Sprinkhuizen-Kuyper, I. *et al.* Adaptive game AI with dynamic scripting. *Mach Learn* 63, 217–248 (2006). https://doi.org/10.1007/s10994-006-6205-6

[11] Everitt T., Hutter M. (2015) Analytical Results on the BFS vs. DFS Algorithm Selection Problem: Part II: Graph Search. In: Pfahringer B., Renz J. (eds) AI 2015: Advances in Artificial Intelligence. AI 2015. Lecture Notes in Computer Science, vol 9457. Springer, Cham. https://doi.org/10.1007/978-3-319-26350-2_15

[12] M. Noto and H. Sato, "A method for the shortest path search by extended Dijkstra algorithm," Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics. 'cybernetics evolving to systems, humans, organizations, and their complex interactions' (cat. no.0, 2000, pp. 2316-2320 vol.3, doi: 10.1109/ICSMC.2000.886462.

[13] A. Candra, M. Budiman and R.Pohan, “Application of A-Star Algorithm on Pathfinding Game”,2021 *J. Phys.: Conf. Ser.* 1898 012047

[14] E. B. Aydoğan and Y. Atay, "Unity Based A* Algorithm Used in Shortest Path Finding Problem for Helicopters," 2021 International Conference on Control, Automation and Diagnosis (ICCAD), 2021, pp. 1-5, doi: 10.1109/ICCAD52417.2021.9638762.

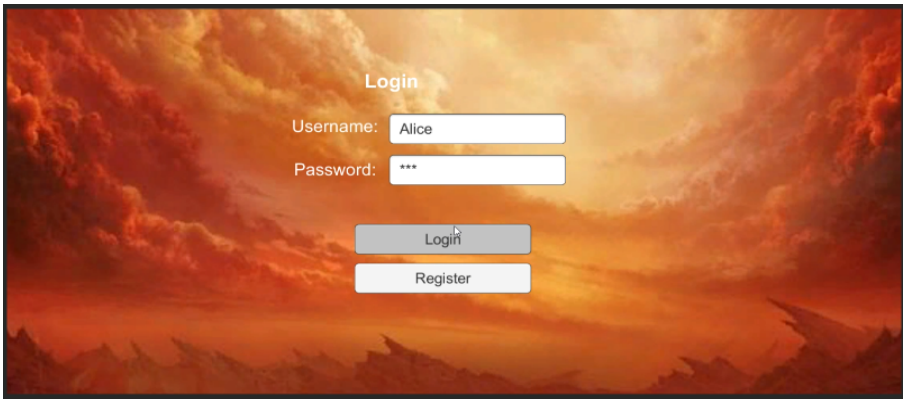
[15] J.Oslin, S.Mitchell and L.Griffin,”The Game Performance Assessment Instruction(GPAI): Development and Preliminary Validation”, Journal of Teaching in Physical Education,1998, Volume 17:Issue 2, page 231-243,DOI: https://doi.org/10.1123/jtpe.17.2.231

[16] Kim, J.-H. (2020). Efficient Controlling Trajectory of NPC with Accumulation Map based on Path of User and NavMesh in Unity3D. *Journal of the Korea Society of Computer and Information*, *25*(4), 55–61. https://doi.org/10.9708/JKSCI.2020.25.04.055

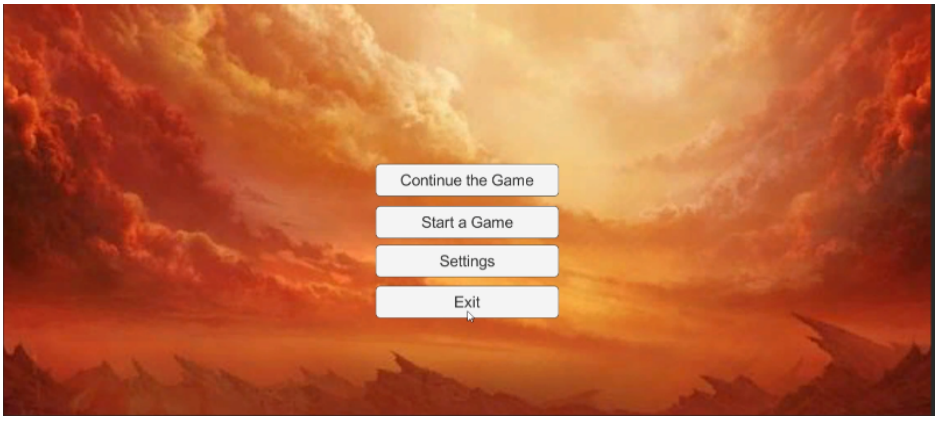
Appendices

A1. Additional UI

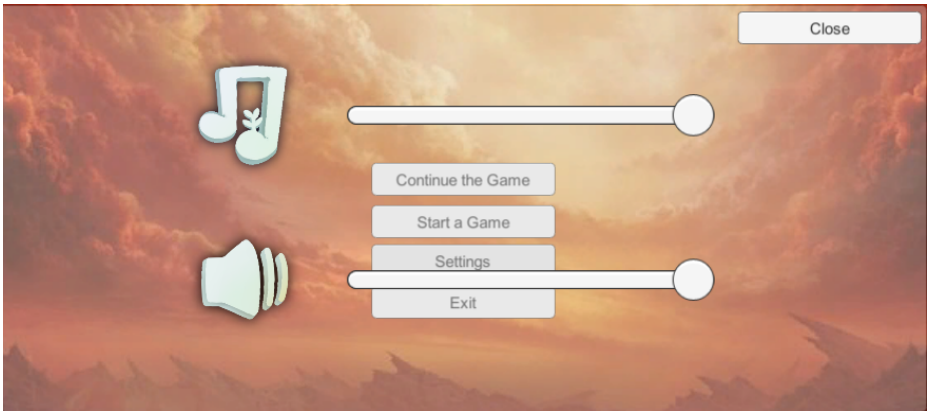
Intro story loading



Login



HomeMenu

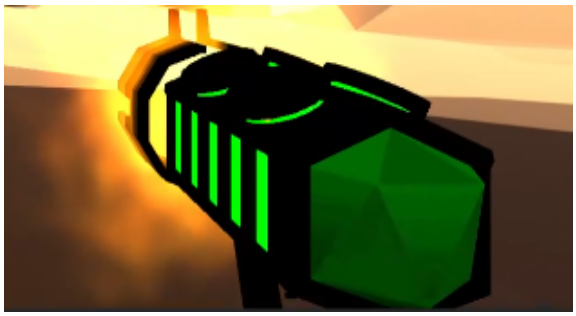
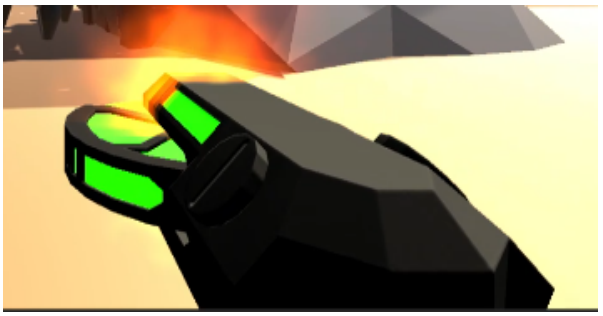


Volume control for Settings






Different types of AI

Weapons 1 & 2 when shooting


Two attacks of BossAI

A2. Test Report

Test report is provided to determine whether all requirements are met. Also, this testing aimed to ensure that the final game is bug-free and without any logical or runtime errors. This report has been divided into two parts, plan and results. The plan shows how the test is structured and the results show the Pass/Fail of each element in the test.

A2.1 Test Plan

The testing of this software mainly uses the BlackBox testing methods. The Expected Result and Actual Result all have laid out in the table. Each expected outcome can be tested by a user to ensure the game works as intended.

A2.2 Test Results

Game Logic

Test Case Number	**Test Case Description**	**Expected Result**	**Actual Result**	**Pass (P)/Fail(F)**
1	Check player moves right when right arrow key or D key is pressed	Player moves right	Player moves right	P

2	Check player moves left when left arrow key or A key is pressed	Player moves left	Player moves left	P
3	Check player moves forward when up arrow key or W key is pressed	Player moves forward	Player moves forward	P
4	Check player moves backward when down arrow key or S key is pressed	Player moves backward	Player moves backward	P
5	Check player jumps from ground when Space is pressed	Player jumps	Player jumps	P
6	Check player can jump over some low objects on maps and move on them	Player jumps to rocks/table and still can move on them	Player jumps to rocks/table and still can move on them	P
7	Check Health Refill prop is able to pick up and the prop works	Player picks up the prop and get full health again	Player picks up the prop and get full health again	P
8	Check prop disappears from the game when picked by a player on map	Power-up disappears when the player picks it	Power-up disappears when the player picks it	P
9	Check the view of degree is moving with the mouse	When the mouse moves, the camera(view)also moves to the left side of the map. Same as the right direction	When the mouse moves, the camera(view)also moves to the left side of the map. Same as the right direction	P
10	Check the shooting of bullets when left click on the mouse is pressed	A bullet is fired when left click of the mouse is pressed	A bullet is fired when left click of the mouse is pressed	P
11	Check the AI monsters patrol on the map if they do not get damaged	The AI monsters walk around the map at each level before the players start attacking	The AI monsters walk around the map at each level before the players start attacking	P

12	Check the AI monsters start following when they receive damage.	When the AI monsters get damaged by the player, they start to speed up.	When the AI monsters get damaged by the player, they start to speed up	P
13	Check the AI monsters start attacking the player when they are inside the attack range	The AI monsters have attacking animations and attacking SFX sound	The AI monsters have attacking animations and attacking SFX sound	P
14	Check the AI monsters also start attacking when the player is very close by their side	The AI monsters start attacking when the player stands by them	The AI monsters start attacking when the player stands by them	P
15	Check the player get damaged when received the attack from the AI monsters	The life count of the AI decreases.	The life count of the AI decreases.	P
16	Check the scenarios of player loses the life	Once the health reaches zero, the “Level Fails” UI will be shown, and players can choose try again	Once the health reaches zero, the “Level Fails” UI will be shown,and players can choose try again	P
17	Check the scenarios of one of the AI monsters loses the life	SFX sound with death will be shown and the AI monster will disappear from the map	SFX sound with death will be shown and the AI monster will disappear from the map	P
18	Check the scenarios of player wins the level	Once all AI monsters benn killed, the UI of “Pass” will be shown	Once all AI monsters benn killed, the UI of “Pass” will be shown	P
19	Check Weapon is changed when Alpha1 or Alpha2 on keyboard is pressed	Weapon changes from 1 to 2 when Alpha2 is pressed, weapon changes from 2 to 1 when Alpha1 is pressed	Weapon changes from 1 to 2 when Alpha2 is pressed, weapon changes from 2 to 1 when Alpha1 is pressed	P
20	Check no weapon changes happens at level1	The user can only use weapon 1 during level 1	The user can only use weapon 1 during level 1	P
21	Check timer updates each second during the game	Timer counts from starting of the game	Timer counts from starting of the game	P

22	Check the star marking system when the player passes 1 level	Star marking appears together with the “Pass” UI	Star marking appears together with the “Pass” UI	P
23	Check Boss AI Idle on map until been attacked	The Boss AI Idle on M	Leaderboard appears at the end of game and show the time and rank for player(s)	P
24	Check Boss AI start Attacking	VFX of fire/ball and SFX sound of Boss attacking will be shown	VFX of fire/ball and SFX sound of Boss attacking will be shown	P
25	Check the scenario of winning the game	Boss AI is been killed and disappear from the map, UI of “Pass” is also shown	Boss AI is been killed and disappear from the map, UI of “Pass” is also shown	P
26	Check return to menu	When corresponding button is pressed, the player will return to main menu	When corresponding button is pressed, the player will return to main menu	P
27	Check the Login & Register System	The player creates an account and login by using it, then main menu screen will be shown	The player creates an account and login by using it, then main menu screen will be shown	P

User Interface

Test Case Number	**Test Case Description**	**Expected Result**	**Actual Result**	**Pass (P)/Fail(F)**
1	Check the intro of the game	The story appears word by word	The story appears word by word	P
2	Check Register button	Account created Successfully with hint message shown up	Account created Successfully with hint message shown up	P
3	Check Login button	Menu Screen should be presented	Menu Screen should be presented	P
4	Check Continue the Game button	The Game Screen of last-failure level should be presented	The Game Screen of last-failure level should be presented	P
5	Check the Start a Game button	The Game Screen of level 1 should be presented	The Game Screen of level 1 should be presented	P

6	Check the sliders for volume adjustment	Volume sliders from 0-100	Volume sliders from 0-100	P
7	Check the Exit button	The program closes	The program closes	P
8	Check the map for each level	Map 1 occurs when the player enters level1; Map 2 occurs when the player enters level2; Map 3 occurs when the player enters level3;	Map 1 occurs when the player enters level1; Map 2 occurs when the player enters level2; Map 3 occurs when the player enters level3;	P
9	Check the objects on maps	Both static and dynamic objects occurs on maps	Both static and dynamic objects occurs on map	P
10	Check the AI monsters on the map to play with	All AI monsters shows up on map	All AI monsters shows up on map	P
11	Check the hint message on screen	Hint message shows on left top of the screen	Hint message shows on left top of the screen	P
12	Check the time counter on screen	Time counter shows on right top of the screen	Time counter shows on right top of the screen	P
13	Check the bullet bar	Bullet bar shows on left bottom of the screen with currently remaining bullets	Bullet bar shows on left bottom of the screen with currently remaining bullets	P
14	Check the Health bar	Health bar shows on left bottom of the screen with currently remaining health	Health bar shows on left bottom of the screen with currently remaining health	P
15	Check the Health Refill prop	The Health Refill Prop shows on set position	The Health Refill Prop shows on set position	P
16	Check the UI of Pass	Pass UI occurs with stars when the player passes the level	Pass UI occurs with stars when the player passes the level	P
17	Check the UI of Fail	Fail UI occurs when the player fails the level	Fail UI occurs when the player fails the level	P
18	Check the try again button	The player reborns at the beginning of the failed level	The player reborns at the beginning of the failed level	P

19	Check the weapons	The model of weapons shows on screen	The model of weapons shows on screen	P
20	Check the VFX of each action	All actions have correct actions: attack, shoot. etc.	All actions have correct actions: attack, shoot. etc.	P
21	Check Boss AI on the map to play with	The Boss AI shows up on the map	The Boss AI shows up on the map	P
22	Check Back to Menu button	The player goes back to the menu screen	The player goes back to the menu screen	P
23	Check the crosshair	The crosshair always in the middle of screen	The crosshair always in the middle of screen	P

Audio

Test Case Number	**Test Case Description**	**Expected Result**	**Actual Result**	**Pass (P)/Fail(F)**
1	Check the background for menu sence/game scene	Correct background music for menu screen/game scene	Correct background music for menu screen	P
2	Check volume adjustment	User can use the sliders to adjust volume, and when setted to 0, the music/sound mutes	Use can use the sliders to adjust volume, and when setted to 0, the music/sound mutes	P
3	Check the sfx-sound for weapons	Shooting sound appears at every fire times; Explosion sound appears after reaching the body of AI; Cool down sound appears after a continuous shooting; Release sound appears after each shooting	Shooting sound appears at every fire times; Explosion sound appears after reaching the body of AI; Cool down sound appears after a continuous shooting; Release sound appears after each shooting	P
4	Check sfx -sound for the AI monsters(including Boss AI)	AI attacking sound appears at every attack; AI Death sound appears after been killed; AI Moving sound appears when they are moving; AI Damage sound appears when AI get damaged;	AI attacking sound appears at every attack; AI Death sound appears after been killed; AI Moving sound appears when they are moving; AI Damage sound appears when AI get damaged;	P

		Different sfx sounds have been set for Boss AI.	Different sfx sounds have been set for Boss AI.	
5	Check the sfx-sound for picking up props	Picking up sound appears when the player crashes the prop	Picking up sound appears when the player crashes the prop	P
6	Check the sfx-sound for weapon changing	Changing-weapon sound appears when the player decide to change the weapon	Changing-weapon sound appears when the player decide to change the weapon	P
7	Check the sfx-sound for the player	Player attacking sound appears at every attack; Player Death sound appears after been killed; PlayerI footstep sound appears when they are moving; AI Damage sound appears when AI get damaged; Land & Jump sound appears when the player does the jump action	Player attacking sound appears at every attack; Player Death sound appears after been killed; PlayerI footstep sound appears when they are moving; AI Damage sound appears when AI get damaged; Land & Jump sound appears when the player does the jump action	P
8	Check the sfx-sound for notifications	Notification sound appears together with UI screen	Notification sound appears together with UI screen	P

Physics

Test Case Number	**Test Case Description**	**Expected Result**	**Actual Result**	**Pass (P)/Fail(F)**
1	Check the jump of the player	The jump has influenced by the gravity	Car accelerates from start and reach a maximum speed, and changes only if slowing down	P
2	Check the movement of the player	The player is able to move in all direction with 360 degrees;	Car decelerates after crashing to a minimum speed and only changes if moving forward	P
3	Check the stopping of the player	The player is able to stop moving whenever during the gameplay	Player can pick up and crash power-ups automatically when moves in the track	P

4	Check the movement of the AI monsters	The AI monsters are able to move in all direction with 360 degrees;		
5	Check the collision with the wall	The player and all AI will stop moving after crashing the wall boundary until they change a direction	Car will stop moving after crashing the wall boundary	P
6	Check the collision between the AI and player	All AI will start attacking due to the little distance	The car which crashes another car rebound a distance, and the crashed car moves forward a distance	P
7	Check the collision between the two AI monsters	Both of them will find a new path and separate	Car moves a distance forward and stops	P
8	Check the weapon when shooting	The weapons have different shooting speed and damage	The weapons have different shooting speed and damage	P

Artificial Intelligence

Test Case Number	**Test Case Description**	**Expected Result**	**Actual Result**	**Pass (P)/Fail(F)**
1	Check the AI monsters moves on the path	The AI monsters can move and change state following the logic	AI car moves around track in anti-clockwise direction	P
2	Check the types of the AI monsters	There are more than 3 types of the AI monsters on one map	The color of AI car is the same type as the chosen of user	P
3	Check the pathfinding(follow state)	The AI monsters are able to find a path to the player	The AI car complete a race without crashing	P
4	Check the obstacle avoidance	The AI monsters are able to detect and avoid obstacles correctly	The AI car speeds up at the start and then reaches a constant speed until passing finish line	P
5	Check theI group Behaviour	Each group of the AI monsters have the same actions	Each group of the AI monsters have the same actions	

6	Check the patrol state	The AI monsters should not attack the player until they get damaged unless the distance between the AI and the player is too short	The AI monsters should not attack the player until they get damaged unless the distance between the AI and the player is too short	
7	Check the Attack state	The AI monsters should starting attacking once they are inside the attack range	The AI monsters should starting attacking once they are inside the attack range	
8	Check the Boss AI	The Boss AI can change to different direction when the player is moving	The Boss AI can change to different direction when the player is moving	
9	Check the Idle State	The Boss AI waits at origin position	The Boss AI waits at origin position	
10	Check the walk state	The Boss AI follows the player at a slow speed	The Boss AI follows the player at a slow speed	
11	Check the Attack of Boss AI	The Boss AI should starting attack right after been attacked	The Boss AI should starting attack right after been attacked	

VFX

Test Case Number	**Test Case Description**	**Expected Result**	**Actual Result**	**Pass (P)/Fail(F)**
1	Check the VFX of map	Particle System works on maps	Particle System works on maps	P
2	Check the VFX on the AI monsters(including the Boss AI)	When get attacked, flash and explosion vfx appears on the body, angry and alert vfx on face; angry steam vfx shows above the head	When get attacked, flash and explosion vfx appears on the body, angry and alert vfx on face; angry steam vfx shows above the head	P
3	Check the VFX on/from the weapons	Gun vfx shows up for weapons; Disk Orbit particle vfx shows for weapon2; Hits spark and muzzlefalsh vfx shows when fire a bullet;	Gun vfx shows up for weapons; Disk Orbit particle vfx shows for weapon2; Hits spark and muzzlefalsh vfx shows when fire a bullet;	P

		Red, blue, green laser vfs shows the trajectory of bullets	Red, blue, green laser vfs shows the trajectory of bullets	
4	Check the light VFX for the whole map	Directional light VFX appears when the game screen is loading, every object is with shadow	Directional light VFX appears when the game screen is loading, every object is with shadow	
5	Check the VFX for props	Health Refill prop showa dynamically on maps; When been picked up, pick up sparkles vfx is shown	Health Refill prop showa dynamically on maps; When been picked up, pick up sparkles vfx is shown	