VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Gong Boyuan

DESIGN AND IMPLEMENTATION OF 2D HORIZONTAL VERSION LEVEL GAME BASED ON UNITY3D ENGINE

Technology and Communication

2020

FOREWORDS

This is my final thesis about design and implementation of hexagon elimination game based on Unity3D engine. I would like to express my appreciation to all people who have given me helped during the whole period I finished my final project.

The first person I would like to express my appreciation is my thesis supervisor Prosi Pirjo. During the whole period, she help to decided my topic of my thesis project and discussed with me in patiently. Under her help and guidance I have already learned a lot about the Unity3D engine and game developing knowledges to make sure I can design and complete my project.

I also want to say a grateful thanks to my family who gave me support let me study in Finland.

And also thanks to the all staff of VAMK for their guidance and teaching.

Finally, thanks to my friends.

Gong Boyuan
Vaasa, Finland

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Degree Program in Information Technology

ABSTRACT

Author	Gong Boyuan
Title	Design and Implementation of 2D Horizontal Version Level Game Based on Unity3D Engine
Year	2020
Language	English
Pages	58
Name of Supervisor	Prosi Pirjo

This topic focuses on the design and implementation of a 2D horizontal version acting game. This level game was made with the Unity3D software and it was combined with the C# language as script and applied to the Visual Studio 2019 to develop.

The main core gameplay of this game is to control the game characters across the terrain obstacles and kill the monsters and Boss to get a certain score to pass the level within the prescribed time. The game sets three different levels, each levels vary from easy to difficult according to the difficulty level of the game. The terrain obstacles, monsters and attack power of each level are different.

For the development of the game, the art materials ware prepared. These art material were collected from the public game art material website, and then, all the material was transformed into a format which unity can use it and import to the development engine.

C# was the programming language for this project, the version is 3.5.0-beta4.And the Visual Studio version is 16.5.4, the version of Unity3D is 2019.3.10f1.

Keywords Unity3D; C#; Level Game

CONTENTS

LIST OF FIGURES AND ABLES

1 INTRODUCTION

1.1 Purpose

This subject is based on the Unity3D game development engine to create an anime character fighting game called "Road of Kings". This game has selected several high-profile animes. These animes are very popular all over the world so the game can attract a wide range of players, and it will be more popular than a game that only relies on a single anime as the theme. All the character materials are from the public material platform(www.6m5m.com), and are paid to download.

Unlike the traditional horizontal version of the game, this game adds a "fighting" element, making the player's exploration process more thrilling, while setting up different enemy characters and special traps, and designing different levels based on the plot of the animation. This greatly improves the game's playability, making the game content richer.

1.2 Background

With the rapid development of politics, economy, culture, and information technology in recent years, China is developing into the direction of online gaming power. The development of online games has also spawned the emergence of anime games and anime game industry. Anime games derived from anime have become an indispensable part of many young people's lives. Judging from the number of broadcasts throughout the year, Japanese animation works with higher rankings include "King of Sailing", "Naruto" and "Detective Conan". The current international animation industry, the United States, Japan, and China are in a leading position. Japan's animation annual output value ranks sixth in the national economy, and the export value of animation products exceeds steel /22/. The

animation industry relies on fictional characters to create a huge wealth. The animation and game industry is developing rapidly world-wide, and the demand and production capacity of the Chinese animation and game market has begun to expand. The animation game industry is a derivative industry of the animation industry. Because of its characteristics and advantages such as low pollution, low energy consumption, multiple employment opportunities and high industrial value, it is hailed as the sunrise industry in the 21st century. It also provides the spirit for the people. One of the important channels for cultural needs.

While the animation game industry is booming, the smart game technology of mobile phones is also constantly developing. The iterative update of smart-phones and the popularization of 5G networks have created a large and perfect stage for smartphone games. At the same time, the country is also actively affirming the game intelligence industry, and many universities have set up related courses. As the development prospects of modern mobile games are growing, people's patience with mobile phones is also increasing. Compared with computers, mobile phones are more convenient to carry and can play an entertainment role in people's lives regardless of location.

1.3 Overview Structure

This thesis project used Unity3D as a game engine. From the beginning of the engine to the current development, its function is very mature and easy to use. This engine, it provides a level editor, so that game developers can make level games more convenient, and at the same time, this game combines the C # language as a script to achieve the development of games This will enable developers to develop games they want to develop very quickly and easily without having to understand and master those complicated technologies.

2 TECHNICAL BACKGROUND

2.1 Introduction of Game Development

Video game development is a process that develops a game program. This work can be done by a developer, or through a team or an international team. Traditional large-scale games are funded by developers and usually take several years to complete. Independent games usually cost less time and money, and can be completed by individuals or small developers. In recent years, the independent game industry has been growing this is facilitated by the growthy of the game development software, for example Unity engine and Unreal Engine or other game development platform. Through years of technology accumulation and cultural precipitation, video games have gradually formed research in multiple frontier fields. /21/

In the field of game design, research in the following areas /23/:

Gamification Study how to apply game concepts to a wider range of fields.

Emotional design Mainly involves how to give the game more artistic meaning, and how to build emotional related topics in the game

Interactive storytelling Study interactive narrative under the interactive premise of the game, this field mainly emphasizes the uniqueness of interactive narrative compared with traditional narrative.

Serious game Mainly represented by educational purposes, often combining narrative, technology and other fields.

Establishment of perfect game design methodology The MDA framework is the representative, and there are many game, and the goal is to establish a perfect and systematic game design methodology.

In the field of game technology, research in the following areas:

PCG(Procedural generation): Procedural generation is the key research object in

recent years, the cutting-edge and meaningful capacity is to generate game design on the basis of technology, and then to the ultimate goal-to generate a complete game

Emergent gameplay: Mainly involved in creating a playability that exceeds the designer's expectations and even exceeds expectations within a controllable range.

Artificial intelligence and Games: It is mainly influenced by the artificial field in recent years to get rid of the artificial intelligence winter step forward. The most cutting-edge direction, mainly focusing on various applications of machine learning in games, including online in-game intelligent enhanced learning, AI director and offline AI training, and content customization.

Cognitive science and Games: The main research direction of this project is to calculate and quantify the psychological and physiological reactions that people produce in the process of playing games, as well as feedback on the actual game process, to help open and design the game.

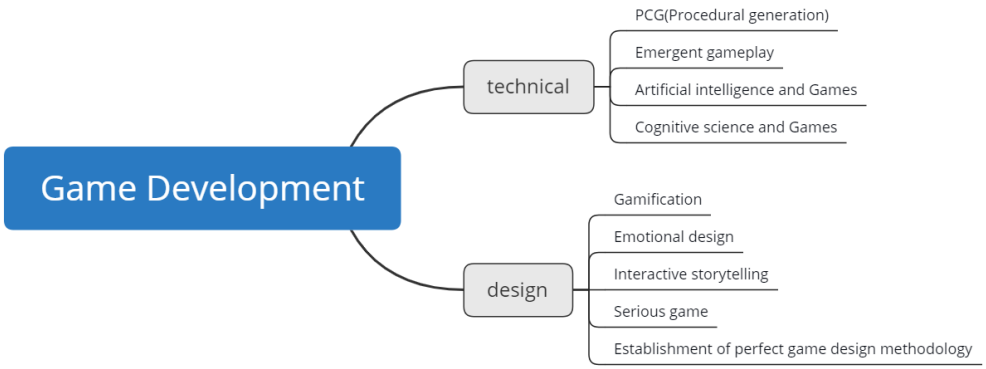

Figure 1. Game Development Research Areas

3 RELATED DEVELOPMENT TOOLS AND TECHNOLOGIES

3.1 Introduction of Unity 3D Game Engine

Unity3D is a software developed by Unity Technologies to make games. When this game software started, it only supported the development of 2D games. With the development of time, 2D games can no longer satisfy people's daily fun. On the 2D plane, the game can only be run in a two-dimensional space, not three-dimensional enough, and there is a big gap with real life, 3D games were discovered and invented, and Unity3D games stand out among many game development software.

The characteristics of Unity3D are as follows /24/:

(1) Strong comprehensive compilation ability

Unity3D is powerful. Only a little basic knowledge is needed to use the Unity3D's compiler. It is suitable for novices to learn, and its compilation ability is strong and simple.

(2) Graphic power:

The Unity3D engine has excellent picture rendering technology, which can help the game scene to quickly render.

(3) Resource import is relatively simple:

Unity3D resource material import is very simple, to import materials or resources in the main scene is done by pressing and holding a button. The general operation is to directly pull materials or resources simply into the scene, suitable for novices to learn.

(4) Simple deployment:

Unity3D can be published on different platforms, showing the work to others on different platforms.

(5) Simpler development:

Unity3D is by far the simplest game development platform on the market, and it is free for personal development.

(6) Shader:

Unity3D has a powerful shader system, which has the advantages of ultra-high performance and simple operation.

Unity's main interface is divided into several modules, and each module has its own unique functions and scope of work. If you want to learn to use Unity to make games, you need to understand and be familiar with the layout of Unity's main interface.

The Unity editor includes six main views and windows:

(1) Project: This view is responsible for storing all the resources of the entire project, including game objects, C # scripts created in Unity, and pictures and music added from the outside.

(2) Hierarchy: Each game will have multiple scenes, such as login scenes and menu scenes. The scope of the hierarchical view is all project resources of the current scene. The project view is to manage all the scenes of the entire game.

(3) Scene: The game interface can be operated and edited in the scene view. Any modification in the scene view will directly affect the final game interface screen.

(4) Game: The game view may be included in the scene view. The game view is in the "run" mode for the scene view, which means that the game view displays the "run interface" of the scene view. What the game player finally sees and experiences is the interface in the game scene.

(5) Inspector: The property view is a window that all objects in the game have. There are various properties of the Game Object, including position, and size, and various components can be added to the Game Object.

(6) Console: The console window mainly responds to and displays the errors and warning messages generated during the running of the game, which can be used by developers for reference.

3.2 Introduction to the C# language

C # is a highly functional computer language. It is an object-oriented programming language developed based on C and C ++. Because of this, it combines the advantages of C and C ++. At the same time, C # operation is very simple, and comes with high efficiency feature; it has become the preferred development language of Unity3D with its simple language style and convenient operation methods.

The C # language can help programmers to save time in developing programs. People who know both C and C ++ can adapt to the C # language in a short time, because the functions between them are roughly the same, The relevance and intersection feature allows programmers to switch between different languages at

will.

As an object-oriented programming language, C # can be used in a variety of development platforms. This software provides many tools and services to help programmers develop game scripts on the largest possible level. The powerful functions of the C # language are also perfectly applied in Visual Studio 2019, and in this project the C# version is 3.5.0-beta4.
/15,20/

3.3 Introduction of Visual Studio 2019

Microsoft Visual Studio (abbreviated as VS) is a series of development kit products of Microsoft Corporation in the United States. Visual Studio is a basic and complete development tool set, which includes most of the tools needed throughout the software life cycle, for example UML tools, code control tools and integrated development environment (IDE). The object code written is applicable to all platforms supported by Microsoft, including Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework, Microsoft Silverlight and Windows Phone.

Visual Studio 2019 installs the Live Share code collaboration service by default to help users quickly write new welcome windows for code, improved search functions, and overall performance improvements; Visual Studio IntelliCode AI help; better Python virtual and Conda support; and support for WinForms and WPF Including .NET Core 3.0 project support.

Visual Studio is the most popular integrated development environment for Windows platform applications. The latest version is Visual Studio 2019, based

on .Net Framework 4.8. In this project, the Visual Studio version is 16.5.4. /19/

3.4 Introduction of Sprites

Sprite is a control used to draw a collection of pictures. Sprites that are usually created in games, such as player characters; have animation effects during movement, and the realization and control of such animations need to be achieved through sprites /9/.

4 GAME PLANNING

All games should have a gaming plan, so in this part, a detailed plan for the game is first developed. The activity of demand analysis is actually a process of communicating with players, allowing developers to understand the requirements mentioned by players.

This process is conducive to improve the efficiency of developers in developing games. In this process of requirements analysis, developers will have to write relevant documents for requirements analysis in accordance with certain specifications and standards, which will enable developers to understand better the various requirements in the requirements analysis report, as well as the future development process.

The specific functions to be implemented are as follows:

(1) The start game module includes:

Game start

Game instructions

Exit the game.

(2) The game level module includes:

Main interface of the game

Pause and continue the game

Sound effect settings

Game score

The time spent

(3) Victory interface modules include:

Next level

Restart

(4) The failed interface modules include:

Restart

Back to start interface

4.1 Plot Design

The anime continent fell apart, and in a mess, Sibo, the sister of the inferior student of the Magic Department High School, Sibo Shenxue, was captured by an unknown masked man who threatened to destroy this anime world. At this time, the situation was at stake, and Spodar also embarked on the path of finding his sister.

Only by defeating the anime characters summoned by the masked people from other secondary units one by one can he rescue his sister Siberian Snow and save the world. This game is based on the background, telling the story of the host company Poda also defeating the enemy in the anime world to save the world on the road to the king.

4.2 Character Design

There are two main characters in this game, the protagonist representing justice and the monster and villain representing evil.

The host company Poda represents the role of justice and the character representing evil are Ninja dogs in Naruto World, navy in the world of King of the Seas, Gageru Leitfox and Naz Doragnier in Fairy Tail World, Qi Zhi in the world of Hunter, and tutor Tsuda Yoshida of the world.

4.3 Scene Design

The design of the game scene is simply to build the scene. This game has a total of three main scenes, which are the initial interface and the main game scene; it also includes three scenes that need to be activated to appear, which are game clearance and game failure scenes. The details are as follows:

(1) The initial scene shows the name of the game "Road to the King", on the bottom there is the start game button and the game description button, the top right is the exit game button, and the upper middle shows the elapsed time.

(2) The main scene of the game is the game level. The main character moves by jumping and walking, the monster can automatically patrol and the monster will attack by the player's walking. There are blood and mana bars of the main character in the upper left corner and display for the number of gold coins and the number of points. The lower left corner has the back button, pause button and sound switch button.

(3) The "Next Level" and "Replay" buttons will be displayed for each level completed

(4) If the game fails, the "Replay" and "Back" buttons will be displayed.

4.4 Level Design

This game requires players to achieve a certain score by killing the villains in each level within a specified time to obtain a certain score: kill a mob and get 1 point, and kill BOSS and get 5 points. The first level requires 10 points, the second level requires 12 points, and the third level requires 18 points. Players can get gold coins

when jumping and moving. When the character got injured or need insufficient mana to release skills, the health amount and mana amount can be recovered by eating the sun and water drop props in the game. There are three levels in this game:

(1) The first level has 8 small monsters and 1 BOSS, adding a box that will disappear when the protagonist steps on it. Players need to kill at least 5 little monsters and solve the BOSS to get 10 points to proceed to the next level.

(2) The second level has 11 monsters and 1 boss, adding moving gears and fired cannonball traps. Players need to kill at least 7 monsters and solve the BOSS to get 12 points to go to the next level.

(3) To the third level with 12 small monsters and 2 BOSS, the moving step terrain has been added. Players need to kill at least 8 monsters and solve 2 BOSS to get 18 points to advance to the next level.

4.5 Worldview Design

The world view of the game represents the game designer's view of the world. Through this game, players can perceive the emotions and ideas that the game designer wants to express. Facing a crisis, are you willing to stand up and protect this loved world?

5 SUMMARY DESIGN

5.1 Game Function Framework

This game is divided into 5 interfaces: game start interface, game main interface, pause interface, clearance interface and death interface that need to be triggered. The main interface of the game includes the main battle scenes.

The Game function framework is as figure 2 shows.

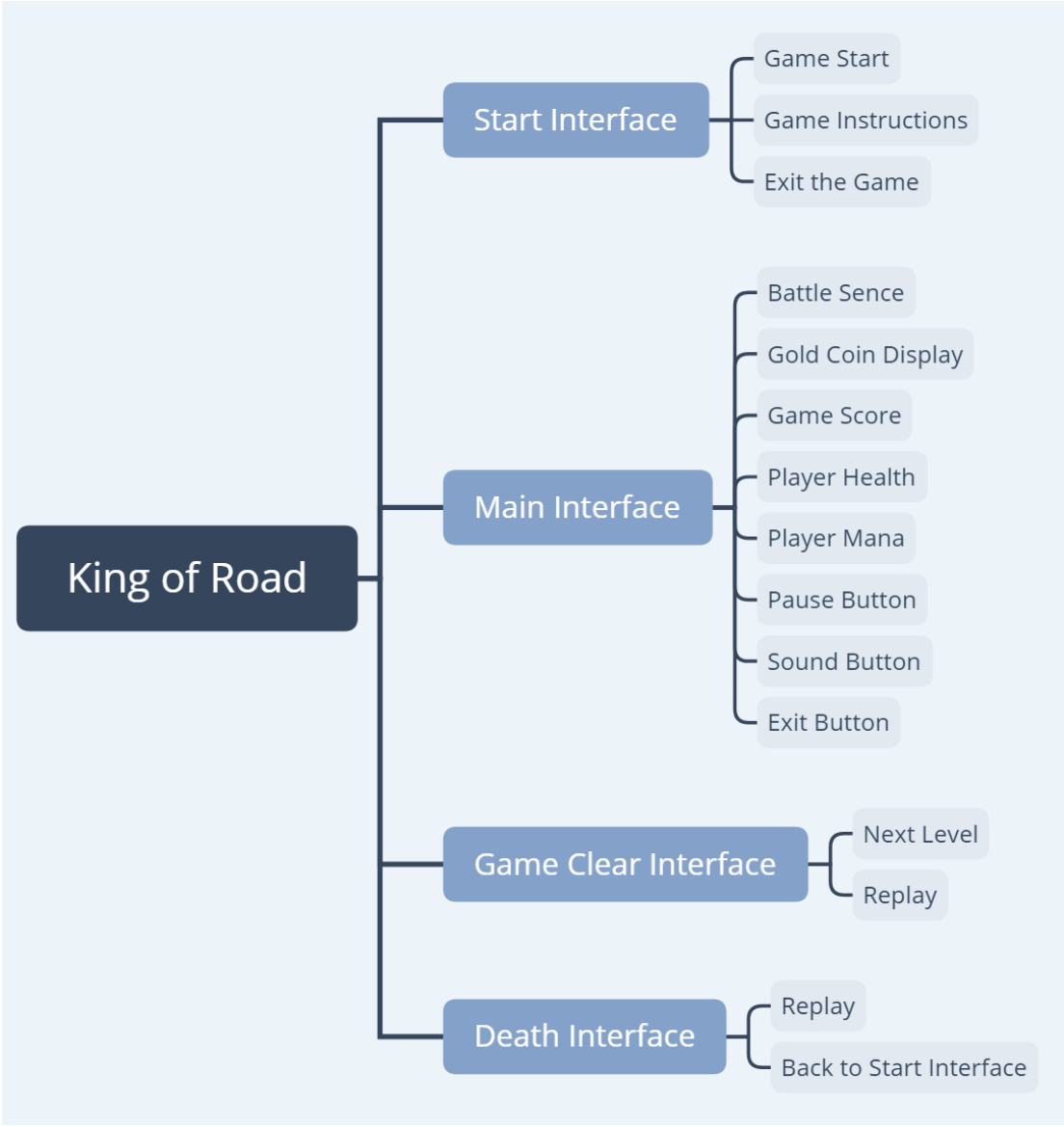

Figure 2. Game function framework

5.2 Game Flow Chart

Since this game is a one-stop game, there is no need to log in and register. Click the program to run, click "Start Game" to enter the game. After entering the main interface of the game, the player can start playing the game. The upper left corner of the game interface displays the character's blood volume, mana quantity information, current gold coin number and score information, and the lower left corner has a pause button, sound setting button and exit game button.

When the player enters the level after starting the game, the aim is to kill the monsters and BOSS in the level within a fixed time to obtain enough points. Then a pop-up window will appear, whether to proceed to the next level or choose to replay the game. Play or choose to quit the game.

It is worth noting that, because this game almost meets the requirements of one-life clearance, when you choose to quit the game, the game needs to start from the beginning. When the player explores all the cards, the game is cleared.

The total flow chart is as Figure 3 shows.

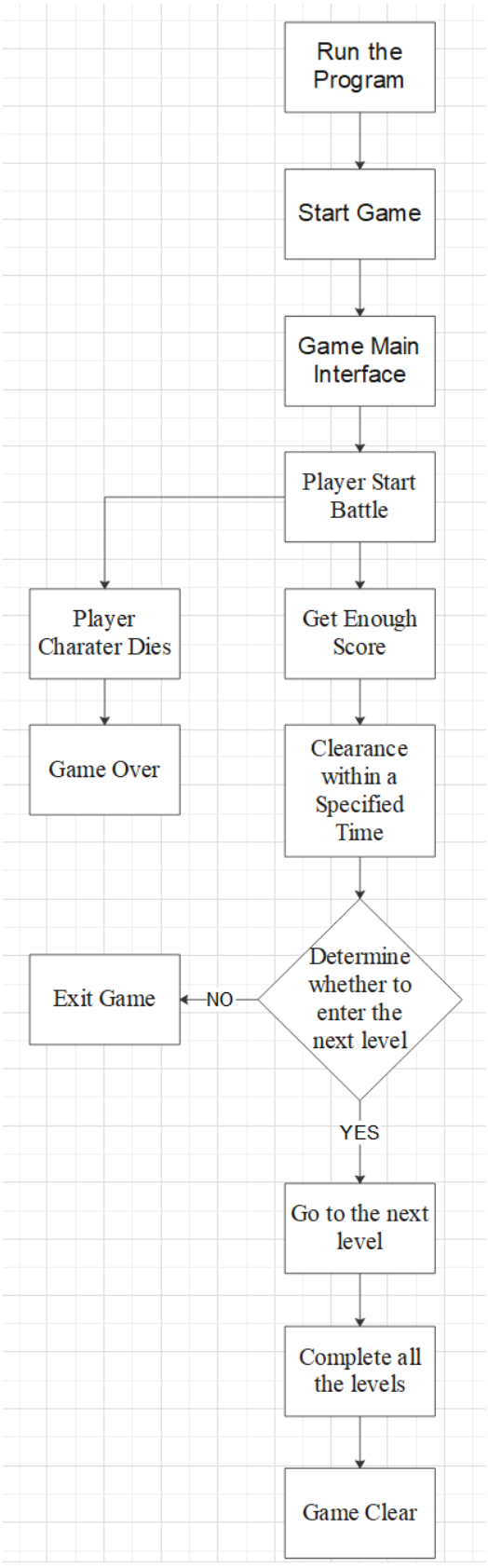

Figure 3. Game Flow Chart

The initial health of the player is 100, and the damage value of the monster attacking the player is 10 each time. The monster will automatically patrol back and forth.

When the player walks near the monster, the monster will detect whether the player enters the attack range. If it enters the attack range, the monster starts to attack the player. The monster's attack is only effective for the player.

When the monster hits the player, the player drops blood and changes the current condition of the blood bar by changing the FillAmont value of the blood bar to fill the picture.

Players need to keep moving and attacking to avoid the loss of their blood volume. When the player's blood volume <= 0, the monster stops attacking, the game is suspended, the player disappears, and the game is judged as a failure.

In order to reduce the difficulty of the game, the option to replay the level is set in this round. Players can choose to replay the level or return to the initial interface to start from the beginning.

The monster attack flow chart is as Figure 4 shows.

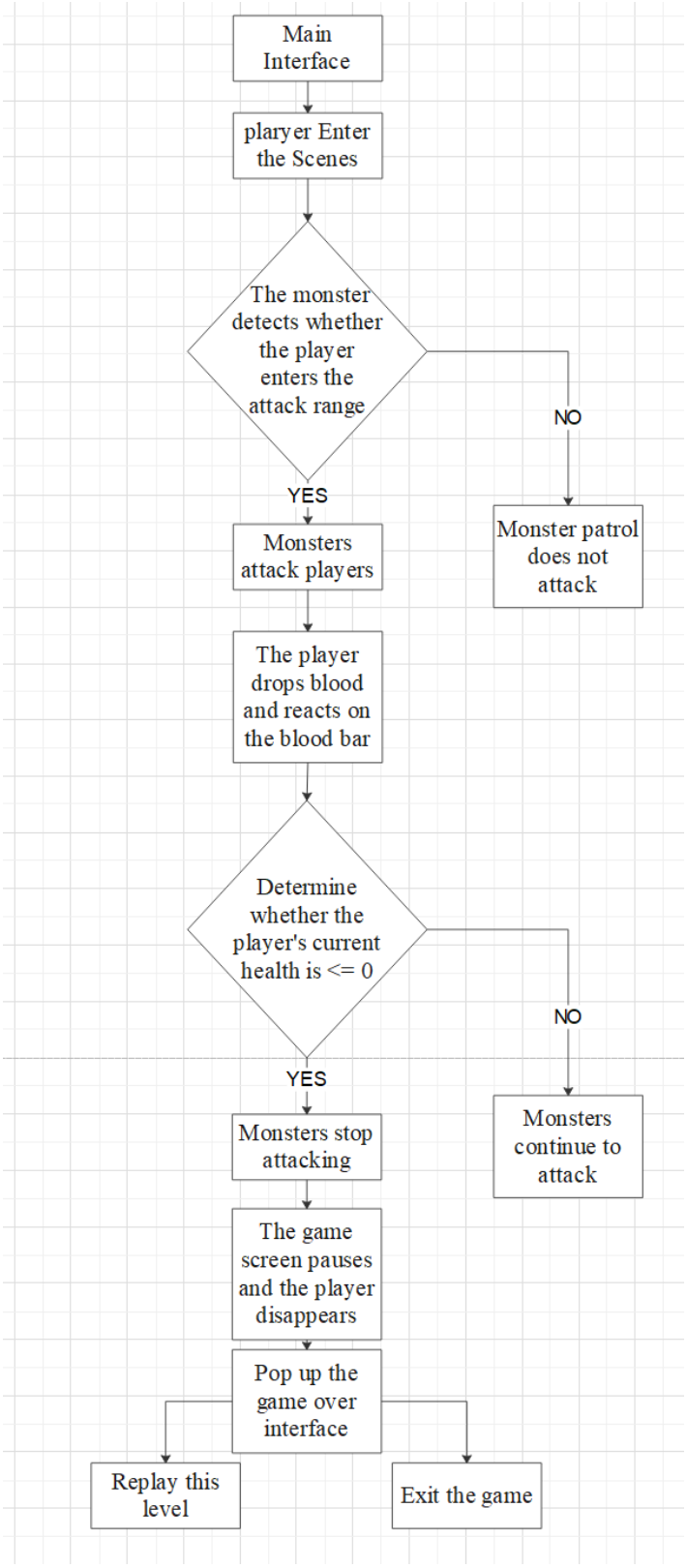

Figure 4. Monster Attack Flow Chart

The enemy's initial health is also 100, and there are ordinary attacks and special attacks on the player character.

The damage value of the ordinary attack is 15, 20, 25 / time randomly. The damage value of the special attack of one skill is 30 / time, two The damage value of the special skill attack is 35 / time, the damage value of the third skill special attack is 40 / time, and the damage value of the four skill special attack is 45 / time. When players enter the battle, they can only pass a certain score by killing monsters within the specified time to complete the level.

Levels 1, 2, and 3 need to get 10, 12, and 18 points in sequence within the specified five minutes to complete the level. When the time to go through the barrier is <= 300 seconds, it is judged that the scores obtained by killing monsters in the first, second, and third levels> = 10, 12, and 18 points. If not, continue to destroy the monsters to obtain points; turn off.

The player attack flow chart is as Figure 5 shows.

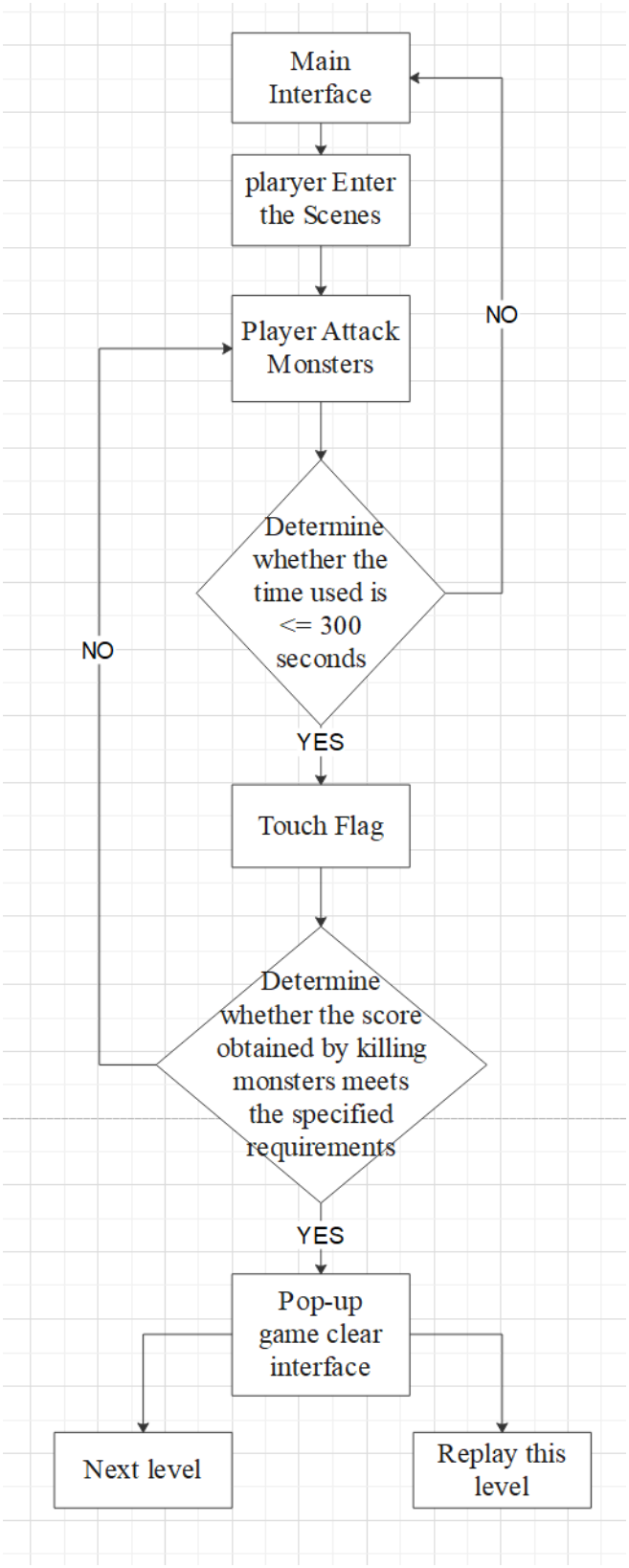

Figure 5. Player Attack Flow Chart

6 DETAILED DESIGN OF THE GAME

6.1 Game Scene Construction

The game scene constriction will be presented in the following sections.

6.1.1 Start the Game Scene and Its Functions

The starting scene of the game mainly has the game descriptions, exiting the game, starting the game three functions, first creating an image, then setting the full screen size, selecting an image to assign it, using this image as the background of the starting scene.

Then an image is created, its position is set in the upper right corner, and then give it a cross image is given as a button to exit the game, a button component is added to the image, and then button event is bound, here through the Application.Quit() method to exit the game.

Then the same method is used to create the start game button and, the SceneManager.LoadScene() method is used to load the game scene.

Finally, a game description page is made. This page consists of an image, a text, and a button. The image is used as a background of the page, text is used to display the description text, and the button uses the gameObject.SetActive() method to close the description page.

The features of the Game Scene and Its Functions are based on references /4/, and /7/.

Figure 6. Game Start Interface

6.1.2 Game Level Scene Construction

The game consists of three levels, each with different backgrounds, roads, props, enemies, and such as

(1) First, the background material is selected for the first level, a sprite is created for background display, the size and position are set, then, this sprite is copied, the position is adjusted backwards until the entire length of the level is covered, and then an empty object is created.

Four box colliders are added on the top and their size is adjusted to the top, bottom, left, and right of the level to limit the border of the level. Then the parcel

material is selected and the parcel material is used to arrange the roads of the game scene. There are multiple parcels and arrange them into suitable roads for the game.

Each parcel must include the corresponding collision body component, and then the tree and grass are joined. After the decorative materials are added to the scene to decorate the game scene, game props are added, including gold coins, water drops, and sun. The position of the props is arranged according to the road of the game level. Of course, the props also need to add collision body components, and finally the enemy is added into the game scene.

(2) The second and third levels of the game are modeled on the scene layout of the first level, changing background images and other materials, changing the layout of roads and props, changing the type of enemy, and finally adding new gears and battery props to the game The scenes of the three levels are arranged. The arranged level is as shown in Figure 7.

Figure 7. Schematic Diagram of Game Levels

The feature of the game level scene is based on references /1/ /9/ and /14/.

6.2 Protagonist Related Function Design and Implementation

The creation of the protagonist related function is described next

6.2.1 Introduction of Animation

Unity has a rich and complex animation system, which provides developers with simple and effective animation import and production functions.

Unity's animation system is based on the concept of "animation clips" and records attribute information about how objects change over time. Through the Animator controller to control and track, the animation will be changed and merged in time.

The character animation of the game is made by using picture frame animation, which is created by animation in unity. First, the corresponding protagonist is selected, then the animation page is opened, a new animation is created, the corresponding picture material is dragged on the ground of the corresponding frame, and the loop set to false .A single-play animation is created, the created animation is shown in Figure 8.

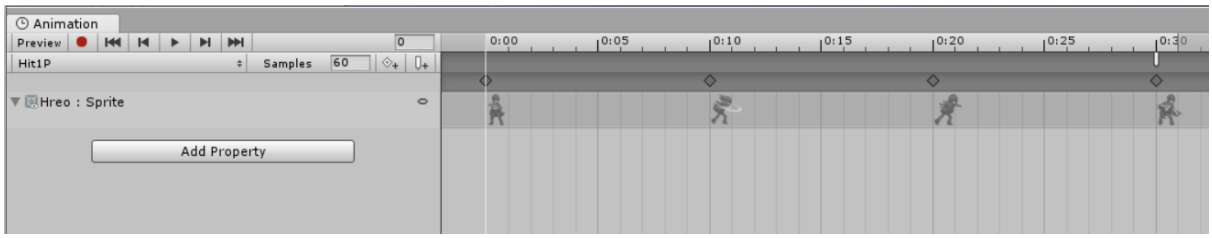

Figure 8. Animation

Then the same method is used to create multiple animations of the protagonist's standing, moving, attacking, skills, etc. After this, an animation controller needs to be created to manage the previously created animation.

Here, the animation controller is selected and then switched to the animator interface. First, all the animations are added, and then transitions is created according to the switching relationship of each animation to connect all the animations. In order to control the animation switching, an int value is created as a

condition for the animation switching, then all transitions are set according to actual needs. The switching conditions of the final animation controller created as shown in Figure 9.

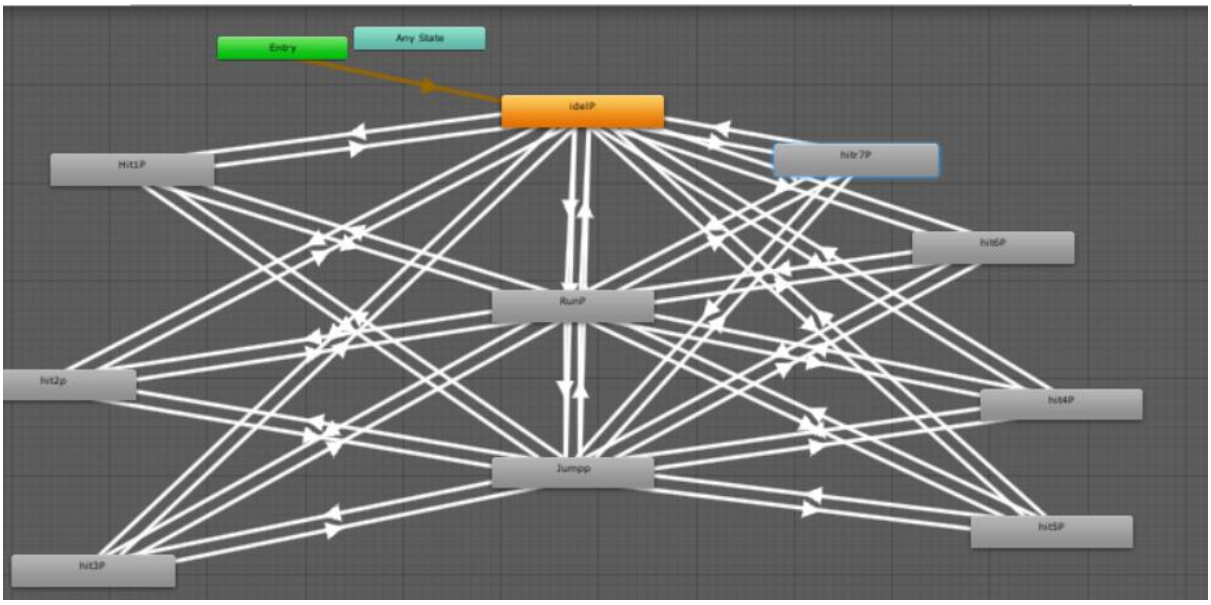

Figure 9. Final Animation Controller

6.2.2 Implementation of Protagonist Function Code

(1)Protagonist moving and jumping function: The protagonist can move through a,d. Here, firstly, input.GetKeyDown () is used to detect the pressing of two keys of a,d.

When pressing is detected, the protagonist's face must first be changed by changing the rotation of the protagonist.

Then the animation state of the protagonist must be changed through GetComponent <Animator> (). SetInteger (), the moving animation of the protagonist is played, and then the value of the current player pressing the ad key is got through Input. GetAxis ("Horizontal"), Finally, Translate method is used to move the main character.

In this way, Input.GetKeyUp () is still needed to be used to detect the player's key lift. When the player lifts the key, it means that it does not move. At this time, the animation is switched to the standing animation.

Next is the protagonist jumping function. Here we first need to create a bool value to show whether the current protagonist is in the jumping state, and the space bar is pressed to detect the key, so that when the space is pressed, it is judged whether the current is in the jumping state.

If it is, this is ignored. If the key is not pressed, the current jumping state is set to true, and then an upward force is given through the "AddForce" method to the protagonist. The protagonist jumps up, and finally protagonist's animation state is changed to jumping statement, and the jumping animation is played.

The same as with the movement, here the jumping state must be stopped and switching back the animation, so OnCollisionStay2D is used for collision detection. When the protagonist is in collision with the ground, the speed of the protagonist was not judged in the y-axis direction. If it is equal to 0, Then it means that it is currently standing on the ground. At this time, the jumping state should be equal to false and if the jumping animation is still playing at this time, the animation state is switched and the standing animation is played.

```
if (Input.GetKeyDown(KeyCode.Space)&& !bJump&& MyHit==0)
{    //如 if enter space
    bJump = true;
    rig.AddForce(new Vector2(0, jumpForce));   // give a force to object
    this.transform.GetComponent<Animator>().SetInteger("State", 2);//animation state changes  2 is jump  1 is run  0 is stand
}

if (Input.GetKeyDown(KeyCode.LeftArrow) && MyHit == 0)
{
    this.transform.rotation = new Quaternion(0, 180, 0, 0);//turn a round
    this.transform.GetComponent<Animator>().SetInteger("State", 1);
}
if (Input.GetKeyDown(KeyCode.RightArrow) && MyHit == 0)//enter rightarrow button

{
    this.transform.rotation = new Quaternion(0, 0, 0, 0);//turn a round
    this.transform.GetComponent<Animator>().SetInteger("State", 1);

}
if (Input.GetKeyUp(KeyCode.LeftArrow) && MyHit == 0)
{

    this.transform.GetComponent<Animator>().SetInteger("State", 0);
}
if (Input.GetKeyUp(KeyCode.RightArrow) && MyHit == 0)
{

    this.transform.GetComponent<Animator>().SetInteger("State", 0);

}
```

Figure 10. Player Character Move Function

(2) Protagonist attack function: The protagonist has two kinds of attacks, namely ordinary attack and 4 skills.

Since the attack needs to have a hit determination, the collision detection used here is used for attack determination, creating two empty objects as the protagonist's sub-objects, and then according to the distance of the protagonist's attack, the position creates the collision body component and sets the position size, and then hides the two empty objects.

A code is added to the empty object, using OnTriggerEnter2D and OnTriggerStay2D in the code for collision detection, when the empty object collides with other objects, the tag of the collided object is determined. If the tag of the collided object is monster, then the monster blood amount is reduced. Then the control code of the attack of the protagonist is implemented. First, the ordinary attack is to press the Q key, so here the pressing of the Q key needs to be detected.

When pressed, it should be determined whether the current state is in the attack state. If the protagonist is not in the attack state, switch to the attack state.

Since there are three animations for ordinary attacks, a random number is needed, and then the corresponding attack animation is played according to the random number, the corresponding collision body sub-object is displayed, and the animation event is added at the end frame of the attack animation to end the attack event.

The object collides with the body and switches the animation so that the current protagonist's attack status becomes false. The protagonist's skill attack is to press the keys 1, 2, 3, and 4, respectively. This method is basically the same as the ordinary attack method, except that the mana amount of the protagonist needs to be determined each time a skill is released , a certain amount of mana need to be reduced. Mana is not enough to activate the skill.

```
if (Input.GetKeyDown(KeyCode.Q) && MyHit == 0)
{
    MyHit = Random.Range(3, 6);
    this.transform.GetComponent<Animator>().SetInteger("State", MyHit);
    this.transform.GetChild(0).gameObject.SetActive(true);
    this.transform.GetChild(0).GetComponent<HitPlayer>().BHit = true;
}
if (Input.GetKeyDown(KeyCode.Alpha1) && MyHit == 0)
{
    if (MP >= 10)
    {
        this.transform.GetChild(0).gameObject.SetActive(true);
        this.transform.GetChild(0).GetComponent<HitPlayer>().BHit = true;
        MP -= 10;
        MyHit = 6;
        this.transform.GetComponent<Animator>().SetInteger("State", 6);
    }

}
if (Input.GetKeyDown(KeyCode.Alpha2) && MyHit == 0)
{
    if (MP >= 15)
    {
        this.transform.GetChild(0).gameObject.SetActive(true);
        this.transform.GetChild(0).GetComponent<HitPlayer>().BHit = true;
        MP -= 15;
        MyHit = 7;
        this.transform.GetComponent<Animator>().SetInteger("State", 7);
    }
}
if (Input.GetKeyDown(KeyCode.Alpha3) && MyHit == 0)
{
    if (MP >= 20)
    {
        this.transform.GetChild(0).gameObject.SetActive(true);
        this.transform.GetChild(0).GetComponent<HitPlayer>().BHit = true;
        MP -= 20;
        MyHit = 8;
        this.transform.GetComponent<Animator>().SetInteger("State", 8);
    }
```

Figure 11. Player Character Attack Function

(3)The protagonist's attribute display: the protagonist has the blood value, mana, gold coin value, and score value, which need to be displayed in the game, so an

attribute display UI must be made for the protagonist through the combination of image, text and slider. The property panel, the finished panel is shown below in Figure 12.

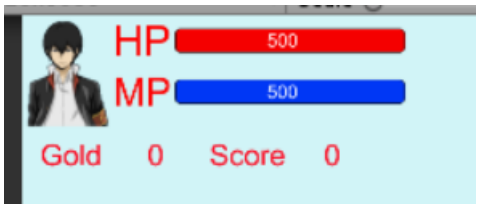

Figure 12. Protagonist UI Panel

Then all attributes need to be changed in real time according to the change of the protagonist attribute value. Here, the value change is achieved by assigning values to all sliders and assigning values to all text in real time. Then when the protagonist's blood volume is zero, the game is suspended through Time.timeScal, and the game end screen appears. When the protagonist's mana is less than 100, there will be a timer, and every 10 seconds, the protagonist's mana is increased by one, which is equivalent to mana's automatic reply.

6.3 The Realization of the Enemy

6.3.1 Enemy Animation and UI Implementation

In the game, there are 7 kinds of enemies, 3 kinds of mobs, and 4 kinds of bosses. First of all, the animation of the mobs is created. In the same way as the animation of the protagonist, animation is used to animate, and then the animation is used to manage the animation.

Here the mobs only has stand , Move, and ordinary attack three animations, and the same method is used to complete the animation of the boss, then because the monster also has attributes such as blood volume, the monster's attribute display UI need to made as the same as the protagonist attribute display, and UI is shown as below Figure 13.

Figure 13. Monster UI Panel

6.3.2 Implementation of Mob Function

In the game, the mobs can first move automatically. Here, two coordinates are first set as the left and right end points of the mob’s movement, and then there are three states in the mobs, standing, moving, and attacking.

When he is not under attack, a timer is used to count the time. When the time is greater than 2 seconds, a random number of zero or one is obtained.

When the random reaches zero, the mobs play a standing animation. When random reaches one, the mobs are in Move state, play the moving animation, and then the Translate function is used to move the mob to the right (Figure 14).

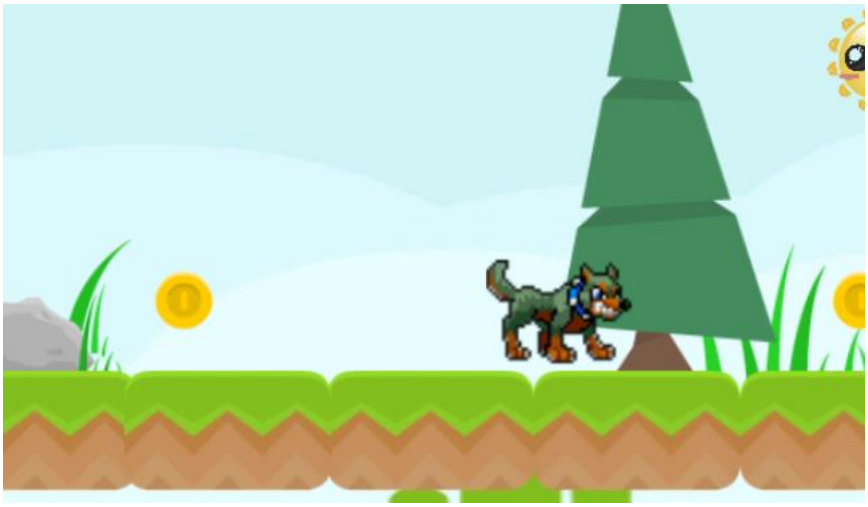

Figure 14. Mods Move to Right Side

At this time, Vector2.Distance is used to determine the distance between the mob and the coordinate point on the right. And the same method is used to determine the distance between it and the target point on the left, so that the mob can automatically move left or right or stand.

The attack of the mob first judges the distance between the protagonist and the mob itself. If it reaches the attack distance, it is judged whether the current protagonist is in an attack state. If it is not, the mob can attack. The mob 's attack is not triggered uniformly (Figure 15).

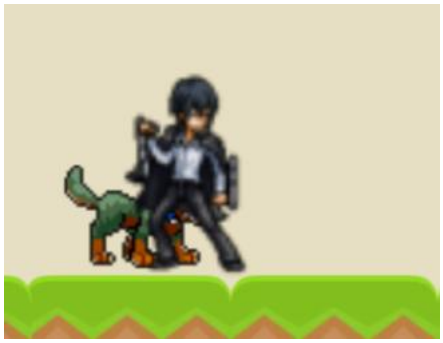

Figure 15. Mod's Attack

A random number between 1 to 10 is drawn and a random number less than 6 will trigger an attack. First, it must be determined whether the mob is facing the

protagonist. If not, it is set to face the protagonist, then the attack animation is played, the protagonist slams the blood, and the mob attacks. After the timer starts, the attack state is cancelled after one second, and the initial standing state is returned. When the mobs 'health is less than or equal to zero, the protagonist's mana increases, the gold coins increase, and the score increases. Finally, the mobs' blood bar UI display disappears, and the mobs themselves are deleted with Destroy.

```
public class Monster1 : MonoBehaviour {

    public int HP;
    public int MP;
    public int MyState;
    public float MyStatetime;
    public int Disx;
    Vector2[] vector2s=new Vector2[2];
    public GameObject Player;
    Vector2[] MoveVec = new Vector2[3];//move vector

    public float Speed;//move speed

    public bool Hit;
    public float Hittime;
    // Use this for initialization
    void Start ()
    {
        vector2s[0] = new Vector2(this.transform.position.x - Disx, this.transform.position.y);
        vector2s[1] = new Vector2(this.transform.position.x + Disx, this.transform.position.y);
        MoveVec[0] = -this.transform.right;
        Player = GameObject.FindGameObjectWithTag("Player");
    }
    public void EndHit()
    {
        MyState = 0;
        this.transform.GetComponent<Animator>().SetInteger("State", 0);
        this.transform.GetChild(0).gameObject.SetActive(false);

    }
```

Figure 16. Mods Random Attack Function

```
if (Vector2.Distance(Player.transform.position, this.transform.position) <= 1f && MyState != 2&&!Hit)
{
    int t = Random.Range(0, 10);
    if (t <= 6)
    {
        if (t <= 1)
        {
            if (Player.transform.position.x > this.transform.position.x)
            {
                this.transform.rotation = new Quaternion(0, 180, 0, 0);//转向
                MoveVec[0] = this.transform.right;
            }
            else
            {
                this.transform.rotation = new Quaternion(0, 0, 0, 0);//z转向
                MoveVec[0] = -this.transform.right;
            }
        }

        MyState = 2;
        Hit = true;
        this.transform.GetChild(0).gameObject.SetActive(true);
        this.transform.GetChild(0).GetComponent<HitMonster>().BHit = true;
        this.transform.GetComponent<Animator>().SetInteger("State", 2);

    }
}
```

Figure 17. Mods Move Function

6.3.3 Implementation of Boss Function

The boss in the game has the same common attacks and skills as the protagonist, so a collision body object has to be added corresponding to the attack in the same way as the protagonist's production method, so that it can be used to determine whether the protagonist is hit during the attack. The body is shown below in Figure 18.

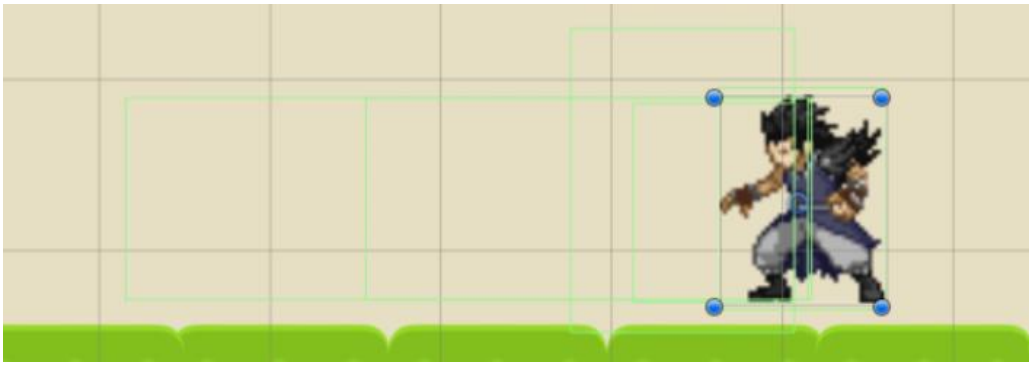

Figure 18. Skills Hit-Box

Then the code is used to implement the various functions of the boss.

First, the mobile standing function of the boss is basically the same as that of the mobs. It is only during the attack, when the boss starts to attack, a random number between 2 to 6 must be obtained. When the number is 2, the boss launches a common attack. When the random number is greater than 2, the boss needs to determine whether the current mana is right enough.

When there is mana, the corresponding mana is deducted, and the corresponding animation is played to display the corresponding skill collision body. The boss' mana also has an auto-recovery function like the protagonist, and the death of the boss is the same as the mobs (Figure 19 and Figure 20).

```
MyState = Random.Range(2,6);
if (MyState > 2)
{
    if (MP >= MyState * 5)
    {
        Player.transform.GetComponent<Run>().UiImg[4].transform.GetChild(0).GetComponent<Slider>().value = HP / 100f;
        Player.transform.GetComponent<Run>().UiImg[4].transform.GetChild(1).GetComponent<Slider>().value = MP / 100f;
        Player.transform.GetComponent<Run>().UiImg[4].transform.GetChild(2).GetComponent<Text>().text = HP.ToString();
        Player.transform.GetComponent<Run>().UiImg[4].transform.GetChild(3).GetComponent<Text>().text = MP.ToString();
        Hit = true;
        this.transform.GetChild(MyState - 2).gameObject.SetActive(true);
        this.transform.GetChild(MyState - 2).GetComponent<HitMonster>().BHit = true;
        this.transform.GetComponent<Animator>().SetInteger("State", MyState);
    }
    else
    {
        MyState = 2;
        Hit = true;
        this.transform.GetChild(MyState - 2).gameObject.SetActive(true);
        this.transform.GetChild(MyState - 2).GetComponent<HitMonster>().BHit = true;
        this.transform.GetComponent<Animator>().SetInteger("State", MyState);
    }
}
else
{
    Hit = true;
    this.transform.GetChild(MyState - 2).gameObject.SetActive(true);
    this.transform.GetChild(MyState - 2).GetComponent<HitMonster>().BHit = true;
    this.transform.GetComponent<Animator>().SetInteger("State", MyState);
}
```

Figure 19. Boss Random Attack Function

```
    if (collision.tag == "Monster2")
    {
        if (this.transform.parent.GetComponent<Run>().MyHit >= 0 && BHit)
        {

            if (collision.GetComponent<Monster2>().HP >= 0)
                collision.GetComponent<Monster2>().HP -= this.transform.parent.GetComponent<Run>().MyHit * 5;
            if (collision.GetComponent<Monster2>().HP <= 0)
                collision.GetComponent<Monster2>().HP = 0;
            BHit = false;
            this.transform.parent.GetComponent<Run>().UiImg[4].SetActive(true);
            this.transform.parent.GetComponent<Run>().UiImg[4].transform.GetChild(0).GetComponent<Slider>().value = collision.GetComponent<Monster2>().HP / 100f;
            this.transform.parent.GetComponent<Run>().UiImg[4].transform.GetChild(1).GetComponent<Slider>().value = collision.GetComponent<Monster2>().MP / 100f;
            this.transform.parent.GetComponent<Run>().UiImg[4].transform.GetChild(2).GetComponent<Text>().text = collision.GetComponent<Monster2>().HP.ToString();
            this.transform.parent.GetComponent<Run>().UiImg[4].transform.GetChild(3).GetComponent<Text>().text = collision.GetComponent<Monster2>().MP.ToString();
            this.transform.parent.GetComponent<Run>().UiImg[4].transform.GetChild(4).gameObject.SetActive(true);
            this.transform.parent.GetComponent<Run>().UiImg[4].transform.GetChild(4).GetComponent<Image>().sprite = collision.GetComponent<Monster2>().sprite;
        }
    }
}
```

Figure 20. Boss Death Function

The function of enemy is based on references /6/, /9/, /12/ and /13/; and UI design is based on references /4/. The function of Hit-Box is based on references /2/.

6.4 Other Functions

The Implementation of other functions is discussed next.

6.4.1 Implementation of Props

There are a variety of props in the game, namely gold coins, water droplets, and the sun. Among them, there are the gold coins plus the protagonist gold coins, the water droplets and the protagonist mana, and the sun and the protagonist's life.

Because these props have collision boxes, the OnTriggerEnter2D collision is added to the script on the protagonist. Detection, and then different props set different tags to it, so when the protagonist collides with the props, it will determine the tag of the collided object, and then increase the corresponding attribute value of the player according to the different tags.

6.4.2 Realization of Map Mechanism

In order to increase the fun of the game, some map mechanism were added to the game.

The first one is the box that will be automatically destroyed. The material of the box here is different from other terrains, which can be clearly seen. Then a script is added to the box. Then the collision detection is done in the script and set a bool value set.

When the collision detection detects that the box has collided with the protagonist,

the bool value is true. At this time, the timing starts. When the time reaches the set value, the box itself is destroyed. The schematic diagram of the boxes that can be automatically destroyed in the game is shown in Figure 21 and the corresponding program code is shown in Figure 22.

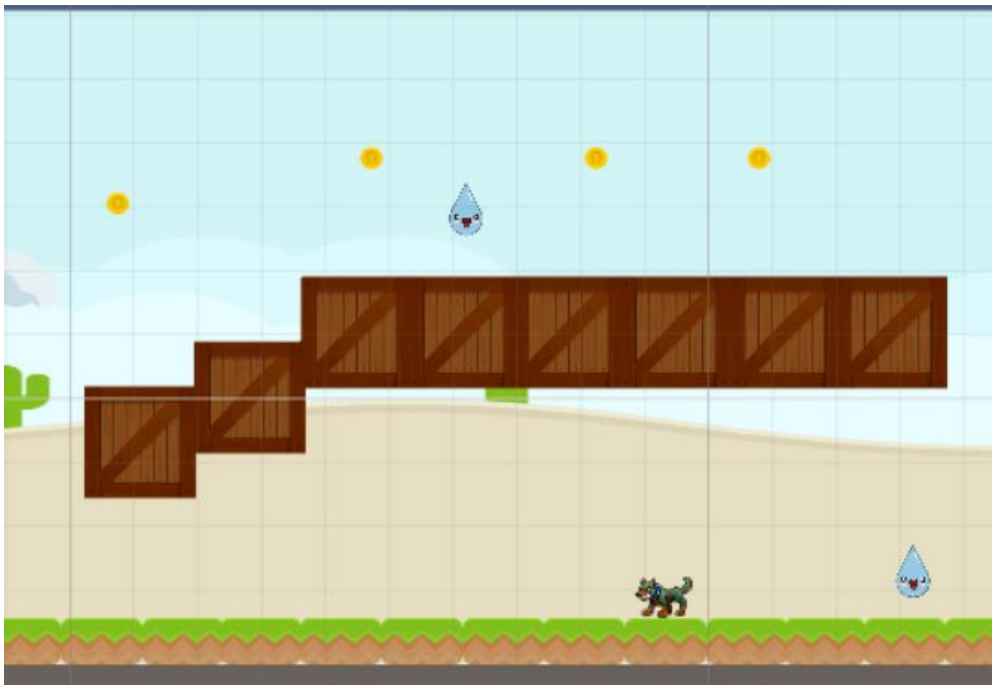

Figure 21. Automatically Destroyed Box

```
public class MuXiang : MonoBehaviour {
    bool bDes;
    public int DesTime;
    // Use this for initialization
    void Start ()
    {
        bDes = false;

    }

    // Update is called once per frame
    void Update ()
    {

    }
    private void OnCollisionEnter2D(Collision2D collision)
    {
        if (collision.gameObject.tag == "Player"&& !bDes)
        {
            Invoke("XiaoShi", DesTime);
            bDes = true;
        }
    }
    public void XiaoShi()
    {
        Destroy(this.gameObject);
    }
}
```

Figure 22. Automatically Destroyed Box Function

The second mechanism is the game clearance flag. The flag is placed at the end of each level to let the player enter the next level(Figure 23). The same script is added to use collision detection to detect whether the protagonist is collided. The score, if satisfied, loads the next level(Figure 24).

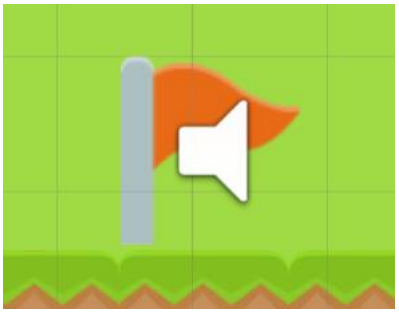

Figure 23 Game Clearance Flag

```
private void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.tag == "Player"&&Run.Win>= WinNumber)
    {
        Win.SetActive(true);
        this.GetComponent<AudioSource>().Play();
        Win.transform.GetChild(0).GetComponent<Text>().text = collision.GetComponent<Run>().JinBi.ToString();
        Time.timeScale = 0;

    }
}
```

Figure 24. End Function on Flag

The third mechanism is the gear. First, the gear will move back and forth between two points. Here, empty objects are created as the two end points of the gear movement, and then “Rotate” is used to rotate the gear itself around its midpoint. A good gear is shown below in Figure 25.

Figure 25. Mechanism Gear

The fourth mechanism is the turret. Here two points are set as the starting point of the turret bullet, and then “Instantiate” is used to generate the bullet preform. After the bullet is generated, it always moves to the end point. When it reaches the end point, it is automatically destroyed. At this time, the protagonist slams the blood, and the fort here is shown below in Figure 26 and the corresponding program code

is shown in Figure 27.

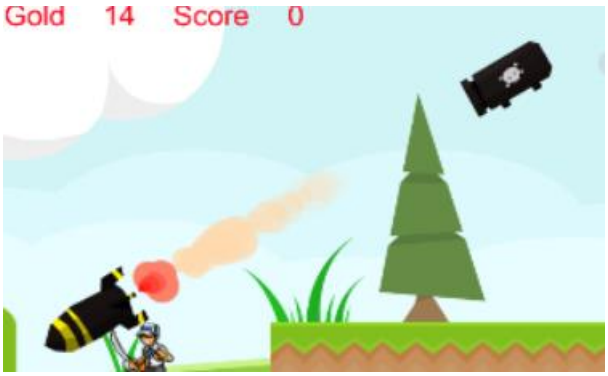

Figure 26. Mechanism Turret

```
blic class PathedProjectileSpawner : MonoBehaviour

    /// the pathed projectile's destination
    public Transform Destination;
    /// the projectiles to spawn
    public PathedProjectile Projectile;
    /// the effect to instantiate at each spawn
    public GameObject SpawnEffect;
    /// the speed of the projectiles
    public float Speed;
    /// the frequency of the spawns
    public float FireRate;

    private float _nextShotInSeconds;

    /// <summary>
    /// Initialization
    /// </summary>
    void Start ()
    {
        _nextShotInSeconds=FireRate;
    }

    /// <summary>
    /// Every frame, we check if we need to instantiate a new projectile
    /// </summary>
    void Update ()
    {
        if((_nextShotInSeconds -= Time.deltaTime)>0)
            return;

        _nextShotInSeconds = FireRate;
        var projectile = (PathedProjectile) Instantiate(Projectile, transform.position,transform.rotation);
        projectile.Initialize(Destination,Speed);

        if (SpawnEffect!=null)
        {
            //Instantiate(SpawnEffect,transform.position,transform.rotation);
        }
    }
```

Figure 27. Function of Mechanism Turret

The function of Map Mechanism is based on /10/ and /17/.

6.4.3 Implementation of Sound Effects

There is background music in the game, the audio source component is added to the camera of each scene, then the background music material is assigned to it, the playonawake and loop options are checked. This is how the background music loop playback can be achieved, and then a music switch button is finally added in the scene.

The button is bound to the button event, in the function to determine whether the current music is playing, then use GetComponent <AudioSource> (). Pause () to pause the music, and play to play the music.

The function of Map Mechanism is based on /10/ and /17/.

6.4.4 Improve the Functions Around the game

A pause function is added in the game, the pause button is created in the UI, the pause function is bound to the button. It has to be determined whether the current game is paused in the function, and then the timescale is changed to achieve the pause and continue of the game.

The function of returning to the home page is added to the game, and a button for returning to the home page is created in the UI to bind the function event that loads the initial scene.

A timing function is added to the game, The text to display the time is created, and a float value is used to time in the game.

Those functions is based on the /9/ and /10/.

6.5 The Publication of the Game

Before the official release, the game scene needs to be set. The build and run settings are found under the file menu bar.

The next thing to do is to add all the four scenes that need to be published, including the game start scene, the first level, the second level, and the third level.

Because the game needs to published on the pc side, so PC, Mac and Linux Standalone are chosen on the menu. This option is the stand-alone version. Then the setting interface is displayed in the inspection panel, the resolution size is set to 1920 * 1080, and then under the resolution window, the default hide is selected. Finally, the exe file is generated by clicking publish. The operation diagram of publishing a game is shown in Figure 28.

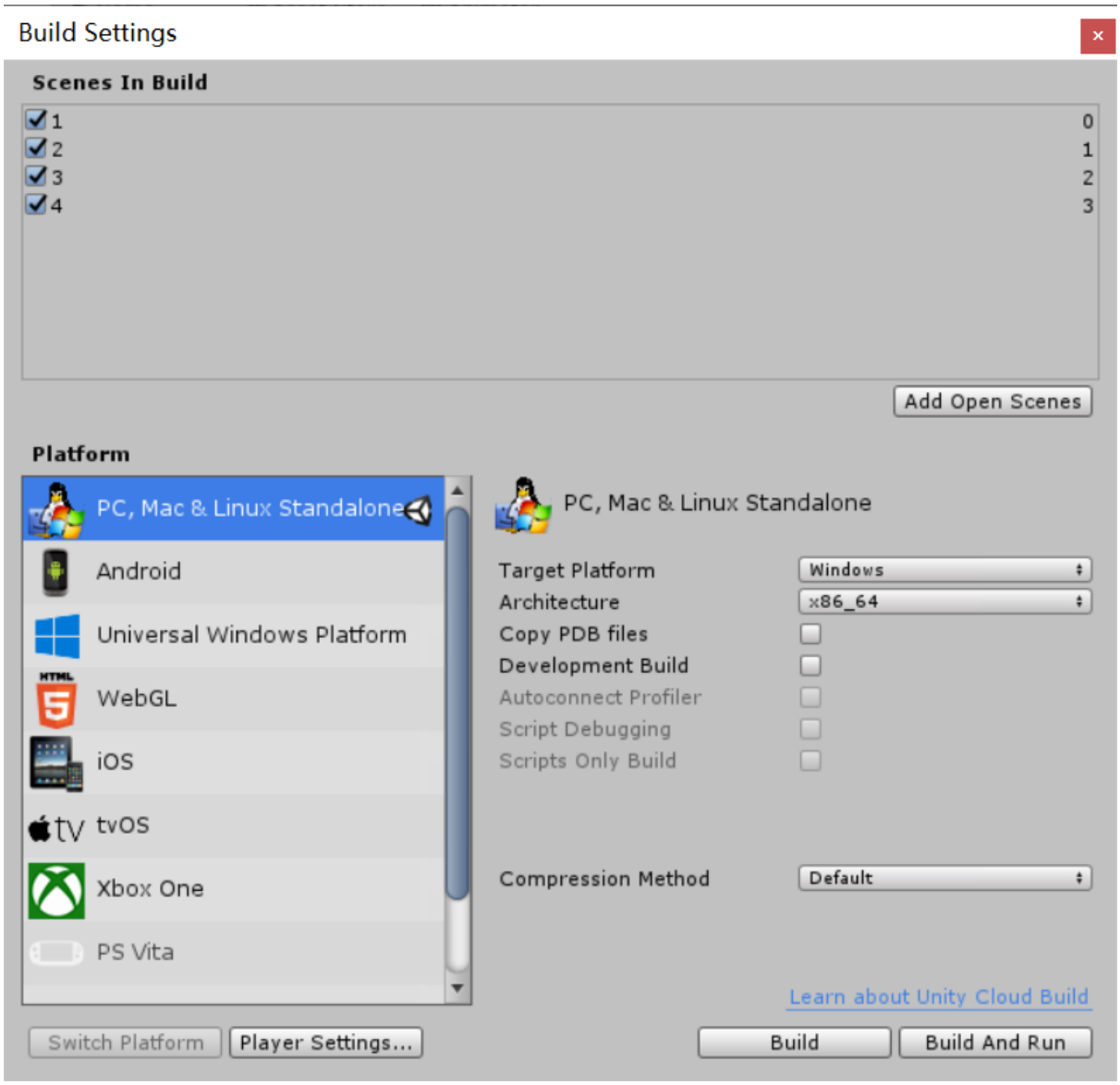

Figure 28. Game Published Operation

7 GAME TESTING

As part of software testing, the game testing has all the common features of software testing.

(1) The purpose of the test is to find defects in the software.

(2) Each test requires the product to run in a real or simulated environment.

(3) Each test requires a systematic method to demonstrate product functionality to prove whether the test results are valid and to find the cause of errors, so that the program personnel can make improvements.

In summary, in order to confirm the development status and running results of this game, this test will be based on the following three aspects:

(1) Vision

(2) Technology

(3) Process

According to the testing principles I set, the following tests were designed to test the overall situation of the game:

(1) Game running status test

(2) Key function test

(3) Monster behavior test

(4) Skill release and its damage test

(5) Map mechanism test

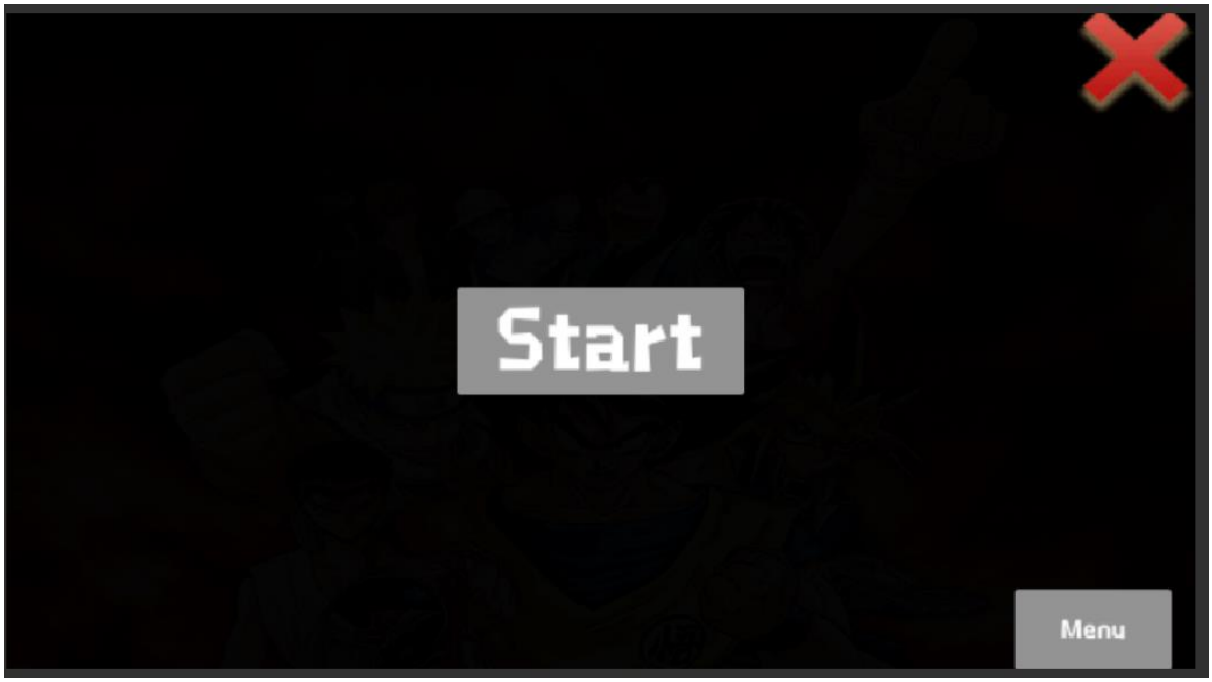

Figure 29. Start Interface

As shown in the Figure 29, the start interface has the game name exit game button, the start game button, and the game description button. Through the click test of each button, the function of each button has been successfully implemented. No obvious bugs are found, and the start interface runs smoothly.

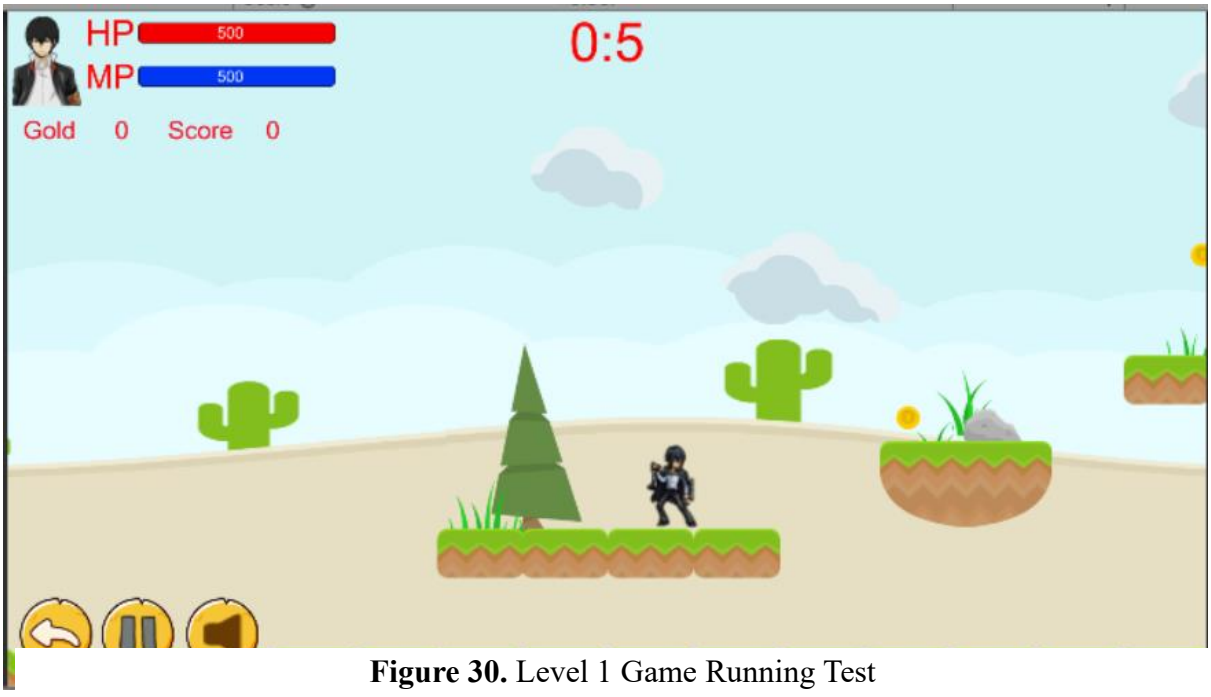

Figure 30. Level 1 Game Running Test

Figure 31. Level 1 Player Character Move Test

Figure 32. Player Character Jump Test

According to the observation, after the key functional tests of the design(Figure 30 and Figure 31), it is concluded that the A, D key movement function of the player character is running normally,. The space bar jump function is running normally(Figure 32). The collision between the protagonist and the ground and the judgment of the end of the protagonist's jump has no problems, so the protagonist's mobile jump function is successfully implemented. The state switching between moving, jumping, standing and the animation playback and switching of each state have reached a smooth state.

Figure 33. Player Character Attack Test

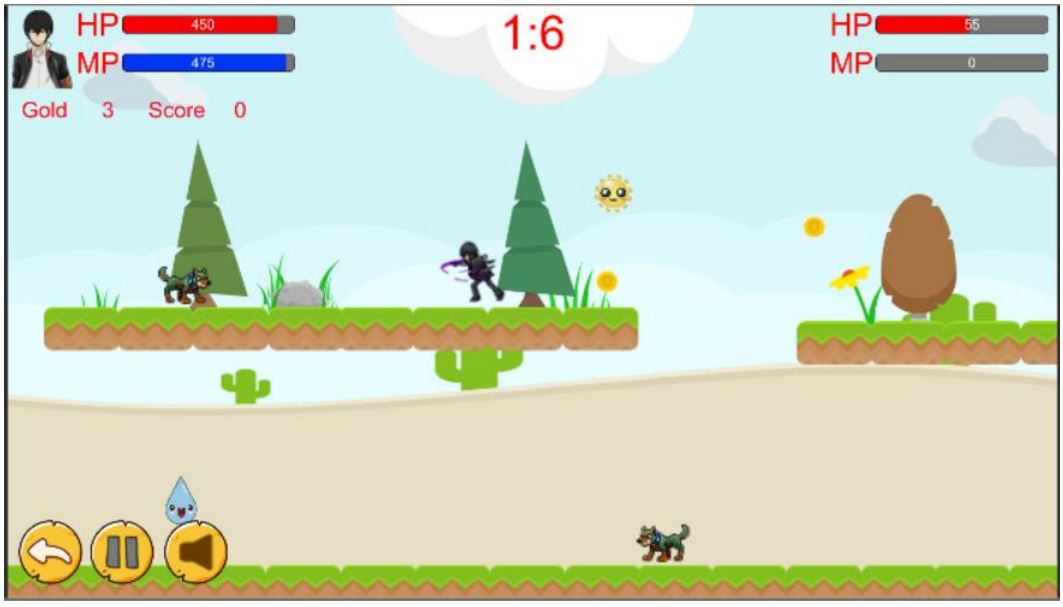

Figure 34. Player Character Skills Test 1

Figure 35. Player Character Skills Test 2

According to the observation, after the key functional tests of the design, (Figure 33), the player character use key “Q” to attack function is running perfect. At the same time, the skills functions are implemented, the animation between standing and skill release is smooth. (Figure 34 and Figure 35)

Figure 36. Player Character Attack Test 3

Figure 37. Monster Attack and Behavior Test

Figure 38. Boss Attack and Behavior Test

Figure 39. Game End Flag Function Test

The Q-key common attack and other skill attacks were tested. The respective attack animations and status switches were normal. In all attack judgments, the attack judgments are valid and the values are consistent with the settings, without obvious bugs (Figure 36).

After that, conducted skills and behavior tests were made for mobs and bosses. (Figure 37 and Figure 38) In this test, the standing and building status of the mobs and bosses are randomly switched to normal. At the same time, when the player character is close to the monster, the mobs will randomly face and attack the player. The boss will randomly release skills or ordinary attacks to attack the player. After testing, the function modules related to mobs and bosses are normal, all animations can be played, and the reduction in blood volume and mana volume are in agreement with the settings; in summary, all the functions of monsters have been obtained good realization.

In the function test of ending the banner, when the player reaches a sufficient score and the player character touches the banner, the banner will successfully end the

current game and pop up the settlement interface (Figure 39). Overall, the banner function is normal and works well.

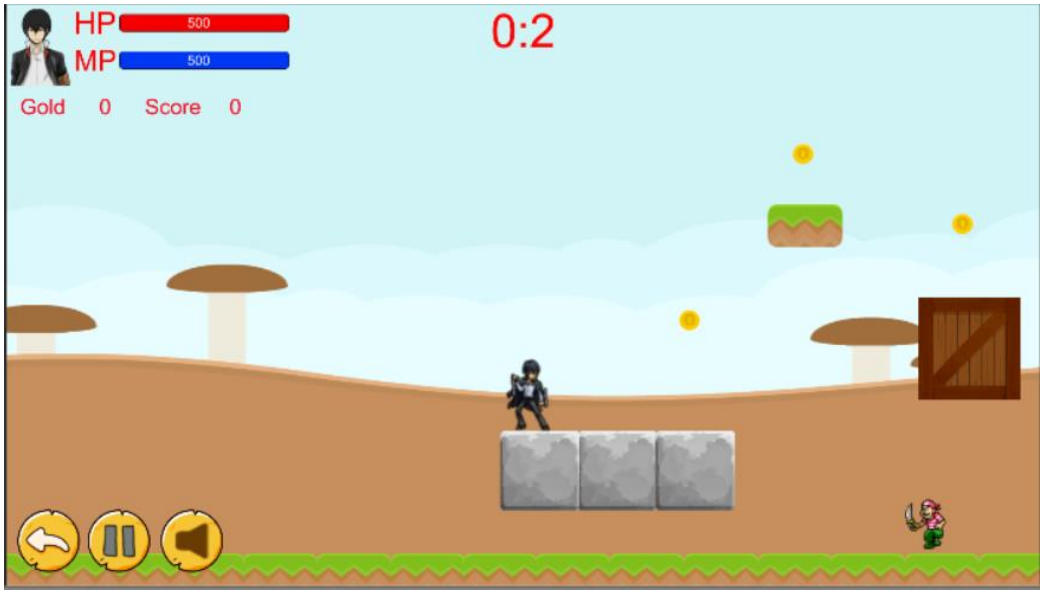

Figure 40 Level 2 Game Running Test

In Level 2, as before, the same test was conducted, the first one is the game running test, as Figure 40 shows, in level 2 is also running.

Figure 41. Map Mechanism Test

Figure 42. Boss Attack and Behavior Test

In Level 2, conducted a trap test was conducted on the map. (Figure 41) In the test, all traps work normally and cause damage equal to the set value, so the trap function is normal.

And then, as show in Figure 42, the mods and boss attack and behavior also works normally.

Figure 43. level 3 Game Running Test

Figure 44. Level 3 Mods and Boss Attack and Behavior Test, Map Mechanism Test

Figure 45. Level 3 Mods and Boss Attack and Behavior Test, Map Mechanism Test 2

In the level 3; As before, the same running and monsters behavior test was conducted, The test result are shown in Figure 43 and Figure 44, The test result is good.

As shown in the Figure 45, in the self-destroy box function test, after the player character touches the box, the box will disappear after a few seconds. This shows that, the self-destroy box function is implemented and running perfect.

At the same time, the switching function was tested between game levels, as well as the attack functions of different mobs and bosses in each level, and finally tested the integrity of the game was tested. The test result is that various functions in each level work well , where the props and traps in the map are also functioning normally. At the same time, the various UI interfaces in the game are displayed, and the UI function buttons are also working properly.

Finally, the game was packaged, a compatibility test was conducted, and the game was run on two other computers. The test result is: the game runs normally, and all functions have passed the test.

In summary, this game basically implements all the functions of the design, and each function works well.

8 CONCLUSIONS

This paper describes in more detail how to design and make a fighting game on the Unity game engine. Starting from analyzing the purpose and significance of the game, to elaborating the characteristics of Unity and the advantages of the scripting language C #, the game planning was then proposed, and the game's module classification was detailed, and the functions below each module were listed and used flow chart to show. Finally, it focuses on the production process of the game, including the creation of the game's protagonist and monster, the layout of the game interface, and the difficulty level setting of the level.

In this project, in addition to challenging myself, I independently developed a game of my own, but also for the internship I have always wanted to make a game myself. Based on this as a starting point, I decided on my motivation for developing this 2D game.

The Unity3D engine was chosen because it has a powerful development engine, a customizable IDE environment, a visual development system, and a Mono-based development script. At the same time, Unity3D's perfect community function has become my reason for choosing it. It can help me learn faster and master Unity3D. Before that, I tried other engines as development tools, such as GMS2, but I encountered a bottleneck in the development, and the engine could not achieve my original design. Therefore, I chose to replace the engine to solve this problem.

After completing the game and passing the test, although the game has achieved many major functions, there are still some functions that are not comprehensive enough. This is mainly due to insufficient time and insufficient personal abilities, so this game needs to be further improved.

REFERENCES

/1/ 徐军，张子墨．,2018, 基于 Unity3D 射击游戏的设计及其核心功能实现, p.111-p113

/2/ 孙可言，陈根．2016(24) ,基于 Unity 射击游戏的人工智能与碰撞检测的研究．科技风，p.11,

/3/ 祝子豪, 2018 ,2D 游戏角色设计与实现．信息与电脑（理论版）,P.80-81

/4/ 根绒切机多吉，邢彪．2017:休闲益智类手机游戏的 UI 与特效开发．信息与电脑,p.65-72

/5/ 马晓萍， 刘静,2018,基于 Unity3D 的坦克大战游戏设计与实现． 数字技术与应用， p.168-169

/6/ 朱喜基，冯振辉,2018,基于 Unity3D 虚拟校园漫游碰撞检测的研究．江苏科技信息， p.66-68

/7/ 王功利，钟健麒,2018,Unity3D 射击游戏中人工智能与摄像机的运用． 电子技术与软件工程

/8/ 梁肇敏, 2017(10)两种基于 Unity3D 的第三人称视角移动的实现方法与比较,电脑迷

/9/ 优美缔软件有限公司，2015, Unity 5.X 从入门到精通． 中国铁道出版社

/10/ 吴亚峰，杜化美,2015 ,Unity 游戏案例开发大全．人民邮电出版社

/11/ 牛犇, 2012,基于 iOS 的 2D 第三人称射击游戏引擎的研究与实现．苏州大学

/12/ Nathan Brewer,2017Computerized Dungeons and Randomly Generated Worlds: From Rogue to Minecraft. Proceedings of the IEEE, p.970-977

/13/ Blackman, Sue,2011, Beginning 3D Game Development with Unity The World's Most Widely Used Multi-platform Game Engine https://learning.oreilly.com/library/view/beginning-3d-game/9781430234227/ Accessed 25.5.2020

/14/ Thorn A,2013, Learn Unity for 2D game development. Learn Unity for 2D Game Development https://www.researchgate.net/publication/316156769_Learn_Unity_for_2D_Game

_Development
Accessed 25.5.2020

/15/ Thorn, Alan, 2014, Pro Unity game development with C#. Pro Unity Game Development with C#.
https://www.researchgate.net/publication/321501092_Pro_Unity_Game_Development_with_C
Accessed 25.5.2020

/16/ Pereira V, 2014, Learning Unity 2D game development by example create your own line of successful 2D games with Unity!.
https://dw1bukz.cf/book.php?id=S-paBAAAQBAJ
Accessed 25.5.2020

/17/ Johnson M, 2014, Learning 2D Game Development with Unity.

/18/All the character materials are from the public material platform, /Online/. Available:
http://www.6m5m.com/service-sid-11609.html
http://www.6m5m.com/service-sid-7251.html
Accessed 25.5.2020

/19/Introduction about visual studio 2019 on Baidu BaiKe, [Online]. Available:
https://baike.baidu.com/item/Microsoft%20Visual%20Studio/4735644?fromtitle=Visual%20Studio&fromid=539453
Accessed 25.5.2020

/20/Introduction about C# language on Baidu BaiKe , [Online]. Available:
https://baike.baidu.com/item/c%23
Accessed 25.5.2020

/21/Video game development description on Wikipedia:
https://en.wikipedia.org/wiki/Video_game_development
Accessed 25.5.2020

/22/ACG Game Report 2019 on joynews.cn:
http://www.joynews.cn/jiaodianpic/201901/2432269.html
Accessed 25.5.2020

/23/ What research topics are involved in the current game field and industry frontiers? On Zhihu.com.cn:
https://zhuanlan.zhihu.com/p/22297857

Accessed 25.5.2020
/24/ What are the advantages of developing games with Unity3D? On csdn.net:
https://blog.csdn.net/JtNbCOC8N2I9/article/details/78557144
Accessed 25.5.2020