Design and Development of Card Games Based on Unity 3D

Yucong Pan[a], Kun Wang[b], Hanqi Shao[c], Zhenghe Chen[d], Wenli Dong[e], Xin Wang[f]

School of Digital Media & Design Arts, Beijing University of Posts and Telecommunications Beijing, China

[a]873846532@qq.com, [b]wang763776553@qq.com, [c]377677840@qq.com, [d]940199956@qq.com, [e]134672609
1@qq.com, [f]2936239615@qq.com

Keywords: Unity 3D; Coroutine; Role-Play Game; Card Game

Abstract: Card games originate from the role-playing card games played by two or more people on the table. Players play the role in the background of the game and play games using cards on a Turn-Base Strategy. Card games are the mainstream in China's mobile game market, and because of their non-real-time characteristics, they are very different from ordinary games in coding: a large part of the calculation is not done in Update (), but using a very convenient function in Unity – Coroutine to solve. This paper introduces in detail the development process of a card game named *Dice of Destiny*, including game design, programming, interface design, card model and mapping, and so on. Among them, programming is the key part of the whole game. How to use the collaboration to realize many functions of the game conveniently is the innovation in the technical link.

1. Introduction

Dice of Destiny is a role-playing game with card as the core of narration. Each time you draw a card, you will come across a story or meet some events that require players to participate. It is possible to encounter a certain number of enemies or some decisions that need to be made. When the last card is drawn, you will enter the final battle and know the ending of your story. There are totally three endings in this game, which depends on the decisions you made during the game. If you deplete your health or are defeated by enemies, you will fail. The Battle and selection in the game are largely completed using six dices on the desktop. There are three different patterns on each of the dices: Sword, Shield, and Lightning. Each represents attack, defense or charge. To prevent the result from relying too much on luck, we set a reset mechanism for the player. After completing the first cast, you can lock any number of desired dices and recast the others.

2. Game Design

2.1 Battle System

The battle system of Dice of Destiny is basically like a Turn-Base game. To win the battle, you need to decide carefully how to deal with the results of each of your cast. As has been mentioned, there are three different patterns of each dice. And the following elaboration is about the details of their functions.

(1) **Defense**: if you get a shield from your cast, then you will get a defense point. After that, you will be rewarded an absorb-damage shield, which equals your armor multiplied by defense point. Defense point will be cleared when each turn starts.

(2) **Charge**: the lightning shows on the dice represents a charge point, when you fill up the charging bar with up to 10 charge points, you can use the unique skill Justice Sword, which will significantly damage one enemy and give you an absorb-damage shield.

(3) **Attack**: The sword means an Attack Point, after casting your dices, you're able to attack your enemies using the attack point. Each attack will cost one attack point and cause damage to a specific enemy, the power of your attack is depending on your attack power.

After finishing player's turn, there will be an enemy's turn, in which each enemy will attack the player and the damage is also depending on their attack power. If possible, some enemy will use their special skill to heavily damage the player. And then there follows another player turn again.

2.2 Content of Cards

The game's card group contains about 25 cards totally:

1.A set of Begin Card, which aims to do the introduction of the story and the character of the game.

2.Two sets of Diversity Card, which include a few selections and will have a significant impact on the ending of the story. And it may also influence the status of your character.

3.A set of Ending Card, which lists the elaboration of which ending you've achieved. The ending of the story completely relies on the choices you've made before.

4.The cards left are called Basic Card, which can be differentiated as Enemy Card, Event Card and Treasure Card:

Enemy Card means you've encountered a few enemies that you must defeat. It will start the Battle System of the game. Event Card will trigger a special event, which may offer some choices for the player to consider. While no matter whether you choose, it won't have any influence on your ending, but will probably change the status of your character, like attack power or health, either positive or negative. As for Treasure Card, it will give you a chance to open a treasure box. However, the treasure inside may also have some harmful effect on your character.

After the player starts a new game, the system will automatically generate a card sequence, which determines the order of card you will draw out. Since it's of low possibility for the player to have the same game as he has played once before.

3. Code Design

3.1 About Coroutine:

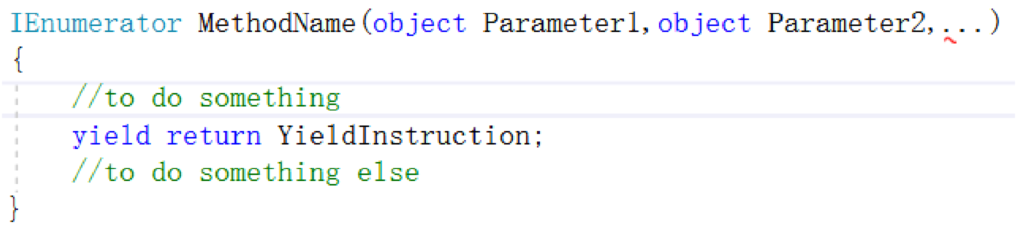
Because of its particularity, the card game is quite different from the general real-time combat game. In real-time combat, a large part of the computation is done in Update (), but for card games, which are like Turn-Base games, we need to use a very convenient function in Unity, Coroutine.

What is a Coroutine? Coroutine in Unity is a function that can pause execution, returns immediately after the pause, and continues execution until the function completes. It's like a child thread that comes out alone to handle some problems, with less consumption, but there can only be one running Coroutine in the main thread provided by MonoBehaviour.

The characteristics of Coroutine are:

1. Coroutine is suspended when it comes across an interrupt instruction (yield return).
2. Once the execution of the Coroutine is suspended, it immediately returns to the main function. Then it continues to execute the remaining functions of the Coroutine after the suspension.
3. After the interrupt instruction is completed, Coroutine will resume at the next line of the interrupt instruction.
4. At the same time, a script instance may have multiple suspended Coroutines, but only one running Coroutine.
5. After the body of the function in Coroutine is completed, the Coroutine ends.
6. Coroutine can easily control behaviors after a certain number of frames.
7. Coroutine has almost no more cost in performance than general functions.

3.2 How to Create a Coroutine:



```
IEnumerator MethodName(object Parameter1,object Parameter2,...)
{
    //to do something
    yield return YieldInstruction;
    //to do something else
}
```

Fig 1. How to create a coroutine.

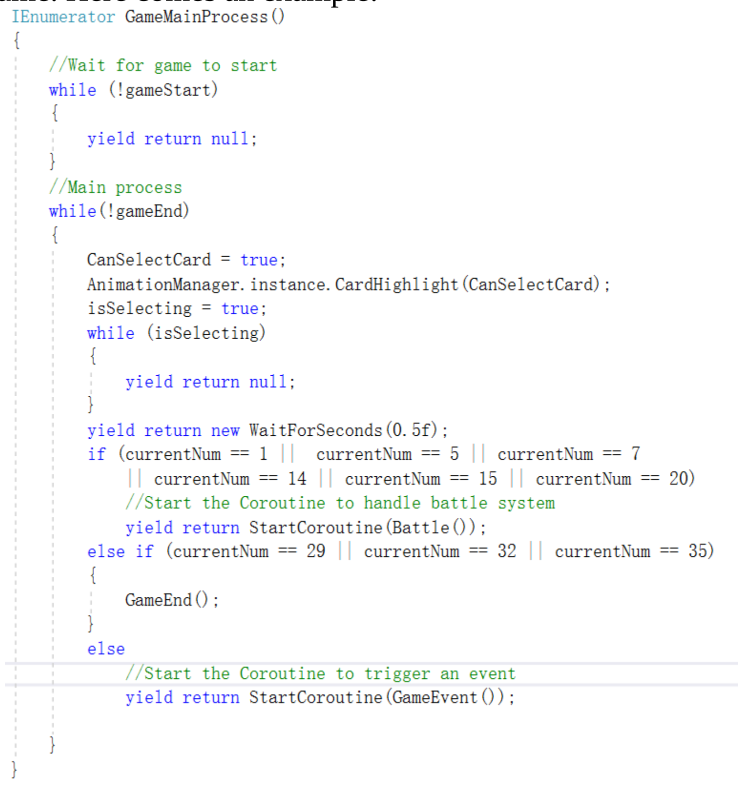
Use the keyword *IEnumerator* to create a Coroutine, then use *StartCoroutine(IEnumerator routine).* You can also use *StopCoroutine()* or *StopAllCoroutine()* to end Coroutine

3.3 About Interrupt Instructions

Tab 1. Interrupt instructions

Instructions	**Description**	**Implementation**
WaitForSeconds	Wait for specified seconds	Yield return new WaitForSeconds(2);
WaitForFixedUpdate	Wait for a FixedFrame	Yield return new WaitForFixedUpdate();
WaitForEndOfFrame	Wait for the end of a frame	Yield return new WaitForEndOfFrame();
StartCoroutine	Start a new Coroutine and wait for its end.	Yield return StartCoroutine(other coroutine);
WWW	Wait for a load to complete	Yield return www;

For instance, if you use *yield return new WaitForSeconds(0.8f)*, the Coroutine will pause for 0.8 seconds then continue to execute. While if you use *yield return StartCoroutine(OtherCoroutine())*, Then a new Coroutine will be started, and the original one will continue to execute after the new one is completed. Due to its flexibility, we can simply achieve lots of difficult target during the development of the game. Here comes an example:



```
IEnumerator GameMainProcess()
{
    //Wait for game to start
    while (!gameStart)
    {
        yield return null;
    }
    //Main process
    while(!gameEnd)
    {
        CanSelectCard = true;
        AnimationManager.instance.CardHighlight(CanSelectCard);
        isSelecting = true;
        while (isSelecting)
        {
            yield return null;
        }
        yield return new WaitForSeconds(0.5f);
        if (currentNum == 1 ||  currentNum == 5 || currentNum == 7
            || currentNum == 14 || currentNum == 15 || currentNum == 20)
            //Start the Coroutine to handle battle system
            yield return StartCoroutine(Battle());
        else if (currentNum == 29 || currentNum == 32 || currentNum == 35)
        {
            GameEnd();
        }
        else
            //Start the Coroutine to trigger an event
            yield return StartCoroutine(GameEvent());

    }
}
```

Fig 2. Example of a coroutine.

3.4 Server & Data Base

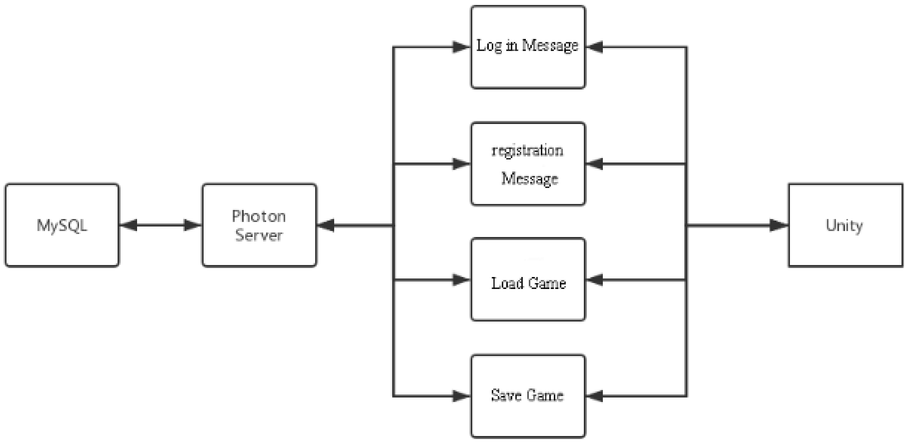


Fig 3. Connection work.

To realize cloud storage and server data processing, we use Photon Server, Visual Studio and MySQL to build servers and databases.

First, we should set up the Photon Server and make sure the corresponding Photon Server runtime environment is configured. Then we can build the server using Visual Studio to realize its corresponding functions of connection and data transmission with the Unity game console. Once after the MySQL database is settled, we can create a database list through MySQL, which stores user login information, character status information, and level information. So, we can read them through the server. Data transfer between Photon Server and MySQL, through NHibernate, to map the database storage list to the server, and then through the server to add, delete and change the data stored in MySQL, and then achieve the corresponding functions. Network connection and data transmission between Unity and Photon Server. On the Unity game side, the server is connected by accessing the IP address of the server, and the data transmission function between the two is realized.

4. Artwork

4.1 User Interface

As the dragon is the ultimate boss of the game, so we choose a picture with a dragon to match the scene, while adding a mysterious atmosphere, adjusting the tone, making the whole background more unified.

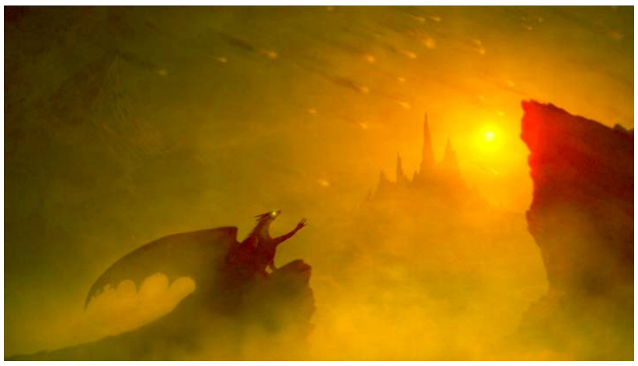


Fig 4. Background of the game.

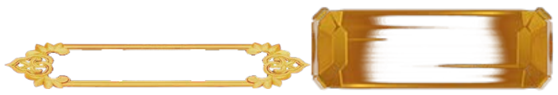


Fig 5. Button.

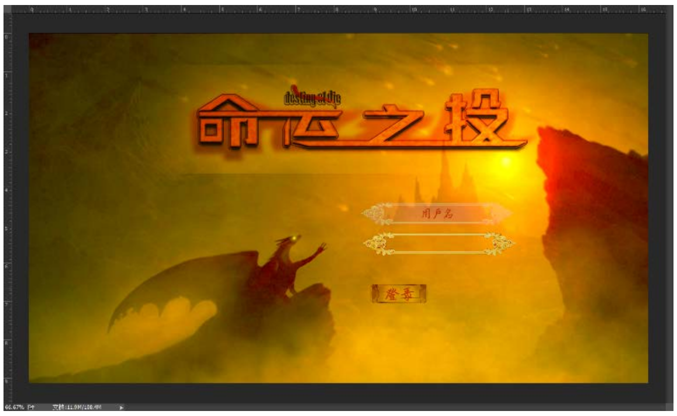


Fig 6. Sign in Page

4.2 Model of Card

Card modeling is done in MAYA. In the early stage, it was a rectangle and thin game card in a traditional concept. To pursue the realistic effect and match it with the whole game style, the chamfering roundness of four card angles is tried. The degree of chamfering is repeatedly tested here. Finally, the results shown in the following figure are determined.

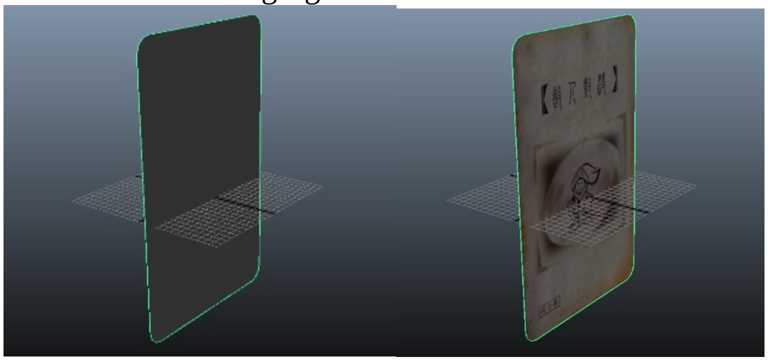


Fig 7. 3D model of card.

In MAYA, we apply the card 3D model map segmentation, export UI mapping into PNG, and then draw it in PhotoShop. That's how we complete the model mapping work. The following illustration shows the unmapped blank card UI image (left) and the mapped card UI image (right) for testing.

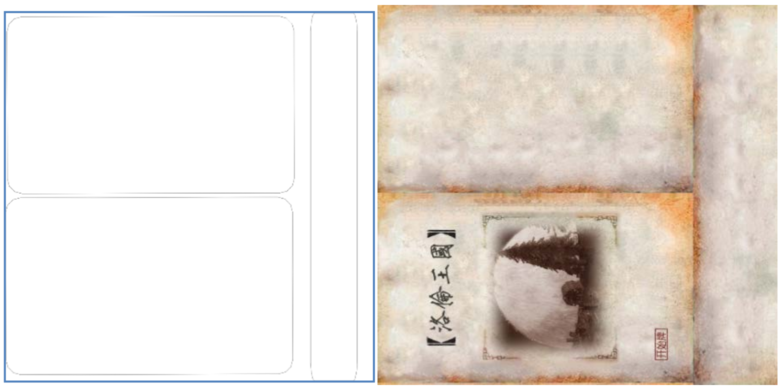


Fig 8. Texture of card.

4.3 Rules of Card

Status cards, a total of 4, belong to the protagonist (that is, the player) character status, mainly for survival and combat, including "health", "defense", "attack" and "charge" four different kinds of card. Charged "lightning" represents skill point, after reaching a certain value, it can allow the player to cast his ultimate skill; defense uses "shield" to represent defense point, which helps the player to absorb damage from enemies; attack uses "sword" to represent attack point, which counts how many times you can attack totally; health uses a "heart" to represent your health, when the player’s health decrease below zero, he fails the game.

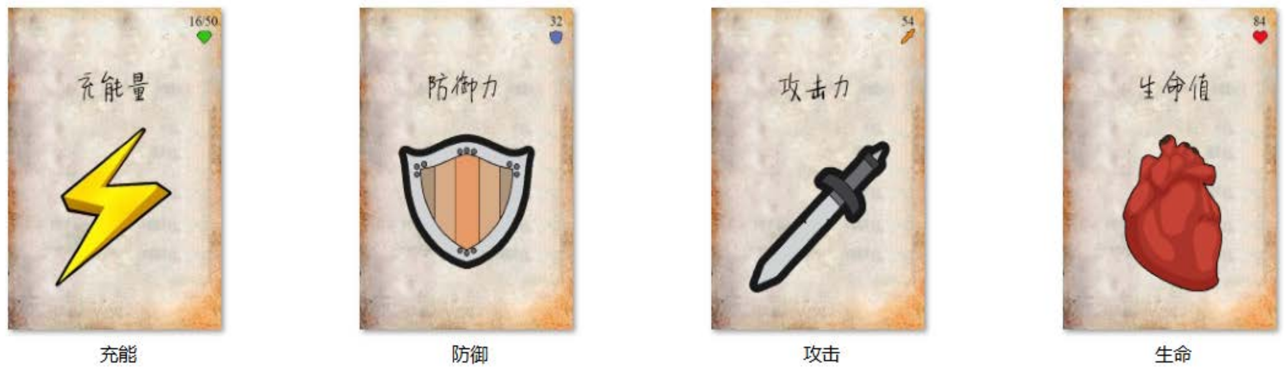


Fig 9. Charge, Defence, Attack and Health

Besides, there are many ordinary cards, such as event card, enemy card or treasure card, which will be added randomly to the card sequence.

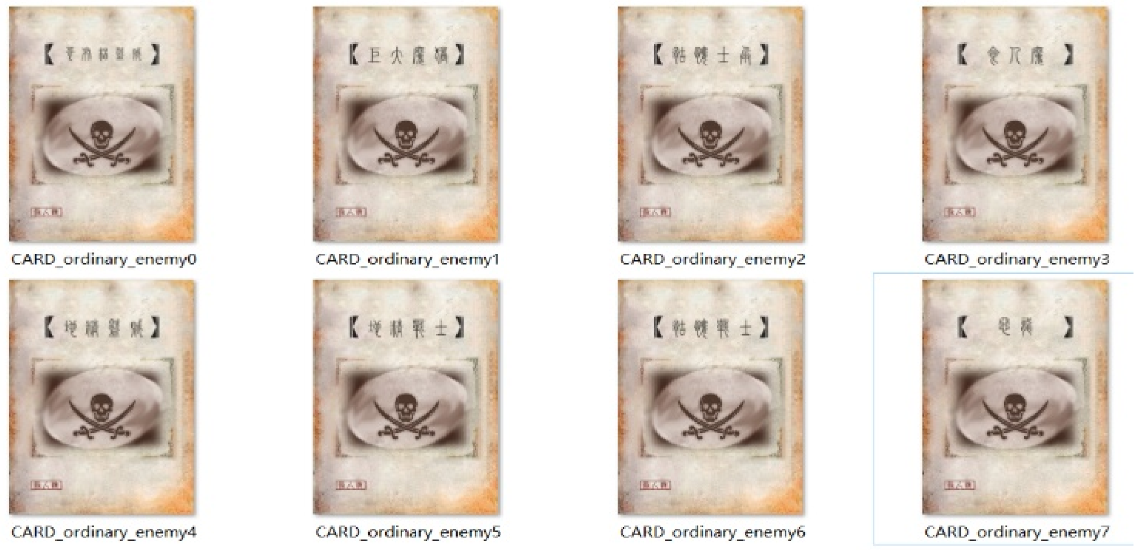


Fig 10. Enemy Cards

Beginning card: A total of three, because the beginning story is too long, it is divided into three parts, respectively, "Loren Kingdom", "Nightmare Coming" and "Rejuvenation March" three story cards corresponding to the three parts.

Diversity cards: there are two turning points. Because the story of the turning point is long, turning point A is divided into three parts, named "strange forest", "mysterious man", "the soul of the dragon". Three story cards correspond to three parts. Turn B is divided into two parts, namely "Old Prophet" and "White Disk". The two story cards correspond to the two parts.

Ending card: There are three kinds of endings. For the same reason, endings A is divided into three parts, which are "defeat dragon", "treasure" and "reincarnation". Three story cards correspond to three parts. Ending B is divided into three parts, named "defeat the dragon", "triumph", "conspiracy". Three story cards correspond to three parts. Ending C is divided into three parts, named "defeat dragon", "truth" and "journey". Three story cards correspond to three parts.

4.4 Design of Poster

Considering the uniform effect of the poster, it was designed in CorelDRAW to make it more like a dice tetragon, and the following draft was obtained.

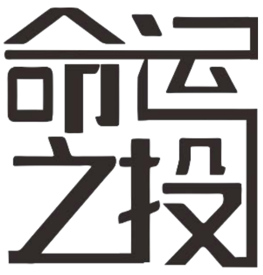


Fig 11. Logo of the game

According to the overall style, we use more rock style charts to overlay the mask and color them. Then adjust the overall tone to golden yellow, and then increase the sense of grandeur, making it bleaker and more vigorous.

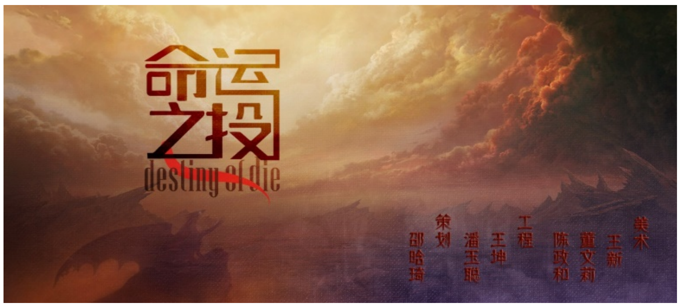


Fig 12. Poster

5. Results

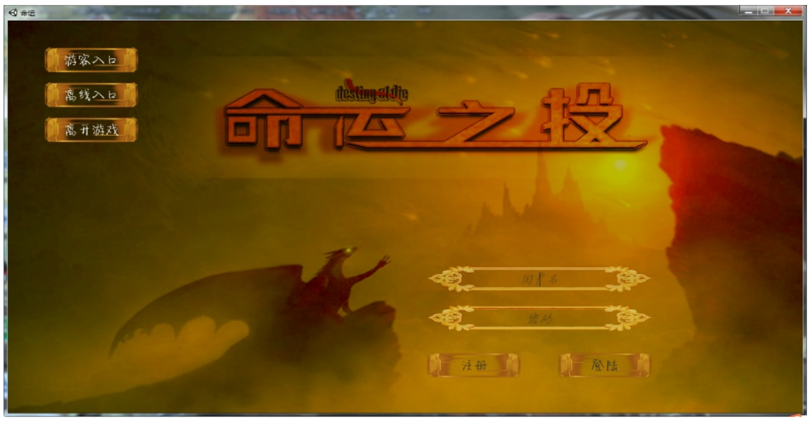


Fig 13. Login Page

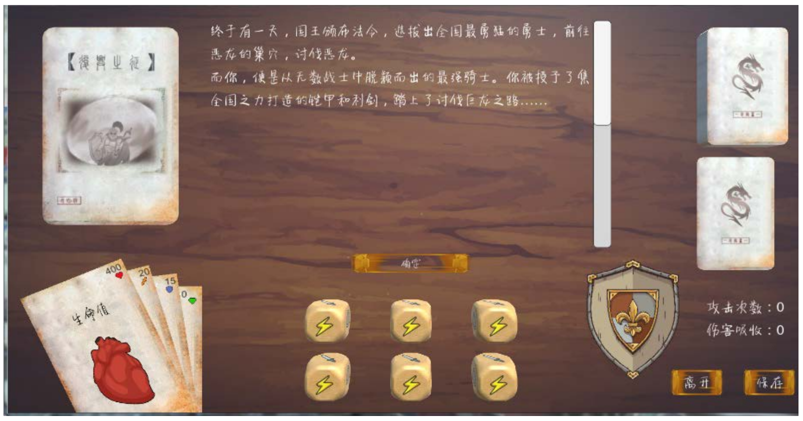

终于有一天，国王颁布法令，选拔出全国最勇猛的勇士，前往恶龙的巢穴，讨伐恶龙。
而你，便是从无数战士中脱颖而出的最强骑士。你被授予了集全国之力打造的铠甲和利剑，踏上了讨伐巨龙之路……

Fig 14. Main Interface

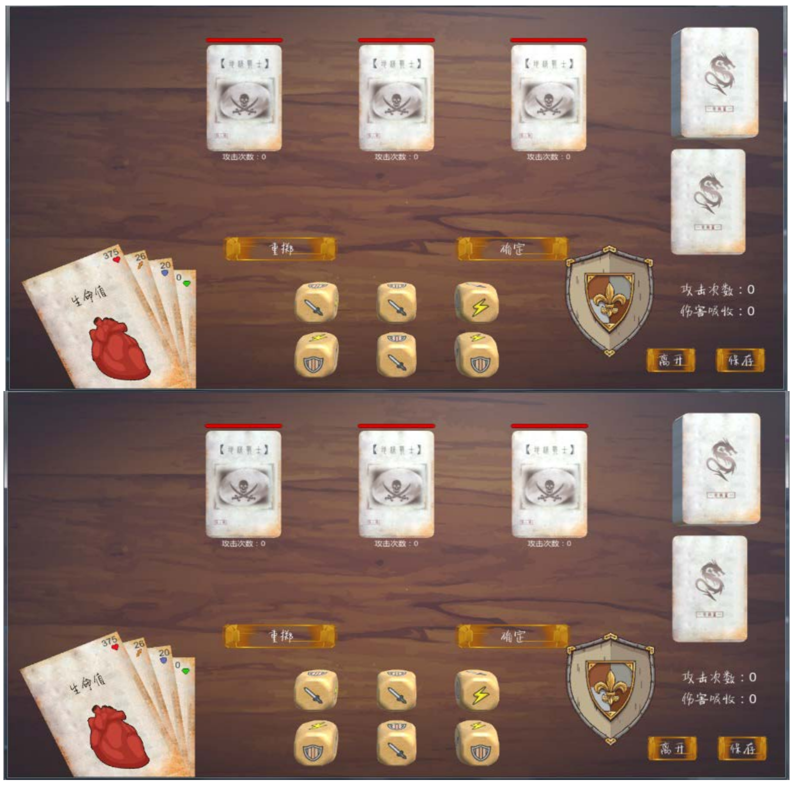


Fig 15. Battle Interface

References

[1] Gregory J. Game Engine Architecture [M]. CRC Press, 2009:45-52

[2] Carlson R E. Universal Gaming Engine:U.S Patent 5,707,286[P]1998-1-13.

[3] Wei Rao. Informal Discussion on the Development of Mobile Games in China [J]. science and education, 2006 (07): 153-154

[4] Yusong Xuan. Unity 3D Game Development [M]. people post and Telecommunications Publishing House, 2012:55-70.

[5] Yun Shang. Card Game at the Crossroads [J]. computer application digest, 2014 (13): 68-69.

[6] Feng Jiang. 3D Game Engine and Implementation [D]. Zhejiang University, 2005:10-13