1 THE OVERALL DESIGN

1.1 Goal of Game

This game should have the following interface and function:

1) Brief and User-friendly interface, The user can easily know how to operate in game and how to change the game settings.
2) Implement the control of character model.
3) Set the obstacles and the game props on the road randomly.
4) Implement the detection and the collision in the game.
5) Build the model pool in the scene game.
6) User data storage on the phone.

1.2 Game Structure

When loading completed, press the play button, the game will start, the character model keeps running, its speed will increase with time goes by. Player should swipe up, down, left and right to control the character to avoid the obstacles. When the character get the game props, it will add different effect to the character according to the type of the props, like multiple scores, magnet, sprint, and so on. You have 2 life, once hitting the obstacle, the life will decrease, after 2 time 's collide, the game is over, you can choose to restart or quit the game, you also can enter into the shop interface,by costing coins you get in the every single playing, you can upgrade the game props as to improve its effect time,shown as Fig. 3.1.

Fig. 3.1 Main Interface

1.3 Character Design

The character prefab contains 2 scripts, one is Controller, the other is Animation manager. It has a component :character controller shown as Fig. 3.2 which is provided by the Unity3D itself, it enables us easy to control the character model without extra scripts, like move, jump etc, this component improve the development efficiency greatly. Controller script as shown in Fig. 3.3 contains the touch-screen control, it decided the character 's state when character get the game props, adding effects to the character. Besides, it creates the best distance effect, when the player break the records of running distance, which is stored in the player's phone, there will be a light wall at the position of the best distance. Also, the time will slow down, when passing the light wall[15~16].

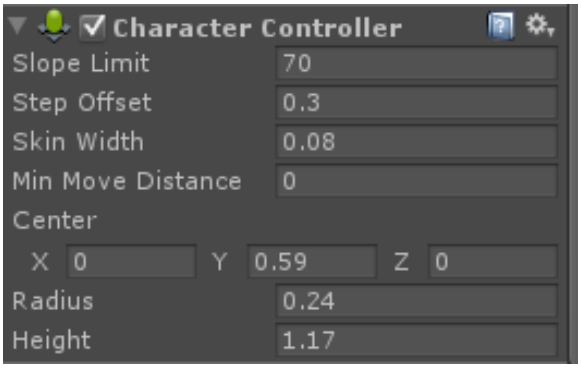

Fig. 3.2 Character Controller

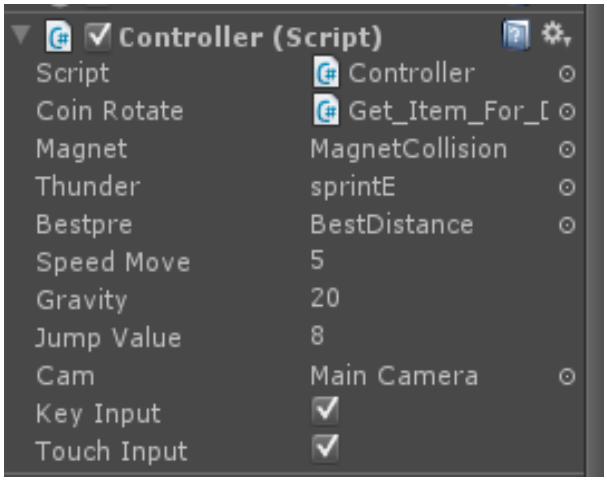

Fig. 3.3 Controller Script

The animation manager script is used to manage the animation of character, when character model jump, turn left or turn right, it will receive the instructions from other scripts to play the relevant animation which has been set in advance.

The character total have 8 kinds of animations shown as Fig. 3.3, the script can use these animation directly by using the method "animation.play()" which is very easy to operate. In addition, every animation has its own duration shown as Fig. 3.4, and speed, it also is set by me in advance according to the user 's state which will make player feel comfortable and not strange when transit from animation to another animation shown as Fig. 3.5.

The character model and the character controller component is like the Fig. 3.6 shows.

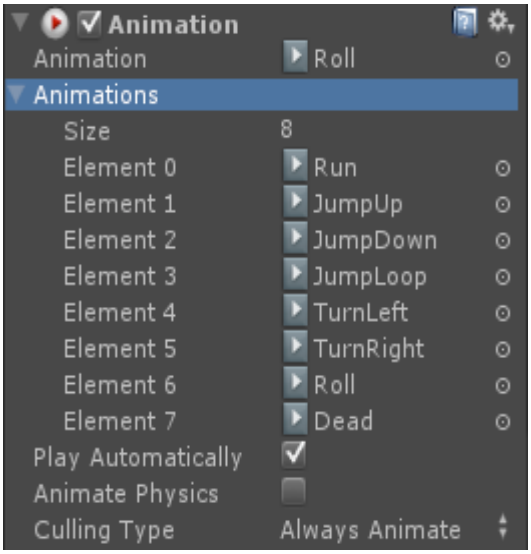

Fig. 3.4 Animation

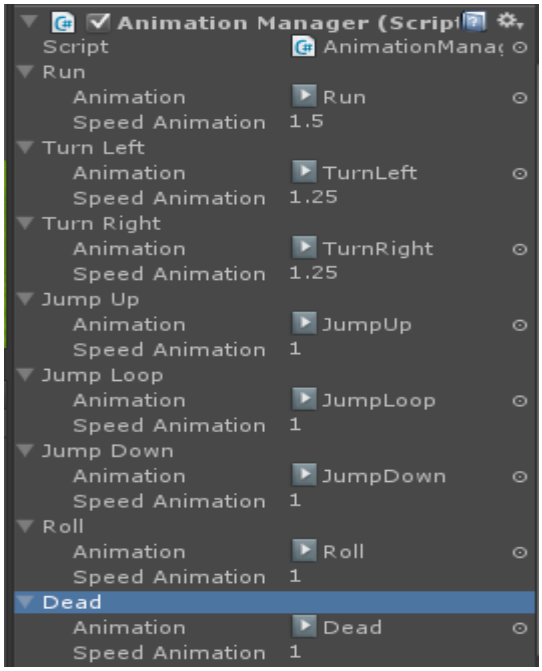

Fig. 3.5 Animation Manager

Fig. 3.6Character Model

1.4 Item Design

In this game, there are 6 kinds of props:Multiple scores, Double jump, Magnet, Sprint, Random, Bounce as shown in Fig. 3.7. They have the same tag: Item, other script can find this prefab in the scene by finding the tag. Every item has an unique Item id, other attributes are showed in the following picture. One significant attribute is Type Item, it decides the prefab's type, varies its function and effect.

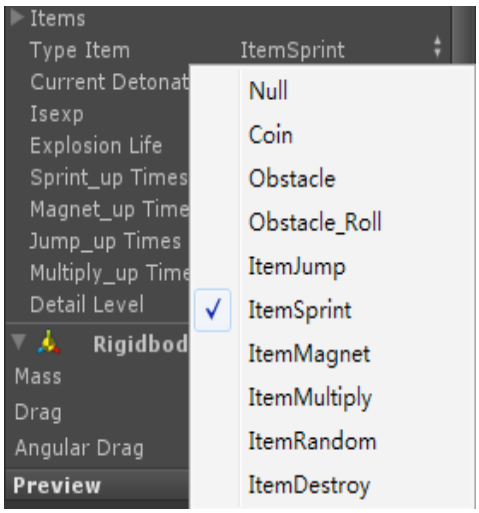

Fig. 3.7 Item Type

The up_times attribute is designed to get the props upgrading times from the user's data which is done in the shop by costing coins, the script will calculate the current props duration, and pass the new value to the other script.

The other kind of items are obstacles, when the character model hit the obstacle item, the character will lose 1 life. This type of props also should be set the Item Type attribute.

Item script contains the attribute, detection of collision and the settings which has been set in

advance by me are as the Fig. 3.8 shows:

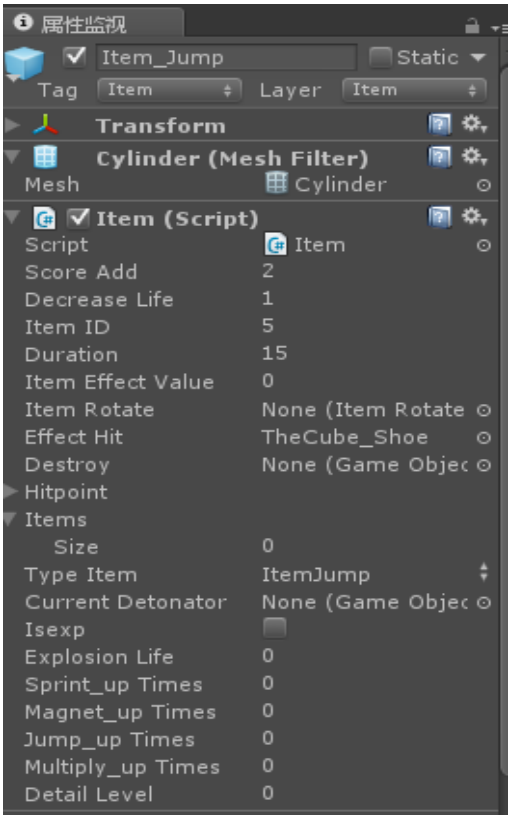

Fig. 3.8 Item script

The specific codes are as following :

```
public void ItemGet(){
if(typeItem == TypeItem.ItemRandom){
    string[] item=new String[]{"ItemJump","ItemSprint","ItemMagnet","ItemMultiply"};
    int i = Random.Range(0,3);
            switch(i)
            {
            case 0:
                typeItem = TypeItem.ItemSprint;
                break;
            case 1:
                typeItem = TypeItem.ItemMagnet;
                break;
            case 2:
                typeItem = TypeItem.ItemJump;
                break;
            case 3:
                typeItem = TypeItem.ItemMultiply;
                break;
                }
            }
```

```
if(GameAttribute.gameAttribute.deleyDetect == false){
        if(typeItem == TypeItem.Coin){
                HitCoin();
                SoundManager.instance.PlayingSound("GetCoin");
        }else if(typeItem == TypeItem.Obstacle){
                HitObstacle();
                SoundManager.instance.PlayingSound("HitOBJ");
        }else if(typeItem == TypeItem.Obstacle_Roll){
                if(Controller.instace.isRoll == false){
                        HitObstacle();
                        SoundManager.instance.PlayingSound("HitOBJ");
                }
        }else if(typeItem == TypeItem.ItemSprint){
                sprint_upTimes=PlayerPrefs.GetInt("sprint");
                Controller.instace.Sprint(itemEffectValue,duration+sprint_upTimes);
                SoundManager.instance.PlayingSound("GetItem");
                HideObj();
                initEffect(effectHit);
        }else if(typeItem == TypeItem.ItemMagnet){
                magnet_upTimes=PlayerPrefs.GetInt("magnet");
                Controller.instace.Magnet(duration+magnet_upTimes);
        //Debug.Log (duration);
                SoundManager.instance.PlayingSound("GetItem");
                HideObj();
                initEffect(effectHit);
        }else if(typeItem == TypeItem.ItemJump){
                jump_upTimes=PlayerPrefs.GetInt("jump");
                Controller.instace.JumpDouble(duration+jump_upTimes);
                SoundManager.instance.PlayingSound("GetItem");
                HideObj();
                initEffect(effectHit);
        }else if(typeItem == TypeItem.ItemMultiply){
                multiply_upTimes=PlayerPrefs.GetInt("multiply");
                //original duration+update times
                Controller.instace.Multiply(duration+multiply_upTimes);
                GameAttribute.gameAttribute.multiplyValue = itemEffectValue;
                //play sfx when get item
                SoundManager.instance.PlayingSound("GetItem");
                HideObj();
                initEffect(effectHit);
        }
        else if(typeItem == TypeItem.ItemDestroy){
                Controller.instace.Destroy();
```

```
HideObj();
initEffect(effectHit);
                }
            }
        }
```

1.5 Game Attributes Design

The GameAttributes script shown as Fig. 3.9 is designed to control all the attributes in the game, including the initialization of the attributes when the game start, and the reset the value of game attributes when the game restart. It has following methods :CountDistance, ActiveShakeCamer, Pause, Resume, Reset;

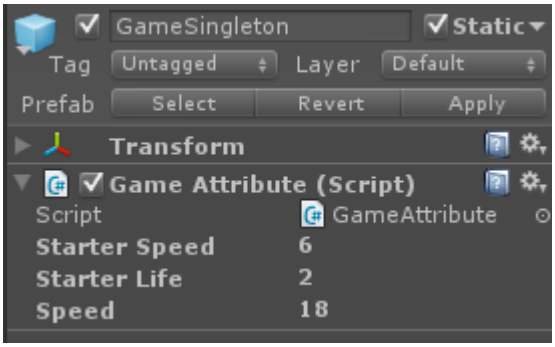

Fig. 3.9 Game Attribute script

The specific attributes in the script are like the Fig. 3.10 shows.

```
public float starterSpeed = 5;
public float starterLife = 2;
                Field
                float starterLife
[HideInInspector]
public float distance;
[HideInInspector]
public float best;
[HideInInspector]
public float coin;
[HideInInspector]
public float totalCoin;
[HideInInspector]
public bool isPlaying;
[HideInInspector]
public bool pause = false;
[HideInInspector]
public bool ageless = false;
[HideInInspector]
public bool menu = false;
[HideInInspector]
public bool deleyDetect = false;
[HideInInspector]
public float multiplyValue;

public float speed = 5;
[HideInInspector]
public float life = 3;
```

Fig. 3.10 Attributes

1.6 Collide Spwan Design

This game object is designed to detect the collision between character model and the collider spawn, the spawn which shown as Fig. 3.11 and Fig. 3.12 is controlled by pattern system manager script, when character model collide with the collider spawn, the spawn 's position will increase automatically, the trees alongside the road and the obstacles in the scene will start initialing in front of the character.

Fig. 3.11 Collider Spawn Prefab

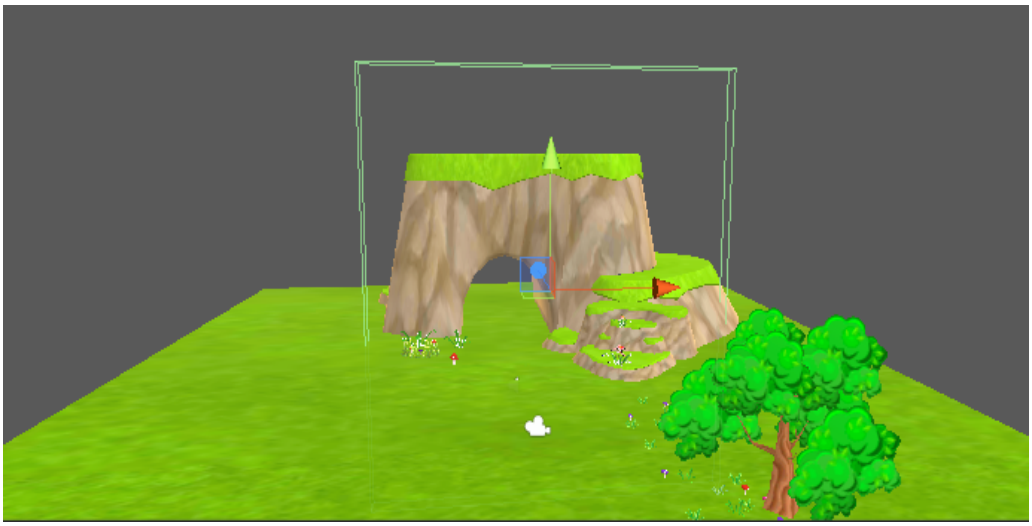

Fig. 3.12 Collide Spawn

2 SCENE SET

2.1 Start Place

The starting place in the Unity3D is shown in Fig. 4.1, this is seen from the workspace which differ from what the player can see.

Fig. 4.1 Starting Platform

When the game start, the scene loading will work first, after that, the main menu will be presented to the player, the background scene is caught by the main camera in the scene. What the player see is like shown in Fig. 4.2.

Fig. 4.2 Loading Interface

The trees and flowers, grass, cave have been set already as the start place where the character

model will initial. When the character is initialized, it will keep running through the cave, and start the adventure in the forest. Restarting the game, the character will come back to this position.

2.2 Game Controller

Game controller is a prefab that has been put into the scene in advance as shown in Fig. 4.3, it is the core of the game which control the process of the game including the game state like pause, playing, game-over.The initialization of the character model is also controlled by this script.

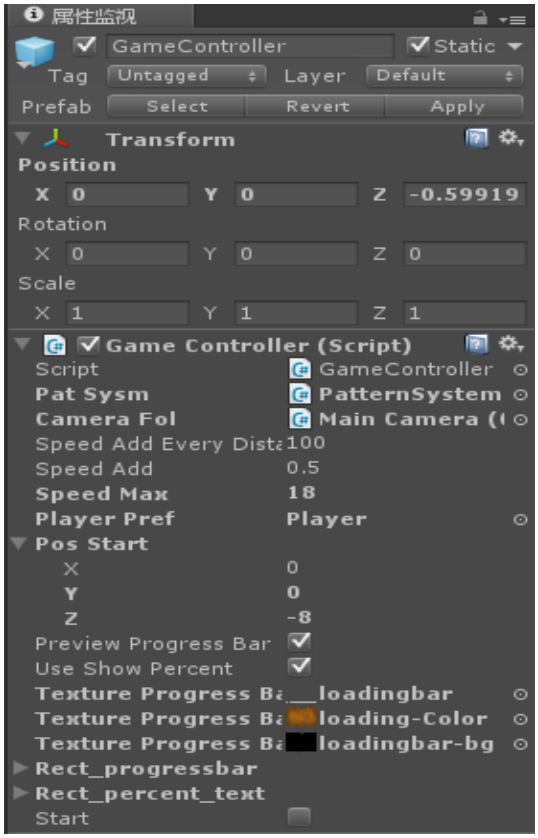

Fig. 4.3Game Controller

3 SPECIAL EFFECT

3.1 Best Distance Effect

When the player break the distance record of himself, the best distance effect prefab shown as Fig. 5.1 will be created on the way in front of the character model like Fig. 5.2, additionally, when the character is closed to this place, the time will slow down half time, then the time will recover to the normal speed after few seconds.

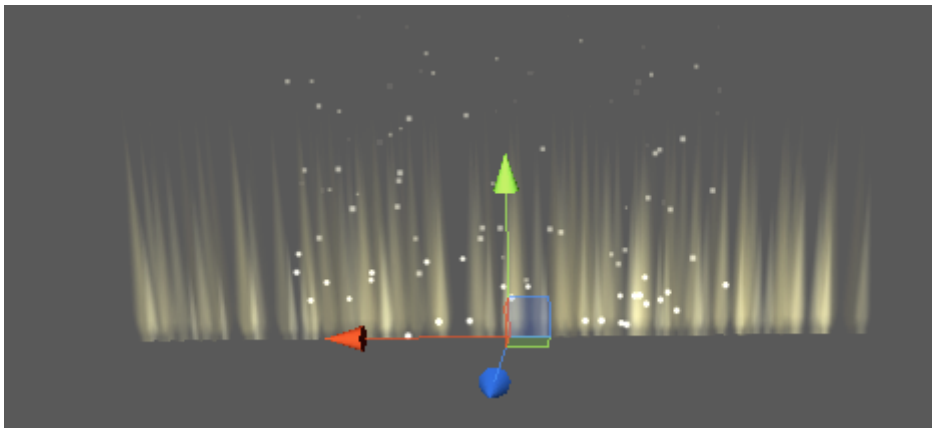

Fig. 5.1 Best Distance Prefab

Fig. 5.2 Best Distance Effect

3.2 Sprint Effect

When the player get the sprint prompt, a special effect will appear on the character model ' body as shown in Fig. 5.3 and Fig. 5.4. The effect 's duration is decided by the sprint 'duration which can be changed by upgrading the prompt in the shop. Also, when sprinting, the camera 'distance will change in the range of 60 to the 75.

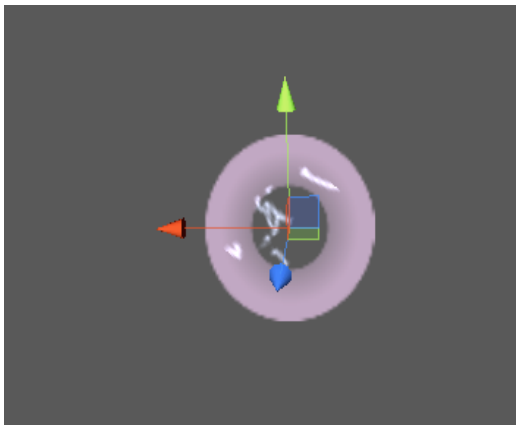

Fig. 5.3Sprint Effect Prefab

Fig. 5.4 Sprint Effect

4 GAME TESTING AND PUBLISHING

4.1 Profile Analysis

When the development was approaching the ending, we realized that the game is run on the android phone, not on PC, the settings are different, the phone ‘s performance is totally different from the PC.The model we use, such as the trees, mountains all are the high quality model, not only the size it takes much bigger, but also the CPU utilization is much higher. We never thought this before, which shocked us when realizing this, and then, I started to learn about the Optimization of the scene, model and the scripts[17~18].

1) Scripts Optimization

The method update is called every frame, if the MonoBehaviour is enabled, it means that, if any code is written in the update method, it will be executed every frame. For the phone, this is really a huge spending on the CPU as shown in Fig. 6.1, which may lead to not smoothly when playing the

game, this is a terrible thing for the player. Then, I tried to put out the codes in the update method using Coroutine，the memory utilization decrease a lot[19].

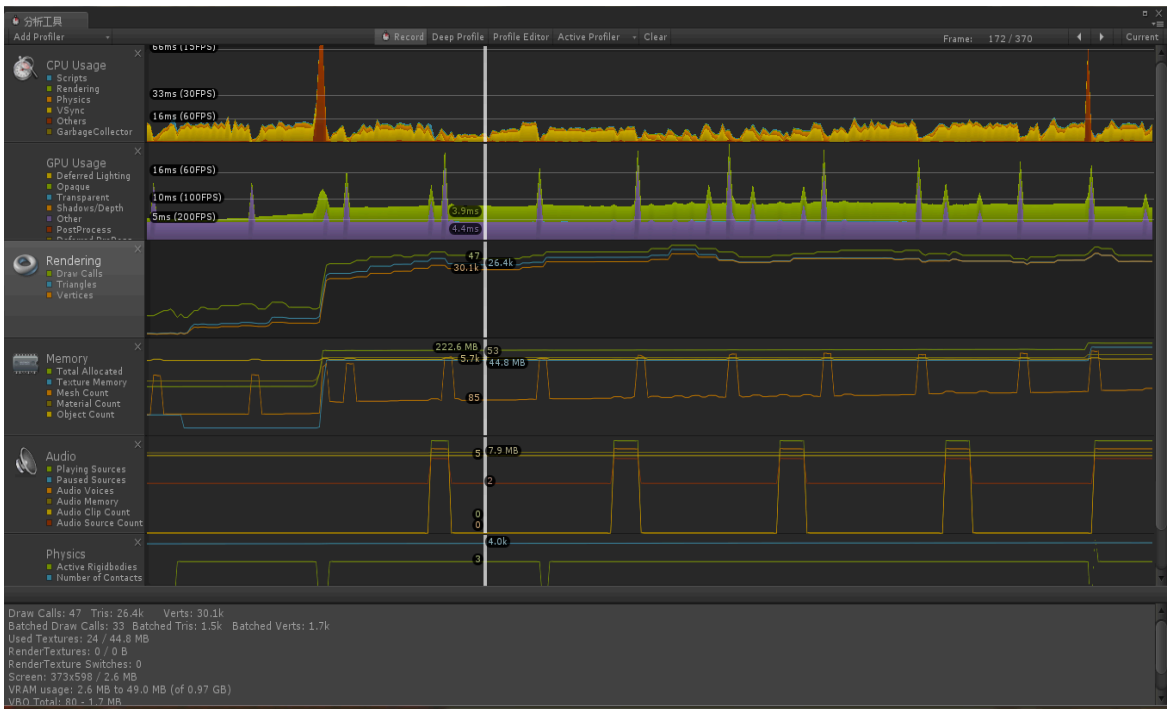

Fig. 6.1 Profile

2) Model Optimization

The model we choose all are high quality models, when testing on the phone, we find the loading rate of the model in the scene is very slow, and the game runs not smoothly. I consult the information about the model optimization on the internet, then I realized that there is a significant parameter called draw calls,To draw an object on the screen, the engine has to issue a draw call to the graphics API (e.g. OpenGL or Direct3D). The graphics API does significant work for every draw call, causing performance overhead on the CPU side.The value of draw calls must be as much low as possible, so as to improve game performance and the fluency on the phone which is vital for phone ’s game.The current value of draw calls is 1500, this is a terrible value which much more than the common phone’ performance. When the draw calls below 80，it will gain a good performance on the phone. Modern graphics cards are really good at handling a lot of polygons but there is a significant overhead for each batch (ie, mesh) that you submit to the graphics card. So if you have a 100-triangle object it is going to be just as expensive to render as a 1500-triangle object. The "sweet spot" for optimal rendering performance is somewhere around 1500-4000 triangles per mesh.

Based on this, I delete the component vertex of the tree model as shown in Fig. 6.2, and use low quality material on the trunk, this really can decrease the cost on draw calls. Additionally, I dim the light, compressed the texture of the model, make it minimum of the size.

After all this have been done, the draw calls has decreased to 60, this value ensure the good performance on the phone.

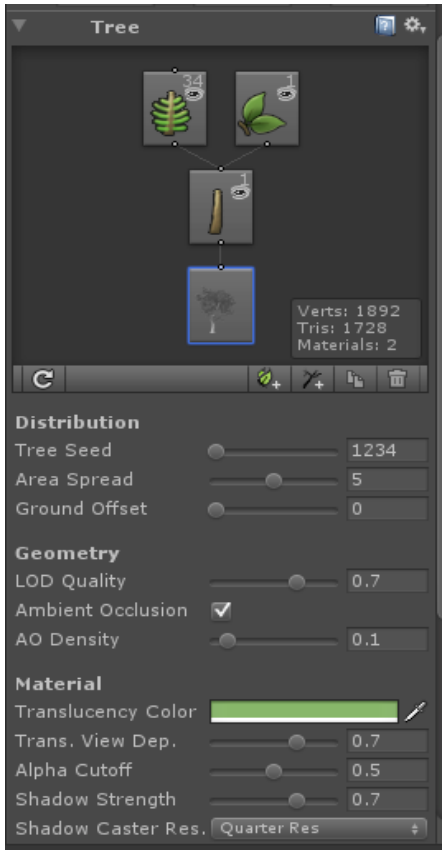

Fig. 6.2 Tree Model Component

3) Scene Optimization

The trees, obstacles and the props all are automatically created, when the character model pass the collide spawn, We use a model pool which is located in the (-100，-100，-100) under the ground, all the prefabs that the game need is stored in here, when needed, these prefabs will be moved to the ground and be placed in order. The platform where the character model has passed will be moved to the model pool and recycled like this[20].

This operation decrease the cost of loading in game,and we set the start place in advance, when the game starts, the start place don't need to be loaded by the script, which decrease the cost as well.

4.2 Phone testing

We use xiaomi3 android phone, the specific state like Fig. 6.3 shows. The total performance of the game is great.

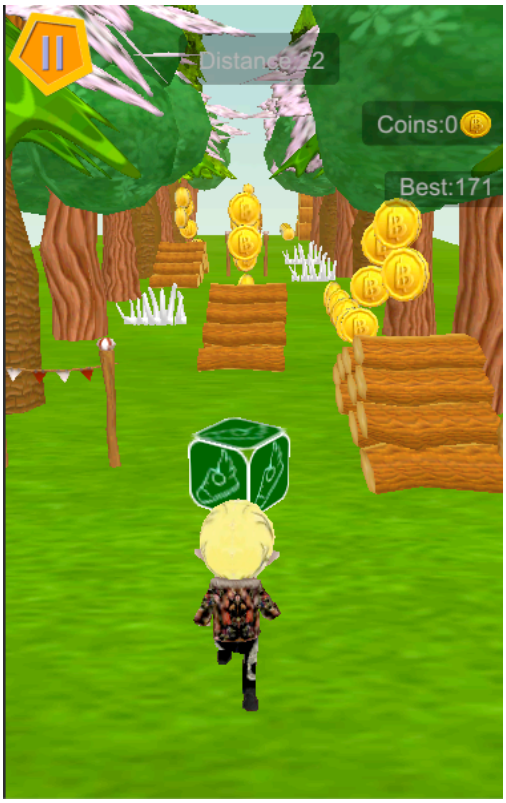

Fig. 6.3 Real phone testing

5 CONCLUSION

The reason why I chose this project as my graduation project one is Unity3D is a new software that is popular among China these years, and it gains interest from me, the other is my own passion for game development. I look forward to developing a personally mobile games, experience the joy in development. I can also learn a lot of useful knowledge during the process of development, although the development is tough, but in the end I will basically solved the problem satisfactory, which made me very happy.

1) Learning experience for Unity3D

In the last semester's project design, my project has been on unity3, so I already have a preliminary understanding of the Unity3D, this software has just entered the Chinese market, so not many people know about it. The information for the Unity3D in China is pretty few, relevant books only have no more than 10, so I log in to a lot of foreign websites and domestic forum to search useful information and tutorial. This is the way how we learn about Unity3D, and it worked. We chose the C #as development language. Foreign companies mostly choose to use JavaScript more, but in the China, c# is generally be chosen, so we chose the c #. This is a new language, but similar with java and c, so it is not difficult for me to learn it.

2) Experience in design of the program

To make a full extent of the game, there almost have none books can help me, what is taught in the

books all are basic knowledge of how to use Unity3D, this made us very headache. Knowledge of the books is not enough, we got a lot of helpful information in the forum, in addition, I got many help from other people in the forum.

Because I also like playing mobile phone games, and the sorts and amount I have ever played are not few, so I have a general understanding of the design of the game development, but in the actual development, I gradually found that although code is essential for a game, but you can't make the game hot only by code. The style of the game, art design and so on are vital to catch people's attention, that is not an easy thing.

When testing in the end of the development, only to find that there are some fatal problems, and these problems had never been noticed or thought before by me, we started to fluster. Through the study, keep trying, the optimization problem is solved in the end. I gained lots of knowledge that the books doesn’t give.

The graduation design promotes my programming ability, comprehension ability as well as for overall architecture to coordinate all aspects of resources. These skills are essential in my later study life, so I think such a project as my final project for my college is enough. I also got the useful knowledge of a lifetime.