Death-Adrenaline

FPS-Game built it with Unity3d

WenXuan Zhang

B.Sc.(Hons) in Software Development

April 14, 2018

Final Year Project

Advised by: Kevin O'Brien

Department of Computer Science and Applied Physics

Galway-Mayo Institute of Technology (GMIT)

Contents

About this project

Abstract This is a LAN[1] game about FPS. Before we start the game, we must have the available servers, otherwise we cannot run the game. So the player can create a client(must type IP address as your servers) or connect the existing client, and then generate a role on the game map (the game is a 3D game so the map is 3D, with X, Y, Z three coordinates), the player can control the role of a series of movements, such as running, jumping, forward, backward, rotating, and so on. It can also control the role of a weapon to fire, attack other players, each player controls the role of HP, when the HP value is 0, the game character dies, and the Game-Over interface will appear. The production of this game is the most popular development software unity3d engine. For LAN games, the unity3d game engine contains a lot of development originals and API[2], developers can not use other development software, all development tools and component unity3d are included.

In this game, we use a series of NETWORK-Component[3] to control LAN Connections, LAN generation and synchronization. For example, using components “NetworkView”, “NetworkPeertype” and so on. I will introduce in detail the technology, API and tools that I used later.

Authors This project has been developed by fourth year student: WenXuan Zhang. I developed this project for our Bachelors of Science Honours Degree in Software Development. Because this project is developed by me alone, and I am not an art designer. So the game map and game characters of this project are purchased on the Internet.

Chapter 1

Introduction

In GMIT, I have been studying for 2 years (level7 and level8). Because I am an international exchange student, I only have 2 years of learning experience. In the 2 years, a lot of professional knowledge of the system was learned, but the most interesting language I was interested in was C#[4], so this project was done on the basis of C#.

So in this project, it is applied to unity3d game development engine production and C# compilation. I believe that many people have played games, such as role playing games, action games, shooting games and so on. So this project is a first person shooter game. Its play is a bit like the popular (Half-life: Counter-Strike)[5] decade years ago. I believe that most people have played CS. A brief introduction to this project: This is a multi - person local online game,in this game, you can create a server or connect to an existing server. Once the server is successful or the server is successful, then the game starts, and you will be able to control the movement, jumping, shooting, and so on. The purpose of the game is to find the others player and kill them. Other players and slaughter. Each character has a HP, and when you are attacked by the enemy, the HP of the role that you control will be reduced, when the HP equals 0, the end of the game will appear. If you kill other players, the game will show "Victory".

A brief introduction to the sequence of development of this game:

Learn the principle of creating local connections

1: create projects and create server side Server using Network. Learned how to use Network components and how does it work.

2: create a client to connect to the server side, use Network components.

3: monitor the event function created by server side and client side. Check whether the method used is run.

4: create a game object on a LAN and use Networkview components to synchronize data.

5: control the movement of game-object(I use cube instead of it) in the LAN.

6: When I figuring out how to create a server and a connection server in the LAN, I look for game materials on the network, such as the material of the game characters, the animation and the game map.

Design control of the game and play rules

7: When I found the game material, next is about the design of the first person shooting control role.

8: learning the first person controller and add the character and learn how to control the soldier's movement, next is add the field of vision control to the character and move left or right.

9: when I finished how to controller movements, then add animation to the character and control the play of character animation.

10:After finished how to control character , need add a weapon ,and modify the control mode of camera's upper and lower field of view.

11: The control field of vision can move along with the movement of the weapon(repair the bug that can't control the rotation of the waist and fix up the tilt of the camera's field of vision)

About design bullets and shoot

12: design the shape of a bullet.

13: control the movement of bullets and the detection of bullets.

14: Design a spark when player open fire and control the spark flash time.

15: realization of shooting function and design the striations in the bullet

And design the bullet animation. And add effect the striations in the bullet.

16: modify the way of shooting bullets.

Adding Game Map and Realizing the function of the game

17: Import the game map ,learned how to use NGUI[6] to design game interface. Like: game logo, game buttons , game meun. And add animation to the game menu interface.

18: monitor and create the press of the server button and initialization of the Game Server.

19: complete the server side to create a character, finish the connection and role creation of the client(Because I had studied and created and connected before, it only took 1 hours to get it done.).

20: display shooting sight and control it .

21: analysis of the role of the role in the local area network, use RPC[7] remote call to control single control of roles and use RPC remote call to control the play of character animation.

22: calculate the damage of bullet shooting, finish the synchronization of the play and injury of the death animation.

23: add background music to the game, add the firing sound to the bullets, and modify the damage of the bullet to the characters.

24: design the end of the game interface

The above is how I develop the game detailed process, during the development of the game, encountered loads of bugs, most of I have solved, and some of the reasons for a variety of reasons, unfortunately not solved it , but from then on, I also learned a lot of knowledge and know how to do a game difficulty, and not

As simple as imagination. Here, I'm very grateful to my supervisor Kevin O 'Brien and my teachers for their support and encouragement.

About unity3d engine

Why I choosed Unity3d engine

Unity3d[8] game development support platform is undoubtedly the most popular platform nowadays. Meet the vast majority of the project needs. Unity the job market is hot and the market needs to be large, Unity has a wide range of employment. The game industry occupies most of the proportion, there are also virtual reality, augmented reality and other directions, and the prospect of employment is hot. Jobs that can be done: Game Development Engineer, mobile application development engineer, game scene designer, game special effect designer, VR development engineer, AR Development Engineer.

Unity3d cross platform release is simple, free (except IOS[9]),unity3d development speed has advantage it has its own network store, most of the effect can be bought, without developing, good editor development function, editor - friendly, that is, I can easily design a set of custom editors. This is vital to me, because in this project I was solo it.

In my level7, because I'm interested in C#, I often use VS[10] to program, unity3d is not only perfectly compatible with C#, but also perfectly compatible with VS, which is very important to help me save a lot of unnecessary time.

About Asset-Store. You can make plug-ins, sell online, earn some benefits, or purchase other plug-ins as a reference or reference. Sometimes, buying some plug-ins can help you quickly out of the current dilemma. One is to solve the schedule problem, the other is to solve the problem of train of thought. This is not before the other engines. It was really helpful, it solved my cant find game materials problem, saved lots of time.

The structure of the game

In this game, Most of the implementation functions are based on C# and small part of JavaScript. For example, shooting function, calculating blood volume function, playing the role animation function is applied to C#. For instance, Move function, jumping function and gravity senor are accomplished by JavaScript. In this project, the scripts of “player” and “GameController” It's the most time I've developed and studied.

All data is used by unity's own components, such as storing and reading data, creating connections, connecting servers, etc .

The media resources of games are all bought on the Internet, not made by myself. Game materials include game scenarios, game roles, game characteristics, animation effects and various game effects

Chapter 2

Context

The general background of this project is a 3D first person shooter game, This game is running under the windows system, As soon as the game is ran, the game logo will be displayed and then entered into a main menu interface. There are 2 buttons and an input text, the IP address can be output in the input text, and then the button of the creation server and the button of the connection server can be clicked. Once the game starts, the player will enter the game scene to get one. A controllable role, the location of the role generated by the server is different from the role generated by the client. The main interface and game end are done with the NGUI plug-in, the role control is implemented by the JAVASCRIPT[11], the game starts, the game and the end of the game are all implemented by C#. It has been applied to many API components in UNITY3D, such as remote call [RPC] components, network setting network components and remote implementation of animation synchronization components, and so on. Because the game market is very hot and the game developer job prospect is good, especially in the VR and AR virtual filed, the main thing is that I have a great interest in the game development. I have got the master offer of game development and design at the University of limerick, so I chose to develop a game alone as my final project, And I can also test my current ability to do projects.

2.1 Objectives

The overall objective of my project is learning and search about how to use plug-ins and C# and JavaScript in unity3d engine together with unity3d's API to achieve various game functions. For example use NUGI to design UI, use NET.WORK component create service and client ,searching how to control character animation and weapon ,bullet animation effects ,add sound effect and so on.

1. **Logo :** The objective for the UI logo is at the beginning of the game, there is a logo will be display. The information is the name of the project, the author of this project and the name of the author.

2. **Main menu interface:** After entering the main menu interface, there are 2 buttons for the player to choose. One is the connection server, the other is the creation of the server, and the other is the input text. If you need to create or

connect the server, you need to output a IP address. The default IP address is 127.0.0.1.

2. **Game scene:** Once the game starts, you will enter a 3D game scene, and you can control your role in a series of movements in this scene, such as moving, jumping, shooting, and so on. The purpose of this game is to find other players in the game scenario and kill them.

3. **Game Over menu:** This objective is each player's controlled role will have a 100 HP. When a player are shot kill by other players, that is, the player - controlled role HP is equal to 0, then the game over interface will be displayed. You can click the EXIT button to exit the game..

4. **Sound effect:** There are three sound effects in this game. The first is that when you enter the game, that is, when you stay on the main menu of the game, you play music. The second is when you create the game success or when the game is successful, then the game map is loaded, another kind of music is played, third One is when you use a weapon to fire, the sound effect of the bullet will appear.

5. **Remote call and synchronization :** Because this game is a LAN game, all the actions that the role does on the server are synchronized to the client, such as a role shooting, moving, and firing on the server. Then the animation of the series of actions will be synchronized to the client, On the contrary, all actions performed by the client must be synchronized to the server side, Use RPC this function for remote call.

2.2 Project Links

All development materials and development codes of this project are in the GITHUB link.

Links:

- https://github.com/neroZWX/Death-Adrenaline

2.3 Chapters Review

In this chapter we I briefly review of the development process and development logic of this project is made, how did I finish this project.

2.3.1 Methodology

In this chapter I will introduce how my project applies to methodology to complete this project. How do I learn the methodology, What's the problem each time, and how the results of each game test are, and project progress every time met with supervisor.

2.3.2 Technology Review

In this chapter, I will introduce in detail what technology is applied to my project and what technologies I use to solve BUG[11] every time.

2.3.3 System Design

In this section I will give a detailed description of how the code I compiled is working together, if the functions are implemented, the functions of the independent code implementation and the functions of multiple codes are implemented together.

2.3.4 System Evaluation

In this chapter I will prove my software is robust, I will also analyze the status of my project from various aspects.

2.3.5 Conclusion

This chapter I will Briefly summarise my context and objectives. and Acomprehensive analysis and summary of the project, and Summarize the knowledge I learned in this project, the self rating of this project and whether I will further develop this project. And I met with how BUG solved and perfected it.

Chapter 3

Methodology

In this chapter I will introduce how I carry out this project, how to search for research on the Internet, how I think at each development stage, and how to complete all the functions step by step. I will also in traduce detail the tools, language and platform used and the way to create this game..

3.1 Test-Driven Development

In the game I developed, I used Test-Driven development methodology. Using this method can greatly reduce the problem that I encountered BUG .It is a new development method which is different from the traditional software development process. It requires writing test code before writing a function, and then only writing the function code that makes the test run through the test to push the whole development through the test. This helps to write concise, usable and high quality code, and speed up the development process. The basic process of test driven development is as follows:

(1) Quickly add a test

(2) run all the tests (sometimes only need to run one or part), and find that the new tests cannot be passed.

(3) make some minor changes and let the test program run as fast as possible, so we can use some unreasonable methods in the program

(4) run all the tests and all pass through, Reconstructing code to eliminate repeated design and optimize design structure.

(5) In short, it is not running / running / refactoring. That's the slogan of test driven development.

It is possible to avoid and find mistakes as soon as possible before the test is referred to and run all the tests frequently, which greatly reduces the cost of subsequent testing and repair and improves the quality of the code. Under the protection of the test, the code is constantly rebuilt to eliminate repetitive design, optimize the design structure, improve the reusability of the code, and improve the quality of the software product. The unit test code produced by Test-Driven Development is the most perfect developer document, showing how all API should be used and how it works, and that they keep in sync with the work code, and are always the latest

3.2 Agile Development

In this project I used agil development .Agile development is centered on user need evolution, and software development is done by iterative and gradual way. In the agile development, the software project is cut into multiple sub items at the beginning of the construction, and the results of each subproject are tested, with the features of visual, integrated and operational use. In other words, it is to divide a large project into a small project that is interconnected, but can also run independently. In the process, the software is always in a state of use.

Tell the truth, As a development process model, agile process has generated many different programming methods that can be applied to practice. ,like extreme programming, to embody some characteristics of agile development process. The extreme programming process is divided into four stages: planning, design, coding and testing, It can be roughly divided into planning stage, design stage, coding stage and testing stage.

But agile development is a methodology that is more likely to be used by a team or a company to develop software. It is not very useful to me as a single project, but I refer to the principles and features of agile development, and find that both team and single software development are very worthy of reference and follow. For example, I can't write all the functions of this project or implement every function, but I can set up all the functions as one independent small project, for example, I can take the shooter in this game as a project, role mobility as a project, and finally step by step. The agile development methodology really accelerates and optimizes a project by solving a small project with one one, and then combining all the functions after the test. (personal opinion)

3.3 Resource Manager

As a single independent project, a part of each project, or a function of the project, should be saved. Because there is no team, the code and resources of the project will not be shared by other people, but the loss rate of the code will be greatly increased, so the use of GITHUB[12] will be better managed. The code has been a resource for various projects. Each test and the modification of each code can be saved by uploading to GITHUB, and even if the project is wrong and the file is lost, we can also read the previous version from the GITHUB. For example, I developed a shooting function in this project. I just realized the shooting function and was satisfied with the function temporarily. But after a few days, I reviewed the function and found it is not very ideal. In this way, it can be used to modify the version of GITHUB and then make the later. Join in. To put it simply, it is to make different modifications according to different development versions. Since there are various versions of the development of a game or software, each version will be tested for debug, so it is very helpful to save each test version for future development.

If you have the opportunity to develop your project in the future, then the resources you save in GitHub will be very useful. If you develop a project from a personal project to a team project, then the resource management methodology will help you to make the next progress, and everyone in the team can be saved and shared in GITHUB.

Use github it can be used as a version control system and collaboration tool to publish work. you can archive projects and share with others, and let other developers help you accomplish this project. The advantage is that he supports many people to complete a project together, so you can talk and talk on the same page. One of the best features of Git is the ability to track errors, which makes it easier to use Github. Bugs can be public, you can submit errors through Github reviews. those features of github is really helpful for a team or single develop a app.

.

Chapter 4

Technology Review

In this chapter, I will introduce all the technologies I have applied , The benefits of using these techniques to play games.

4.1 Development Environment

In this whole project, many different technologies are used to implement the various functions of the game, but most of the technologies are based on unity3d, and all of the technologies in unity3d are learned in the unity3d official network. Programming languages are also used in different languages.

4.1.1 Microsoft Visual Studio

Microsoft Visual Studio is the full name of VS. VS is the Microsoft Corp development kit series. VS is a basically complete set of development tools that include most of the tools needed throughout the software lifecycle, such as UML tools, code control tools, integrated development environment (IDE), and so on. The target code is applicable to all the platforms supported by Microsoft, including the Microsoft Windows, Windows Mobile, Windows CE,.NET[13] Framework,.Net Core,.NET Compact. It also perfectly supports unity3d's programming language, C# and Javas, which is very suitable for programming with unity3d engine.

4.1.2 Unity3d

Unity3D is a multi platform integrated game development tool developed by Unity Technologies, which allows players to easily create interactive content such as 3D video games, architectural visualization, real-time 3D animation and other types of interactive content. It is a comprehensive and integrated professional game engine. Unity is similar to Director, Blender game engine, Virtools or Torque Game Builder and so on. Its editor runs on Windows and Mac OS X, and can release games to Windows, Mac, Wii, iPhone, WebGL (requiring HTML5[14]), Windows phone 8, and the platform. You can also use Unity web player plug-in to publish web games, and support Mac and Windows web browsing. Its Web player is also supported by Mac.

Unity3d It can build projects separately from client, art, model and so on, and then upload and update them separately by SVN[15]. The words of fine art only manage to upload UI well, and then the client side updates the UI itself and then handles it in the client, and there is something unsuitable to communicate with the art.

For unity3d: AssetsProject: mainly store models, special effects and other art materials. Some test scripts are also put in order to test that art materials can match the script correctly.

Data Project: store planned values, level editors, etc. You also need to put some test scripts so that the checkpoints can run.

Scripts Project: programmers specially maintain projects. It is mainly used for logical development and storing some art resources.

- **Final Project**: final merger.

In the development process, according to the specific circumstances, the above two items can be combined.

4.1.3 GitHub

GitHub is a hosting platform for open source and private software projects, because Git is only supported as the sole version repository format, so it is called gitHub.

GitHub was formally launched in April 10, 2008. Besides the GIT code warehouse hosting and the basic Web management interface, it provides the functions of subscriptions, discussion groups, text rendering, online file editors, collaboration Atlas (reports), code fragment sharing (Gist) and so on. At present, its registered users have exceeded 3 million 500 thousand, and the number of hosting versions is also very high. There are many well-known open source projects such as Ruby on Rails, jQuery[16], Python[17] and so on.

GitHub can host a variety of GIT libraries and provide a web interface, but unlike other services like SourceForge or Google Code, the unique selling point of GitHub lies in the simplicity of branching from another project. The code to contribute to a project is very simple: first click on the "fork" button of the project site, then check the code and add the changes to the split code library, and finally apply for code merge to the project leader through the built "pull request" mechanism.

Execute the following command to create a clone version of the local repository.:

```
git clone /path/to/repository
```

If it's a repository on the remote server, your command will look like this:

```
git clone username@host:/path/to/repository
```

Update your local repository to the latest changes and implementation:

```
git pull
```

add files in your repository

```
git add <filename>
```

commit message

```
git commit -m”test”
```

4.1.4 C#

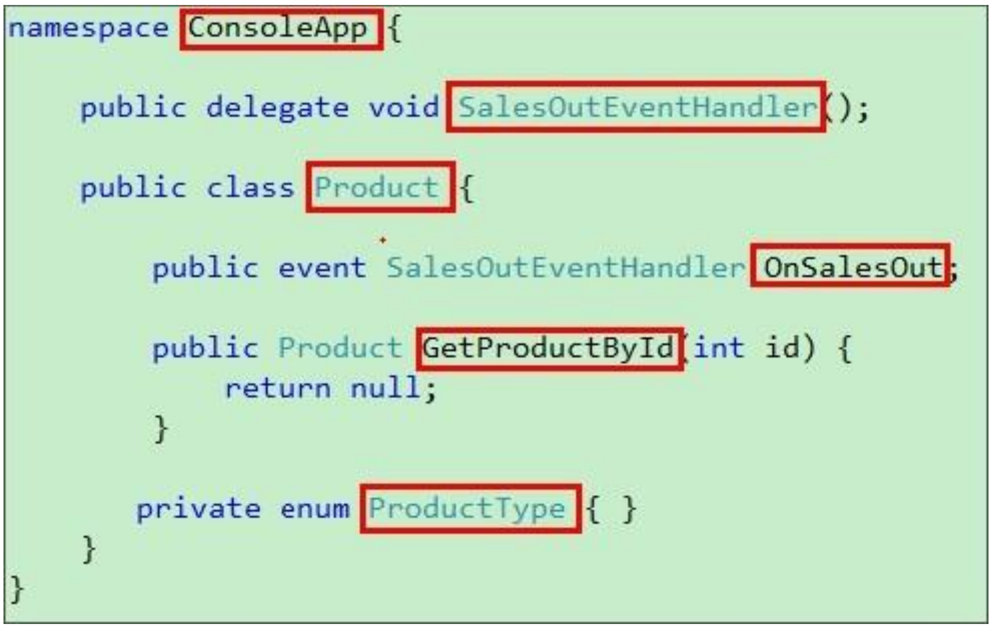
C# is an object-oriented, advanced programming language running on.NET Framework issued by Microsoft Corp. And is scheduled to appear on the Microsoft career developers Forum (PDC). C# is the latest achievement of Anders Hejlsberg, a Microsoft Corp researcher. C# looks surprisingly similar to Java; it includes a process such as single inheritance, interfaces, almost the same syntax with Java, and compiled into intermediate code to run again. But C# is quite different from Java, and it draws on one of the features of Delphi, which is directly integrated with COM (component object model), and it is the leading role of the Microsoft Corp.NET windows network framework.

C# is a safe, stable, simple and elegant object-oriented programming language derived from C and C++[18]. It inherits the powerful functions of C and C++, and removes some of their complex characteristics (for example, no macro and multiple inheritance is not allowed). C# combines simple VB visualization and C++'s high efficiency, and becomes the preferred language for.NET development with its powerful operational capabilities, elegant grammar style, innovative language features, and convenient component oriented programming support.

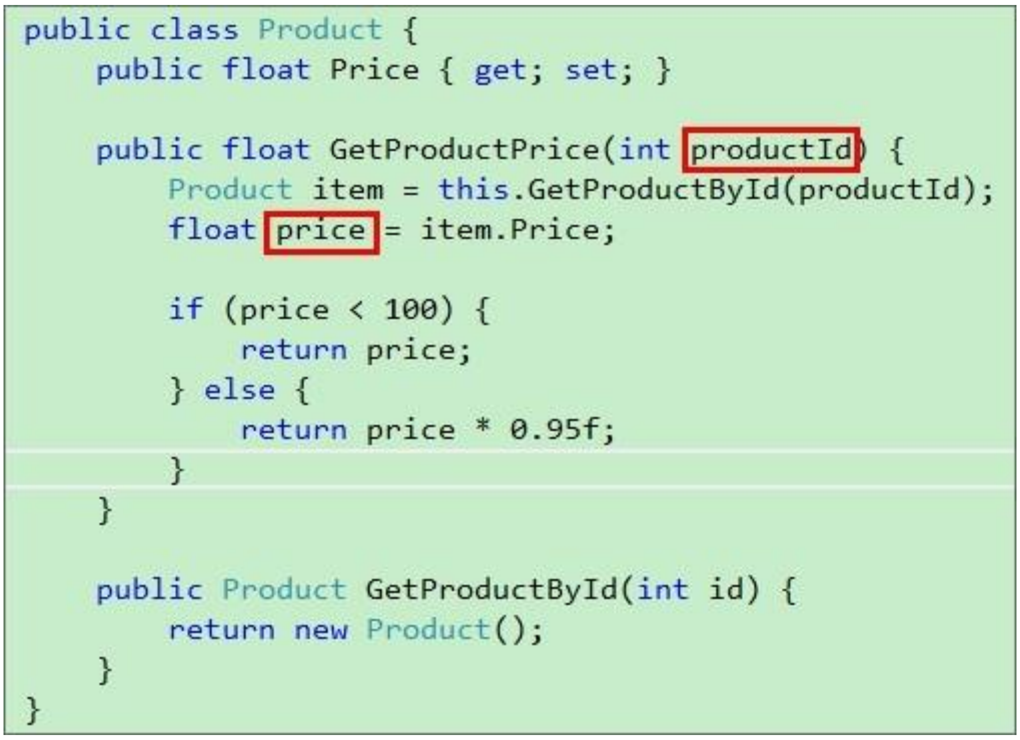
C# is an object oriented programming language. It allows programmers to quickly write applications based on the MICROSOFT.NET platform, and MICROSOFT.NET provides a series of tools and services to maximize the development and utilization of computing and communications.

C# enables C++ programmers to develop programs efficiently, and because they can call native native functions written by C/C++, so it will never lose the powerful functions of C/C++. Because of this inheritance relationship, C# and C/C++ have great similarity. Developers who are familiar with similar languages can quickly turn to C#..

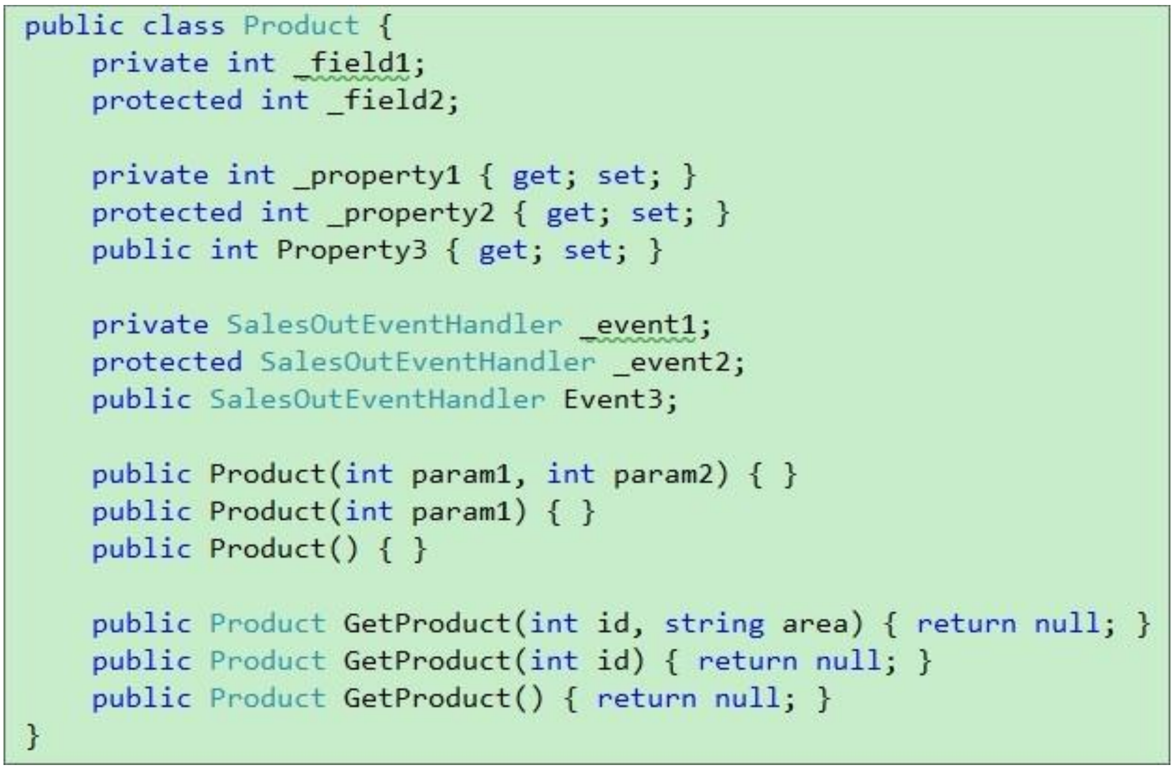
A simple C# code display:



```
namespace ConsoleApp {

    public delegate void SalesOutEventHandler();

    public class Product {

        public event SalesOutEventHandler OnSalesOut;

        public Product GetProductById(int id) {
            return null;
        }

       private enum ProductType { }
    }
}
```



```
public class Product {
    public float Price { get; set; }

    public float GetProductPrice(int productId) {
        Product item = this.GetProductById(productId);
        float price = item.Price;

        if (price < 100) {
            return price;
        } else {
            return price * 0.95f;
        }
    }

    public Product GetProductById(int id) {
        return new Product();
    }
}
```



```
public class Product {
    private int _field1;
    protected int _field2;

    private int _property1 { get; set; }
    protected int _property2 { get; set; }
    public int Property3 { get; set; }

    private SalesOutEventHandler _event1;
    protected SalesOutEventHandler _event2;
    public SalesOutEventHandler Event3;

    public Product(int param1, int param2) { }
    public Product(int param1) { }
    public Product() { }

    public Product GetProduct(int id, string area) { return null; }
    public Product GetProduct(int id) { return null; }
    public Product GetProduct() { return null; }
}
```

4.1.5 javascript

JavaScript a literal translation script language, is a dynamic type, weak type, prototype based language, built-in support type. Its interpreter, known as the JavaScript engine, is widely used as a part of the browser and is widely used in the client's scripting language. It was first used on a web page in HTML (an application under standard General Markup Language) to add dynamic functions to the HTML web page..

JavaScript It's an interpreted scripting language (code is not pre compiled), It is mainly used to add interactive behaviors to pages in HTML (standard generic markup language) application, It can be directly embedded into HTML pages, but writing a separate JS file is conducive to the separation of structure and behavior. Cross platform features, under the support of most browsers, can run on multiple platforms (such as Windows, Linux, Mac, Android, iOS, etc.), avascript script language, like other languages, has its own basic data types, expressions, arithmetic operators and basic program framework of programs. Javascript provides four basic data types and two special data types to process data and text. While variables provide places for storing information, expressions can accomplish complex information processing.

A simple display of JavaScript code:

The following JavaScript statement outputs the text "Hello World" to the HTML element of id= "demo".

```
document.getElementById("demo").innerHTML="Hello World";
```

AvaScript code (or only JavaScript) is a sequence of JavaScript statements, Browsers execute each statement according to the order of writing:

```
document.getElementById("demo").innerHTML="Hello World";

document.getElementById("myDIV").innerHTML="How are you?";
```

JavaScript can operate the functions of two HTML elements:

```
function myFunction()

{

document.getElementById("demo").innerHTML="Hello World";

document.getElementById("myDIV").innerHTML="How are you?";

}
```

4.2 NGUI Plugins

NGUI is a powerful UI system and event notification framework for Unity (both Pro and Free) written in C# that closely follows the KISS principe[19]. It features clean code and simple, minimalistic approach to everything. Many behaviour classes are kept under 200 lines of code. For a programmer this means a much easier time when it comes to working with the kit — from extending its functionality to tweaking the existing one. For everyone else this means better performance, less frustration, and more fun

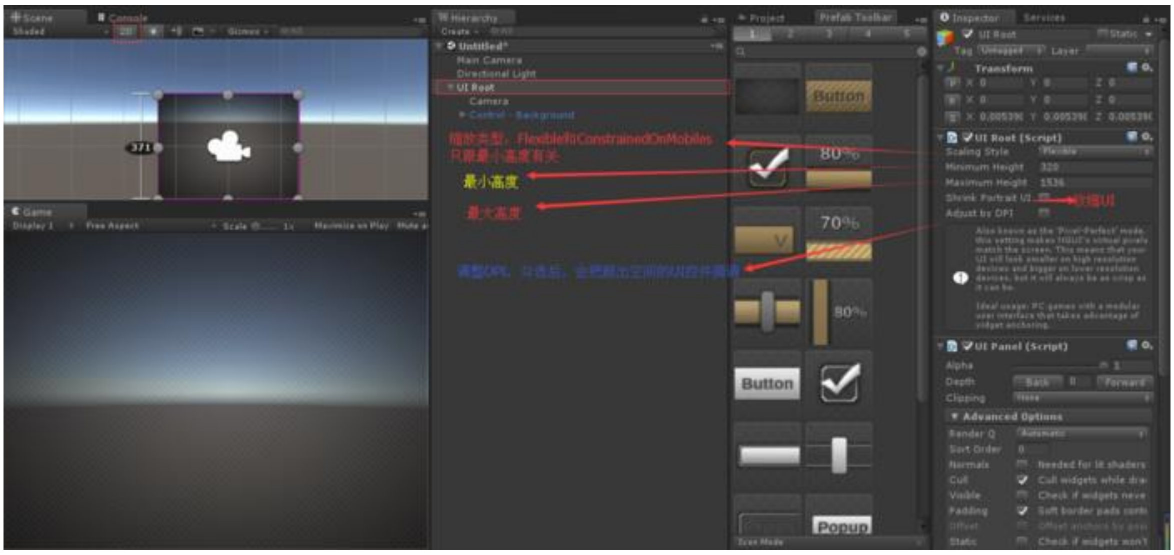
1. **DrawCall**: in Unity, the process that every engine prepares data and notifies GPU is called Draw Call once. The lower the Draw Call value is, the better rendering performance will be Draw Call Tool:NGUI--, Open--, Draw Call Tool, you can see some information about DrawCall. Factors that affect DrawCall: Number of Atlas, The number of Font dynamic fonts, Render order: depending on the depth of Sprite Depth, Unity will render according to the Depth of the control. Render from front to back. When using the same material, the control will merge into a Draw Call. If it is different from the previous material, it will regenerate a Draw Call and Number of Panel

2. **UI Root**: If Create is 2D UI., it automatically generates a UI Root and generates a Camera under its sub layer. UI Root game objects are always placed on the top of NGUI UI level.

   UI Root game objects include Transform components, +UIRoot script components, +UIPanel script components, +Rigidbody components.

   The UIRoot script component provides some parameters to scale the UI interface according to the inverse ratio of screen height.

   NGUI for every UI scenario, it takes a UIRoot as the root of UI game object tree, There are only 4 attributes of UIRoot game object, namely, scaling rule, manual height, minimum height and maximum height respectively.



UI Root (Script)
Scaling Style: Flexible
Minimum Height: 320
Maximum Height: 1536
Shrink Portrait UI
Adjust by DPI

UI Panel (Script)
Alpha: 1
Depth: Back 0 Forward
Clipping: None
Advanced Options
Render Q: Automatic
Sort Order: 0

UIRoot is a screen width of 2 Int height = Mathf.Max (2, Screen.height); ManualHeight = Screen.height * 1024 / Screen.width; //Width based screen resolution adaptive

3. **Camera**: If Create is 2D UI., it automatically generates a UI Root and a Camera. 2D UI is projected by an orthogonal camera. The NGUI custom camera makes the projection range in a square visual range, and is ignored in the visual performance of Z axis.

   Camera Tool: This tool can see all the cameras in the current scene, including Main Camera and UI camera.

4. **Altas**: Atlas is a container, which contains many coordinate information of Sprite. Compared with using many small maps to render UI, using a large map that contains all the small maps is much more efficient. These small maps are called Sprite, and the big map is called Atlas. Altas Maker:NGUI provides this tool to create a Altas atlas.

5. **Font**: Font Maker: is used to observe the existing Font fonts, or to make static Font (Bitmap font) and dynamic Font.

   Static Font is usually the image text obtained from art department. It needs conversion to be used in NGUI. The static Font must belong to a certain Atlas.

NGUI indicates that dynamic fonts can't be part of Atlas, and using dynamic fonts will increase DrawCall at least once.

6. **The controls provided by the NGUI**: Prefab Toolbar: This tool provides a lot of commonly used controls, which can be directly used.

   Components: UI components are basically script components starting with UI, such as UISprite, UILable, UIWidget, etc.

   Control: I understand so that an empty GameObject+, one or more components, can form a control, for example:

   Button:Transform+UISprite (Script) +UIButton (Script) +Box Collider+UIPlay Sound.

   Sprite:Transform+UISprite (Script) / / So Sprite is a control, UISprite is a component.

7. **Panel controls, and UIPanel components**: The UIPanel is responsible for creating the actual collection graphics. You don't need to manually add UIPanel-. Once you create a control, it will automatically be added. If you want to split your UI rendering into different DrawCall, you can create your own UIPanel manually, for example, you want to create a split screen game, each screen is rendered with a camera, and you need 2 UIPanel to avoid overlapping controls.

8. **Widget（Container）**:Widget Tool (Widget Wizard): control creation wizard, you can add various controls to UI game objects, such as UI Root.

9. **NGUI event mechanism:** NGUI event triggers must be added to Box Collider, and Is Trigger is selected, Box size is set in the Inspector window, and auto-adjust to match can be selected at Widget Collider. One more important parameter needs to be set up correctly, that is, the Camera parameter under UI Root, in the Inspector window, to determine that Event Type in UICamera selects 3D UI, and Event Mask selects Everything.

   Box Collider:Button itself has a OnClick event, but Sprite, Label, and so on (also tied to Widget) do not trigger an event, so you have to add Box Collider manually, Event registration method:

   Directly bind the script on MonoBehaviour or Microsoft Visual Studio to Notifiy. This method is not flexible enough and is not recommended.

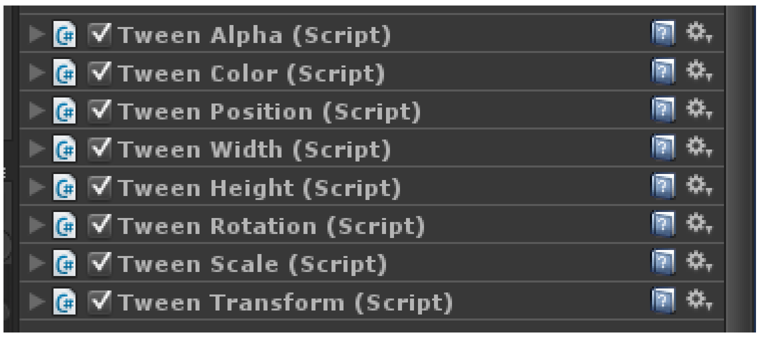
Append the Event Listener to the GameObject that needs to receive events. Add component->NGUI->Internal->Event Listener.

In any script or class, you can get the click event of the button and put the following code in any class or script.

A sample demo how to use NUGI plguins:

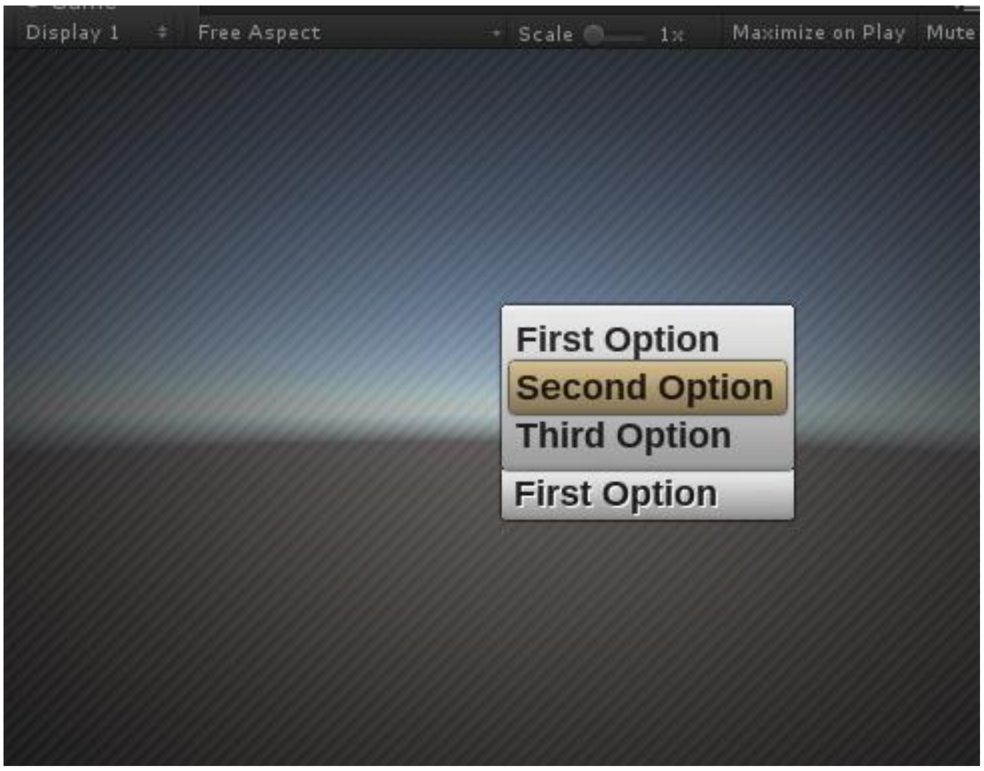
```
void Awake ()
{
    // Get the button object that you need to monitor
    GameObject button = GameObject.Find("UI Root (2D)/Camera/Anchor/Panel/LoadUI/MainCommon/Button");
    // Set the monitor of this button to point to the ButtonClick method of this class.。
    UIEventListener.Get(button).onClick = ButtonClick;
}
// Calculate the click event of a button
void ButtonClick(GameObject button)
{
     Debug.Log("GameObject " + button.name);
}
```

10. Tween: The animation system of NGUI adds animations to UI game objects, Including: Alpha animation, Position animation, Rotation animation, Scale animation, Transform animation.



Tween Alpha (Script)
Tween Color (Script)
Tween Position (Script)
Tween Width (Script)
Tween Height (Script)
Tween Rotation (Script)
Tween Scale (Script)
Tween Transform (Script)

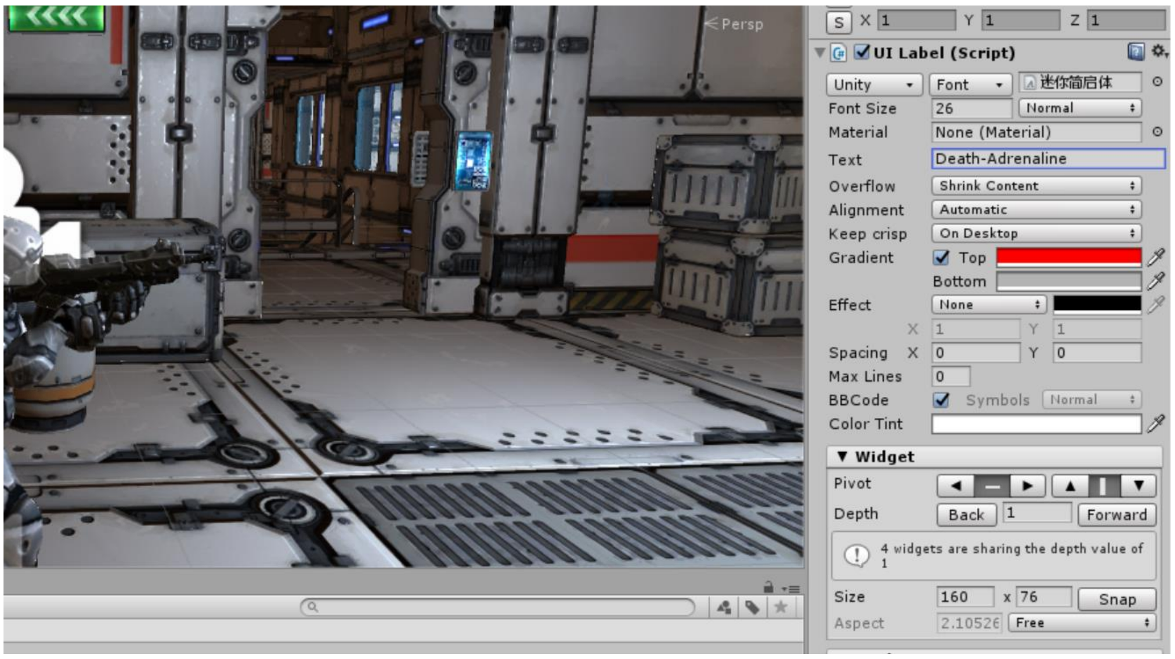
PopupList:



11. **Common UI script components**: **UICamera**: send NGUI events to all objects that are attached to the camera. In addition, it does not do anything to UI. In fact, if you want objects in your game to receive NGUI like OnPress, OnClick, and other events, what you need to do is to add a UICamera script to your MainCamera, that is, to hang to the Camera object under UIRoot.

    UISprite: it's the life of NGUI. It draws sprite from the texture atlas. All Sprite requires you to create an atlas (Atlas) first. It inherits all the functions of the UIWidget.

    UILable: script that displays text, inherited from UIWidget. All label needs Font to work properly. This font can make Dynamic (reference Unity Font) or Bitmap font -- the font embedded into Atlas. The dynamic font is more stable because it doesn't require you to create a symbol in advance, but the Bitmap font can be rendered on the same draw call with the rest of your atlas and these fonts can be beautified by Photoshop.



S X 1 Y 1 Z 1

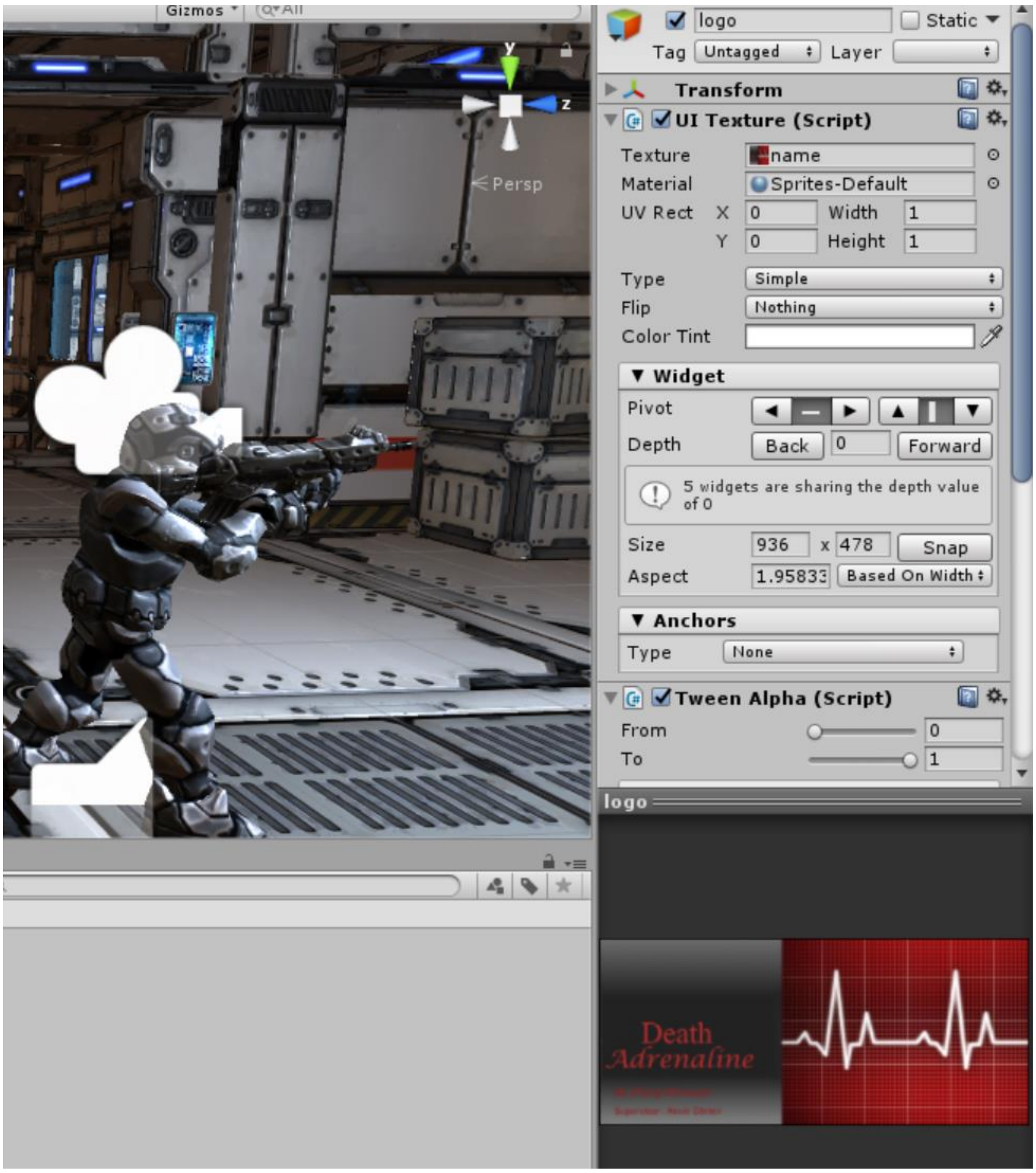
UI Label (Script)

Unity | Font | 迷你简启体
Font Size 26 Normal
Material None (Material)
Text Death-Adrenaline
Overflow Shrink Content
Alignment Automatic
Keep crisp On Desktop
Gradient ✓ Top 🟥
Bottom
Effect None ⬛
X 1 Y 1
Spacing X 0 Y 0
Max Lines 0
BBCode ✓ Symbols Normal
Color Tint

▼ Widget
Pivot
Depth Back 1 Forward
4 widgets are sharing the depth value of 1
Size 160 x 76 Snap
Aspect 2.10526 Free

UIWidget: simply speaking, it's a rectangle box that you can place anywhere on the screen. Widget has a certain area (such as the range of the white frame below), but it is completely invisible when running (GameView), so it is very suitable to be a container for other components (all sprite or label and so on for a variety of alignments). UIWidget is also used as the base class of all NGUI elements -- all the sprites and labels you create. UILabel, UISprite, UITexture and UI2DSprite (Unity3D current version) all inherited from UIWidget.

UITexture: it doesn't have the concept of atlas. It's very flexible to use. It only needs to hang up the picture. In this way, memory will only occupy the size of your map. Though memory is small, DrawCall will go up. Because each UITexture is a DrawCall.



logo Static
Tag Untagged Layer
Transform
UI Texture (Script)
Texture name
Material Sprites-Default
UV Rect X 0 Width 1
Y 0 Height 1
Type Simple
Flip Nothing
Color Tint
Widget
Pivot
Depth Back 0 Forward
5 widgets are sharing the depth value of 0
Size 936 x 478 Snap
Aspect 1.95833 Based On Width
Anchors
Type None
Tween Alpha (Script)
From 0
To 1
logo

UIPanel: used to collect and manage all the components of Widget below. Create a real DrawCall through the Geometry of Widget. Nothing can be rendered without Panel. If you are familiar with Unity, you can regard UIPanel as Renderer.

UIScroll View: it is used to draw a rolling view area in the interface, and we can control the content of the displayed area through the scroll bar.

The principle of UIGrid and UITable:UIGrid and UITable is simple, sorting the List of the sub Transform, and then arranging the location of the different rules (UIGrid and UITable are quite different).

UIGrid and UITable define 5 permutations (in fact, 3, None default unsort is the default sort of Transform, while Custom provides a virtual that can be overloaded):

```
public enum Sorting
{
    None, // Default sort is the default sort of Transform
    Alphabetic, // Sort by name string
    Horizontal, // Sort by localPosition
    Vertical,   // Sort by localPosition
    Custom,
      }
```

UIAnchor: you can fix GameObjects on the screen or on one side or other corner of Widgets. This is a key component to create modular UI in NGUI.

4.3 Unity3d RPC component

Remote procedure call (RPC) lets you call a function of a remote computer. It's as easy as calling a common function, but it needs understanding Some important differences.
RPC calls can have many parameters. All the parameters will be sent through the network. These parameters will increase the occupancy of your network. because In this case, you should try to reduce your parameters. You need to make sure who will receive your RPC. There are several RPC invocation patterns, which cover all the common types. You can call it easily The RPC function is in every machine, or only on the server, or everyone except you, or a specific client / player.

RPC calls are usually used to execute some events, according to all game clients or event information on both sides of a specific range. But, if you wish It means you can use it creatively. For example, after a game starts, only 4 clients connect to start the game. A client can Send PRC to each marked group member. A server can send RPC to a specific client and make it initialized after connection

For example, give him user number, local generator, team color and so on. A client can send to the server in turn, specifying the beginning of its selection.

Using RPC.

Calling RPC requires two steps: definition and call. You need a prefix when declaring a function. As follows:
All the function declaration needs / / [RPC]

```
[RPC]

Function PrintText (text: String)

{

Debug.Log (text);

}
```

nce you have declared RPC, you can call him. All networks can view scripts attached to the same object through the network view. Let me put it another way,
You need to attach the script that contains the RPC function to the game object, and you must include the network view component. The type of parameter you can use:

Int; float; string; NetworkPlayer; NetworkViewID; Vector3; Quatemion;

The RPC function is called as follows:

NetworkView.RPC ("PrintText", "RPCMode.All", "Hello world").

This code will use the attached NetworkView to invoke all PrintText () functions, as long as the script is attached to the same game object.
The first parameter is the name of the function. The second parameter is who calls the function. Here we call all PRC connected to the server. But no tell it to buffer the call after the client connection.
All the following parameters will be passed to the RPC function through the network. In this case, "Hello World" will be passed as a parameter.
You can also add extra internal functions, such as a NetworkMessageInfo structure, including the details of who sent RPC. As mentioned, to make RPC function work in script of a game object, there must be a Network View. The status synchronization of View can be set to shut down unless you use the state synchronization under the same Network View.

RPC Buffer:

RPC calls can also be buffered, and buffered RPC calls are stored and executed for every new connection client. They make a player join an existing one game is easier. A common case is that each player first loads a special level. So you're going to send a RPC function give each person and set it as a buffer. By buffering, whenever a new player joins the game, he will automatically send it to the player. This way, you don't need to track the player's loading and sending dynamically.

There are also some functions to eliminate the RPC buffer. In the case of manned level, if the game processes multiple levels, what you really care about is A new player joins the current card. In this case, you will delete any previously added RPC calls.

In this project a RPC demo:

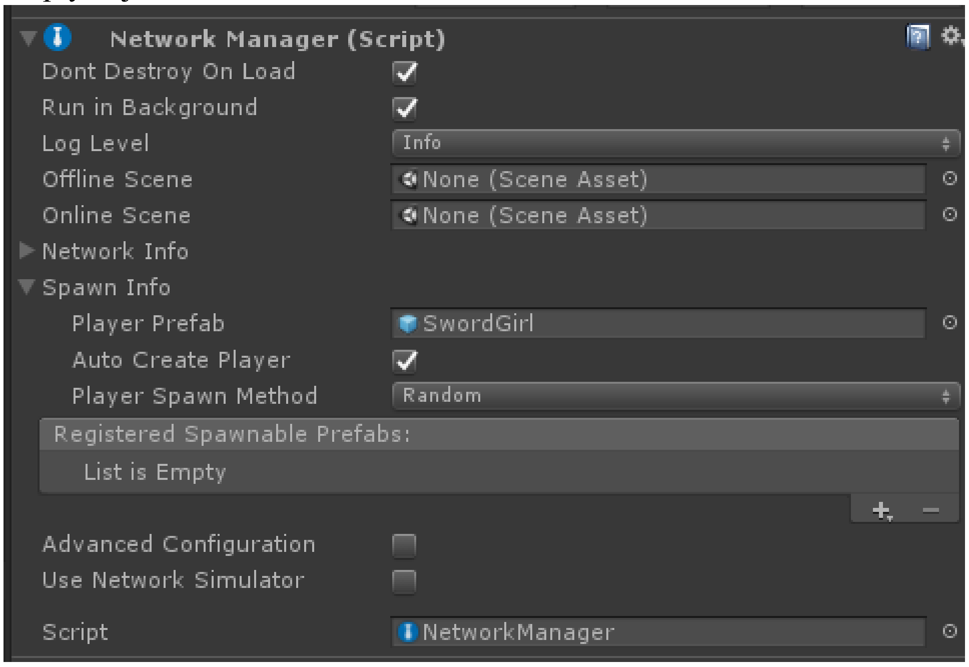
```
[RPC]//declare this is RPC method to unity
    public void SetOwnerPlayer(NetworkPlayer player) {
        this.ownerPlayer = player;
        if (Network.player!= player) {
            LoseControl();//cant ccontrol if this character isnt created by you
        }
    }
    public void TakeDamage(float damage) {
        hp -= damage;
       GetComponent<NetworkView>().RPC("SysncHp", RPCMode.All, hp);
    }
    [RPC]
    public void SysncHP() {
        //同步血量
        this.hp = hp;
    }
}
[RPC]
    void PlayState(string animName) {
        GetComponent<Animation>().CrossFade(animName, 0.2f);

    }
}
```

.

4.4Unity3d Network component

1. network management objects:

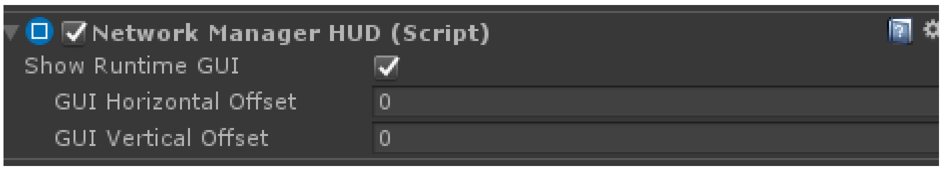
To build a network, we first need to create an empty object to add network management components. First we need to add the following two components to the empty object.



Network Manager (Script)

Dont Destroy On Load	✓
Run in Background	✓
Log Level	Info
Offline Scene	None (Scene Asset)
Online Scene	None (Scene Asset)
Network Info	
Spawn Info	
Player Prefab	SwordGirl
Auto Create Player	✓
Player Spawn Method	Random
Registered Spawnable Prefabs:	
List is Empty	
Advanced Configuration	☐
Use Network Simulator	☐
Script	NetworkManager

Precautions: 1. OffLine Scene represents the scene waiting for the client to connect to the server (the game hall). OnLine Scene represents the scenario after the client connects to the server.
2.Spawn Info server information: drag the network preset (must have a network component) into PlayerPrefab, and the server will twinning the game object into all connected clients



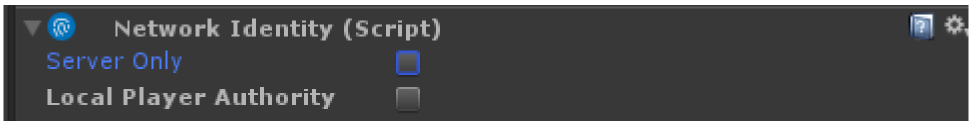
Network Manager HUD (Script)

Show Runtime GUI	✓
GUI Horizontal Offset	0
GUI Vertical Offset	0

2. The object of the game:

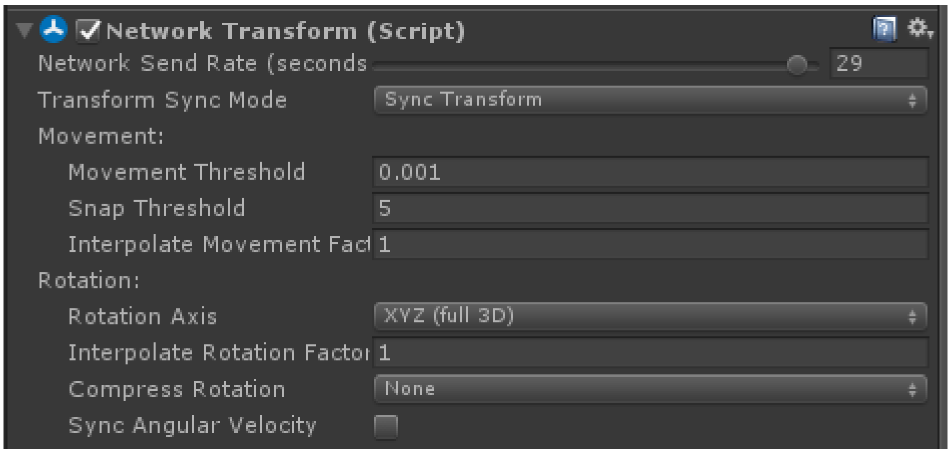
2.1.Network Identity**:** The game object (network preset) needs to hang the Network Identity component, which is the core of the network, and the object of the server Spwan must be available, and the component automatically assigns assetID and permissions at the time of it**.**

1. After the ServerOnly check, the object exists only in the server
2. After Local Player Authority is checked, it exists in the client.


Network Identity (Script)
Server Only ☐
Local Player Authority ☐

2.2 Realizing state synchronization: The control scripts of game objects need to inherit NetWorkBehaviour components (depending on NetWorkIdentity) to implement RPC technology and state synchronization attributes.

2.2.1 Transform synchronization: This module is responsible for synchronizing the player's movement to all clients after sending the mobile instructions to the client.


Network Transform (Script)
Network Send Rate (seconds 29
Transform Sync Mode Sync Transform
Movement:
Movement Threshold 0.001
Snap Threshold 5
Interpolate Movement Fact 1
Rotation:
Rotation Axis XYZ (full 3D)
Interpolate Rotation Factor 1
Compress Rotation None
Sync Angular Velocity ☐

Control scripts to write code:

```
public class ControlMove : NetworkBehaviour {

    void Update() {
        if (!isLocalPlayer)   // check whether it is a local client
        {
            return;
        }
        float x = Input.GetAxis("Horizontal");
        float y = Input.GetAxis("Vertical");
        if (x != 0 || y != 0)
        {
            transform.position += new Vector3(x, 0, y);
        }
```

In this project I learned how to use it in my game ,so below code is my test verson, Use Cube instead of game character:

```
public class Mycube : MonoBehaviour {
    public float speed = 5;
    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        float h = Input.GetAxis("Horizontal");
        float v = Input.GetAxis("Vertical");
        transform.Translate(new Vector3(h * speed * Time.deltaTime, 0, v * speed *
Time.deltaTime));
    }
}
```

How to use network components to achieve connection between client and server in this project(C#):

```
public class MyNetwok : MonoBehaviour {
    public int connections = 10;
    public int listenport = 8899;
    public bool useNat = false;
    public string ip = "127.0.0.1";
    public GameObject playerPrefab;


    void OnGUI()
    {
       if(Network.peerType==NetworkPeerType.Disconnected)// When the current state
is not connected, the following buttons will be displayed. Network.peerType is an
end type state: Connecting, server / disconnected, or client。


        if (GUILayout.Button("Create Server"))
        {
            // Create sevice
            NetworkConnectionError error= Network.InitializeServer(connections,
listenport, useNat);
            //print(error);
        }
        if (GUILayout.Button("Connecte Server"))
        {
            NetworkConnectionError error = Network.Connect(ip, listenport);
            print(error);

          }
else if (Network.peerType == NetworkPeerType.Server)
        {//当前状态为连接服务器那么显示以下内容, The current state displays the
following contents for connecting servers.。
            GUILayout.Label("Server has been created");
        }
        else if (Network.peerType == NetworkPeerType.Client) {
            GUILayout.Label("client connected");
        }
    }
```

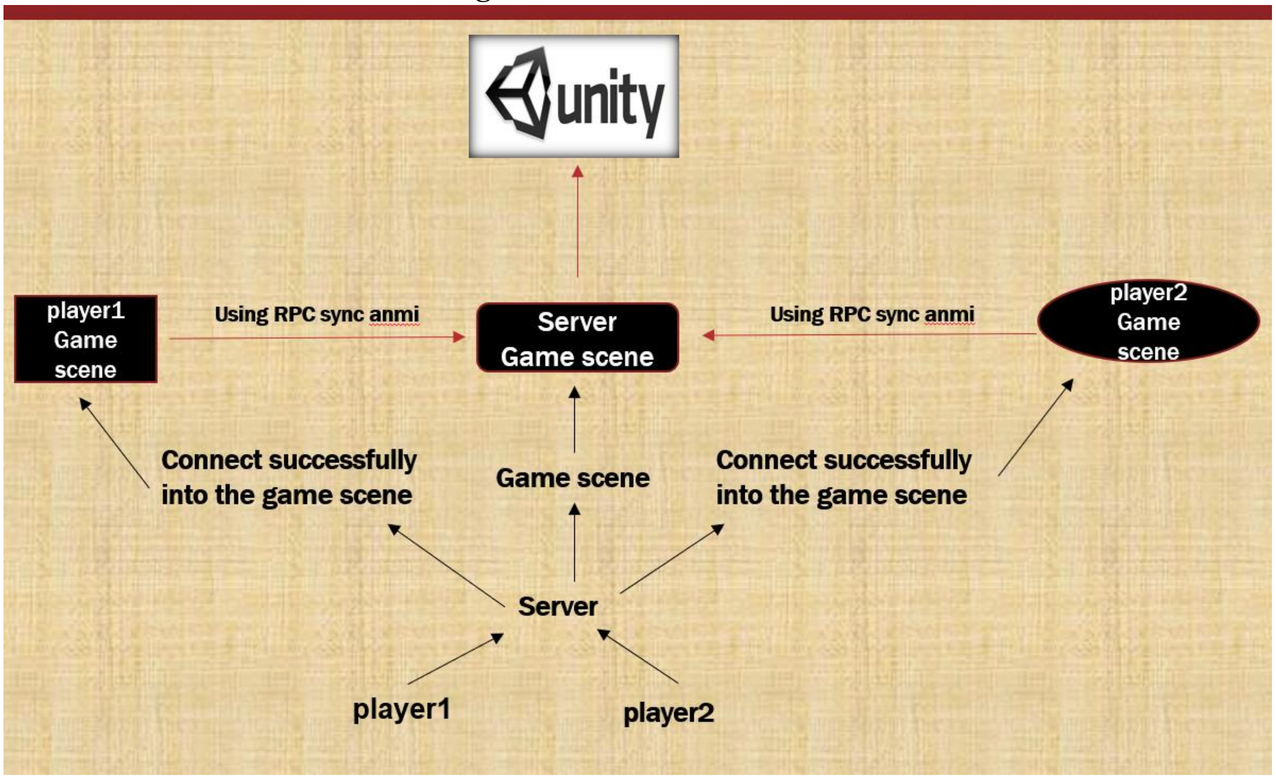
```
     void OnServerInitialized()//display on server
    {
        print("Completed initialize");
        //Network.player;//访问到当前的客户端
        int group = int.Parse(Network.player + "");//直接访问network.player会得到客户
端的索引, Direct access to network.player will get the index of the client.
        Network.Instantiate(playerPrefab, new Vector3(0, 10, 0),
Quaternion.identity,group);
    }
   void OnPlayerConnected(NetworkPlayer player)//显示在SErVER,display on server
    {
        print("A New player has been connected,Index number:" + player);
    }
    void OnConnectedToServer()//显示在CLIENT,display on client
    {
        print("Successful connected");
        int group = int.Parse(Network.player + "");//直接访问network.player会得到客户
端的索引, Direct access to network.player will get the index of the client.
        Network.Instantiate(playerPrefab, new Vector3(0, 10, 0),
Quaternion.identity, group);
    }
  }
```

Chapter 5

System Design

In this part, I will introduce the design of this project, how to implement every game function, and also show the code UML diagram.

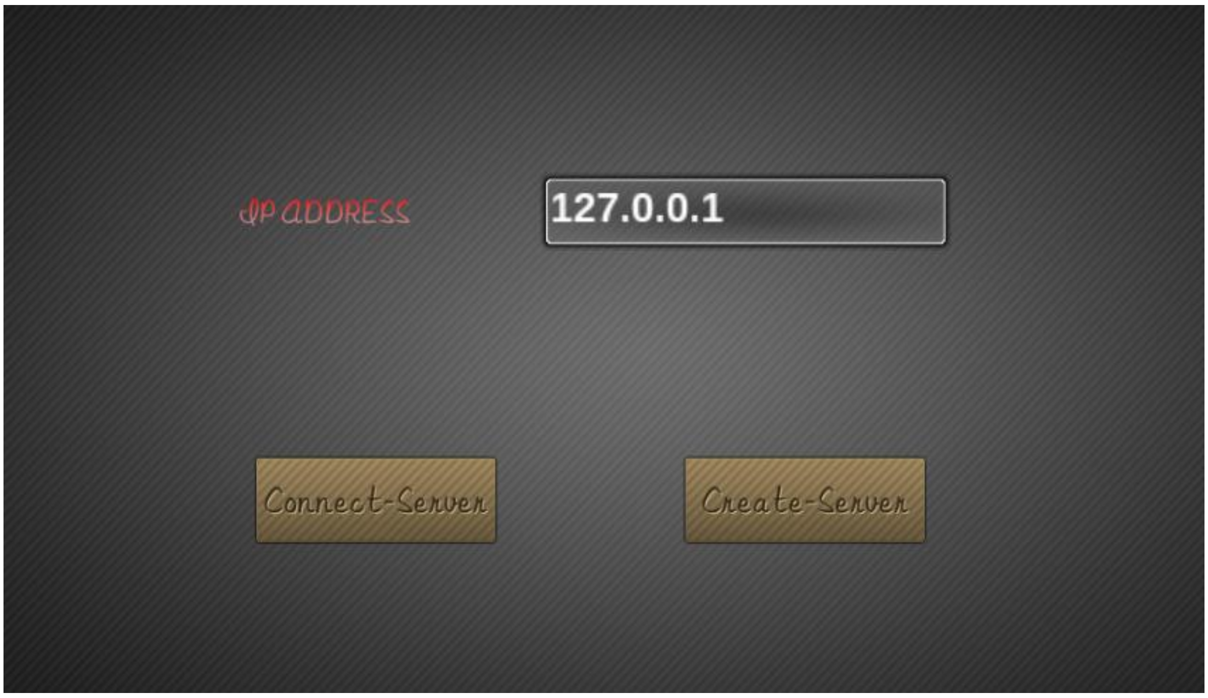
Game design architecture:



5.1 create a server and connect server

Because this game is a local area network game. All games start with a player must to create a server, and then other players can connect to the server, so the game is designed to get the game server function and the client connection function first..

A game start interface with two selections button, Create server and connect server respectively:

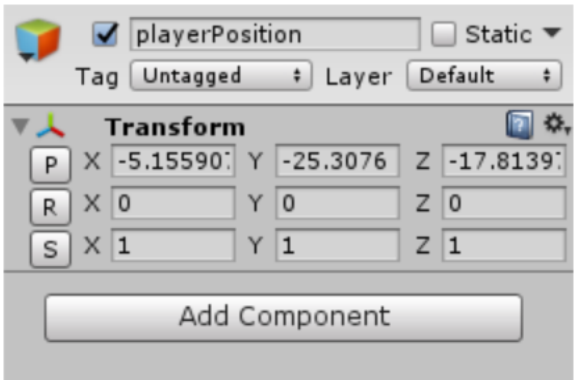


```
public int connections = 10;// maximum connection
      public int listenPort = 8899;
//当服务器按钮按下的时候运行, Run it when the server button is pressed
    public void OnButtonCreateServerClick() {
        Network.InitializeServer(connections, listenPort);//创建服务器,create server
        uiRoot.SetActive(false);
        PlayMusic();
      }
public void OnButtonConnectServerClick()
    {
        Network.Connect(ipInput.value, listenPort);//连接服务器
        uiRoot.SetActive(false);
        PlayMusic();

    }
```

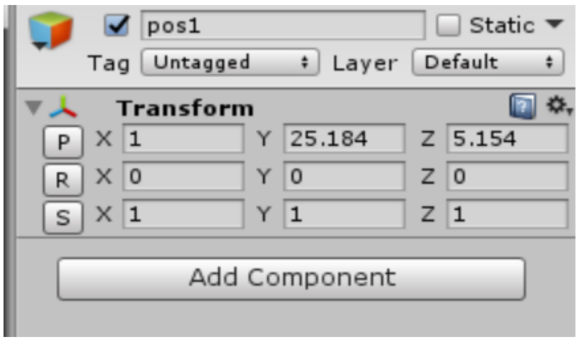
5.2 Get the character in the game map

Once the game server has been created or connect the server is successful, then The main character of the game is to be created in the game map, where the game server's role is located in a different place in the game map from the position of the client to generate the game role.

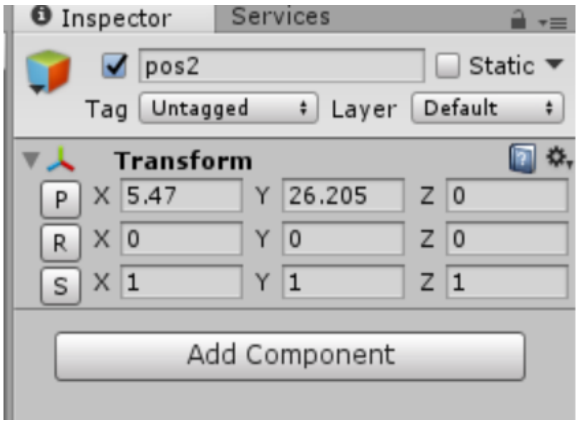
For example: The role generated by the game server is on the east side of the game map, so the role that the client generates is in the west of the game map. Which means different players have different coordinate, so we have the player position, like :



```
playerPosition        Static
Tag Untagged   Layer Default
Transform
P  X -5.15590   Y -25.3076   Z -17.8139
R  X 0          Y 0          Z 0
S  X 1          Y 1          Z 1
Add Component
```



```
pos1        Static
Tag Untagged   Layer Default
Transform
P  X 1     Y 25.184   Z 5.154
R  X 0     Y 0        Z 0
S  X 1     Y 1        Z 1
Add Component
```



```
Inspector   Services
pos2        Static
Tag Untagged   Layer Default
Transform
P  X 5.47  Y 26.205   Z 0
R  X 0     Y 0        Z 0
S  X 1     Y 1        Z 1
Add Component
```

The transform component is decide the birth position of the role, X,Y,Z which's the different location on the game map, C# code below:

```
void OnServerInitialized()//get character on server
    {
        //GameObject.Instantiate(soldierPrefab, pos1.position, Quaternion.identity);
        NetworkPlayer player= Network.player;//创建角色,create character
        int group = int.Parse(player + "");
        GameObject go = Network.Instantiate(soldierPrefab, pos1.position,
Quaternion.identity, group) as GameObject; //get the character coordinate
in the game map
```

```
void OnConnectedToServer() { //get a character on client


        NetworkPlayer player = Network.player;//创建角色,create a character
        int group = int.Parse(player + "");
        GameObject go = Network.Instantiate(soldierPrefab, pos2.position,
Quaternion.identity, group) as GameObject;// //get the character coordinate
in the game map
```

5.3 Control of the movement and vision of the character

When the role is generated on a map, the next step is to control a series of actions, such as walking, running, jumping, firing, moving perspectives, and so on. Because this is the first person shooter game, so the game perspective is displayed in the first person. Using javascript to control character move:

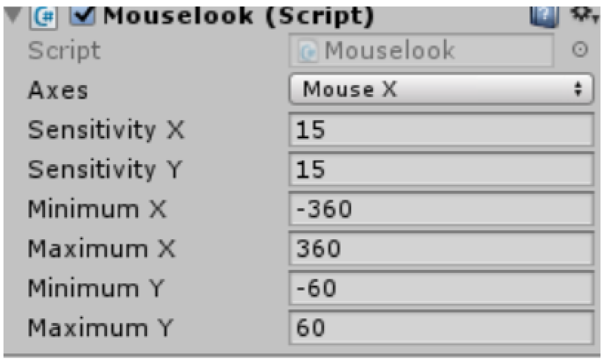
```
var canControl : boolean = true;

var useFixedUpdate : boolean = true;

// For the next variables, @System.NonSerialized tells Unity to not serialize the variable
or show it in the inspector view.
// Very handy for organization!

// The current global direction we want the character to move in.
@System.NonSerialized
var inputMoveDirection : Vector3 = Vector3.zero;

// Is the jump button held down? We use this interface instead of checking
// for the jump button directly so this script can also be used by AIs.
@System.NonSerialized
var inputJump : boolean = false;

class CharacterMotorMovement {
    // The maximum horizontal speed when moving
    var maxForwardSpeed : float = 10.0;
    var maxSidewaysSpeed : float = 10.0;
var maxBackwardsSpeed : float = 10.0;

// Curve for multiplying speed based on slope (negative = downwards)
    var slopeSpeedMultiplier : AnimationCurve = AnimationCurve(Keyframe(-90, 1),
Keyframe(0, 1), Keyframe(90, 0));

    // How fast does the character change speeds?  Higher is faster.
    var maxGroundAcceleration : float = 30.0;
    var maxAirAcceleration : float = 20.0;

    // The gravity for the character
    var gravity : float = 10.0;
    var maxFallSpeed : float = 20.0;
    // For the next variables, @System.NonSerialized tells Unity to not serialize the
variable or show it in the inspector view.
    // Very handy for organization!

    // The last collision flags returned from controller.Move
    @System.NonSerialized
    var collisionFlags : CollisionFlags;

    // We will keep track of the character's current velocity,
    @System.NonSerialized
    var velocity : Vector3;
var movement : CharacterMotorMovement = CharacterMotorMovement();

class CharacterMotorJumping {
    // Can the character jump?
    var enabled : boolean = true;

    // How high do we jump when pressing jump and letting go immediately
    var baseHeight : float = 1.0;

    // We add extraHeight units (meters) on top when holding the button down longer while
jumping
    var extraHeight : float = 4.1;

    // How much does the character jump out perpendicular to the surface on walkable
surfaces?
    // 0 means a fully vertical jump and 1 means fully perpendicular.
    var perpAmount : float = 0.0;

    // How much does the character jump out perpendicular to the surface on too steep
surfaces?
    // 0 means a fully vertical jump and 1 means fully perpendicular.
    var steepPerpAmount : float = 0.5;
    var jumping : CharacterMotorJumping = CharacterMotorJumping();

class CharacterMotorMovingPlatform {

var movingPlatform : CharacterMotorMovingPlatform = CharacterMotorMovingPlatform();

class CharacterMotorSliding {
private function ApplyInputVelocityChange (velocity : Vector3) {
    if (!canControl)
inputMoveDirection = Vector3.zero;

      And so on …………
```

Unity3d game engine for the convenience of programmers to develop games, so the above Javascripts code is done by the official unity3d, so the role of the mobile function is the official JavaScript code, but I also learned how to use JavaScript to control the movement of an object.

Character's horizons:

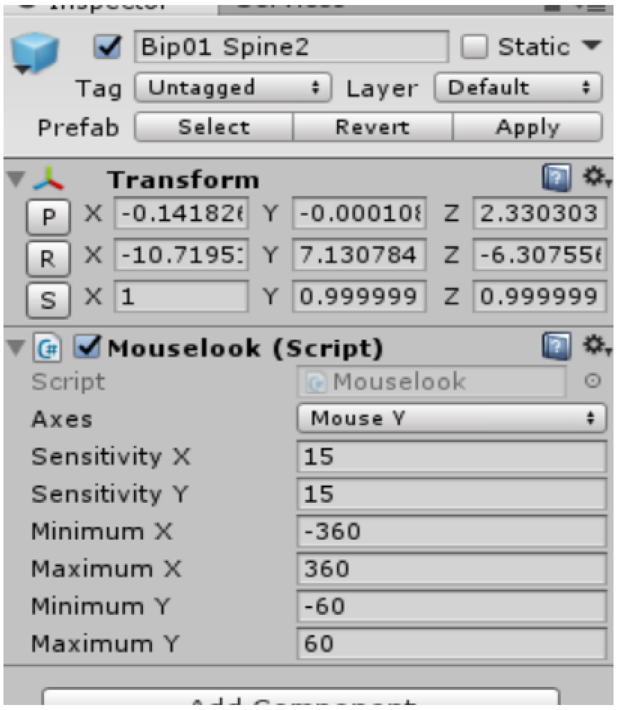
There are several questions to be paid attention to how to control the role's perspective: the camera needs to move with the character , and the character can only rotate around the field, the character can not be moved, the X, Y coordinates of the role's vision should be controlled.so in this project, we need set two script to control character view, need control X and Y That is control character’s Left and right horizons , Upper and lower horizons

FOR control X direction:


Mouselook (Script)

Script	Mouselook
Axes	Mouse X
Sensitivity X	15
Sensitivity Y	15
Minimum X	-360
Maximum X	360
Minimum Y	-60
Maximum Y	60

A mouseLook script is added to the role's spine to control the direction of Y:


Bip01 Spine2 Static
Tag Untagged Layer Default
Prefab Select Revert Apply

Transform

P	X	-0.14182(	Y	-0.00010(	Z	2.330303
R	X	-10.7195:	Y	7.130784	Z	-6.30755(
S	X	1	Y	0.999999	Z	0.999999

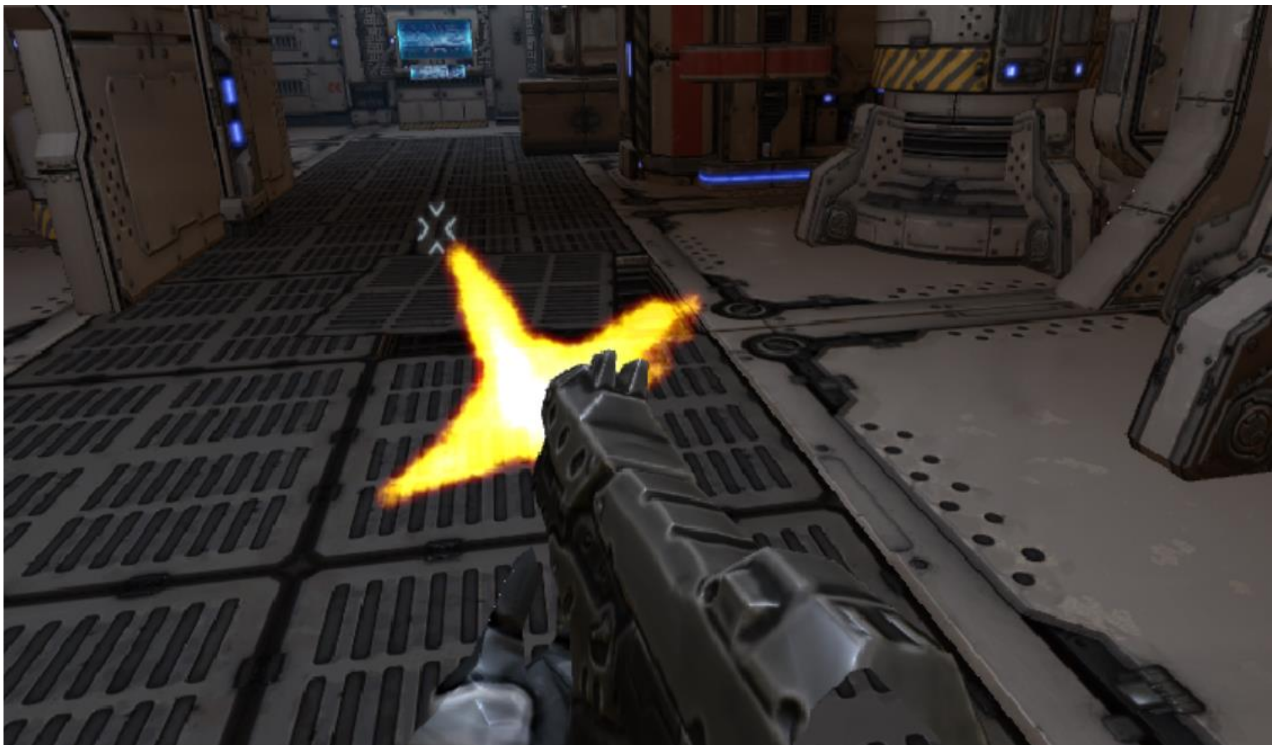
Mouselook (Script)

Script	Mouselook
Axes	Mouse Y
Sensitivity X	15
Sensitivity Y	15
Minimum X	-360
Maximum X	360
Minimum Y	-60
Maximum Y	60

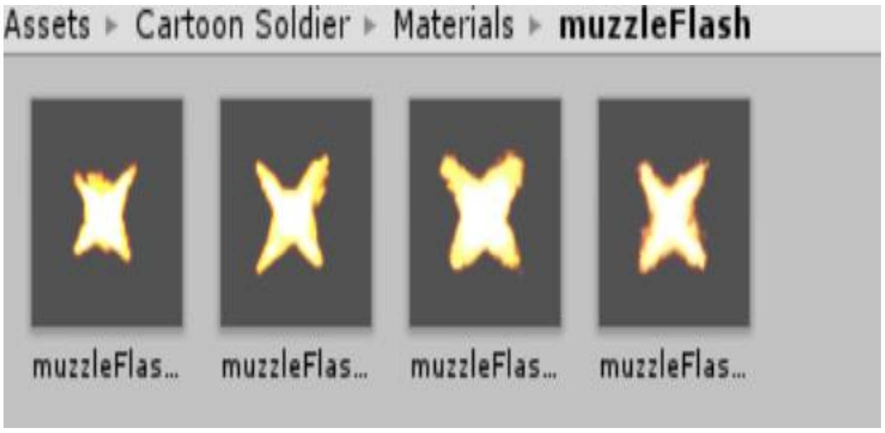
```
public enum RotationAxes { MouseXAndY = 0, MouseX = 1, MouseY = 2 }
    public RotationAxes axes = RotationAxes.MouseXAndY;
    public float sensitivityX = 15F;
    public float sensitivityY = 15F;
    public float minimumX = -360F;
    public float maximumX = 360F;
    public float minimumY = -60F;
    public float maximumY = 60F;
    //float rotationX = 0F;
    float rotationY = 0F;
    //float rotationZ = 0F;
    private Vector3 eulerAngles;
    //Quaternion originalRotation;

    void LateUpdate()
    {
        if (axes == RotationAxes.MouseXAndY)
        {
            float rotationX = transform.localEulerAngles.y + Input.GetAxis("Mouse X") *
sensitivityX;


            rotationY += Input.GetAxis("Mouse Y") * sensitivityY;
            rotationY = Mathf.Clamp(rotationX, minimumX, maximumX);
            transform.localEulerAngles = new Vector3(-rotationY, rotationX,
0);


        }else if (axes == RotationAxes.MouseX)
        {
            transform.Rotate(0, Input.GetAxis("Mouse X") * sensitivityX, 0);

        }
        else
        {
            rotationY += Input.GetAxis("Mouse Y") * sensitivityY;
            rotationY = Mathf.Clamp(rotationY, minimumY, maximumY);

            transform.localEulerAngles = new Vector3(eulerAngles.x, eulerAngles.y,
eulerAngles.z+rotationY);


        }
    }
    void Start()
    {
        // Make the rigid body not change rotation
        if (GetComponent<Rigidbody>())
            GetComponent<Rigidbody>().freezeRotation = true;
        eulerAngles = transform.localEulerAngles;
    }

}
```

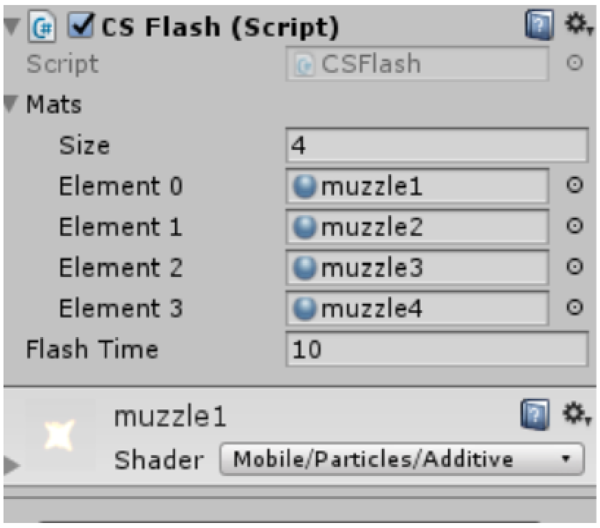
5.4 The effect of weapon firing

When the player uses the left mouse button to fire each time then a spark effect will be generated,



To achieve this effect, we need to show these four pictures to the muzzle:




CS Flash (Script)
Script CSFlash
Mats
Size 4
Element 0 muzzle1
Element 1 muzzle2
Element 2 muzzle3
Element 3 muzzle4
Flash Time 10
muzzle1
Shader Mobile/Particles/Additive

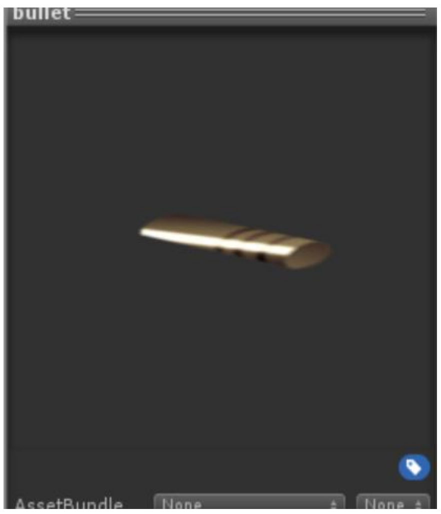
First we need get a component that can put Multiple materials to muzzle, and then Use the timer to display the special effects of the 4 pictures in turn, and finally click the event with the mouse to realize this function:

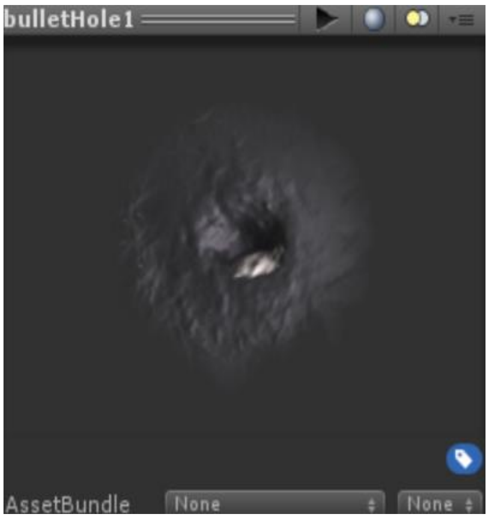
```
public Material[] mats;

public float flashTime = 0.1f;//闪光时间,set flash time as 0.1s
    private float flashTimer = 0; // need timer
    private int index =0;
    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        if (Input.GetMouseButtonDown(0)) {// when player click left mousebutton then
display the spark effect
            Flash();
        }
            //FOR flash time
        if (GetComponent<Renderer>().enabled) {
            flashTimer += Time.deltaTime;
            if (flashTimer > flashTime) {// if timer > flash time,flash effect will
disappear
                GetComponent<Renderer>().enabled = false;
            }
        }
    }
    public void Flash(){
        index++;// get materials number
        index %= 4; // there are four pictures need be displayed
        GetComponent<Renderer>().enabled = true;
        GetComponent<Renderer>().material = mats[index];
        flashTimer = 0;
    }
}
```

5.5 The formation of bullets and bullet holes

First we need to get the bullet and bullet holes materials, and then to implement it

On game:





Then we need to animate the bullet, set the speed of the bullet, when the player opens fire, the bullet will be generated, when the bullet hits the object, the bullet is disappearing, and the bullet has a special effect. After a period of time, the bullet holes disappear. If we need to complete the above effects, we need to create a ray for the bullet to detect whether it hits the object.

```
public
float speed = 800;
    public GameObject[] bulletholes; //Create a multi - material game object for bullet
holes
    public float damage = 10; //set bullet damage
    void Start()
    {
        Destroy(this.gameObject, 5); //after 5s bullet disappear
    }
    // Update is called once per frame
    void Update () {
        Vector3 oriPos = transform.position; //get bullet original position
        transform.Translate(Vector3.forward * speed * Time.deltaTime); //the movement of
bullet
        Vector3 direction = transform.position - oriPos; Get the bullet position after
opened fire
        float length = (transform.position - oriPos).magnitude; // Calculating the distance
between bullets
        RaycastHit hitinfo;//存储碰撞信息, The first object information that stores ray
collisions
       bool isCollider= Physics.Raycast(oriPos, direction, out hitinfo,length);
        if (isCollider) {

            if (hitinfo.collider.tag == "Player")
            {
                //子弹射击到角色身上, The bullet shot on the character
                player player = hitinfo.collider.GetComponent<player>();
                if (player.hp > 0) {//only character's hp more than 0 then can shoot
                    player.TakeDamage(this.damage);
                }
            }
            else {
                //如果射线碰撞到物体要做的动作,If the ray hits an object to do
                int index = Random.Range(0, 2);
                GameObject bulletHolePrefab = bulletholes[index];
                Vector3 pos = hitinfo.point;//得到碰撞的位置,Get the position of the
collision
                GameObject go = GameObject.Instantiate(bulletHolePrefab, pos,
Quaternion.identity);
                go.transform.LookAt(hitinfo.point - hitinfo.normal);
                go.transform.Translate(Vector3.back * 0.01f);
                //hitinfo.normal;//可以得到碰撞的垂线向量, 面的法线

            }
            Destroy(this.gameObject);
        }
    }
      }
```

Then we need the formation and shooting rate of a bullet:

```
public int shootRate = 10;//每秒可以射击多少子弹,How many bullets can be fired per second
    public CSFlash flash;
    public GameObject bulletPrefab;
    private float timer = 0;
    private bool isFiring = false;//default is not fire
    // Use this for initialization


    // Update is called once per frame
    void Update () {
        if (Input.GetMouseButtonDown(0)) {//Press the left button of the mouse
            isFiring = true;
        }
        if (Input.GetMouseButtonUp(0)) {//Lift the left button of the mouse
            isFiring = false;
        }
        if (isFiring) {//is firing
            timer += Time.deltaTime;
            if (timer > 1f / shootRate) {
                Shoot();
                timer = 0;
                //timer -= 1f / shootRate;
            }
        }
    }
    void Shoot() {
        //flash.Flash();
        GameObject.Instantiate(bulletPrefab, transform.position, transform.rotation);
//generate bullets
    }
```

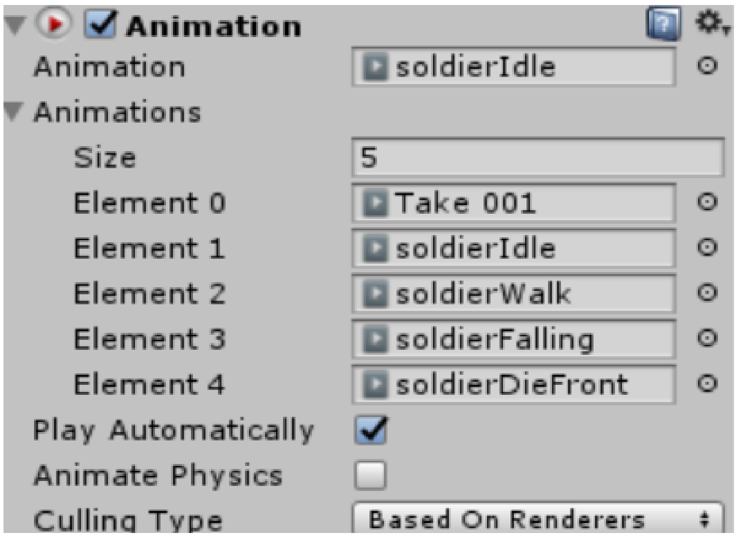
```
}
              For the bullet holes effect:

public float speed = 0.3f;
    private float timer = 0;
    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {//The bullets will fade away.
        timer += Time.deltaTime;
        if (timer > 2) {
            GetComponent<Renderer>().material.color =
Color.Lerp(GetComponent<Renderer>().material.color, Color.clear, speed * Time.deltaTime);
        }
        if (timer > 10) {// after 10s The bullet completely disappeared
            Destroy(this.gameObject);
        }
    }
}
```

Because all the bullets are hit by the bullet damage, the HP value of the character will decrease:

```
public float hp = 100;//主角的血量,the character HP

public void TakeDamage(float damage) {
        hp -= damage;
       GetComponent<NetworkView>().RPC("SysncHp", RPCMode.All, hp);
    }
    [RPC]
    public void SysncHP() {
        //同步血量,using RPC sysnc HP
        this.hp = hp;
    }
}

if (isCollider) {

            if (hitinfo.collider.tag == "Player")
            {
                //子弹射击到角色身上,attack the character

                player player = hitinfo.collider.GetComponent<player>();
                if (player.hp > 0) {//only character's hp more than 0 then can shoot
                    player.TakeDamage(this.damage);
                }
            }
```

5.6 Character animation effects

In this project, role animation is very important. Only after adding a series of animation effects can let the game become more vivid. For example, when the role open fired, there will be an action to raise character's hand. If the role is not controlled, it will display idel animation. So I set up a series of animation for this role:



Animation

Animation	soldierIdle
Animations	
Size	5
Element 0	Take 001
Element 1	soldierIdle
Element 2	soldierWalk
Element 3	soldierFalling
Element 4	soldierDieFront
Play Automatically	✓
Animate Physics	☐
Culling Type	Based On Renderers

But we still need to use code to achieve animation of each action step by step.:

```
private CharacterController characterController;
    private player player;
    private bool havePlayDieAnimation = false;// If the death animation has been played
    //private Animation anim;
    // Use this for initialization
    void Start() {
        characterController = this.GetComponent<CharacterController>();
        player=this.GetComponent<player>();
       // anim = GetComponent<Animation>();
    }

    // Update is called once per frame
    void Update()
    {
        if(player.hp>0){
            if (characterController.isGrounded == false)
            //播放下落的动画, Play the falling animation
            {
                //远程调用 玩家与玩家之间的动画同步, Remotely invoke animation synchronization
between players and players
                GetComponent<NetworkView>().RPC("PlayState", RPCMode.All,
"soldierFalling");
                PlayState("soldierFalling");
            }
            else
            {
                float h = Input.GetAxis("Horizontal");
                float v = Input.GetAxis("Vertical");
                if (Mathf.Abs(h) > 0.03f || Mathf.Abs(v) > 0.03f)
                {
                    //PlayState("soldierWalk");
                    GetComponent<NetworkView>().RPC("PlayState", RPCMode.All,
"soldierWalk");
                }
                else
                {
                    // PlayState("soldierIdle");
                    GetComponent<NetworkView>().RPC("PlayState", RPCMode.All,
"soldierIdle");
                }
            }
        }else{
            if (havePlayDieAnimation == false) {
                GetComponent<NetworkView>().RPC("PlayState", RPCMode.All,
"soldierDieFront");
                havePlayDieAnimation = true; //只播放一次,only play once
            }
        }
    }
        [RPC] //using RPC Synchronize all animations
    void PlayState(string animName) {// Use crossFade to make some smoothing between each
animation, making animation more natural.
        GetComponent<Animation>().CrossFade(animName, 0.2f);

    }
}
```

5.7 Players control the character of the game

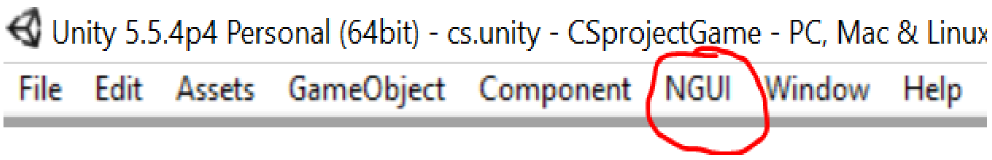
When my server and client have roles, then there will be 2 roles on the map, but the client can only control the role generated by the client, and the server can only control the role of the server, that is, the player's 1 can not control the player's 2 role. When the role is generated, we need to check who has been created it, and if it is not created by myself, we should disable a series of functions of the control role:

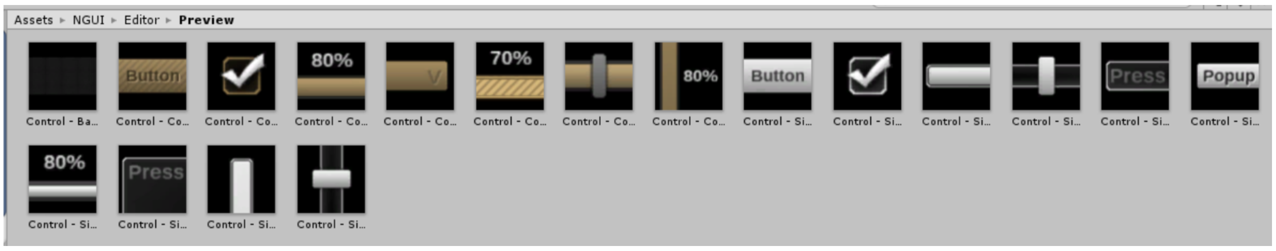
```
public void LoseControl() {
        motor = this.GetComponent<CharacterMotor>();//Get the movements of role script
        playerAnimation = this.GetComponent<PlayerAnimation>();
        camera.gameObject.SetActive(false);//only have one CAMERA;
        //cant move
       motor.cancontrol= false;
        playerAnimation.enabled = false;//cant run animaiton
        //can shoot
        bulletgenrator.enabled = false;
        //cant control mouse
        xMouselook.enabled= false;
        yMouselook.enabled = false;
        flash.enabled = false;


    }
    [RPC]//declare this is RPC method to unity
    public void SetOwnerPlayer(NetworkPlayer player) {
        this.ownerPlayer = player;
        if (Network.player!= player) {
            LoseControl();//cant ccontrol if this character isnt created by you
        }
       }
```

5.8 General game management

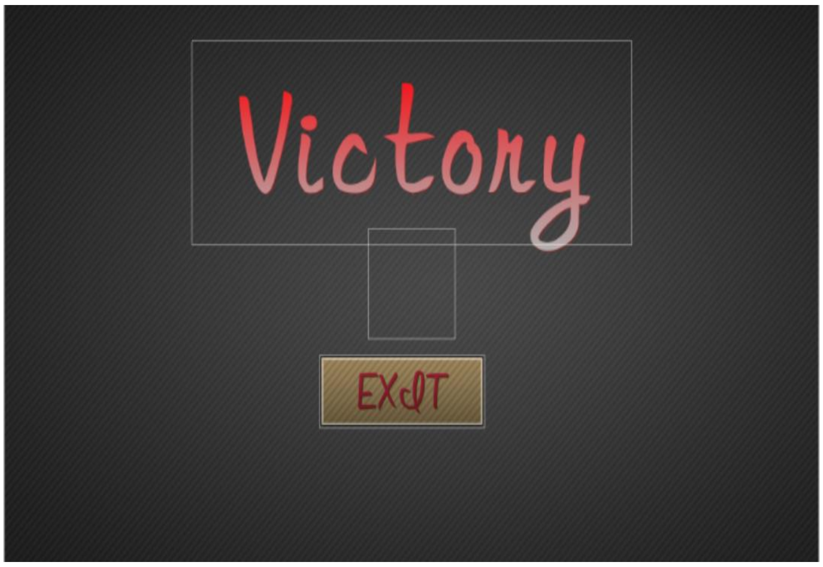
Game management is to set up a series of game UI management. For example, the sound effect of the game, the beginning and end of the game, the menu option of the game, the position control of the game role and so on, About all games UI, I use NGUI plug-in to do it:

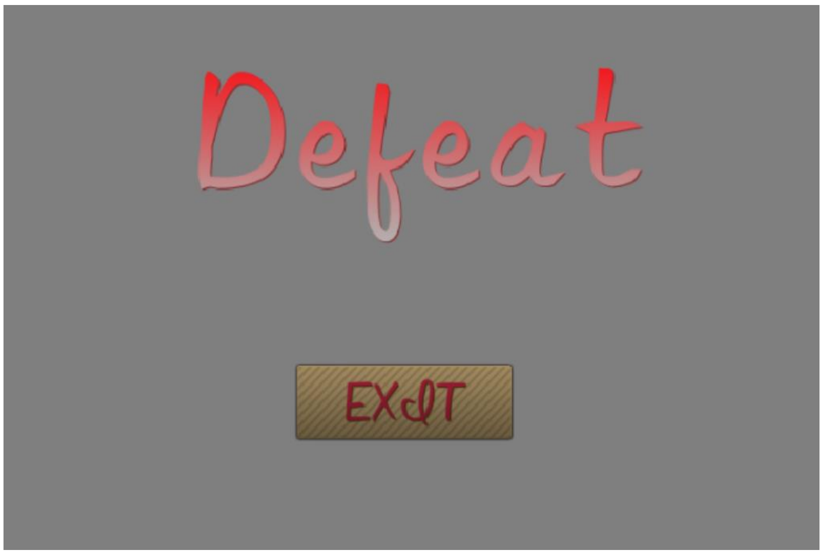
NGUI-plug:


Unity 5.5.4p4 Personal (64bit) - cs.unity - CSprojectGame - PC, Mac & Linux
File
Edit
Assets
GameObject
Component
NGUI
Window
Help

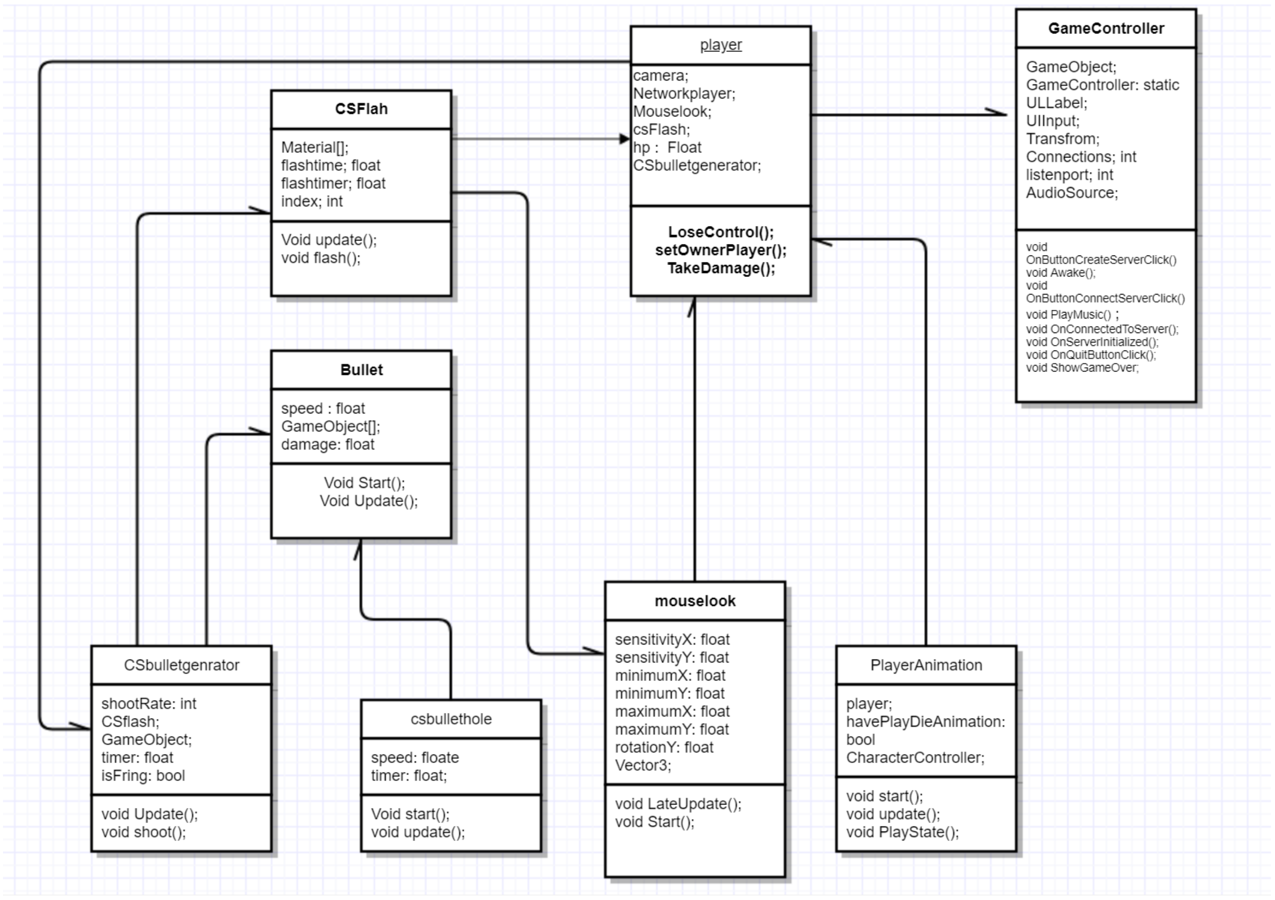


```
public GameObject bgGameObject;
    public GameObject logoGameObject;
    public GameObject menuGameObject;
    public GameObject gameoverpanelGameObject;
    public static GameController _instance;
    public UILabel gameoverlabel;
    public UIInput ipInput;
    public GameObject uiRoot;
    public Transform pos1;
    public Transform pos2;
    public GameObject soldierPrefab;
    public int connections = 10;//最大连接数
    public int listenPort = 8899;
    public AudioSource BGMenuMusic;
    public AudioSource BGPlayMusic;
    //当服务器按钮按下的时候运行
    public void OnButtonCreateServerClick() {//create server
        Network.InitializeServer(connections, listenPort);//创建服务器
        uiRoot.SetActive(false);
        PlayMusic();
    }
    private void Awake()
    {
        _instance = this;
    }
    //当按下链接服务器按钮的时候运行
    public void OnButtonConnectServerClick()//connect server
    {
        Network.Connect(ipInput.value, listenPort);//连接服务器
        uiRoot.SetActive(false);
        PlayMusic();

    }
    void PlayMusic() {//play BG
        BGMenuMusic.Stop();
        BGPlayMusic.Play();
    }
    void OnServerInitialized()
    {
        //GameObject.Instantiate(soldierPrefab, pos1.position, Quaternion.identity);
        NetworkPlayer player= Network.player;//创建角色,create player
        int group = int.Parse(player + "");
        GameObject go = Network.Instantiate(soldierPrefab, pos1.position,
Quaternion.identity, group) as GameObject; //get player position
        go.GetComponent<player>().SetOwnerPlayer(Network.player);// Only set the
OWNERPlayer attribute of the role in the current creator. The property of other clients is
empty。
        //RPC：
        go.GetComponent<NetworkView>().RPC("SetOwnerPlayer", RPCMode.AllBuffered,
Network.player);//完成一个远程调用，会执行所有连接客户端上的SETOWNERplayer方法；Completing a
remote call will execute all the SETOWNERplayer methods on the connection client.
        Cursor.visible = false;//游戏开始隐藏鼠标, The game begins to hide the mouse
    }
    void OnConnectedToServer() {
        NetworkPlayer player = Network.player;//创建角色,create player
        int group = int.Parse(player + "");
        GameObject go = Network.Instantiate(soldierPrefab, pos2.position,
Quaternion.identity, group) as GameObject;//得到角色位置,get player position
        //RPC：远程调用
        go.GetComponent<NetworkView>().RPC("SetOwnerPlayer", RPCMode.AllBuffered,
Network.player);//完成一个远程调用，会执行所有连接客户端上的SETOWNERplayer方法；
        Cursor.visible = false;//游戏开始隐藏鼠标
    }
   public void OnQuitButtonClick() {
        Application.Quit();//EXIT GAME
    }
    public void ShowGameOver(bool isWin) {//display gameover interface, need hide others
interface
        //Game over interface
        uiRoot.gameObject.SetActive(true);
        bgGameObject.SetActive(false);
        logoGameObject.SetActive(false);
        gameoverpanelGameObject.SetActive(true);
        menuGameObject.SetActive(false);
        if (isWin) //if player win then display
        {
            Cursor.visible = true;
            gameoverlabel.text = "Victory";
        }
        else {//if player lose then display
            Cursor.visible = true;
            gameoverlabel.text = "Defeat";
        }
        }
    }
```





5.9 UML


player
camera;
Networkplayer;
Mouselook;
csFlash;
hp : Float
CSbulletgenerator;
LoseControl();
setOwnerPlayer();
TakeDamage();
GameController
GameObject;
GameController: static
ULLabel;
UIInput;
Transfrom;
Connections; int
listenport; int
AudioSource;
void
OnButtonCreateServerClick()
void Awake();
void
OnButtonConnectServerClick()
void PlayMusic() ;
void OnConnectedToServer();
void OnServerInitialized();
void OnQuitButtonClick();
void ShowGameOver;
CSFlah
Material[];
flashtime; float
flashtimer; float
index; int
Void update();
void flash();
Bullet
speed : float
GameObject[];
damage: float
Void Start();
Void Update();
mouselook
sensitivityX: float
sensitivityY: float
minimumX: float
minimumY: float
maximumX: float
maximumY: float
rotationY: float
Vector3;
void LateUpdate();
void Start();
CSbulletgenrator
shootRate: int
CSflash;
GameObject;
timer: float
isFring: bool
void Update();
void shoot();
csbullethole
speed: floate
timer: float;
Void start();
void update();
PlayerAnimation
player;
havePlayDieAnimation:
bool
CharacterController;
void start();
void update();
void PlayState();

Chapter 6

System Evaluation

In this section, I am going to evaluate this game project based on different attributes. They are as follows:

- Scalability
- Robustness
- Maintainability
- Extensibility

Scalability: At present, 3D games are very popular in the game market, The game created by unity3D engine is very expansibility. Using RPC script can control the connection of LAN very well. The use of NETWORK component can not only create the connection of the LAN, but also create the Internet connection.

Robustness: This game has a certain robustness, all the code has no problems, all the game functions are no problem, through various aspects of the test, the system logic is correct, the game resources have been placed on the GITHUB, the resources on the unity3d engine can be detected and modified.

Maintainability: There are a lot of resources in this project, the resources of each plate are clear and the structure is very stable. When the program code is written, each CLASS is very clear, and the various functions are very convenient to modify on the unity3d, and all codes can be found on github link, so the maintenance will be very convenient.

Extensibility: The game has a very high scalability, the LAN connection has been fully implemented, the next direction should create a connection on the Internet, because local resources, functions have been completed, so only a database is needed to store the game player's data, and a network server can achieve online game battle.

6.1 Testing

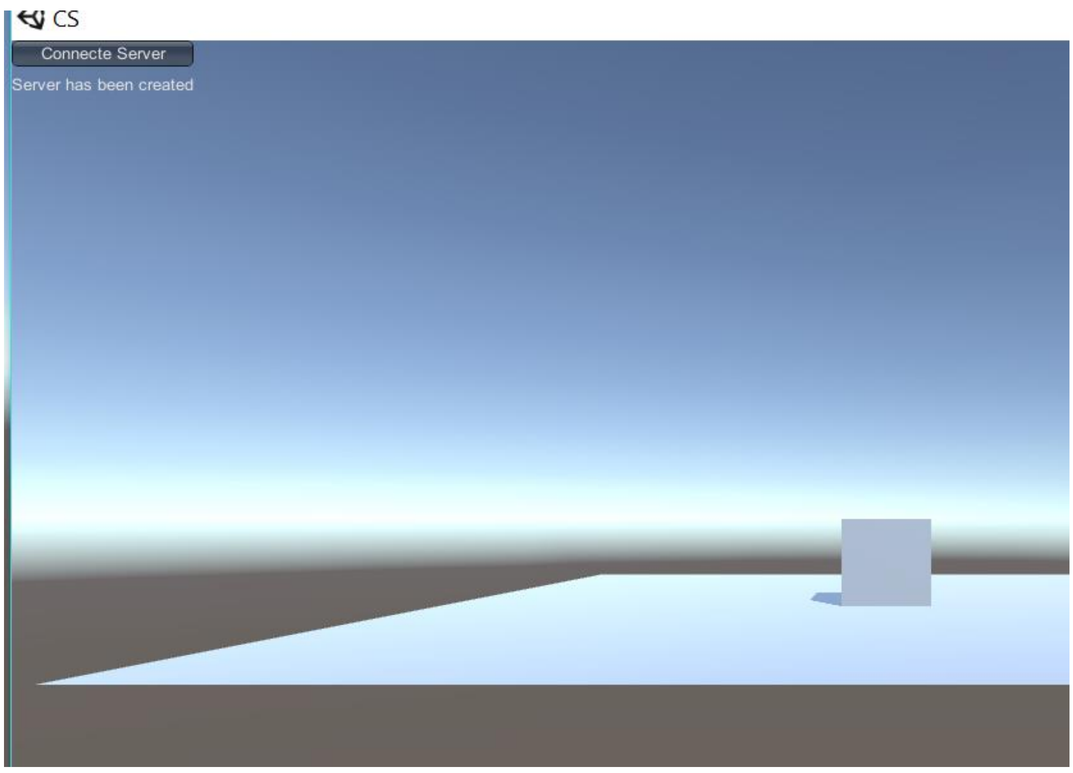
When I using the network component to create a server, there are some problems at the beginning, such as creating a successful server, the client cannot find the server or is unable to connect. Or the client can be able to connect successfully, but the server cannot receive the client's data, and I through many aspects of testing, it has now completely solved this kind of problem, and has ensured that this problem will not happen again, and the following is the test code that I use the NETWORK component to create the server and the connection server. It has already used in this game:

```
public class MyNetwok : MonoBehaviour {
    public int connections = 10;
    public int listenport = 8899;
    public bool useNat = false;
    public string ip = "127.0.0.1";
    public GameObject playerPrefab;


    void OnGUI()
    {
       if(Network.peerType==NetworkPeerType.Disconnected)//当当前状态为未连接
那么会显示以下的按键,If the current state is not connected, the following
buttons will be displayed.
        if (GUILayout.Button("Create Server"))
        {
            // Create sevice
            NetworkConnectionError error=
Network.InitializeServer(connections, listenport, useNat);
            //print(error);
        }
        if (GUILayout.Button("Connecte Server"))
        {
            NetworkConnectionError error = Network.Connect(ip, listenport);
            print(error);

        }



        else if (Network.peerType == NetworkPeerType.Server)
        {//当前状态为连接服务器那么显示以下内, If the current state is
connected, the following buttons will be displayed.。
            GUILayout.Label("Server has been created");
        }
        else if (Network.peerType == NetworkPeerType.Client) {
            GUILayout.Label("client connected");
        }
    }
     void OnServerInitialized()//display on Server
    {
        print("Completed initialize");
        //Network.player;//访问到当前的客户端
        int group = int.Parse(Network.player + "");//直接访问network.player会
得到客户端的索引, Direct access to network.player will get the index of the
client.
        Network.Instantiate(playerPrefab, new Vector3(0, 10, 0),
Quaternion.identity,group);
    }
   void OnPlayerConnected(NetworkPlayer player)//display on Server
    {
        print("A New player has been connected,Index number:" + player);
    }
    void OnConnectedToServer()//display on CLIENT
    {
        print("Successful connected");//for test
        int group = int.Parse(Network.player + "");//直接访问network.player会
得到客户端的索引
        Network.Instantiate(playerPrefab, new Vector3(0, 10, 0),
Quaternion.identity, group);
    }
}
```

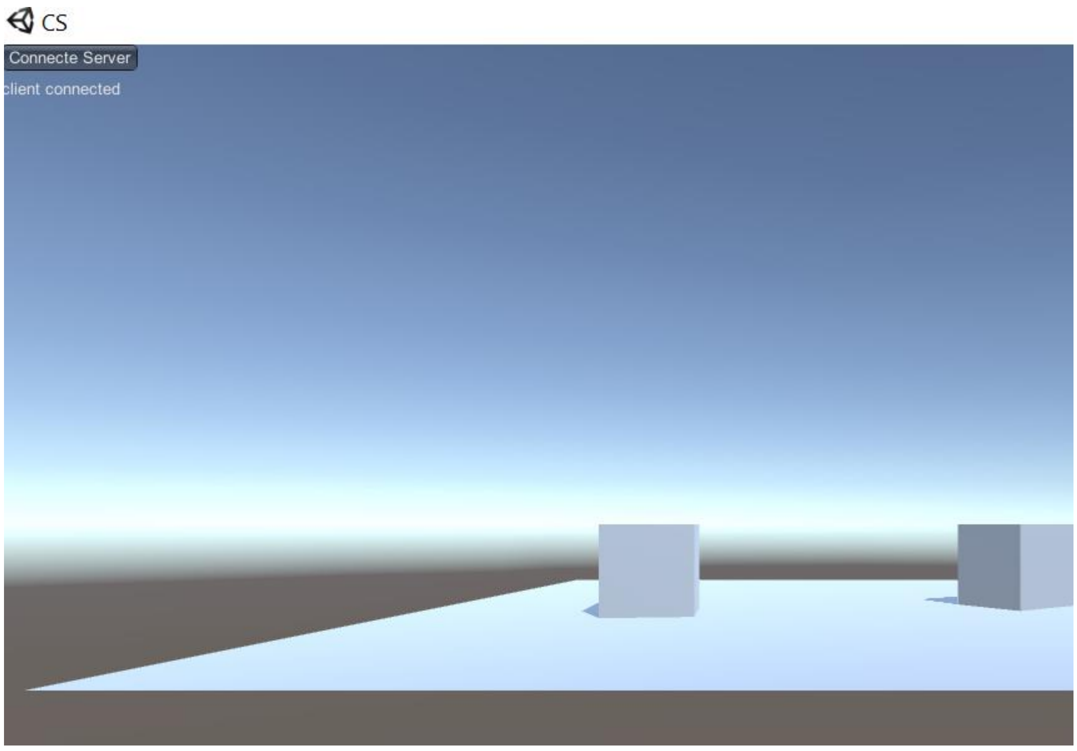
Using Cube instead of character:

```
public class Mycube : MonoBehaviour {
    public float speed = 5;
    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        float h = Input.GetAxis("Horizontal");
        float v = Input.GetAxis("Vertical");
        transform.Translate(new Vector3(h * speed * Time.deltaTime, 0, v *
speed * Time.deltaTime));
    }
}
```

When you click to create a server, a CUBE will be generated in the game scene, so when we use the client to connect, then the server will receive the message from the client and synchronize the screen, which will have 2 cube in the game scene:



Server has been created



client connected

6.2 Limitations

It does have a limit. If developer use the NETWORK component to create a server, developer have to use RPC to synchronize the action of the role. If not, the server can only accept the message sent to the client, that is Server can only receive a role created from the client and display it on the map., and the server cannot sync character's animation which is created by client . On the contrary, the same is the same for the client. So it's still not flexible enough to create servers in this way.

.

6.3 Opportunities

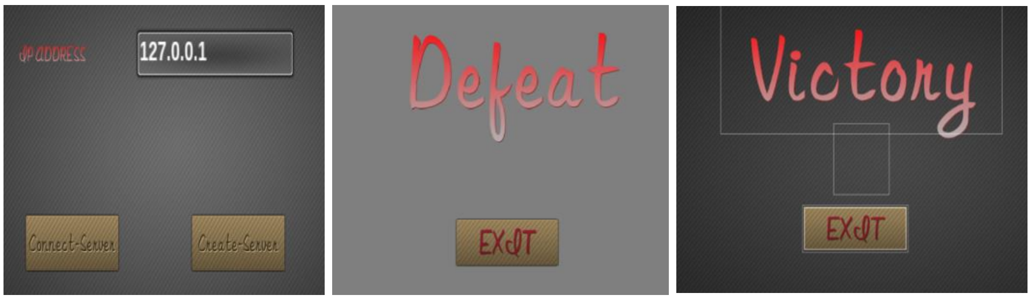
The game is focused on the ability to implement a LAN connection, which is important for many games, so this function can also be developed in any other game. In the current game market, many games pay attention to the network connection, but ignore the local connection, which is not good for the game experience. Many players have suggested that the game official can create a local connection, so there is a great opportunity to connect this function locally in the future. Used on all kinds of games.

Chapter 7

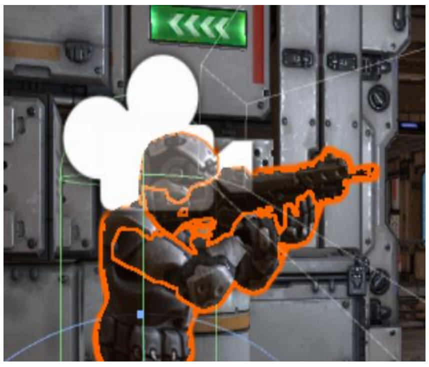
Conclusion

For this project, I have searched a lot of resources and teaching on the Internet, including the use of NGUI to make game interfaces, to realize the animation effects of the game roles, to control the game's vision, to add guns to the game characters, to add bullets and to generate bullets. Use network component to create server, use RPC to remote invocation and synchronized game screen.

NGUI: In the game's start menu and game over menu, the game buttons are completed by NGUI, when player click create-server button then player will create a server, then another player click connect-server will connect the server.



Game view: because this is first person shooter game, we need to use the main camera in the head position of the game character, and let it move with the role and move.



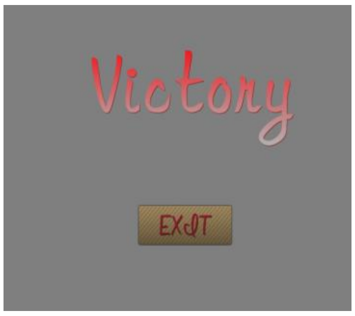
Bullet effect and generate: Every time a character uses a weapon to fire, a bullet will be generated and every bullet has damage

Server and client synchronization: When the server and client are created and connected successfully, it is necessary to synchronize the game screen with the RPC function.

7.1 Game Bug

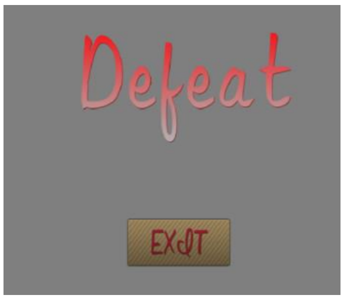
Although all the functions of this game have been completed, but there is still a bug, that is, when the game role is killed, the game victory can show the game over interface, but the loser can not show the game over interface, I tried many methods to solve it, but it is very regrettable, I failed. I guess Maybe it's the unity version problem that I can't call GUIText or unity can't be compatible with 2 guitext. but I'm sure my code is no problem.

Can be displayed when game over:



Cannot be displayed when game over:

But it's useful to detect the role of death



7.2 Future development

The functions that need to be implemented by the game have been completed. The important functions of the game have been tested after many tests. So in the future development, this game has a huge development space, function, role special effects have already have, then only need to create the network server, in addition to some play, such as single person SOLO mode, multi person team cooperation mode and so on, I believe that after learning more technology, I will improve the project Better.

I've got a conditional offer for the master of game development and design at Limerick University, so I will go to the master of UL in September this year if I can get 2.2 grade this year. After that, I will continue to improve and develop this game project. After all, this is my first 3D game project. I will attach great importance to this project.

.

Chapter 8

Appendix

Project-Source-Link: https://github.com/neroZWX/Death-Adrenaline
Project-Documentation-Link:https://github.com/neroZWX/Death-Adrenaline/report.pdf
Unity3d API: https://docs.unity3d.com/ScriptReference/

Bibliography

[1] "LAN"(Local Area Network) Available: https://searchnetworking.techtarget.com/definition/local-area-network-LAN [Accessed: 07- 04- 2018].

[2] "API"(Application-Programming-Interface) Available: https://medium.freecodecamp.org/what-is-an-api-in-english-please-b880a3214a82[Accessed: 07- 04- 2018].

[3] "Network-component." Available: https://unity3d.com/cn/learn/tutorials/s/multiplayer-networking?_ga=2.159262714.1568031307.1523759955-1107744089.1523231727 [Accessed: 07- 04- 2018].

[4] "C#" Available: https://docs.microsoft.com/en-us/dotnet/csharp/ [Accessed: 07- 04- 2018].

[5] "half-life-counter-strike" Available: http://store.steampowered.com/app/70/HalfLife/, [Accessed: 14-04-2018].

[6] "NGUI-plugins" Available: https://assetstore.unity.com/packages/tools/gui/ngui-next-gen-ui-2413 [Accessed: 14- 04-2018].

[7] "RPC" Available: https://docs.unity3d.com/Manual/net-RPCDetails.html [Accessed: 14- 04-2018].

[8] "unity3d" Available: https://unity3d.com/cn/ [Accessed: 14-04-2018].

[9] "IOS" Available: https://en.wikipedia.org/wiki/IOS [Accessed: 14-04-2018].

[10] "VS"(Visual-Studio) Available: https://www.visualstudio.com/zh-hans/ [Accessed: 14-04-2018].

[11] "BUG" Available: https://www.codesimplicity.com/post/what-is-a-bug/ [Accessed: 14-04-2018]

[12] “.NET” Available: https://www.microsoft.com/net/ [Accessed: 14-04-2018]

[13] “GITHUB” Available: https://github.com/ [Accessed: 14-04-2018]

[14] “HTMNL5” Available: https://www.w3schools.com/html/html5_intro.asp [Accessed: 14-04-2018]

[15] “SVN” Available: https://subversion.apache.org/ [Accessed: 14-04-2018]

[16] “Jquery” Available: https://jquery.com/ [Accessed: 14-04-2018]

[17] “python” Available: https://www.python.org/ [Accessed: 14-04-2018]

[18] “c++” Available: http://www.runoob.com/cplusplus/cpp-tutorial.html [Accessed: 14-04-2018]

[19] “KISS-principle”(**K**eep **I**t **S**imple, **S**tupid) Available: https://www.interaction-design.org/literature/article/kiss-keep-it-simple-stupid-a-design-principle [Accessed: 14-04-2018]