# 摘要

随着生活水平与科技的发展，游戏已经成为现代人缓解压力的一大利器，使用好这一利器，有利于锻炼自己的反应能力拓宽自己的朋友圈等等，而市面上游戏引擎五花八门，其中最吸引眼球的是 Unity 引擎，它凭借三大优势，孕育了成百上千款高品质游戏，而这三大优势为：第一，Unity 引擎功能丰富，插件资源庞大，大大地减少了开发者的对游戏的开发时间；第二，它的可视化界面对开发新手老手都十分友好，容易上手；第三，支持跨平台的开发。

本文将构思并实现一个基于 Unity 引擎技术的多人在线射击游戏，并将其实现。之所以会选择这个课题，是因为我认为射击游戏在过去，现在，未来都是属于不会没落的游戏类型，在世面上，各种射击游戏让人眼花缭乱，反恐精英、穿越火线、绝地逃生、守望先锋等游戏都深受人们喜欢，而联机游戏，既可以让玩家感受游戏内容，还可以与小伙伴一起享受开黑的乐趣，在开发过程中，我使用了 Unity 中一套称为 Mutiplayer 的网络体系结构，以局域网的形式进行联机，是一种基于 UDP 协议的高性能传输层，包含以 NetworkManager 为核心的管理工具，我可以在脚本中进行网络开发，在开发过程中，这个网络架构非常有效地降低了程序在时间上的复杂度,提高了游戏运行时的实效性，解决了开发联机游戏过程中常会发生的同步问题。

**关键词：**Unity3D；射击类游戏；多人联机；功能实现

# Abstract

With the development of living standards and technology, games have become a great weapon for modern people to relieve stress. The use of this weapon is good for exercising their responsiveness and widening their circle of friends. The game engines on the market are diverse. One of the most eye-catching Unity engines, it has nurtured hundreds of thousands of high-quality games by virtue of three advantages, and these three advantages are: First, the Unity engine is rich in functions and the plug-in resources are huge, greatly reducing the number of developers The development time of the game; second, its visual interface is very friendly to novices and veterans, easy to get started; third, one-time development supports multi-platform development.

This article will conceive and implement a multiplayer online shooting game based on Unity engine technology and implement it. I use a network architecture called Mutiplayer in Unity, which contains NetworkManager as the core management tool. Developers can perform network development in scripts, for example: NetworkIdentity can be used for non-player game objects, a series of components For example, NetworkTransform and NetworkAnimator can be used for assistance. ClientRpc, SynCar and other packages are very comprehensive. Secondly, in terms of how to make the game run, this network architecture effectively reduces the complexity of the program in time, improves the effectiveness of the game when it runs, and greatly solves this problem.



**Keywords**：Unity3D; shooting game;Multiplayer ;Function realization

# 目录


**第一章 绪论** ..... 1

**1.1 选题的目的和意义** ..... 1

**1.2 研究现状** ..... 1

**1.3 研究内容** ..... 2

**1.4 章节安排** ..... 2

**第二章 Unity 引擎与相关软件介绍** ..... 3

**2.1 Unity3D 游戏引擎** ..... 3

2.1.2 Unity 引擎的特点 ..... 3

2.1.3 Unity 引擎的优势 ..... 3

2.1.4 Unity 引擎重点常用函数介绍 ..... 4

**2.2 插件介绍** ..... 5

**2.3 开发过程中使用的软件介绍** ..... 5

**2.4 输出游戏平台** ..... 5

**第三章 游戏需求设计** ..... 6

**3.1 游戏基本介绍** ..... 6

**3.2 游戏功能** ..... 6

3.2.1 游戏界面 ..... 6

3.2.2 游戏控制 ..... 7

3.2.3 游戏设计 ..... 8

**第四章 游戏功能实现** ..... 9

**4.1 Unity Network** ..... 9

4.1.1 Unity Manager ..... 9

4.1.2 Unity Manager HUD ..... 9

**4.2 创建 Player 角色** ..... 10

4.2.1 角色模型及控制脚本.....10
4.2.2 角色绑定网络并进行 Transform 同步.....11
4.2.3 角色随机出生点和重生点.....11
4.2.4 角色血条控制并进行检测同步.....12
**4.3 射击功能**.....13
4.3.1 子弹的生成.....13
4.3.2 子弹绑定网格并进行同步.....13
**4.4 创建敌人预制体**.....14
4.4.1 敌人模型.....14
4.4.2 敌人绑定网格并进行同步.....15
4.4.3 敌人随机生成.....15
**第五章 美术实现**.....16
**5.1 场景**.....16
5.1.1 外部场景.....16
5.1.2 内部场景.....16
**5.2 游戏 UI**.....17
5.2.1 菜单.....17
5.2.2 UI.....18
**5.3 游戏音效**.....19
5.4.1 游戏开始前中后音效.....19
5.4.2 游戏内音效.....19
**5.4 游戏灯光**.....20
**第六章 游戏测试**.....21
**6.1 游戏测试**.....21
**第七章 总结与期望**.....23
**参考文献**.....25
**致谢**.....26

# 第一章　绪论

## 1.1 选题的目的和意义

随着生活水平与科技的发展，游戏已经成为现代人缓解压力的一大利器，使用好这一利器，有利于锻炼自己的反应能力，拓宽自己的朋友圈。所以当人们对游戏要求提高，开发游戏的要求也随之变高。市面上开发游戏的引擎五花八门，根据GameTracker 数据排行榜显示，Unity3d 与 Cocos2D 系列引擎，成为全球最主流的两大第三方引擎的供应商，其中最吸引眼球的 Unity 引擎，它凭借三大优势，孕育了成百上千款高品质游戏，根据有效数据显示，Unity 用户超过 330 万人，每月活跃用户高大 70 万，市场比例接近百分之四十五，它的优势分为三部分，第一，Unity 引擎功能丰富，插件资源庞大，大大的减少了开发者所需的对游戏的开发时间；第二，它的可视化界面对开发新手老手都十分友好，即便是不会编程的人也能快速适应并上手；第三，高度的可跨越平台性，减少了为了适应平台所投入的人力物力资源，作为开发者来说，不管是哪个引擎，都必须为爱好者提供交流的平台，尽量给开发者自己的空间，可以提供丰富的开源组件，也能提供开发者自行设计编辑器的组件。

## 1.2 研究现状

中国人口基数大，游戏的市场需求大，玩游戏的人口基数大，但大多数人都围绕着热门游戏玩，所以热门游戏就占据了游戏市场中较大的份额，而热门游戏中，例如绝地求生（TPS 类）和英雄联盟（MOBA 类）就属于国内热门游戏，已经变成了大家交流的一个媒介。游戏除了让人们缓解生活带来的压力外，还能以此为话题进行沟通交流，而游戏市场也是围绕着这两大类去更新玩法，给人带来新鲜的东西，吸引别人进行游戏，本文就 TPS 类游戏展开设计开发，第三人称射击游戏，也就是我们说的 TPS 游戏类型，玩家可以看到自己的角色进行行走射击等操作，即为 TPS 游戏，该类游戏收获了国内外许多玩家的喜爱。相对于第一人称游戏的紧张感和操作感，TPS 游戏往往更加有策略性和操作性，玩家通过第三视角全方位的进行掌握局势，通过击杀大量敌人产生刺激感、成就感，从而获得游戏体验。

## 1.3 研究内容

从最初的小想法到写需求分析、游戏文档，到分析游戏并进行最终开发测试，阅读了许多 Unity 专业知识书籍以及许多基于 Unity3D 技术的开发者论坛博客，设计并实现了多人在线射击游戏。首先，为了实现多人联机，我用的是 Unity Networking 网络技术。基于 UDP 的高性能传输层是他最大的特点，可以支持所有游戏类型，底层通过 LLPI 进行全面控制，在面临如何改善游戏的网络同步问题上，我查阅了非常多的资料,解决了游戏运行时的服务器端与客户端的实效性问题，该游戏在保证运行效果的同时，可以实现客户端与服务器的同步，具有设备要求低，跨平台性能好，运行流畅稳定的特点，其次，为了提高游戏体验，使用 TextMesh pro 字体组件，有效地丰富玩家的视觉效果，还通过 NavMeshAgent 组件，用来实现角色进行寻路导航的功能开发，在开发过程中根据游戏功能进行逻辑推理，实现功能后测试中调整游戏数值参数，使游戏更加完善。

## 1.4 章节安排

本论文分为七章，结构安排如下：

第一章：绪论，将会从本课题研究的目的及意义出发，讲述研究现状与研究课题。

第二章: 介绍开发环境和 Unity 引擎、还有其他涉及到本次开发的插件和软件。

第三章：介绍本次游戏开发在前期的需求设计和构想，包括于设定的游戏背景和游戏玩法规则，考虑该游戏需要拥有的功能和如何实现。

第四章：介绍游戏的开发阶段，从功能出发，讲述如何编写脚本实现，包括使用 Unity 中一套称为 Mutiplayer 的网络体系结构，以 NetworkManager 为核心的管理工具，从创建人物、敌人角色，编辑控制脚本，角色的血条控制、子弹的形成、计数器的逻辑，联网的代码逻辑，服务器端和客户端之间的同步问题等。

第五章：场景、UI、音效、灯光等的设计与实现，从寻找合适的模型到，进行搭建场景，对界面 UI 的设计、合适的游戏音效和背景音乐、添加灯光渲染气氛。

第六章：游戏完成后，对游戏进行多次的测试，包括界面测试、功能测试，打包封装。

第七章：结论和感想，作为论文最后一章，对本次毕业设计开发中的总结，包括学到的知识，开发中的感想。

# 第一章　开发工具介绍

## 1.1 Unity 引擎

### 1.1.1　Unity 引擎来源

Unity 拥有着交互舒适的操作界面，让开发者能够快速上手[4]，Scene 场景窗口是开发者最熟悉的部分，你可以在这里进行大胆创作，例如摄影机、光源、模型、材质、粒子系统、天空盒等，你可以在这里对你的游戏预制体进行编辑，包括不限于选择物体、旋转物体、移动物体、缩放物体；Game 游戏窗口，这个窗口主要用来呈现出完整的动画效果，不能进行编辑，用来渲染 Scene 中的动画；如果你想知道你正在编辑的对象的具体数据，你对应在 Inspector 编辑窗口查找，包括不限于名字、标签坐标、角度、大小、身上的组件、脚本等各种属性信息；Hierarchy 清单窗口，陈列所有 Scene 中的预制体；Project 项目窗口，你的所有模型素材、组件、脚本等都在这。

### 1.1.2　Unity 引擎的特点

Unity 具备着高效、高度优化、高度集成等优点。

### 1.1.3　Unity 引擎的优势

第一，Unity 引擎功能齐全，插件资源庞大，大大缩短了开发时间，Unity 引擎在开发上，为开发人员提供了可视化工具和 3D 补偿引擎；在脚本上，有着多样的语言环境，还有着非常庞大 的 IDE 社区，在用户体验上，提供了严谨的系统用于处理项目，给合作开发的团队提供了版本管理，只要用过这个引擎的人都会知道 Asset store，这个是他们公司自己所经营的，包含着超过三千种工具、组件。为后续的开发工作提供了非常大的便利，特别注意的是，里面的免费产品的质量也及其高。拥有高度集成的特点，带有粒子、物理、简单模型系统，调用方便，全程可视化调整。

第二，所谓所见即所得，即便是不会编程的人也能快速适应并上手，例如说开发时你并不需要对物理了解得多深，引擎的物理系统已经接近完善了；当你想有碰撞检测时，你不需要自行编辑代码，你只需要通过组件和鼠标，就可以实现自己想要的效果；当你希望你的人物可以进行导航寻路，敌人可以自动巡逻，都可以用unity 里面强大的组件进行设计吗，虽然不足与 3dmax 相比专业，但是 unity 进行初步场景设计也是绰绰有余的，包括场景的烘焙，渲染，制作粒子效果，无论是哪个单拿出来都属于比较有技术性的模块功能，但在该引擎中，你只需要下载几个组件，进行操作设置就能实现大部分你想要的功能，这样也是在减少开发游戏的资金、资源，降低开发游戏的难度，缩短开发游戏的时长。而官方也提供了非常详细的文档说明，并且在 18 年版本之后新增了中文语言，方便我们进行开发游戏。

第三，高度的可跨越平台性，减少了为了适应平台所投入的人力物力资源。该引擎在常见的 Win、Linux 和 Mac 系统上都可以进行开发，开发的游戏可以发布到对应的平台。甚至你也可以通过他发布网页游戏，支持常用系统的网页浏览。它的网页播放器也被 Mac 所支持。

### 1.4.1 Unity 引擎重点常用函数介绍

**表 2-1 函数表**

| 函数名 | 内容 |
| --- | --- |
| Update | 当 MonoBehaviour 启用时，其 Update 在每一帧被调用。 |
| LateUpdate | 当 Behaviour 启用时，其 LateUpdate 在每一帧被调用 |
| FixedUpdate | 当 MonoBehaviour 启用时，其 FixedUpdate 在每一帧被调用,常用与物理引擎的调用，以及需要固定时间更新的函数 |
| Awake | 一个初始化函数，当一个脚本势力被载入时，Awake 被调用，一般用于创建变量。 |
| State | 一个初始化函数，仅在 Update 函数第一次被调用前调用，一般用于富裕变量值。 |
| Reset | 重置与默认值。 |

| | |
|---|---|
| OnTriggerEnter | Collider 进入 trigger 时调用 OnTriggerEnter，一般用于判定物体碰撞到角色。 |
| OnTriggerExit | Collider 停止触发 trigger 时调用 OnTriggerExit。 |
| OnTriggerStay | 当碰撞体接触触发器时，OnTriggerStay 将在每一帧都调用。 |

## 1.5 插件介绍

1、TextMesh pro：是该引擎 Unity 收购的插件，足以看出他的强大，强大的 UI 功能受不少开发者喜爱,这款组件解决了国内开发者在 Unity 里无法进行文字渲染，不会出现乱码情况，你可以经过网格渲染，也可以经过 UI 渲染，其中最著名的一个技术——SDF，在我们传统的渲染技术中，都是通过像素代表字符形状，而这个新技术是是把输入的字符串放到一个集合中，形成字符状态，它的优点是字符不管如何缩放都可以精准的渲染出清晰的字符。

2、NavMeshAgent：作为寻路系统的核心组件，有了这个容器，可以生成导航网格，这个 NavMesh 可以设置我们预制体的活动范围。

## 1.6 开发过程中使用的软件介绍

1、Unity3D：功能插件资源都非常成熟，能有效的缩减开发时间，该引擎界面交互友好，及时不懂编程的人也可以轻松做出一款小游戏，常见的平台都可支持。

2、3ds Max：如果说到 3dmax 三维动画渲染和制作的软件，我想 3dmax 有它自己的一席之地，开发中可以通过他建造各种场景模型和人物模型，可以对材质进行处理。

3、Visual Studio ：拥有非常成熟的开发工具，在主流的编码软件算得上是非常成熟了，拥有着开发者所需要的工具，例如代码管控、UML、IDE 等等，VS 对各种平台有着高度的适应性。

4、Photoshop：PS 是一款可以处理图像的工具，可以利用他对图片做需要的编辑操作，PS 支持多个常用平台。

## 1.7 输出游戏平台

通过调查，Windows 平台性能游戏在性能的限制和平台使用量占有优势，所以决定将会输出为 Windows。

# 第一章　游戏需求设计

## 1.1 游戏基本介绍

主角 Henry 是宇宙中一万亿颗星球中的一员，作为一名维护自己星球和平的保卫队员，他的日常就是在自己的领域上巡逻，有天他接到总部的通知，称 Tw-1 空间站失去了连接，在最后一次连接时，总部收到了该空间站的求救信号，信息表明，空间站被外星种族 MAX 突袭，并侵占了所有资源，Henry 的任务就是到达 Tw-1 空间站，消灭所有敌人，并收集所有被抢占的资源，当然你可以邀请你的战友 Jack 与你协同作战！

在游戏菜单中选择开始游戏，一名玩家作为局域网中的主机，以参数的形式建立主机名，其他玩家寻找该局域网并进行连接，连接完成后进行数据的传递与同步，联机完成则玩家会出生在停机坪，等待小伙伴们连接成功，就可以和小伙伴控制自己的角色进行上下左右移动，空格键为射击键，穿过一座桥，就到达空间站，空间站里有不少敌人，玩家需要操作自己的人物消灭所有的敌人，同时玩家还需要收集空间站中因掠夺散落一地的资源，操控人物触碰即可拾取，当所有敌人被消灭，资源都回收，游戏就获得胜利。

## 1.2 游戏功能

### 1.2.1　局域网的建立、连接

主服务器端建立后，通过 UDP 协议广播地址，内容包括服务器的 id、服务器组、主机的名字还有 ip 地址，而客户端通过接受到广播后，会在列表中添加相关信息，同时同步自身信息到主服务器端，从而建立连接。

### 1.2.2　游戏界面

游戏需要一个游戏菜单来引导玩家进行开始和退出游戏，在游戏过程中，有血条的显示，让玩家清晰的知道自己的血量和敌人的血量，也能知道已经收集多少被强占的资源。

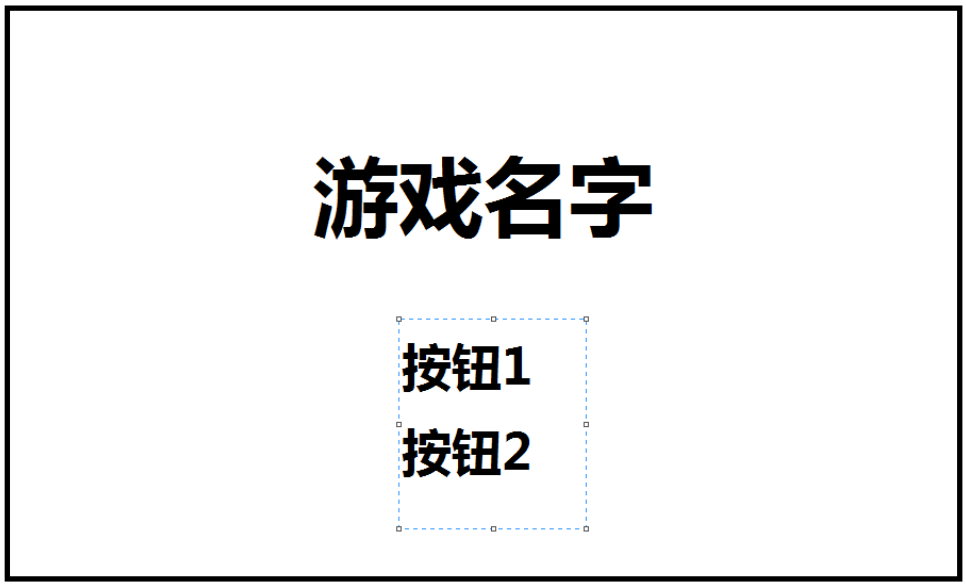


图 3-1 游戏菜单

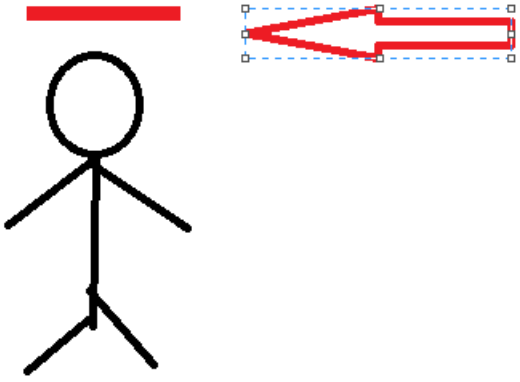

图 3-2 血条设想

### 1.7.1 游戏控制

玩家控制键盘，控制上下左右，空格射击，触碰资源即拾取。

### 1.7.2 游戏设计

（1）玩家需要控制角色进行射击敌人，躲避敌人攻击，通过上下左右移动，空格射出子弹，

（2）敌人自动巡逻并检测到角色进入特定范围内会主动进行攻击

（3）玩家和敌人显示血条，当死亡时销毁并播放死亡音效

（4）当检测碰撞到资源时，玩家资源数+1

（5）玩家清理完敌人，收集完全部空间站中的资源即游戏胜利

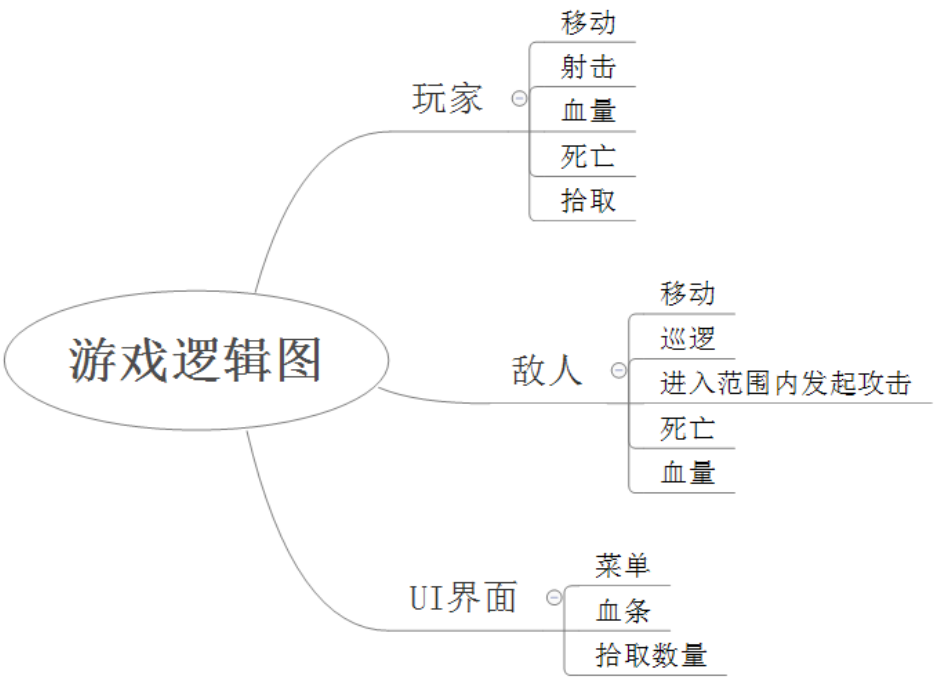


图 3-3 游戏逻辑图

# 第二章　游戏功能实现

## 2.1 Unity Network

Network Manager [1]（如图 1）,就如他的名字，是一个网络管理器，用于管理从预制体进行实例化网络对象，开发者可从派生一个类去继承又 HLAPI 实现的 Network Manager，这个组件可以进行游戏的状态管理、游戏的派生管理、游戏的场景管理、游戏中网络的各项自定义等等。

### 2.1.1　Unity Manager

添加网络管理器，Network Manager 是 unity 的组件（如图 1）,就如他的名字是一个网络管理器,开发者可从派生一个类去继承又 HLAPI 实现的 Network Manager，并且可以通过自己的需求去编写任何他可以提供的虚拟函数，这个组件可以进行游戏的状态管理、游戏的派生管理、游戏的场景管理、游戏中网络的各项自定义等等。

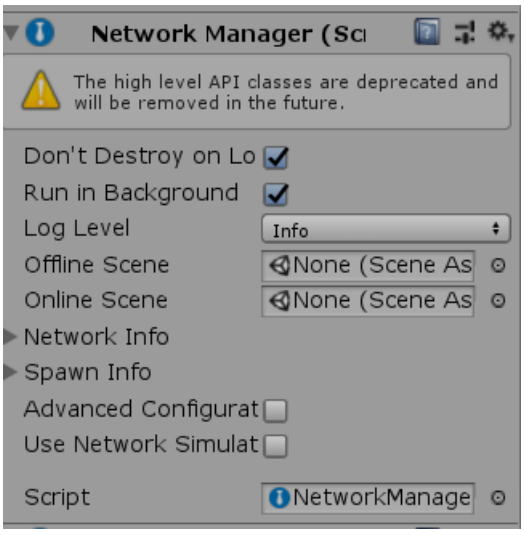


图 4-1 网络管理器

### 2.1.2　Unity Manager HUD

首先创建如图 2 的网络管理器，需要在 Inspector 面板上配置和编辑网络相关状态，有一个网络管理器可视化组件，后缀是 HUD，为开发者打开了一个高效率的操作界面，十分友好，在游戏中控制显示当前的网络状态,方便开发时进行测试，图里 host 为本机的服务器段，client 为客户端，当我们要处理一些事件时，只需要在服务器上处理，不需要重复处理。

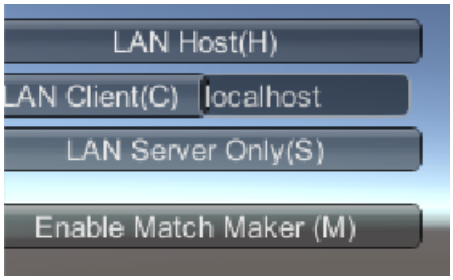


图 4-2 网络可视化组件

## 1.1 创建 Player 角色

### 1.1.1 角色模型及控制脚本

1、角色模型如图，进行注册联网对象，找到 Spawn Info 属性集从组件面板中，挂上我们的玩家预制体在 player prefab 上

图 4-3 角色模型

1、编写角色控制移动脚本、空格发射子弹，我们的前提是 Unity Engine.Networking，再改为 NetworkBehaviour，识别到本地玩家后进行区分，不然会控制到不同的客户端的玩家对象。

```
1.  void Update () {
2.          if (isLocalPlayer == false)    {
3.              return;
4.      }
5.          float h = Input.GetAxis("Horizontal");
6.          float v = Input.GetAxis("Vertical");
7.          transform.Rotate(Vector3.up * h * 120 * Time.deltaTime);
```

```
8.          transform.Translate(Vector3.forward * v * 3 * Time.deltaTime);

9.          if (Input.GetKeyDown(KeyCode.Space))
10.         {
11.             CmdFire();
12.         }
13.   }
```

### 2.1.3 角色绑定网络并进行 Transfrom 同步

由于是多人联机，角色不管在服务器端生成还是客户端生成，都需要进行角色的位置同步，而 Network SendRate 是表示网络数据的同步，如果频率太高会使游戏的帧数过高导致网络延迟，但是频率低又会影响玩家游玩体验，具体代码如下：

```
1.    [Command]
2.    void CmdFire()//这个方法需要在 server 里面调用
3.    {
4.        GameObject bullet = Instantiate(bulletPrefab, bulletSpawn.posi
tion, b             ulletSpawn.rotation) as GameObject;
5.        bullet.GetComponent<Rigidbody>().velocity = bullet.transform.f
orward              * 10;
6.        Destroy(bullet, 2);
7.        NetworkServer.Spawn(bullet);
8.    }
9.  }
```

### 2.1.4 角色随机出生点和重生点

有两个随机的出生点和重生点，NetworkStartPosition 组件可以用来控制玩家的出生点，原理是访问 NetworkStartPosition 可以让列表通过 Network.StartPositions,帮助在 NetworkManager 实现 OnServerAddPlayer，如图

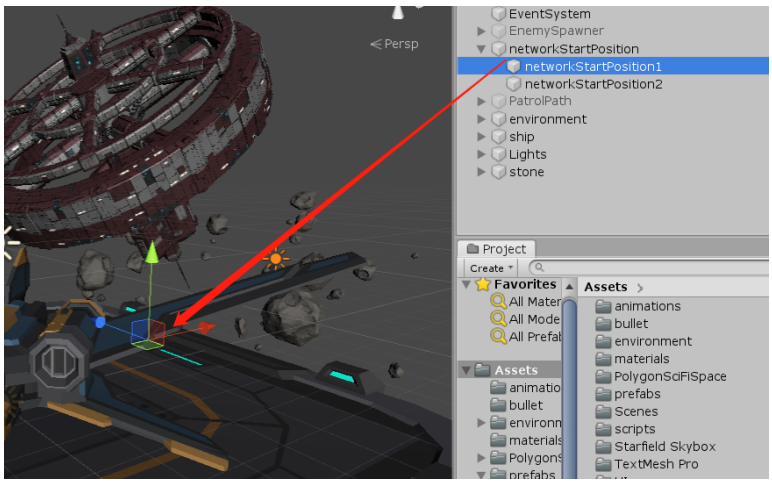


图 4 -4 随机出生点

重生时，第一步应检测是否为本地角色，第二步进行血量检测，当血量＜＝0时，则死亡，代码如下：

```
1.   public void TakeDamage(int damage)
2.       {
3.           if (isServer == false) return;// 检测血量这部分只在服务器端
4.           currentHealth -= damage;
5.           if (currentHealth <= 0)    {
6.               if (destroyOnDeath)        {
7.                   Destroy(this.gameObject); return;
8.               }
9.               currentHealth = maxHealth;
10.              Debug.Log("Dead");
11.              RpcRespawn();
12.          }      }
```

检测到角色死亡后，检测重生的随机点，在这里添加 ClientRpc，是为了是调用双端同步以执行任务，方法是序列化客户端的数据并将其发送给客户端以执行，首先服务器将检测到挂在对象 Components 上的 NetworkIdentity 然后将 RPC 命令发送大各个所监视的客户端上，代码如下：

```
1.   [ClientRpc]
2.   void RpcRespawn()//客户端调用   {
3.       if (isLocalPlayer == false) return;
4.       Vector3 spawnPosition = Vector3.zero;
5.       if (spawnPoints != null && spawnPoints.Length > 0)     {
6.           spawnPosition = spawnPoints[Random.Range(0, spawnPoints.Leng
   th)].transform.position;
7.       }
8.       transform.position = spawnPosition;
9.   }
```

### 2.1.5 角色血条控制并进行检测同步

当前角色血量为 100，首先要做一个碰撞检测，当子弹碰撞到角色后，子弹需要自我销毁，再处理角色的血量，调用 takedamage 这个方法，把血量-10，这里有一种逻辑，当销毁某些东西时，应该调用 NetworkIdentity.OnDestroy，最好的方法是在服务器端销毁它，然后与每个客户端同步，这样就可以减少如果客户端销毁时，服务器端的物体会突然消失的问题，调用服务器检测到后，预制件销毁，服务器端也同步销毁。

```
public class Bullet : MonoBehaviour {
```

```
    void OnCollisionEnter(Collision col)    {
        GameObject hit = col.gameObject;
        Health health = hit.GetComponent<Health>();
        if (health != null)     {
            health.TakeDamage(10);
        }    Destroy(this.gameObject);
    } }
```

血条始终保持面向相机，代码如下：

```
  void Update () {
        transform.LookAt(Camera.main.transform);
  }
```

## 2.2 射击功能

### 2.2.1 子弹的生成

首先实例化子弹,子弹的位置，子弹的攻击力为 10，当子弹射出并碰撞后，销毁子弹

```
public GameObject bulletPrefab;//子弹
public Transform bulletSpawn;//子弹位置
 void OnCollisionEnter(Collision col)    {
        GameObject hit = col.gameObject;
        Health health = hit.GetComponent<Health>();
        if (health != null)    {
            health.TakeDamage(10);    }
        Destroy(this.gameObject);
    }
  }
```

### 2.2.2 子弹绑定网络并进行同步

做完前面的工作后，测试发现，子弹射中后可以出发掉血，但是当由于涉及后回触发 TakeDamage 方法，如果其中一方接触到对方并且销毁时，由于网络延迟，另一方在打中对方前就进行了销毁处理，导致双方数据的不同步，这是因为先前子弹只是在客户端生成，并没有进行同步，所以我们需要把子弹绑定网格再同步到各个客户端，首先，子弹要在网络生成，需要把子弹注册到网络管理器中，在此之前，要在子弹上挂上 NetworkIdentify，增加 Command 特性，原理是代码在服务器端运转，但需要在客户段调用，把子弹同步到各个客户端，这时候可以用 Spawn 方法，可以把指定生成在各个客户端，当具有网络徽标的预制件挂在服务器 Spawn 上时，该预制件将调用 Onstar Server，而后进行调配 netid，当完成这个注册这个流程后，会更改对应的网络物体列表，接着查找所有客户端带有网络标识的物体，如果状态为 TRUE，则将 MsgType.ObjectSpawn 或是 MsgType.ObjectSpawnScene 传递到预制体上，在注册列表中的预制体里面寻找对应的物体，找到后更新注册到 NetworkScene 的列表中，需要更新相关信息，这时客户端上的子弹没有速度，我们得先进行数据的同步，先在子弹上挂上组件获取他的位置和状态，同步刚体组件。因为子弹的速度不是时刻变化，所以只需要同步一次即可，从客户端调用参数的前提条件是可以对网络进行序列化，以便可以对参数进行反序列化。 在开发时，由于我的粗心，发生了无法调用的错误，在我查阅资料后发现，我的客户端上忘记了挂上 NetworkIdentity 组件，像 spawn、destroy 这些都需要有一个 NetworkIScene 为前提。SynCar 特性是解决这个方法的核心，通过这个方法，如此一来可以使服务器的值同步到客户端上，让双端数据保持一致，防止因为一端改变而影响另外一端，出现错误。

```
1.   [Command]
2.    void CmdFire()//这个方法需要在 server 里面调用
3.    {
4.        GameObject bullet = Instantiate(bulletPrefab, bulletSpawn.posi
       tion, bulletSpawn.rotation) as GameObject;
5.     bullet.GetComponent<Rigidbody>().velocity = bullet.transform.for
ward *              10;
6.        Destroy(bullet, 2);
7.        NetworkServer.Spawn(bullet);
8.    }
9.   }
```

## 1.1 创建敌人预制体

### 1.1.1 敌人模型

设置有两种敌人，一种血量为 100，一种血量为 200，为了增加难度，不同敌人的伤害也不同。