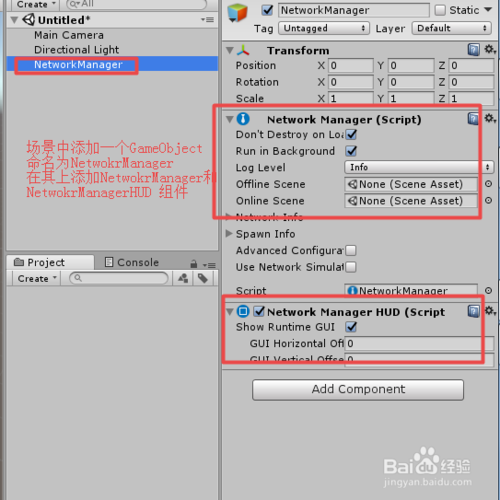
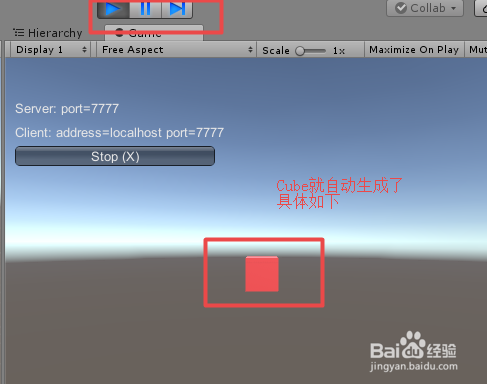
# Unity Network教程之 Network的配置Player生成

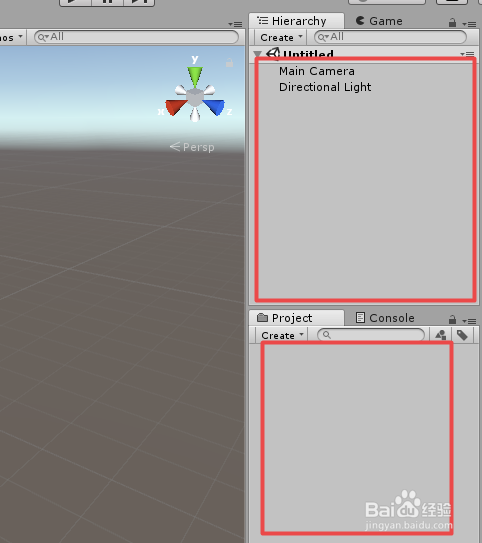
**Unity Network教程之 Network的配置Player生成。本节介绍，使用Unity原生的 Network 进行网络开发第一课，Network 的配置和Player预制体连接主机时候的自动生成，具体如下(效果图预览)**

[](http://jingyan.baidu.com/album/eb9f7b6d4a3765869364e815.html?picindex=1)[](http://jingyan.baidu.com/album/eb9f7b6d4a3765869364e815.html?picindex=2)

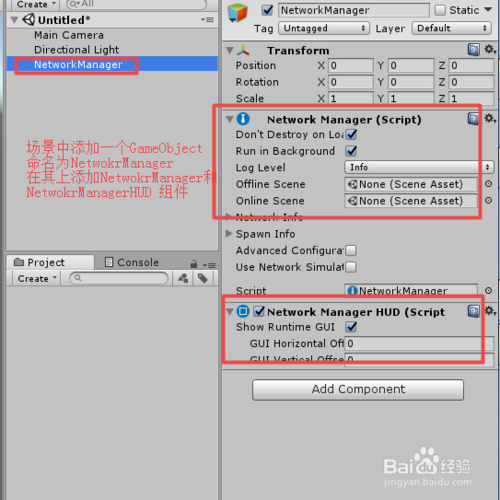
工具/原料

Unity方法/步骤

打开unity，新建一个空工程，具体如下图

[](http://jingyan.baidu.com/album/eb9f7b6d4a3765869364e815.html?picindex=3)

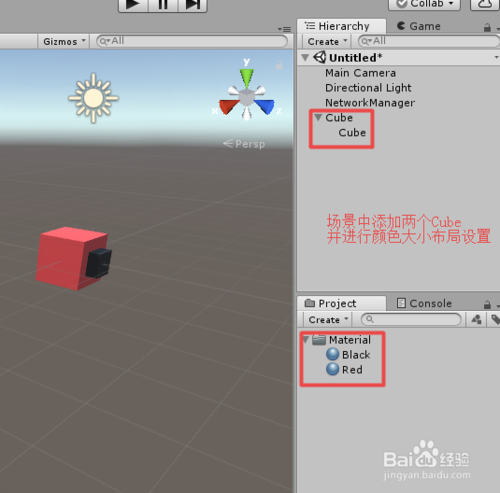
在场景中添加一个空游戏物体 GameObject，命名为 NetworkManager，进行网络管理，在其上添加 NetworkManager组件和 NetworkManagerHUD组件，具体如下图

[](http://jingyan.baidu.com/album/eb9f7b6d4a3765869364e815.html?picindex=4)

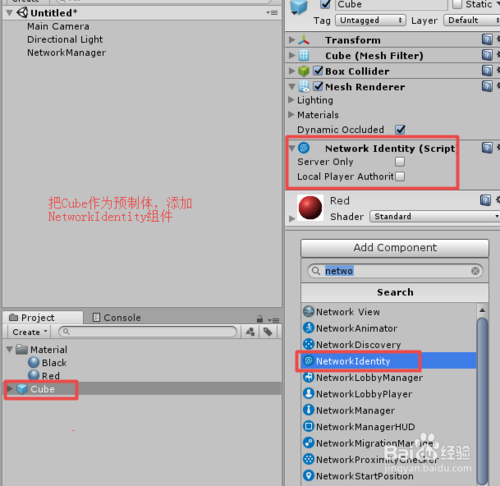
运行场景，即可看到一个连接进入网络的UI，具体如下图，到这里Network的简单配置完成

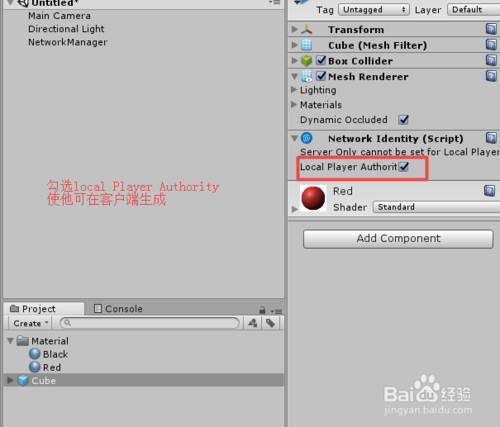
[](http://jingyan.baidu.com/album/eb9f7b6d4a3765869364e815.html?picindex=5)

在场景中添加两个Cube，设置他们的颜色大小，布局如下图

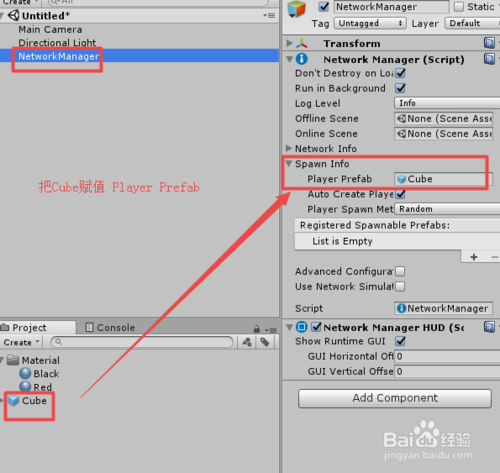
[](http://jingyan.baidu.com/album/eb9f7b6d4a3765869364e815.html?picindex=6)

把Cube拖入工程作为预制体，在其上添加 NetworkIdentity，并且勾选 Local Player Authority，具体如下图

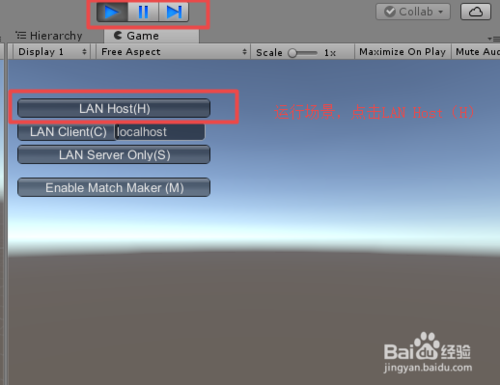
[](http://jingyan.baidu.com/album/eb9f7b6d4a3765869364e815.html?picindex=7)

[](http://jingyan.baidu.com/album/eb9f7b6d4a3765869364e815.html?picindex=8)

把 Cube预制体赋值给 NetworkManager的PlayerPrefab，并且勾选 Auto Create Player，具体如下图

[](http://jingyan.baidu.com/album/eb9f7b6d4a3765869364e815.html?picindex=9)

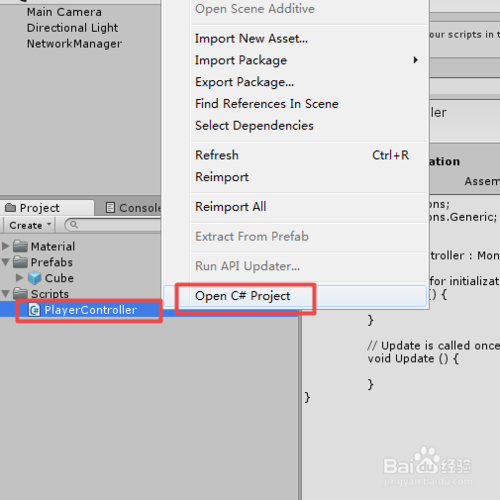
运行场景，点击 LAN Host（H），创建服务和房间(既作为服务器也作为客户端)，就会看到 Cube在房间自动生成，具体如下图

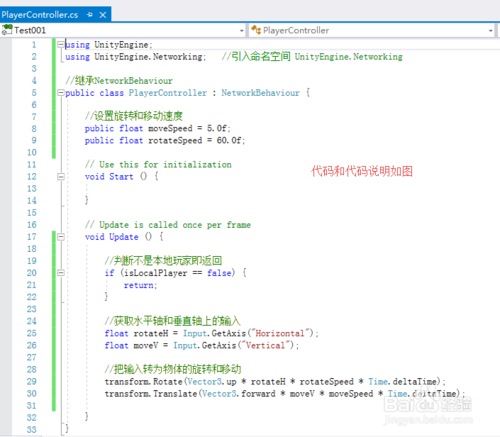
[](http://jingyan.baidu.com/album/eb9f7b6d4a3765869364e815.html?picindex=10)

[](http://jingyan.baidu.com/album/eb9f7b6d4a3765869364e815.html?picindex=11)

# Unity Network教程之二 Player的控制和状态同步

在工程中添加一个脚本 PlayerController，用来控制玩家的移动旋转，代码和代码说明具体如下图

[](http://jingyan.baidu.com/album/08b6a59190835c14a809228e.html?picindex=2)

[](http://jingyan.baidu.com/album/08b6a59190835c14a809228e.html?picindex=3)

PlayerController 脚本的具体内容如下：

using UnityEngine;

using UnityEngine.Networking;   //引入命名空间 UnityEngine.Networking

//继承NetworkBehaviour

public class PlayerController : NetworkBehaviour {

    //设置旋转和移动速度

    public float moveSpeed = 5.0f;

    public float rotateSpeed = 60.0f;

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

        //判断不是本地玩家即返回

        if (isLocalPlayer == false) {

            return;

        }

        //获取水平轴和垂直轴上的输入

        float rotateH = Input.GetAxis("Horizontal");

        float moveV = Input.GetAxis("Vertical");

        //把输入转为物体的旋转和移动

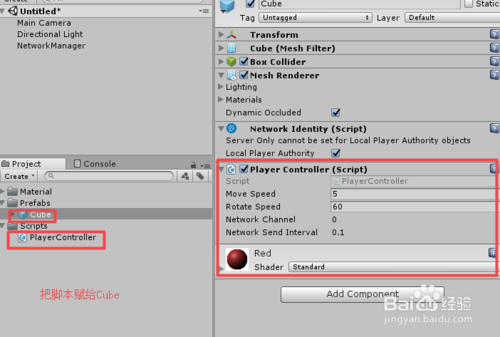
        transform.Rotate(Vector3.up \* rotateH \* rotateSpeed \* Time.deltaTime);

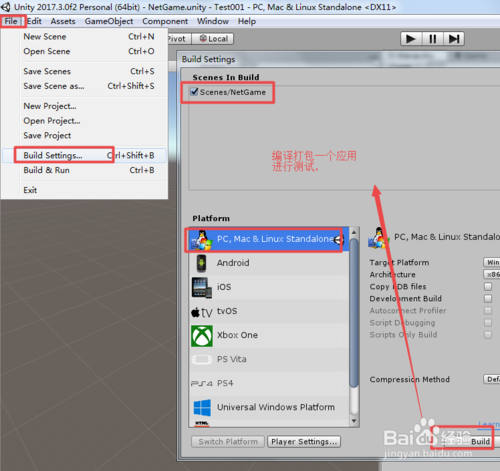
        transform.Translate(Vector3.forward \* moveV \* moveSpeed \* Time.deltaTime);

}

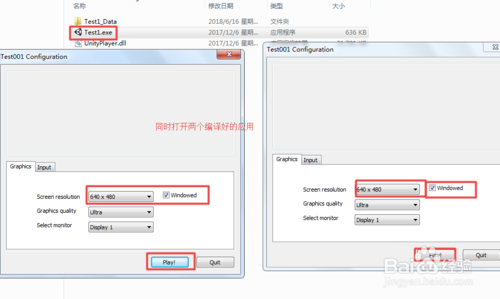
}

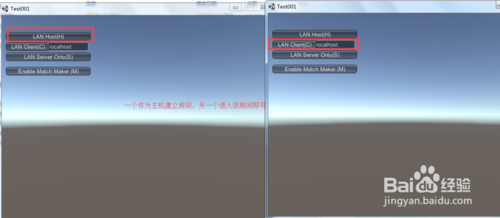
脚本编译正确，回到Unity，把脚本挂载到Cube上，然后保存成场景，Build打包编译一个应用exe测试，具体如下图

[](http://jingyan.baidu.com/album/08b6a59190835c14a809228e.html?picindex=4)

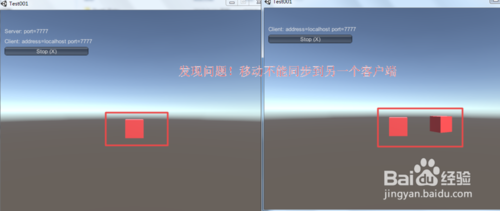
[](http://jingyan.baidu.com/album/08b6a59190835c14a809228e.html?picindex=5)

打开编好的应用exe两次，一次作为服务器端和客户端，另一次作为客户端

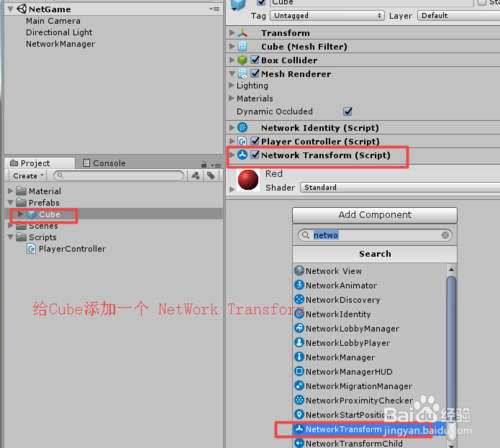
[](http://jingyan.baidu.com/album/08b6a59190835c14a809228e.html?picindex=6)

[](http://jingyan.baidu.com/album/08b6a59190835c14a809228e.html?picindex=7)

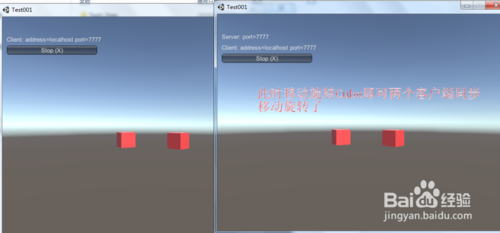
移动旋转Cube时发现另一个客户端不同步移动旋转，具体如下图

[](http://jingyan.baidu.com/album/08b6a59190835c14a809228e.html?picindex=8)

回到Unity界面，给Cube添加NetworkIdentity组件，具体如下图

[](http://jingyan.baidu.com/album/08b6a59190835c14a809228e.html?picindex=9)

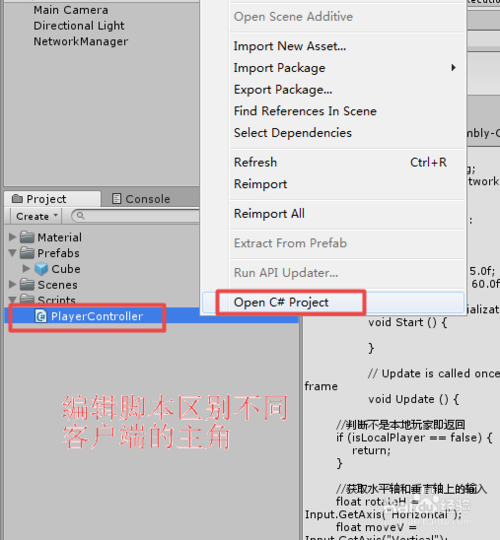
再次Build打包编译一个应用exe测试，此时移动旋转物体，各个客户端同步了，具体如下图

[](http://jingyan.baidu.com/album/08b6a59190835c14a809228e.html?picindex=10)

# Unity Network教程之三 Player颜色区别子弹同步

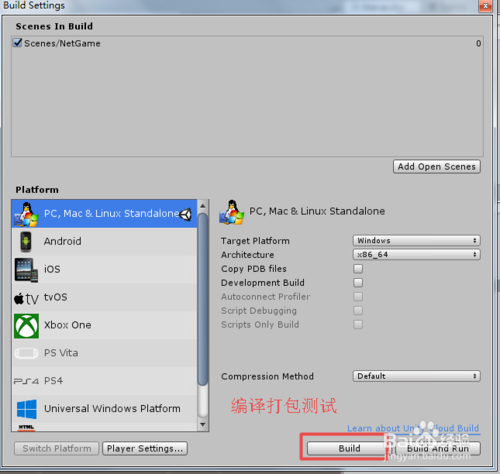
方法/步骤

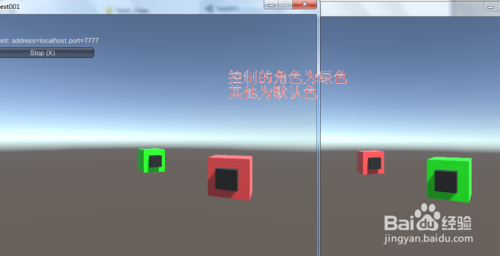
打开PlayerController脚本，进行编辑，控制客户端Player生成颜色的不同，代码和代码说明，具体如下图

[](http://jingyan.baidu.com/album/624e74590a014434e8ba5a19.html?picindex=2)

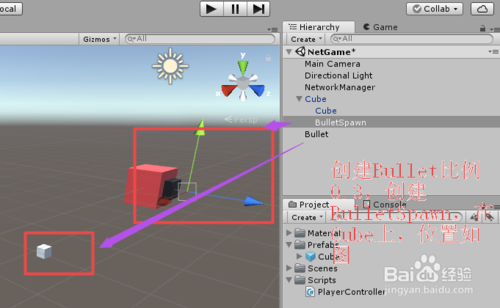
[](http://jingyan.baidu.com/album/624e74590a014434e8ba5a19.html?picindex=3)

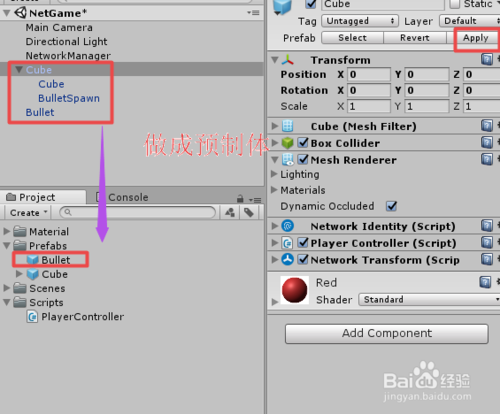
脚本编译正确，回到unity，编译打包，运行APP，发现客户端生成的Player已经有了颜色区别，便于玩家操作，具体如下图

[](http://jingyan.baidu.com/album/624e74590a014434e8ba5a19.html?picindex=4)

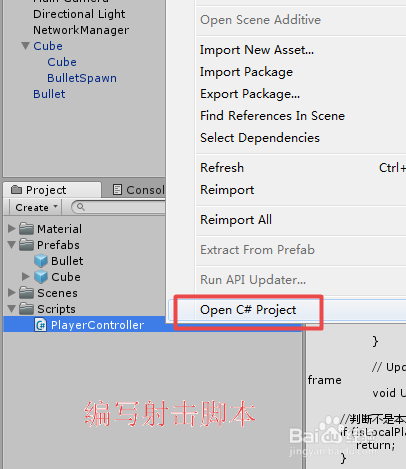
[](http://jingyan.baidu.com/album/624e74590a014434e8ba5a19.html?picindex=5)

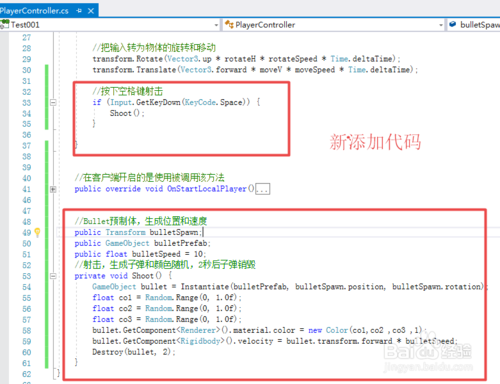
回到unity，在场景中添加一个Cube，修改比例0.3，作为Bullet，添加一个GameObject调整位置，作为BulletSpawn，并把它们做成预制体，具体如下图

[](http://jingyan.baidu.com/album/624e74590a014434e8ba5a19.html?picindex=6)

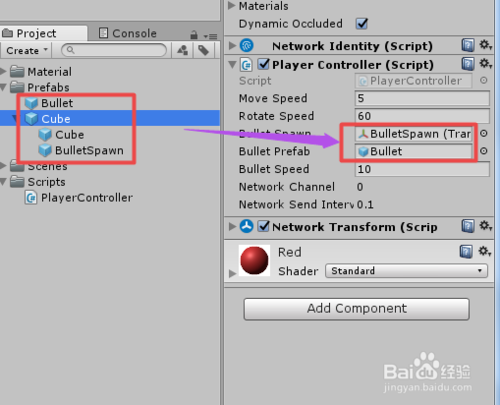
[](http://jingyan.baidu.com/album/624e74590a014434e8ba5a19.html?picindex=7)

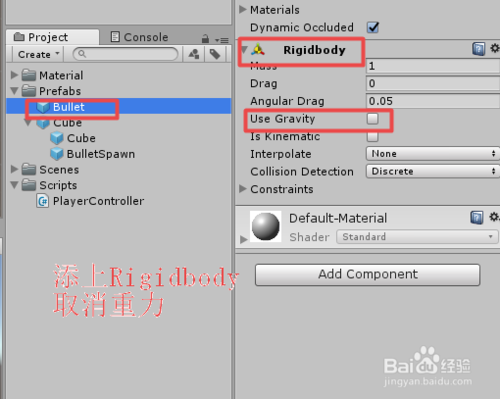
给PlayerController 添加射击代码，，具体的代码和代码说明如下图

[](http://jingyan.baidu.com/album/624e74590a014434e8ba5a19.html?picindex=8)

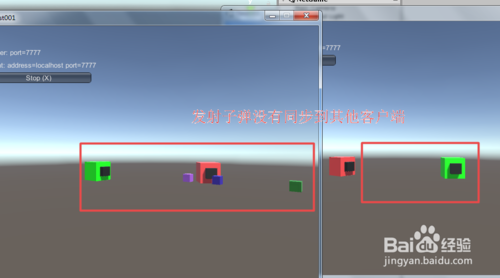
[](http://jingyan.baidu.com/album/624e74590a014434e8ba5a19.html?picindex=9)

脚本编译正确，回到Unity界面，给Cube添加上Rigidbody，并取消重力影响，然后给PlayerController把变量赋值，具体如下图

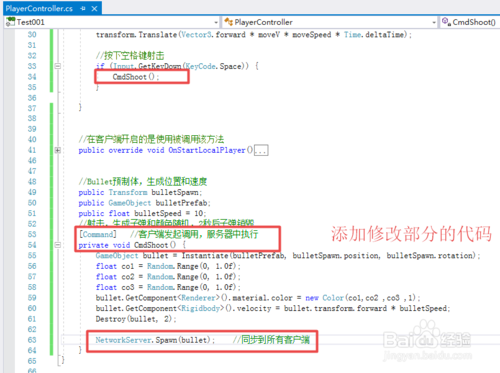
[](http://jingyan.baidu.com/album/624e74590a014434e8ba5a19.html?picindex=10)

[](http://jingyan.baidu.com/album/624e74590a014434e8ba5a19.html?picindex=11)

打包运行APP，发现子弹并不能同步到各个客户端，具体如下图

[](http://jingyan.baidu.com/album/624e74590a014434e8ba5a19.html?picindex=12)

回到PlayerController脚本，给Shoot()方法添加 Command，修改方法名为CmdShoot，具体代码和代码说明如下图

[](http://jingyan.baidu.com/album/624e74590a014434e8ba5a19.html?picindex=13)

PlayerController脚本具体内容如下：

using UnityEngine;

using UnityEngine.Networking;   //引入命名空间 UnityEngine.Networking

//继承NetworkBehaviour

public class PlayerController : NetworkBehaviour {

    //设置旋转和移动速度

    public float moveSpeed = 5.0f;

    public float rotateSpeed = 60.0f;

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

        //判断不是本地玩家即返回

        if (isLocalPlayer == false) {

            return;

        }

        //获取水平轴和垂直轴上的输入

        float rotateH = Input.GetAxis("Horizontal");

        float moveV = Input.GetAxis("Vertical");

        //把输入转为物体的旋转和移动

        transform.Rotate(Vector3.up \* rotateH \* rotateSpeed \* Time.deltaTime);

        transform.Translate(Vector3.forward \* moveV \* moveSpeed \* Time.deltaTime);

        //按下空格键射击

        if (Input.GetKeyDown(KeyCode.Space)) {

            CmdShoot();

        }

}

    //在客户端开启的是使用被调用该方法

    public override void OnStartLocalPlayer()

    {

        //角色创建的时候，把他的颜色设置为绿色,区别其他角色

        transform.GetComponent<Renderer>().material.color = Color.green;

    }

//Bullet预制体，生成位置和速度

    public Transform bulletSpawn;

    public GameObject bulletPrefab;

public float bulletSpeed = 10;

//注意：方法名都要为Cmd前缀

    //射击，生成子弹和颜色随机，2秒后子弹销毁

    [Command]   //客户端发起调用，服务器中执行

    private void CmdShoot() {

        GameObject bullet = Instantiate(bulletPrefab, bulletSpawn.position, bulletSpawn.rotation);

        float co1 = Random.Range(0, 1.0f);

        float co2 = Random.Range(0, 1.0f);

        float co3 = Random.Range(0, 1.0f);

        bullet.GetComponent<Renderer>().material.color = new Color(co1,co2 ,co3 ,1);

        bullet.GetComponent<Rigidbody>().velocity = bullet.transform.forward \* bulletSpeed;

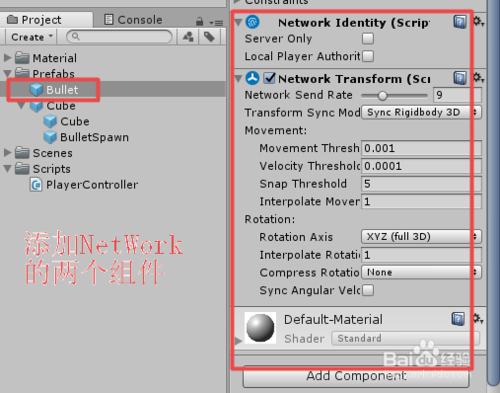
        Destroy(bullet, 2);

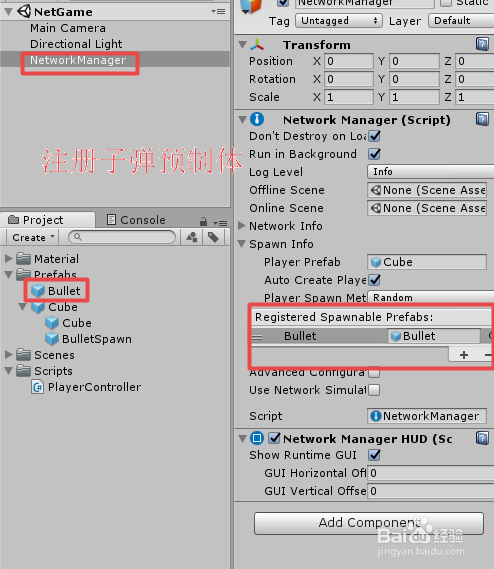
        NetworkServer.Spawn(bullet);    //同步到所有客户端

    }

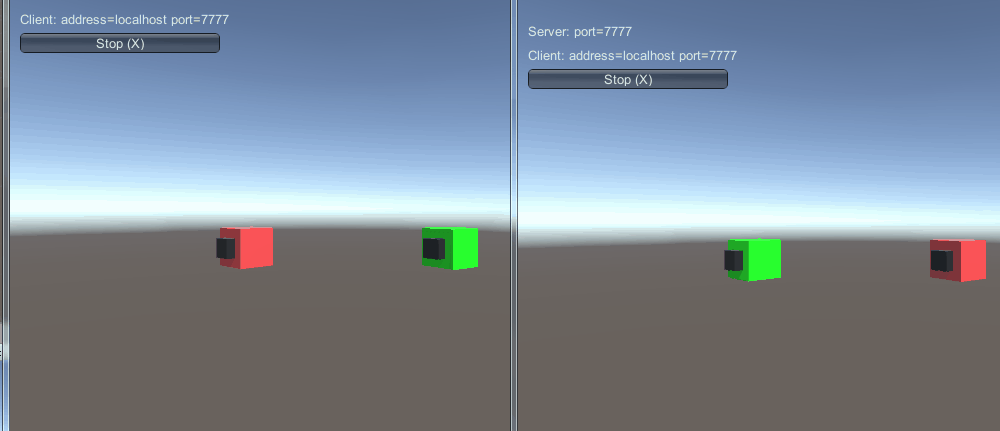
}

回到Unity，给Bullet添加NetWork两个组件，具体如下，并且注册Bullet预制体到NetworkManager，具体如下图

[](http://jingyan.baidu.com/album/624e74590a014434e8ba5a19.html?picindex=14)

[](http://jingyan.baidu.com/album/624e74590a014434e8ba5a19.html?picindex=15)

编译打包，运行APP，子弹得意同步到各个客户端，具体如下图

[](http://jingyan.baidu.com/album/624e74590a014434e8ba5a19.html?picindex=16)

# NetWorkBehaviour的常用特性

[SyncVar] 用于标识序列化变量,实现变量同步 例: (把Hp标识,就可以实现同步减血)

[Client] 表示只能在客户端调用

[ClientCallBack] 表示客户端执行的回调

[Command]表示客户端向服务端发送的命令,在服务端执行,注意方法名都要为Cmd作为前缀

[ClientPrc] 表示服务端向客户端发送的命令,在客户端执行

## 变量同步例子：

如变量的同步(血量)

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.Networking;

using UnityEngine.UI;

public class ControlMove :NetworkBehaviour {

[SyncVar]

float hp = 100;

private Slider slider;

//Bullet预制体，生成位置和速度

void Start () {

slider = GameObject.Find("Slider").GetComponent<Slider>();

}

// Update is called once per frame

void Update () {

if(!isLocalPlayer){

return;

}

slider.value = hp / 100f;

if(Input.GetKeyDown(KeyCode.A)){

CmdReduceHP();

}

}

//服务器端执行的方法，方法名的前缀都要为Cmd开头

[Command]

private void CmdReduceHP() {

hp -= 10;

}

}