在上一课的内容中，我们添加了武器开火的粒子特效。本课的内容相对比较简单，在这一课的内容中，我们将给所添加的粒子特效设置对应的脚本。

打开Unity，在Project视图中找到\Assets_Scripts\ShootEnemy.cs脚本，并在MonoDevelop中将其打开。

更改其中的代码如下：

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
//import namespace
using UnityEngine.UI;

public class ShootEnemy : MonoBehaviour {

	//创建到Button对象的引用
	public Button shootBtn;
	//创建到主摄像机的引用
	public Camera fpsCam;

	//设置敌人每次受到伤害的数值
	public float damage = 10f;

	//敌人受伤的粒子特效
	public GameObject bloodEffect;

	//攻击的粒子特效

	public GameObject shootingEffect;

	//添加的攻击力度
	public int forceAdd = 300;

	//定义两个音源对象
	AudioSource shootSound;
	AudioSource reloadSound;


	//创建到弹药UI元素的引用
	public Text ammo1Text;
	public Text ammo2Text;
	public int ammo1;
```

```
public int ammo2;

//判断弹药是否已空
private bool ammoIsEmpty;

//1.创建到开火粒子系统的引用
public ParticleSystem muzzleFlash;

// Use this for initialization
void Start () {


     //Debug.Log ("Activated!");
     //添加按钮的响应事件
     shootBtn.onClick.AddListener (OnShoot);

     //获取音源组件
     AudioSource[] audios = GetComponents<AudioSource>();

     //设置音源
     shootSound = audios [0];
     reloadSound = audios [1];

     //设置弹药数量的初始值
     ammo1 = 20;
     ammo2 = 100;
}

public void OnShoot(){

     //仅在ammoIsEmpty为真时才可执行逻辑判断中的操作
     if (!ammoIsEmpty) {

          if (ammo1 == 1) {

               ammo1 = 21;
          }



          //弹药数量减少
          ammo1 -= 1;
          string ammo1String = (ammo1).ToString ();
          ammo1Text.text = ammo1String;
```

```
                ammo2 -= 1;
                string ammo2String = (ammo2).ToString ();
                ammo2Text.text = "/"+ ammo2String;


                //如果弹药总数量为0，则设置ammoIsEmpty为true
                if (ammo2 == 0) {

                     ammoIsEmpty = true;
                     ammo1 = 0;
                     string ammoTempString = (ammo1).ToString ();
                     ammo1Text.text = ammoTempString;

                }


                //播放音效
                shootSound.Play();

                Debug.Log ("shooting!");

                //定义一个RaycastHit类型变量，用于保存检测信息
                RaycastHit hit;

                //判断是否检测到命中敌人
                if (Physics.Raycast (fpsCam.transform.position,
fpsCam.transform.forward, out hit)) {


                     //获取所受攻击的敌人
                     Enemy target =
hit.transform.GetComponent<Enemy>();

                     //destroy enemy

                     if (target != null) {

                          //instantiate blood effect

                          target.TakeDamage (damage);
                          //创建敌人受伤的粒子特效

                          GameObject bloodBurst = Instantiate
(bloodEffect, hit.point, Quaternion.LookRotation (hit.normal));
```

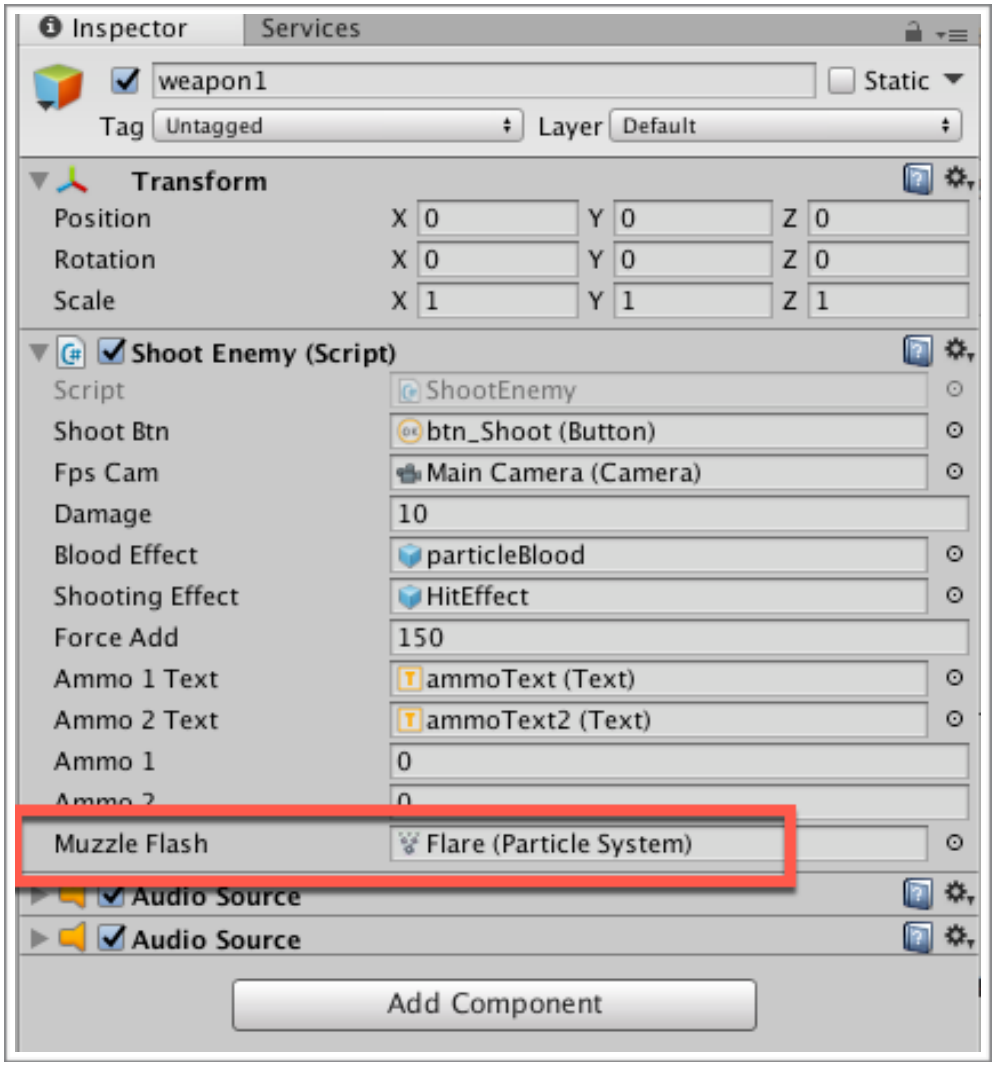
```
                    //0.2秒后销毁粒子特效

                    Destroy (bloodBurst, 0.2f);
                }  else {
                    //load shooting effect

                    //如果没有击中敌人，则创建攻击时的粒子特效

                    GameObject shootingGo =  Instantiate
(shootingEffect, hit.point, Quaternion.LookRotation
(hit.normal));

                    //0.2秒后销毁粒子特效
                    Destroy (shootingGo,0.2f);

                }


                //攻击敌人时添加一个额外的冲击力

                if (hit.rigidbody != null) {

                    hit.rigidbody.AddForce (-hit.normal *
forceAdd);

                }


                //输出所命中的对象名称
                Debug.Log (hit.transform.name);
            }
            //2.播放开火的粒子特效
            muzzleFlash.Play ();

        }

    }

}
```

以上我们只添加了两行代码，按照注释行的数字编号来解释一下：

1.创建了到开火粒子系统的引用

2.在满足逻辑条件的前提下播放开火的粒子特效。

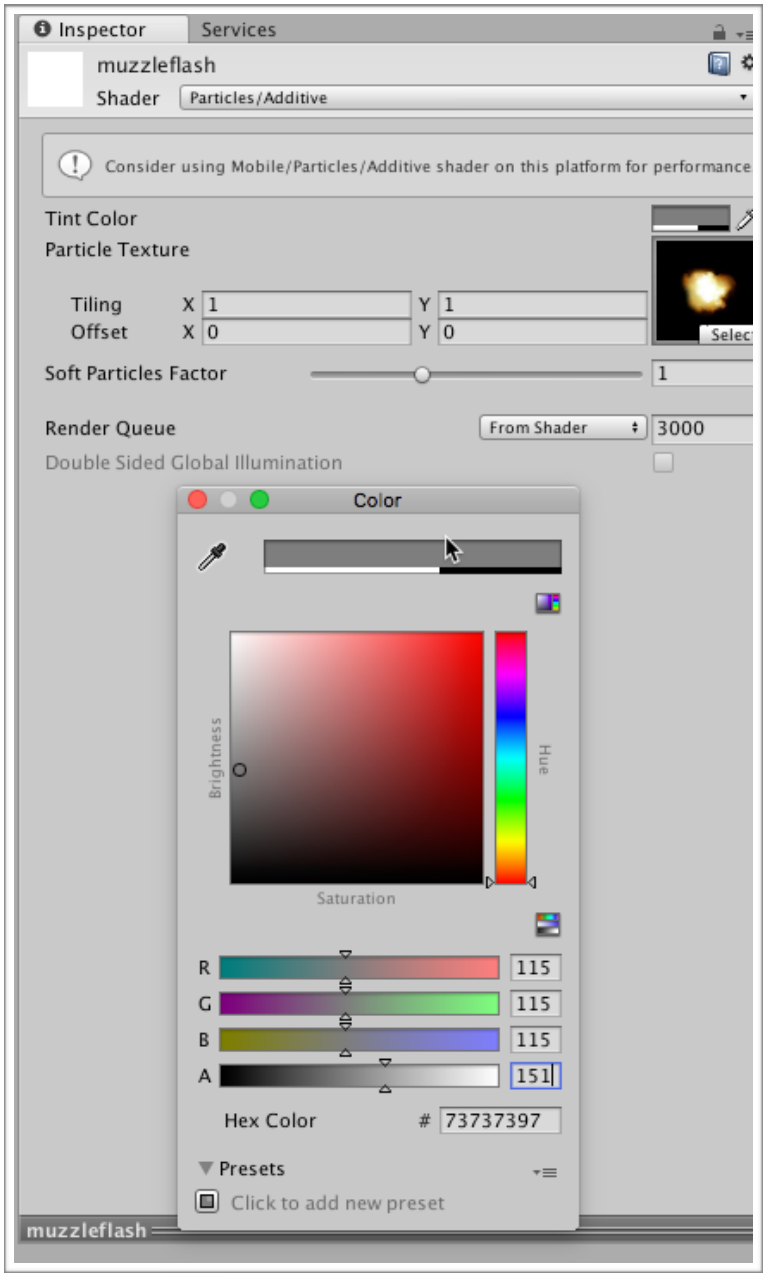
接下来回到Unity编辑器，在Hierarchy视图中选中CameraParent下Main Camera的子对象weapon1，然后在Inspector视图中，将Shoot Enemy组件中的Muzzle Flash属性设置为Flare，如图所示。



点击工具栏上的Play按钮，预览一下游戏的运行效果。

感觉开火的颜色似乎有点不满意，让我们来调整一下。

在Project视图中找到WeaponPack文件夹中的muzzleFlash材质，然后在Inspector视图中点击Tint Color旁边的色彩拾取器，并适当调整一下其中的色彩。



大概调整到类似上图的程度就好了。当然，具体如何调整其实是比较主观的。

调整完成后，点击Unity编辑器工具栏上的Play按钮，观察下效果。

好了，这下似乎观感要好一点。

本课的内容就这么点，惊不惊喜，意不意外？~让我们下一课再见。