

2011-05-08

# 使用 Unity 3D 进行游戏开发 从入门到精通 (第六章)

一颗子弹引发的(javascript)

www.docin.com

# 目录

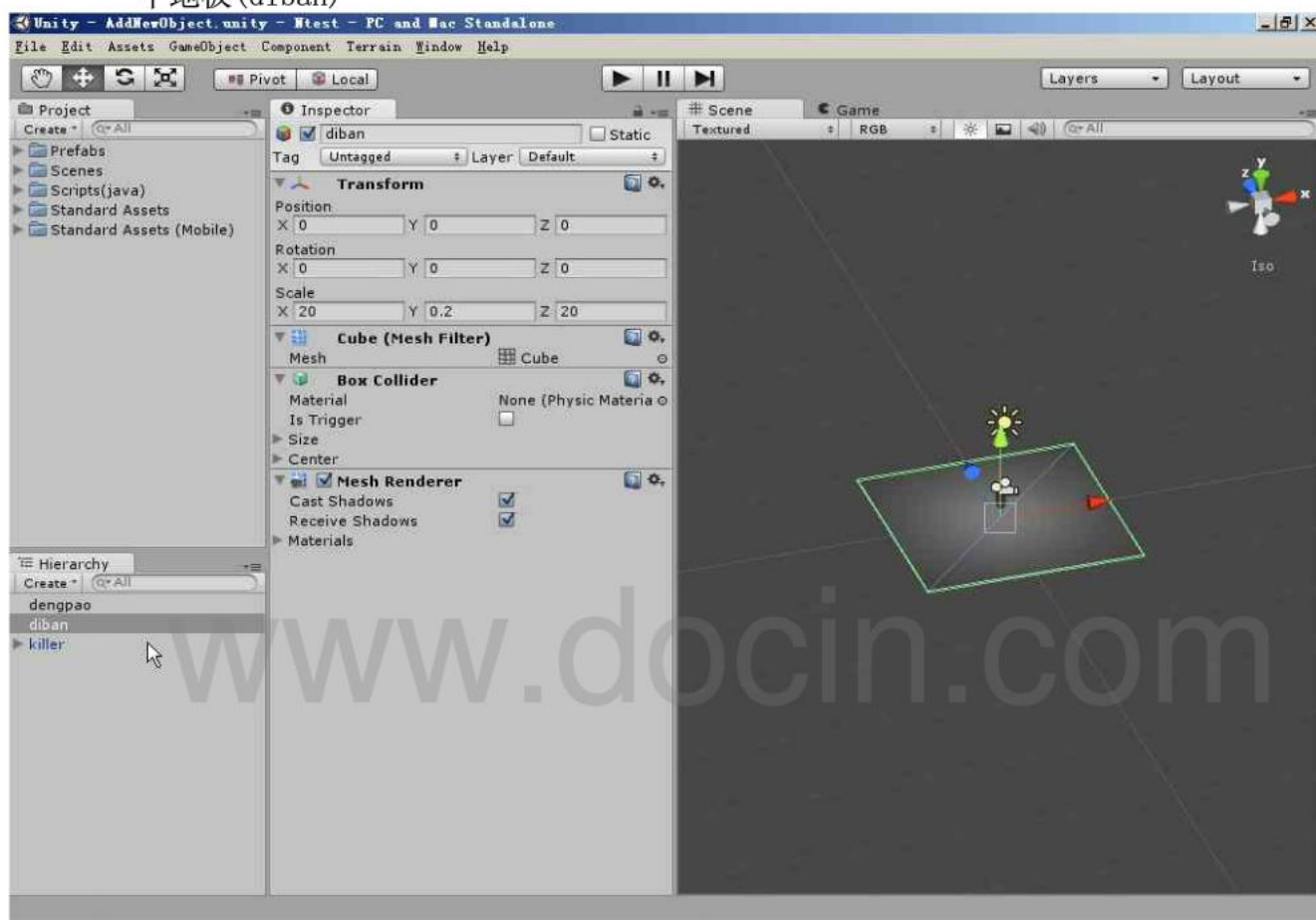


# 场景说明

第一人称视角，当用户按下鼠标左键或输入配置中设定的开火/攻击键时，在当前视角新增一个预设好的 prefab，让它沿着当前视角以指定的速度前进，并在运动指定的距离后自动销毁。

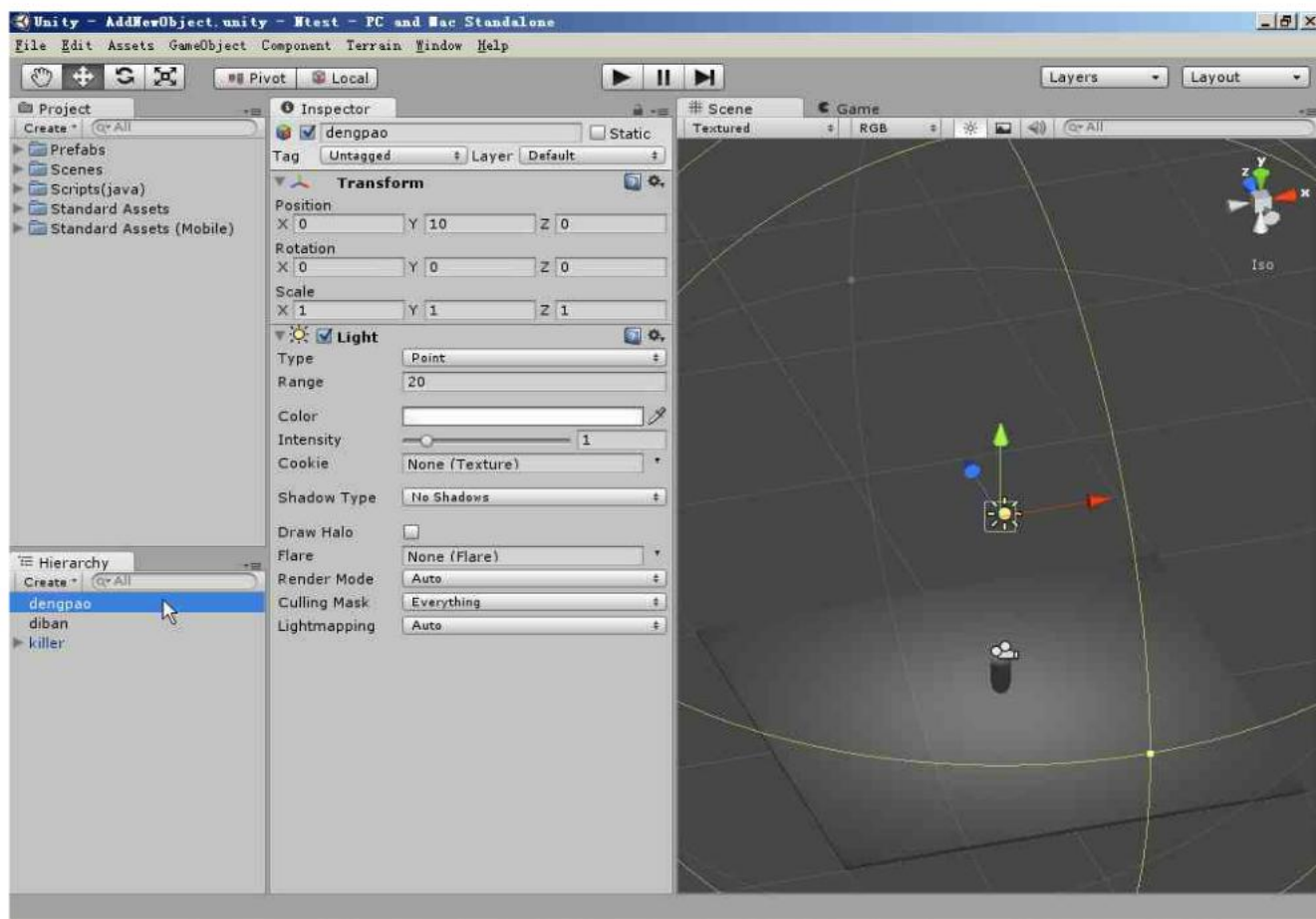
## 基本场景创建

一个地板(diban)



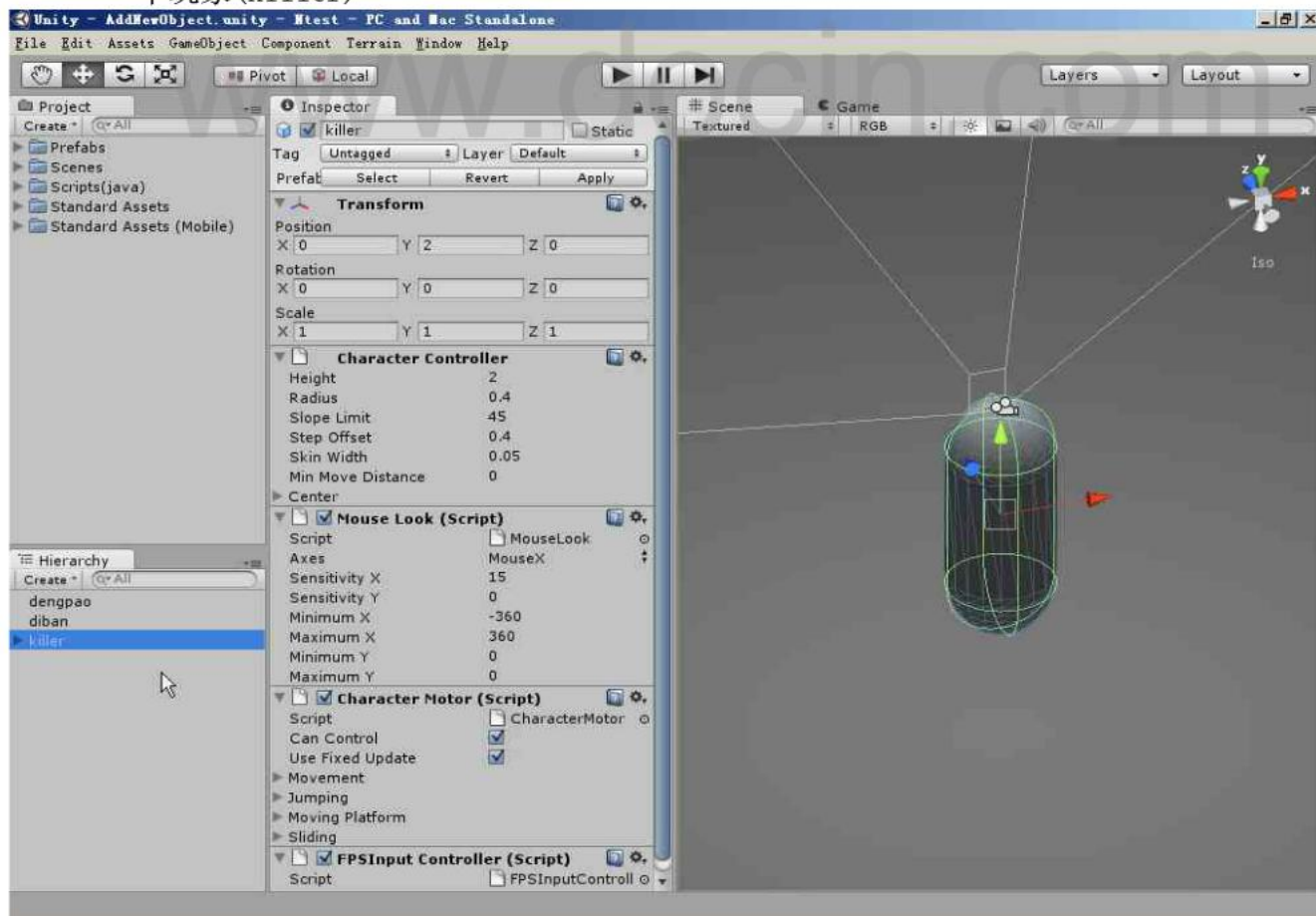
其实就是一个 cube，长宽各 20，厚度 0.2。

一个灯泡 (dengpao)



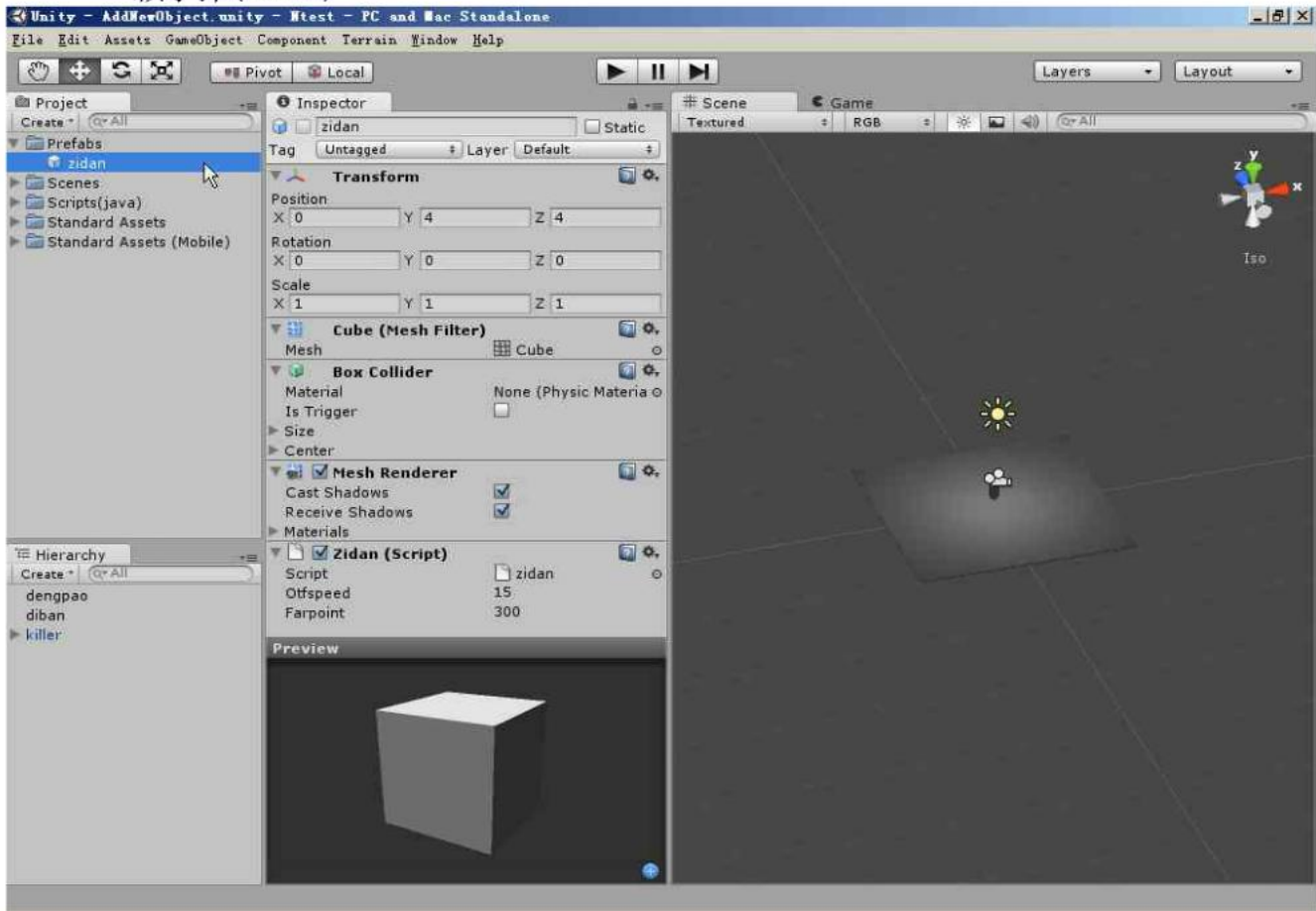
就是一个 point light，调整它的光照范围为 20。

一个玩家(killer)



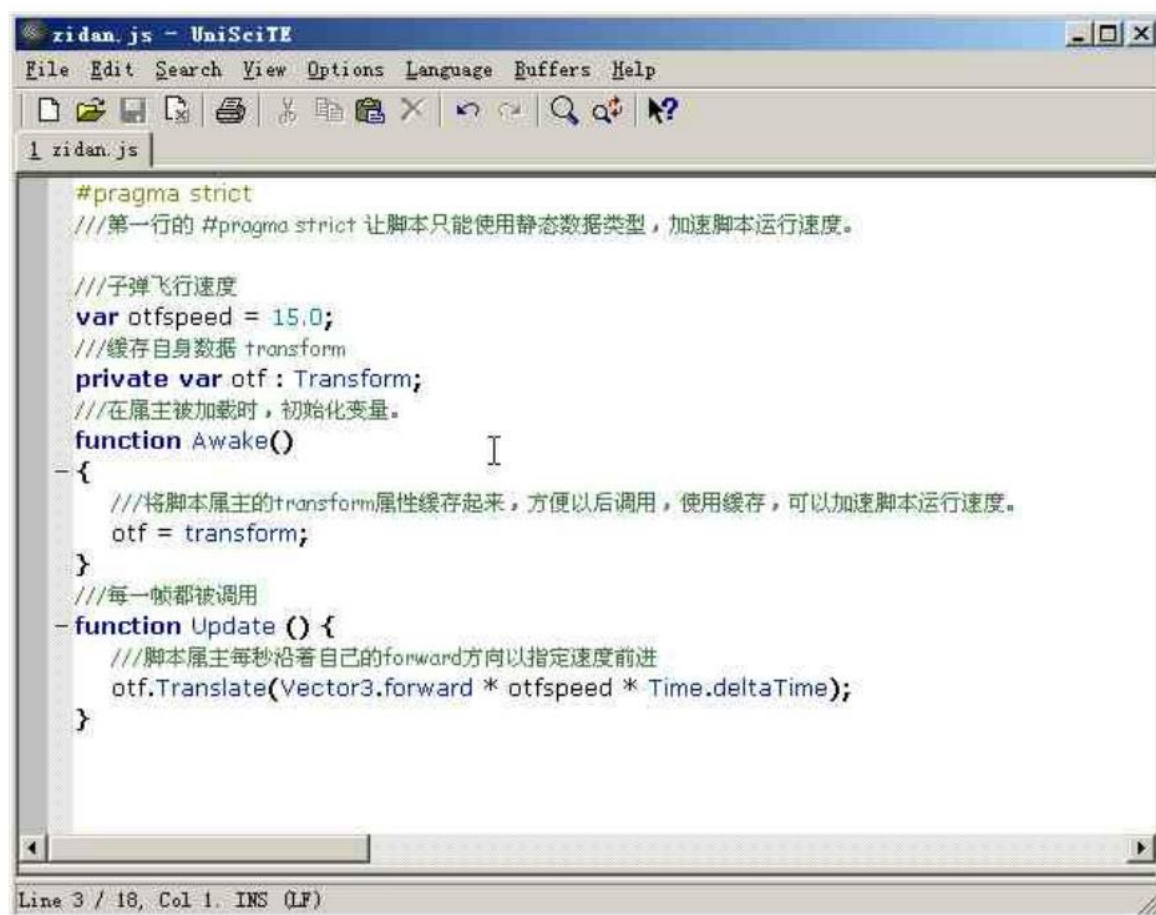
就是一个 unity 预置的 prefab，名称是 First Person Controller 。

一颗子弹(zidan)



就是一个 cube，自定义成一个 prefab，名称是子弹(zidan)。

子弹脚本(zidan.js)



```
zidan.js - UniSciTE
File Edit Search View Options Language Buffers Help
1 zidan.js

#pragma strict
///第一行的 #pragma strict 让脚本只能使用静态数据类型，加速脚本运行速度。

///子弹飞行速度
var ofspeed = 15.0;
///缓存自身数据 transform
private var otf : Transform;
///在属主被加载时，初始化变量。
function Awake()
- {
    ///将脚本属主的transform属性缓存起来，方便以后调用，使用缓存，可以加速脚本运行速度。
    otf = transform;
}
///每一帧都被调用
- function Update () {
    ///脚本属主每秒沿着自己的forward方向以指定速度前进
    otf.Translate(Vector3.forward * ofspeed * Time.deltaTime);
}

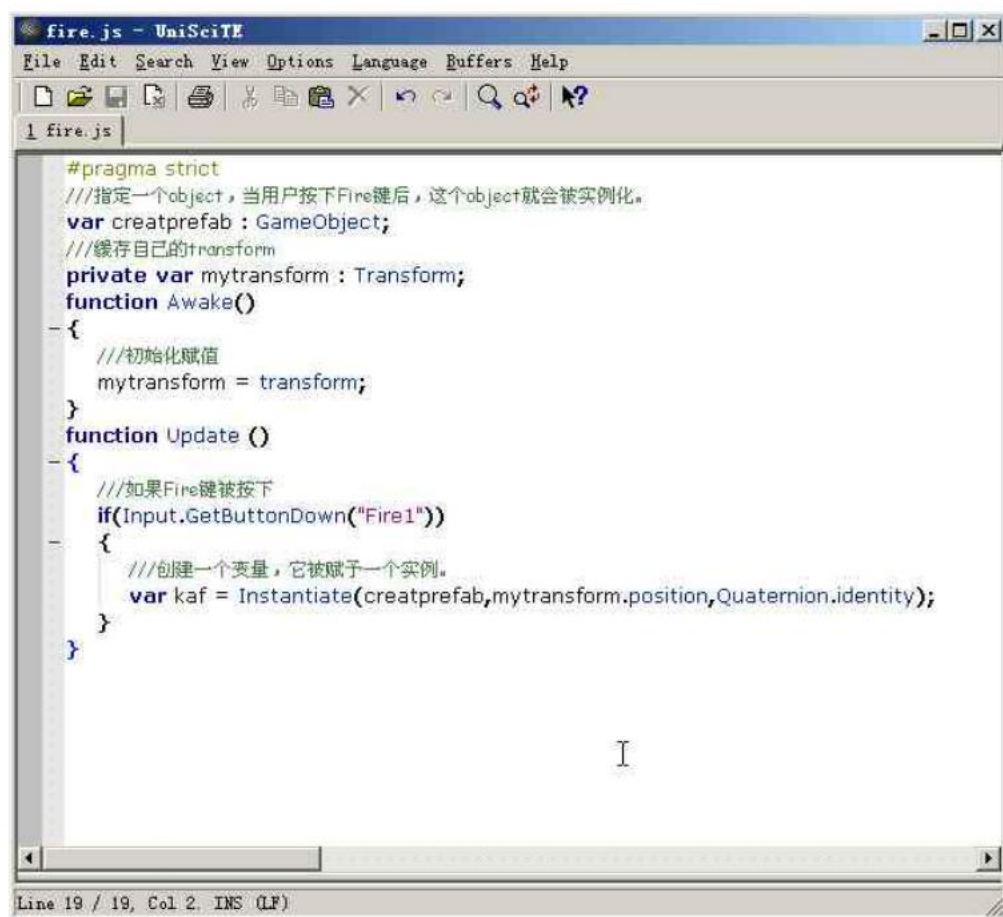
Line 3 / 18, Col 1, INS (LF)
```

这是子弹自身的属性脚本，它是被赋予 zidan 这个 prefab 的，当 zidan 这个 prefab 被实例化以后，实例在被加载时就会执行 Awake 函数，完成初始化及渲染后，会调用 Update 函数。

开火脚本(fire1.js)

www.docin.com





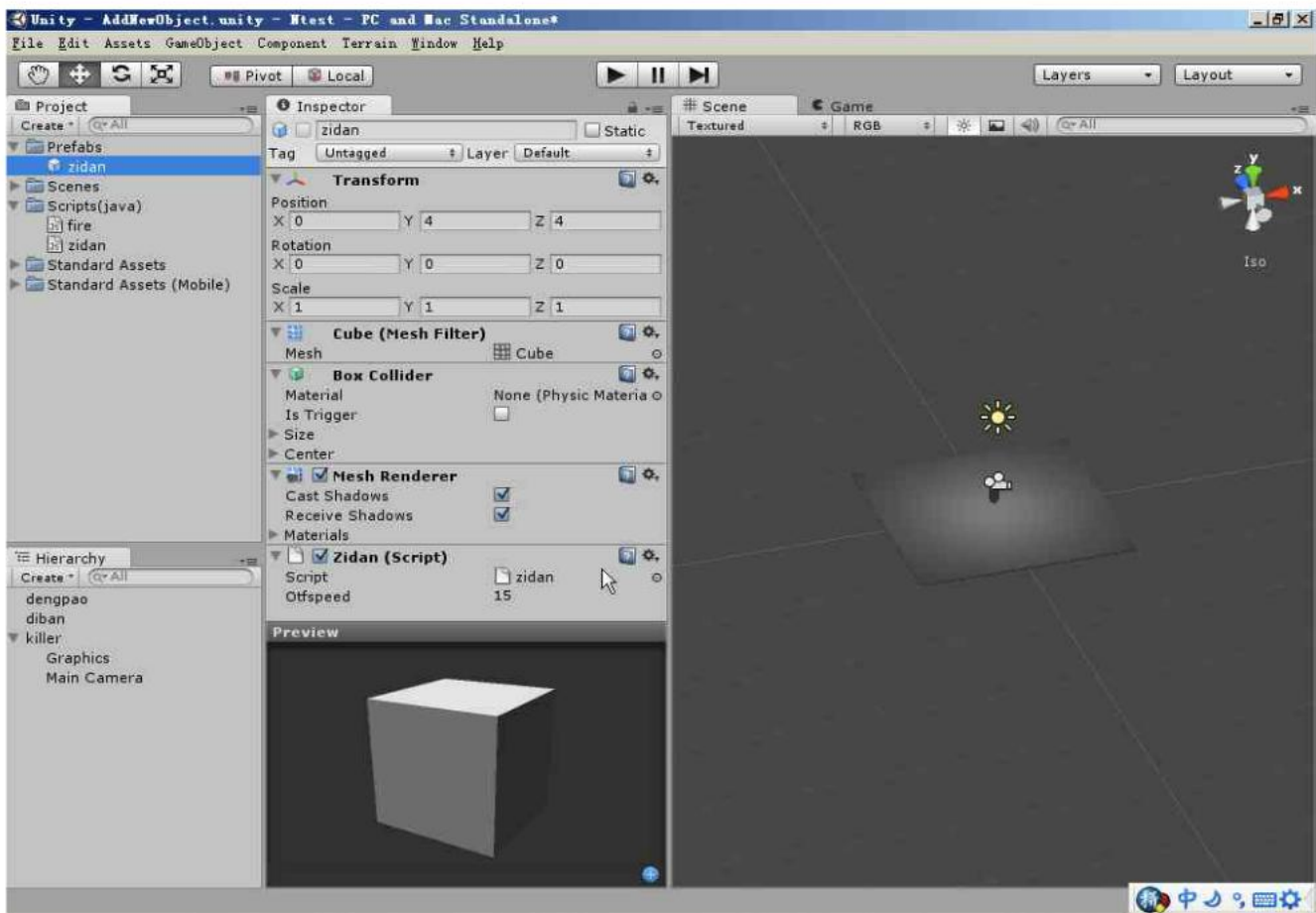
```
fire.js - UniScript
File Edit Search View Options Language Buffers Help
1 fire.js

#pragma strict
///指定一个object，当用户按下Fire键后，这个object就会被实例化。
var creatprefab : GameObject;
///缓存自己的transform
private var mytransform : Transform;
function Awake()
- {
    ///初始化赋值
    mytransform = transform;
}
function Update ()
- {
    ///如果Fire键被按下
    if(Input.GetButtonDown("Fire1"))
    - {
        ///创建一个变量，它被赋予一个实例。
        var kaf = Instantiate(creatprefab,mytransform.position,Quaternion.identity);
    }
}
```

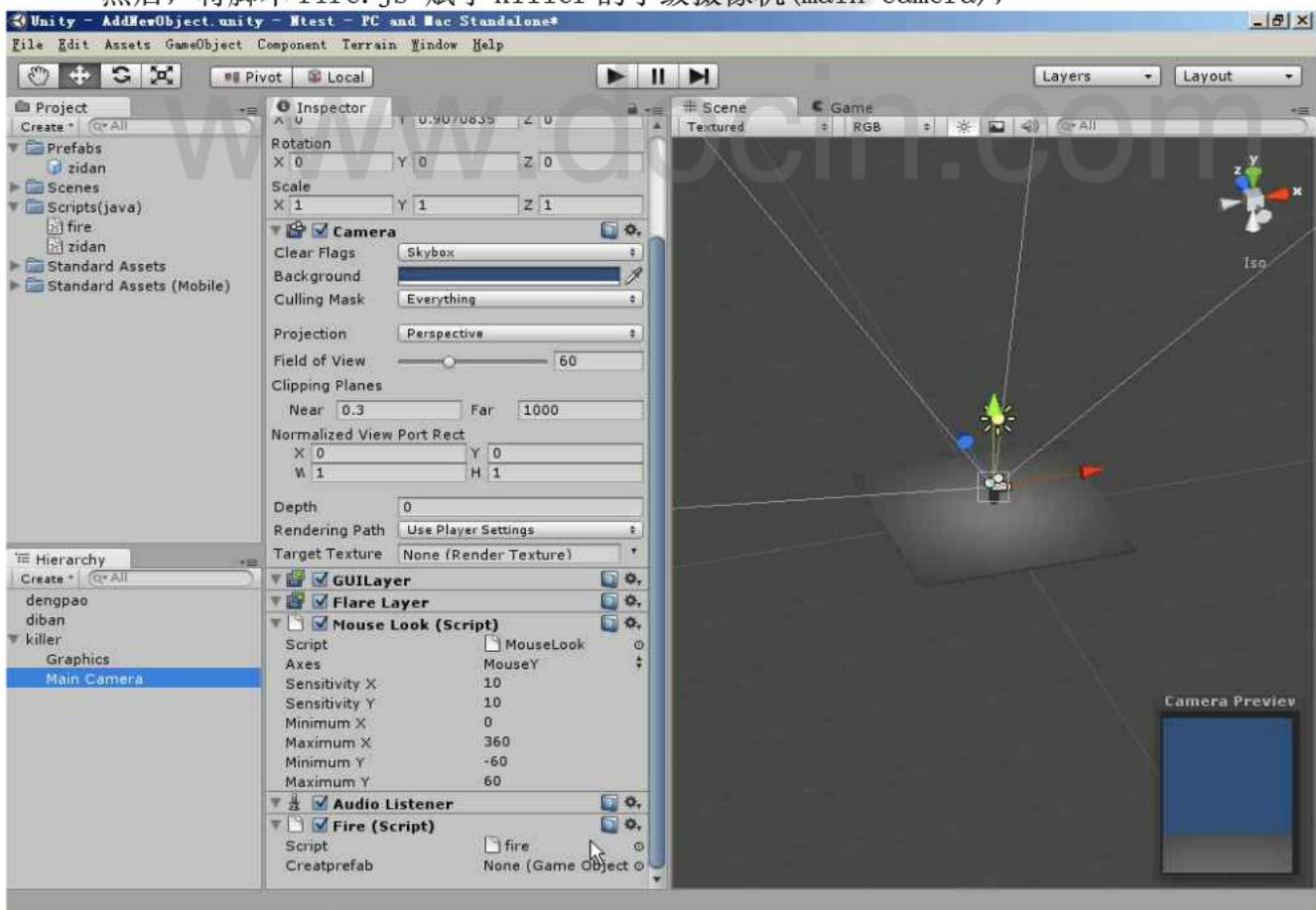
Line 19 / 19, Col 2, INS (LF)

## 组装

ok，现在将脚本 zidan.js 赋予名为 zidan 的 prefab。

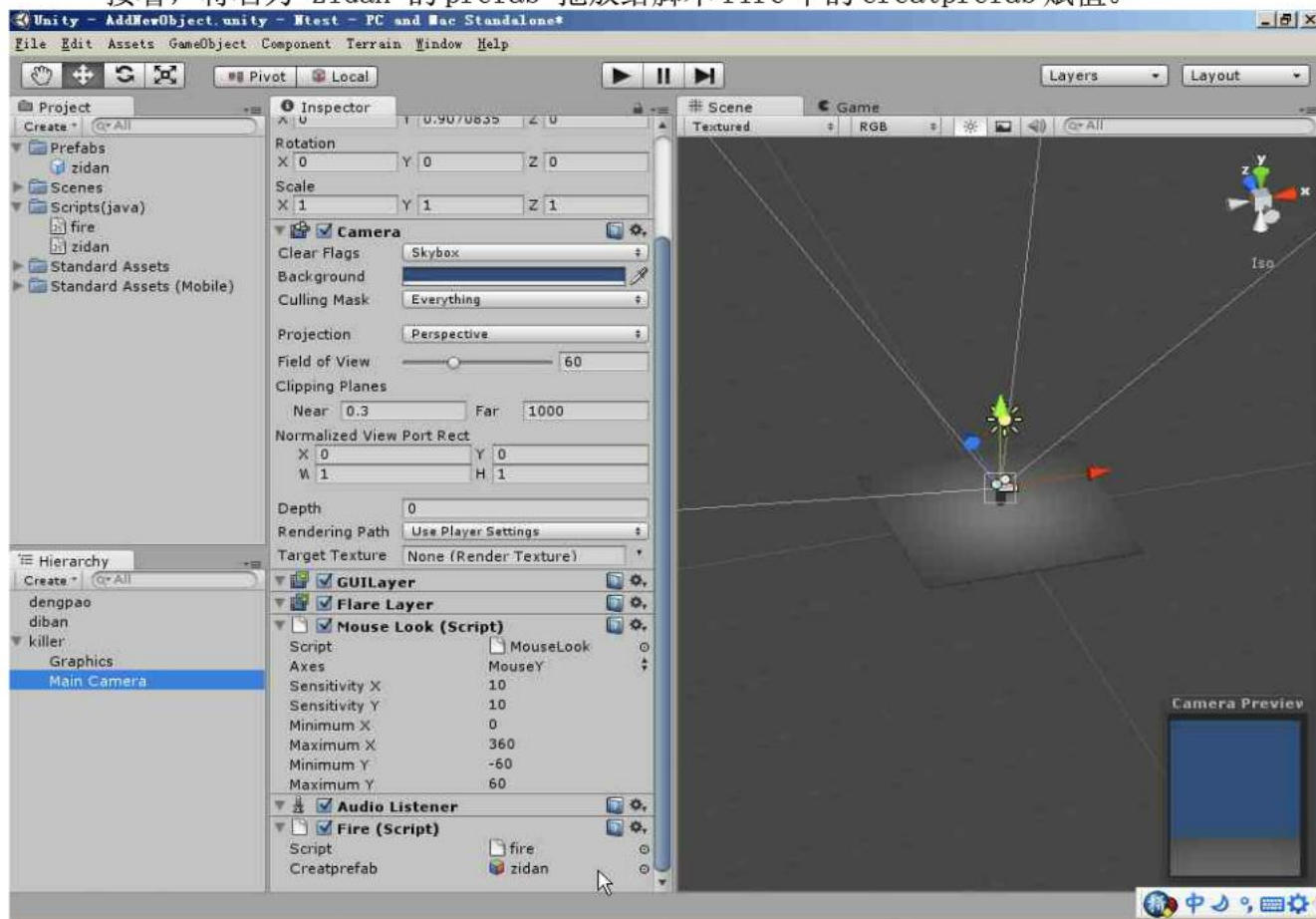


然后，将脚本 fire.js 赋予killer 的子级摄像机(main camera)，





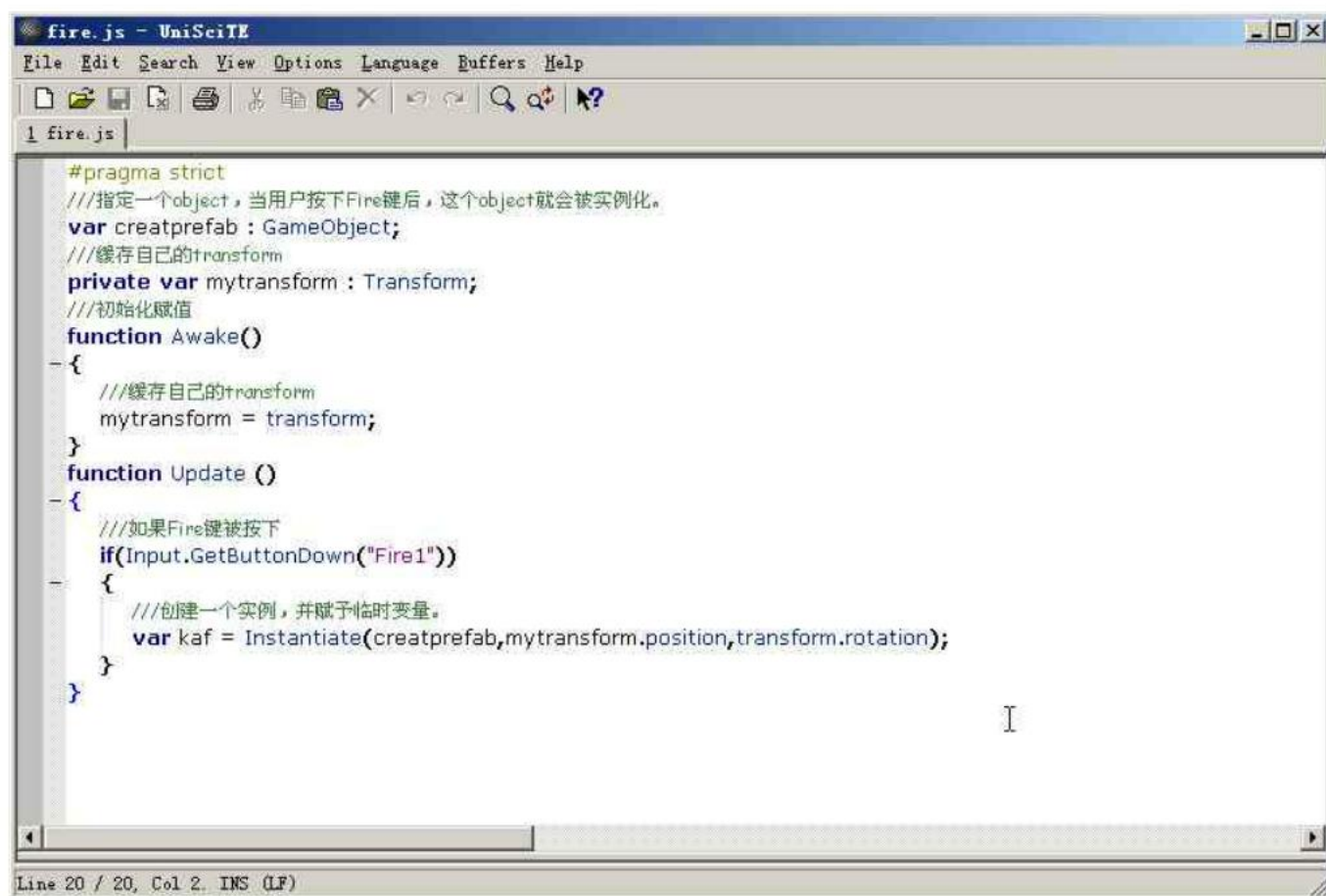
接着，将名为 zidan 的 prefab 拖放给脚本 fire 中的 creatprefab 赋值。



现在，就可以运行游戏看下效果。ok，继续。

## 让枪口随着视野来

在上面，子弹不管是在哪个位置被创建出来，都是朝着一个方向前进，这说明子弹的 `transform.rotation` 数据没有按我想象的被设置。需要让子弹拥有和摄像机相同的 `rotation`。



```
#pragma strict
///指定一个object，当用户按下Fire键后，这个object就会被实例化。
var creatprefab : GameObject;
///缓存自己的transform
private var mytransform : Transform;
///初始化赋值
function Awake()
- {
    ///缓存自己的transform
    mytransform = transform;
}
function Update ()
- {
    ///如果Fire键被按下
    if(Input.GetButtonDown("Fire1"))
    - {
        ///创建一个实例，并赋予临时变量。
        var kaf = Instantiate(creatprefab,mytransform.position,transform.rotation);
    }
}
```

Line 20 / 20, Col 2. INS (LF)

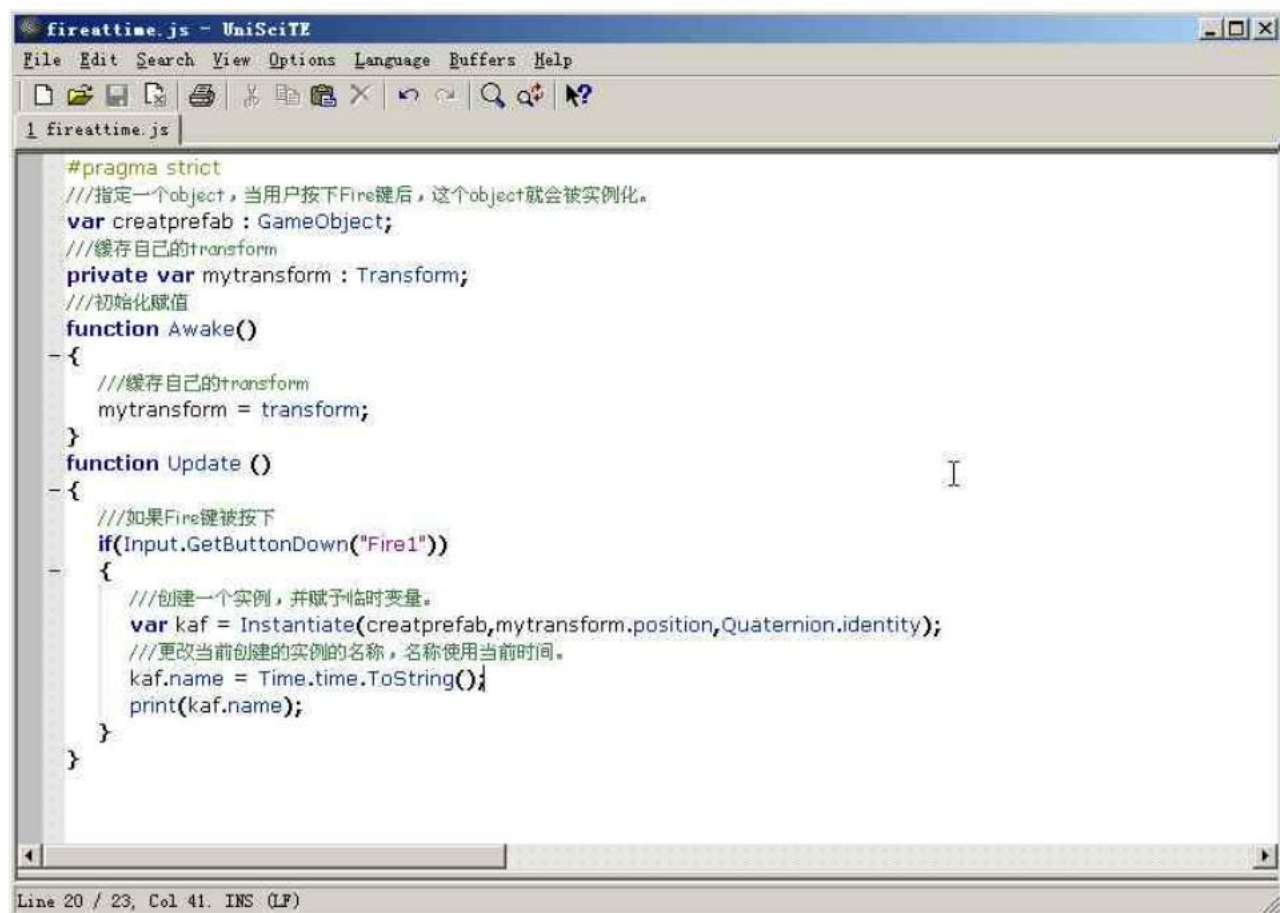
运行游戏实验一下吧。

## 记住自己的子弹

在运行游戏时，所有的实例化的子弹都用的同样的名字。能否将它们区分开？

用名称，名称可以使用当前的时间。

新建一个脚本 fireatime.js ，用它替换掉原来的 fire.js 脚本。

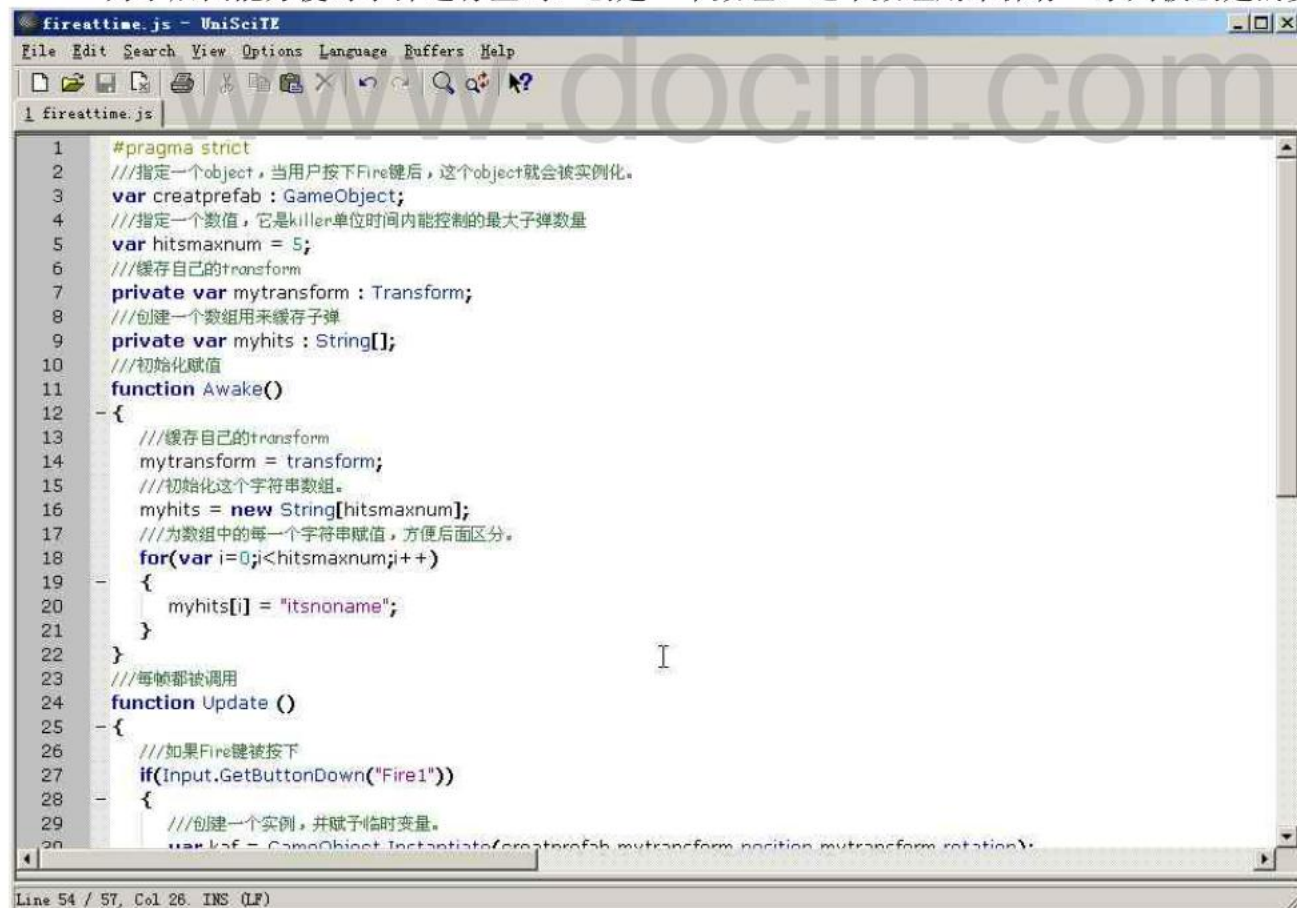


```
#pragma strict
///指定一个object，当用户按下Fire键后，这个object就会被实例化。
var creatprefab : GameObject;
///缓存自己的transform
private var mytransform : Transform;
///初始化赋值
function Awake()
- {
    ///缓存自己的transform
    mytransform = transform;
}
function Update ()
- {
    ///如果Fire键被按下
    if(Input.GetButtonDown("Fire1"))
    - {
        ///创建一个实例，并赋予临时变量。
        var kaf = Instantiate(creatprefab,mytransform.position,Quaternion.identity);
        ///更改当前创建的实例的名称，名称使用当前时间。
        kaf.name = Time.time.ToString();
        print(kaf.name);
    }
}
- }
```

Line 20 / 23, Col 41. INS (LF)

这样，在HierarchyView中就不再是清一色的同样的名字了。

为了后面能方便对子弹进行查询，创建一个数组，这个数组用来保存一系列被创建的实例的名。

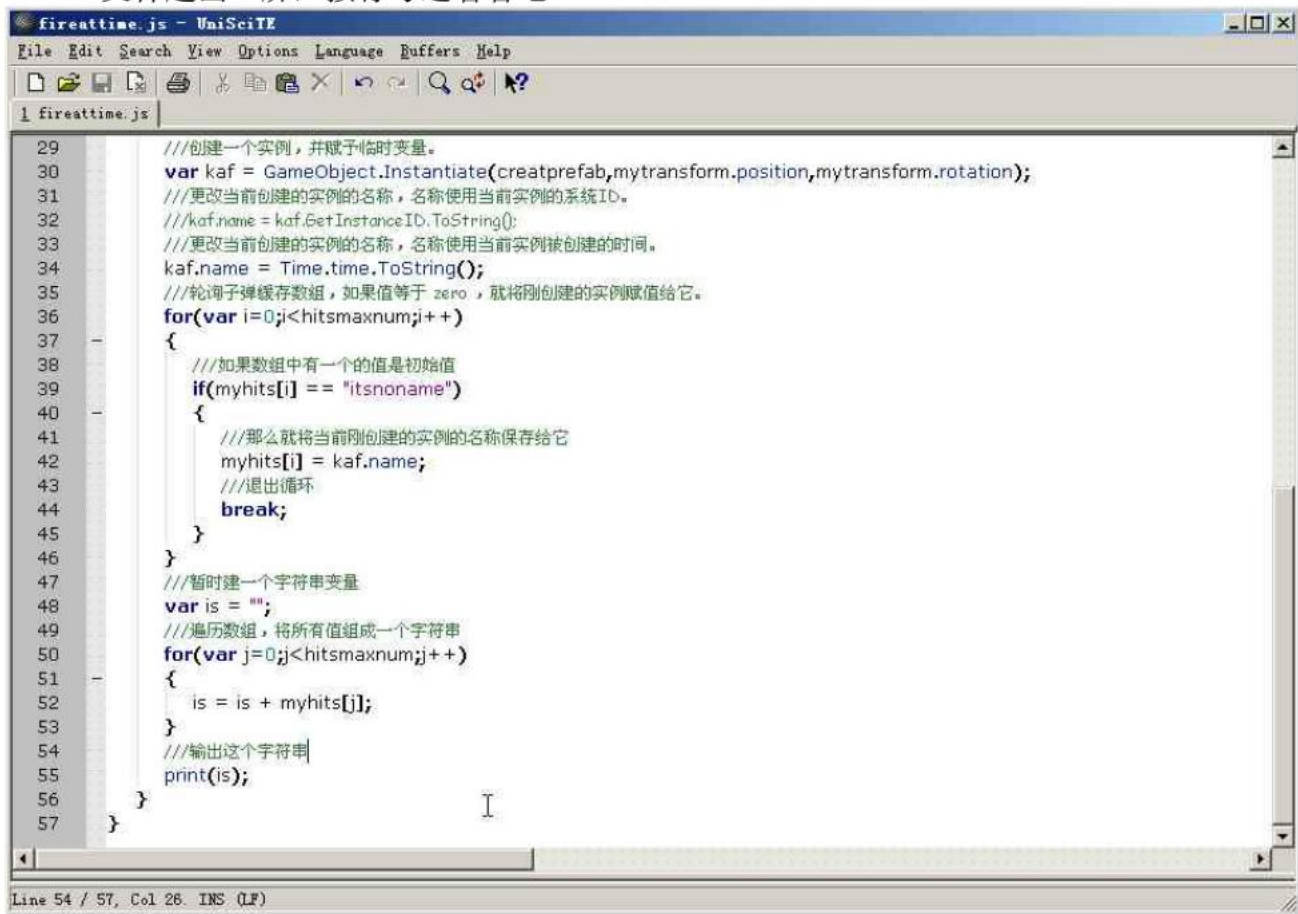


```
1 #pragma strict
2 ///指定一个object，当用户按下Fire键后，这个object就会被实例化。
3 var creatprefab : GameObject;
4 ///指定一个数值，它是killen单位时间内能控制的最大子弹数量
5 var hitsmaxnum = 5;
6 ///缓存自己的transform
7 private var mytransform : Transform;
8 ///创建一个数组用来缓存子弹
9 private var myhits : String[];
10 ///初始化赋值
11 function Awake()
12 - {
13     ///缓存自己的transform
14     mytransform = transform;
15     ///初始化这个字符串数组。
16     myhits = new String[hitsmaxnum];
17     ///为数组中的每一个字符串赋值，方便后面区分。
18     for(var i=0;i<hitsmaxnum;i++)
19     - {
20         myhits[i] = "itsnoname";
21     }
22 }
23 ///每帧都被调用
24 function Update ()
25 - {
26     ///如果Fire键被按下
27     if(Input.GetButtonDown("Fire1"))
28     - {
29         ///创建一个实例，并赋予临时变量。
30         var kaf = GameObject.Instantiate(creatprefab,mytransform.position,mytransform.rotation);
31     }
32 }
33 - }
```

Line 54 / 57, Col 26. INS (LF)



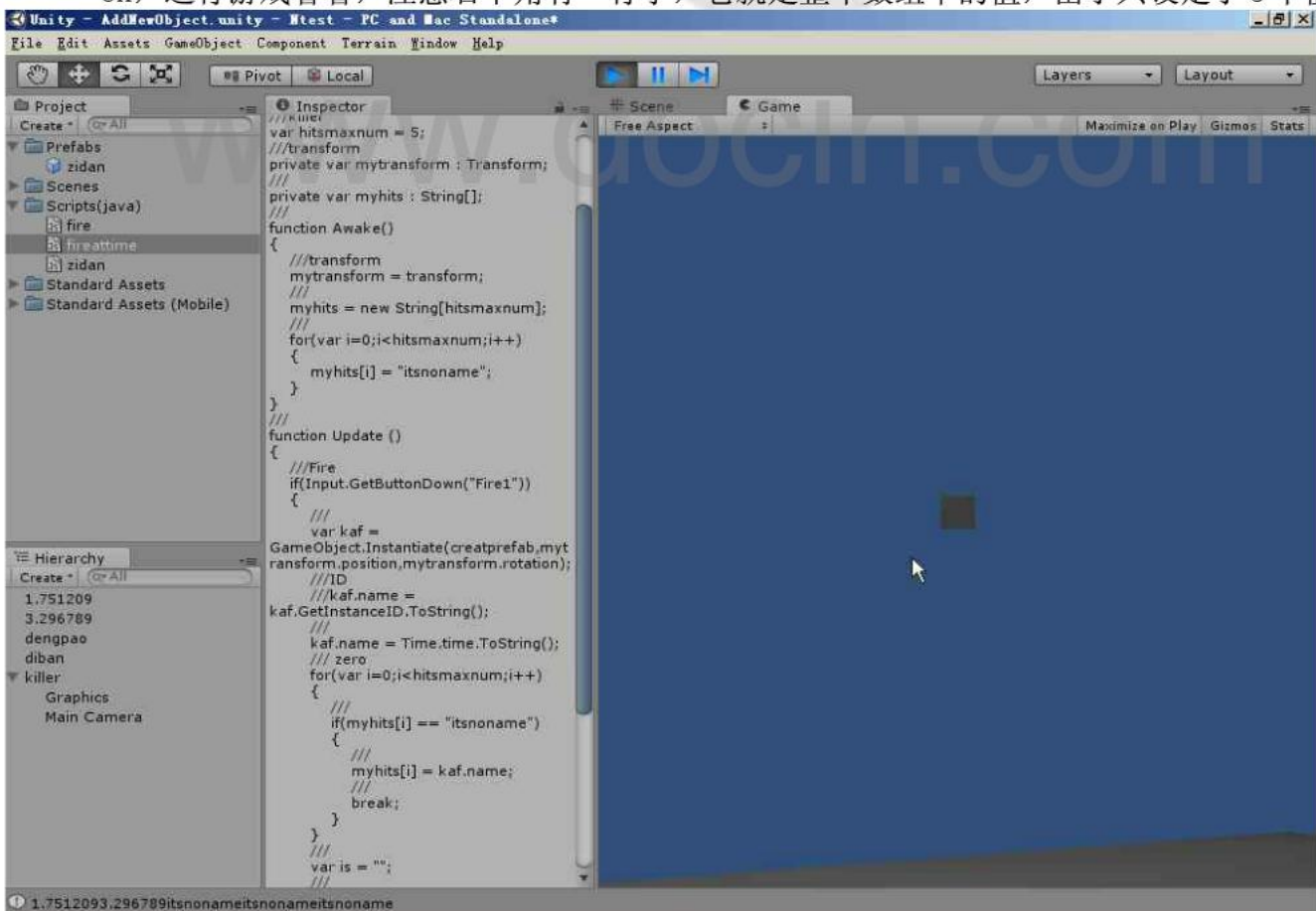
文件超出一屏，按行号连着看吧



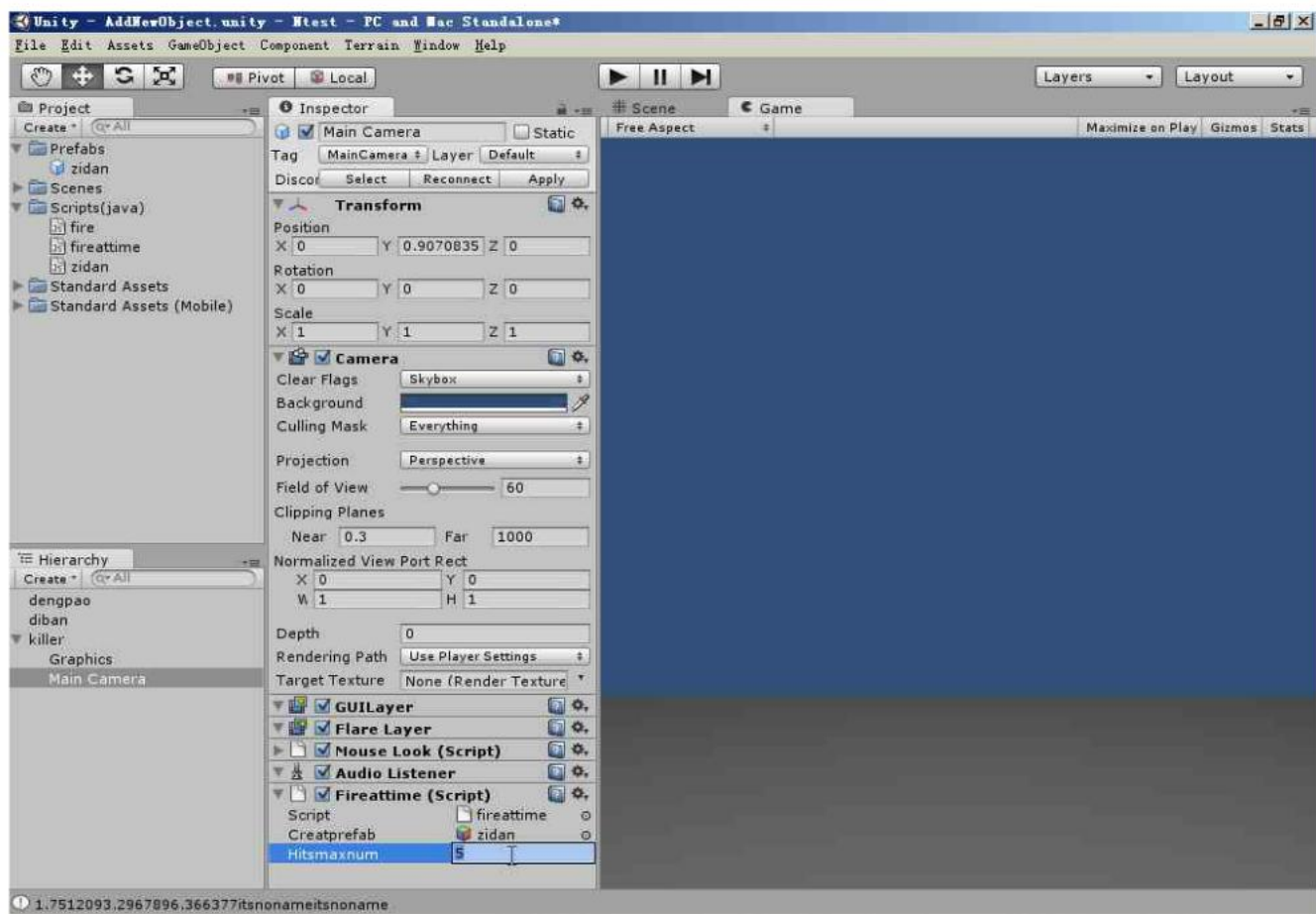
```
29  //创建一个实例，并赋予临时变量。  
30  var kaf = GameObject.Instantiate(creatprefab,mytransform.position,mytransform.rotation);  
31  //更改当前创建的实例的名称，名称使用当前实例的系统ID。  
32  //kaf.name = kaf.GetInstanceID.ToString();  
33  //更改当前创建的实例的名称，名称使用当前实例被创建的时间。  
34  kaf.name = Time.time.ToString();  
35  //子弹子弹缓存数组，如果值等于 zero，就将刚创建的实例赋值给它。  
36  for(var i=0;i<hitsmaxnum;i++)  
37  {  
38  //如果数组中有一个的值是初始值  
39  if(myhits[i] == "itsnoname")  
40  {  
41  //那么就将当前刚创建的实例的名称保存给它  
42  myhits[i] = kaf.name;  
43  //退出循环  
44  break;  
45  }  
46  }  
47  //暂时建一个字符串变量  
48  var is = "";  
49  //遍历数组，将所有值组成一个字符串  
50  for(var j=0;j<hitsmaxnum;j++)  
51  {  
52  is = is + myhits[j];  
53  }  
54  //输出这个字符串  
55  print(is);  
56  }  
57  }
```

Line 54 / 57, Col 26. INS (LF)

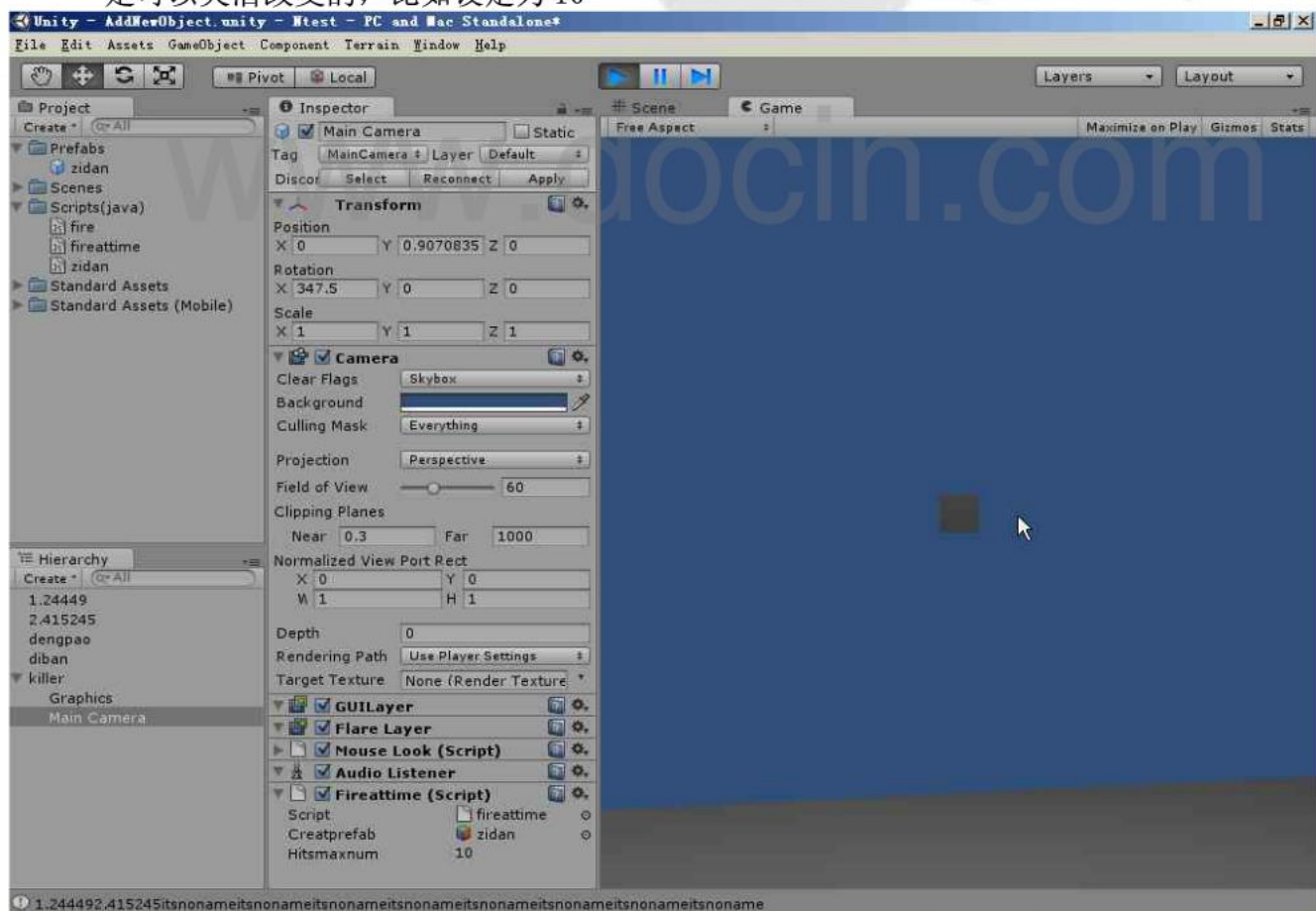
ok，运行游戏看看，注意右下角有一行字，它就是整个数组中的值，由于只设定了5个值，



所以现在只可以保存5可子弹的名字，不过，在这里



是可以灵活改变的，比如设定为 10



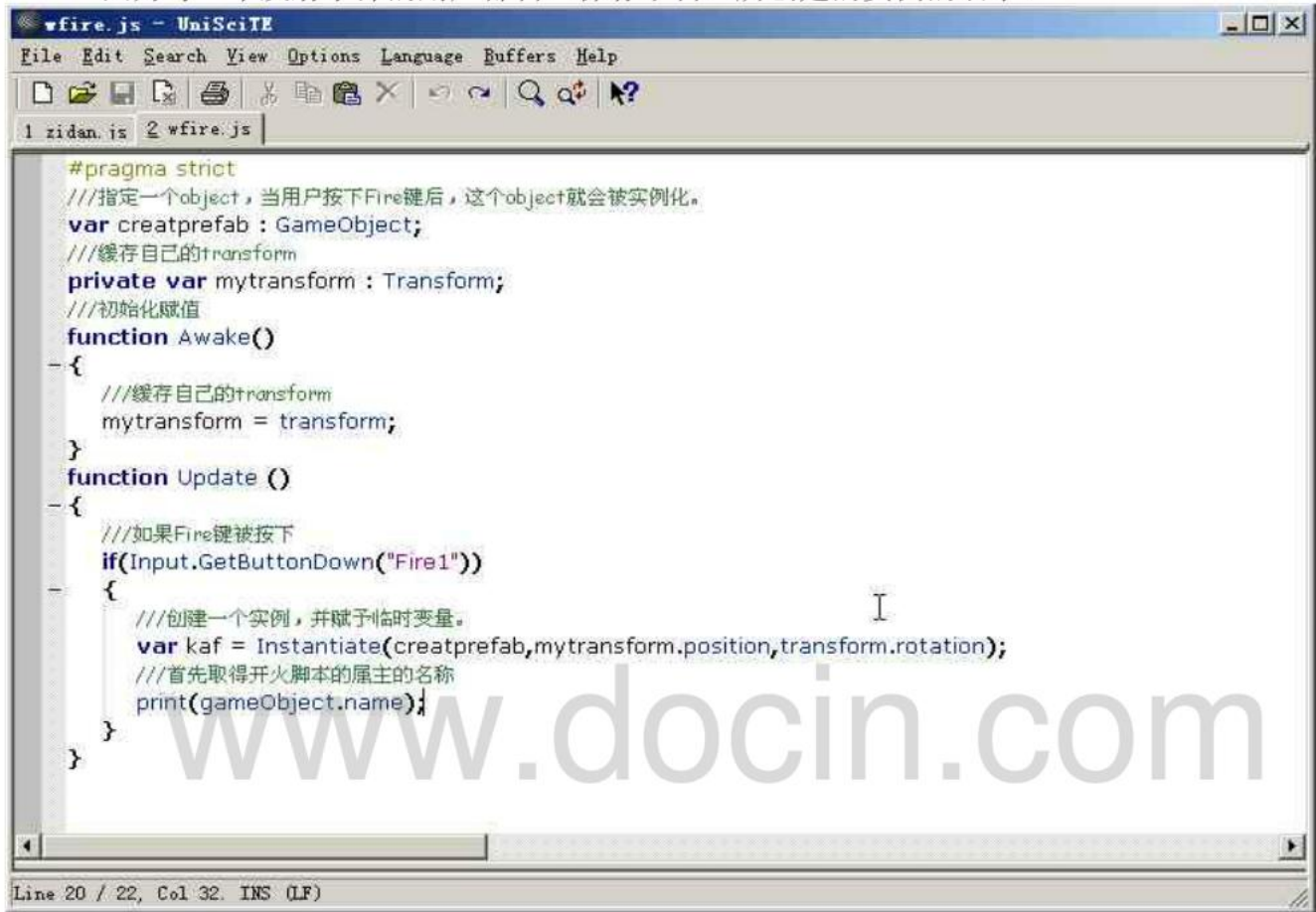


# 打飞机的子弹是无效的

子弹在飞行一段时间以后，如果没有击中目标，不管是地板还是墙壁，或者是其他什么，那么这颗子弹就应该消失，因为飞行足够时间之后，子弹已经远离了有效的游戏环境，如果它继续存在于游戏环境中，会浪费不少系统资源。

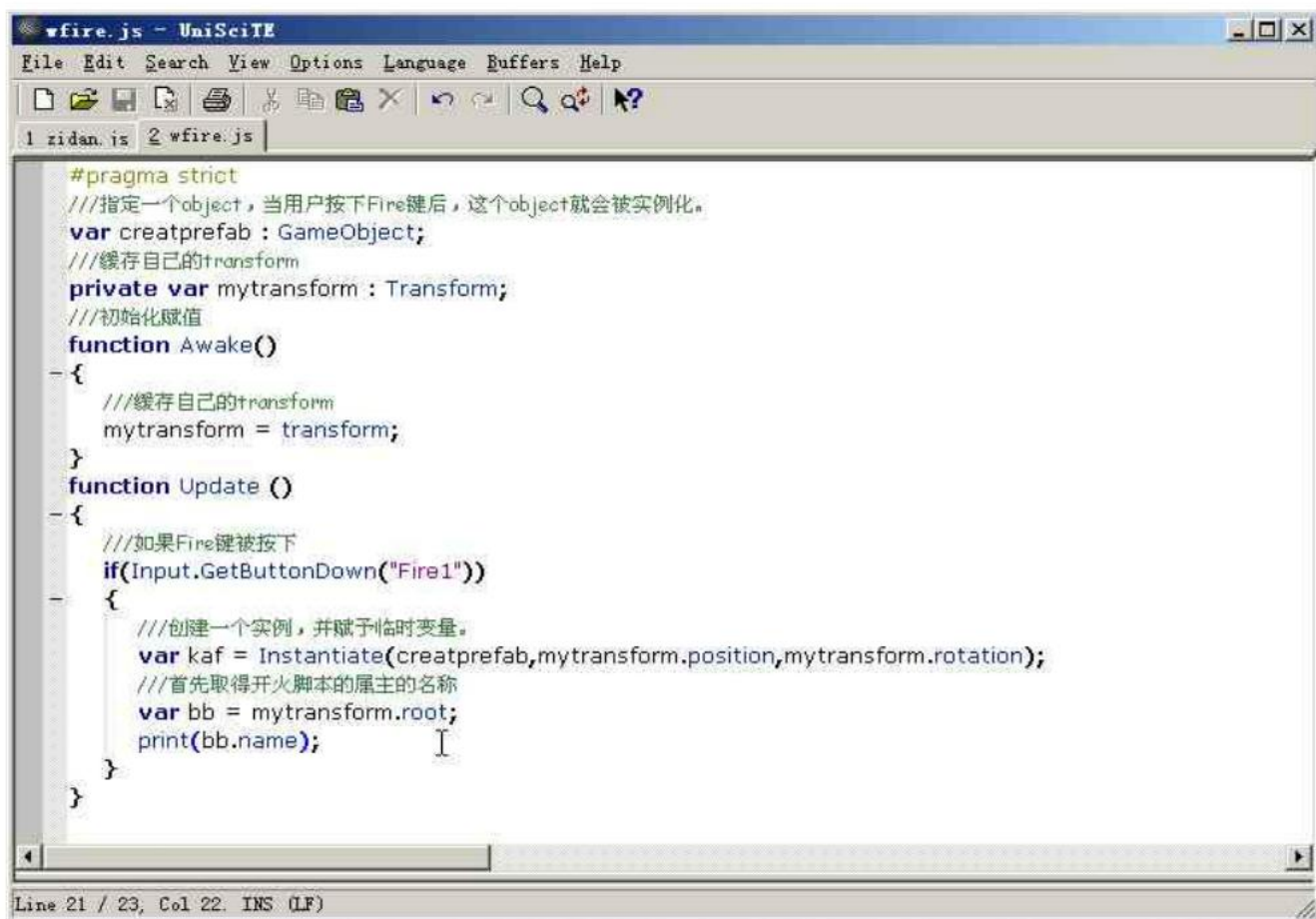
首先，确认是谁发射了子弹。

因为每一个发射子弹的用户都自己保存了自己所创建的实例的名字。



```
#pragma strict
///指定一个object，当用户按下Fire键后，这个object就会被实例化。
var creatprefab : GameObject;
///缓存自己的transform
private var mytransform : Transform;
///初始化赋值
function Awake()
- {
    ///缓存自己的transform
    mytransform = transform;
}
function Update ()
- {
    ///如果Fire键被按下
    if(Input.GetButtonDown("Fire1"))
    - {
        ///创建一个实例，并赋予临时变量。
        var kaf = Instantiate(creatprefab,mytransform.position,transform.rotation);
        ///首先取得开火脚本的属主的名称
        print(gameObject.name);
    }
}
}
```

运行游戏，发现，其实 `gameObject.name` 是保存的本脚本属主的信息，



```
wfire.js - UniSciTE
File Edit Search View Options Language Buffers Help
1 zidan.js 2 wfire.js

#pragma strict
///指定一个object，当用户按下Fire键后，这个object就会被实例化。
var creatprefab : GameObject;
///缓存自己的transform
private var mytransform : Transform;
///初始化赋值
function Awake()
- {
    ///缓存自己的transform
    mytransform = transform;
}
function Update ()
- {
    ///如果Fire键被按下
    if(Input.GetButtonDown("Fire1"))
    - {
        ///创建一个实例，并赋予临时变量。
        var kaf = Instantiate(creatprefab,mytransform.position,mytransform.rotation);
        ///首先取得开火脚本的屋主名称
        var bb = mytransform.root;
        print(bb.name);
    }
}

Line 21 / 23, Col 22, INS (LF)
```

而从 transform 向上的追溯就能找到它的最终属主。

给子弹脚本新增一个变量，用来保存自己是被谁创建出来的。



```
zidan.js - UniSciTE
File Edit Search View Options Language Buffers Help
1 zidan.js

#pragma strict
///第一行的 #pragma strict 让脚本只能使用静态数据类型，加速脚本运行速度。

///子弹飞行速度
var otfspeed = 15.0;
///子弹的最大射程，注意，是最大射程，不是有效射程。
var otfmaxle = 100;
///一个子弹发射人员的变量
var wfireme = "";
///缓存自身数据 transform
private var otf : Transform;
///子弹已经飞行的距离
private var otffle = 0.0;
///在属主被加载时，初始化变量。
function Awake()
- {
    ///将脚本属主的transform属性缓存起来，方便以后调用，使用缓存，可以加速脚本运行速度。
    otf = transform;
}
///每一帧都被调用
function Update ()
- {
    ///脚本属主每秒沿着自己的forward方向以指定速度前进
    otf.Translate(Vector3.forward * otfspeed * Time.deltaTime);
}

Line 9 / 39, Col 18, INS (LF)
```

恩，现在这颗子弹就能自己飞行了，如果给它附加上钢体属性，附加碰撞后销毁自己并再创建

一个爆炸效果，就完美了。

不过今天这个就到这里吧。

