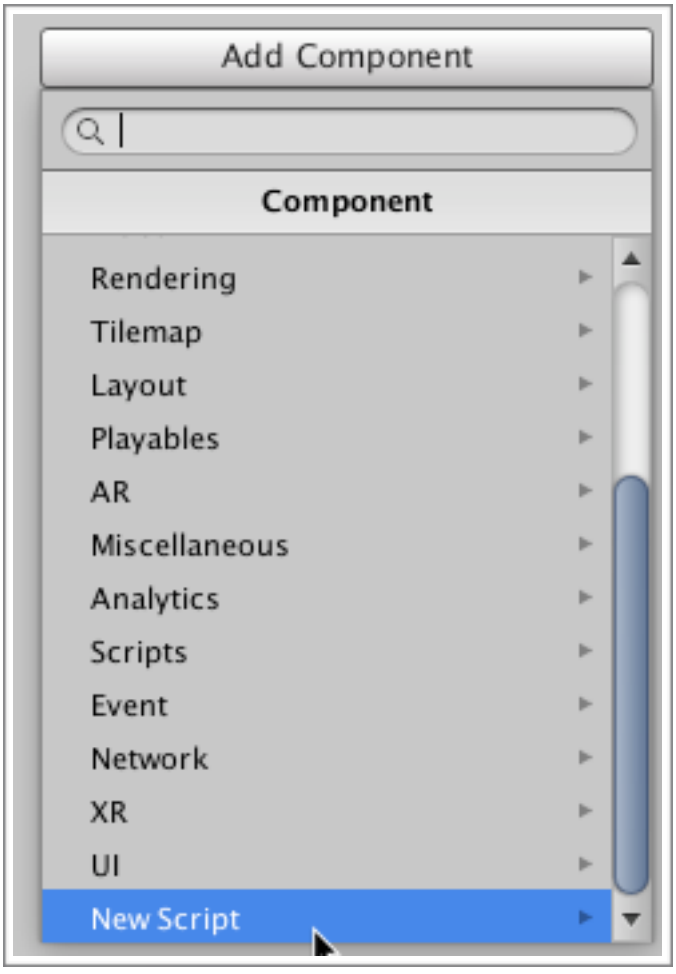
在上一课的内容中，我们往场景中添加了敌人，同时还设置了它的动画控制器。但是奇怪的是虽然僵尸敌人已经在循环播放动画了，但是我们也在预览场景中看到僵尸并没有真正的向前走动。

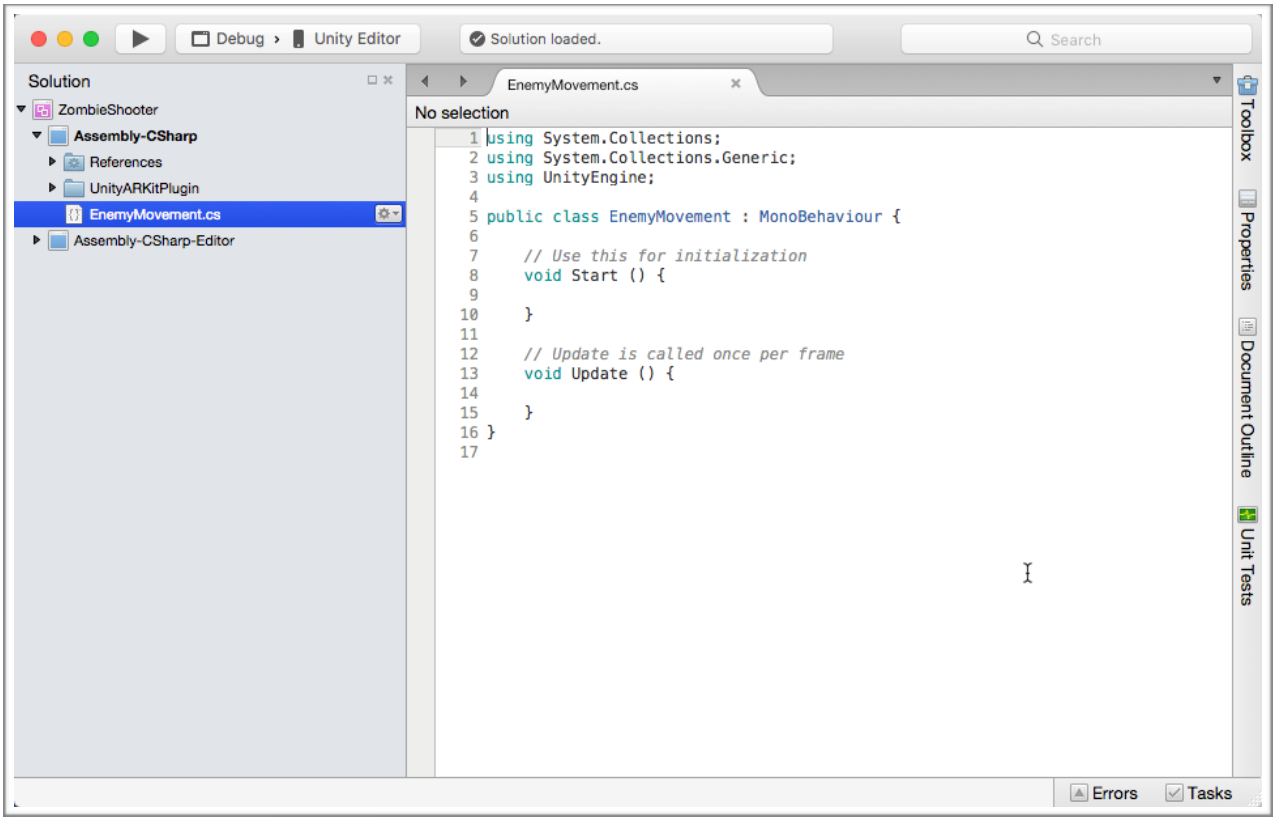
因此，在这一课的内容中我们将让敌人在场景中正常行走。为了实现这一点，我们将需要借助代码的神奇魔力。

在Hierarchy视图中选择z@walk游戏对象，在Inspector视图中点击Add Component，选择New Script，

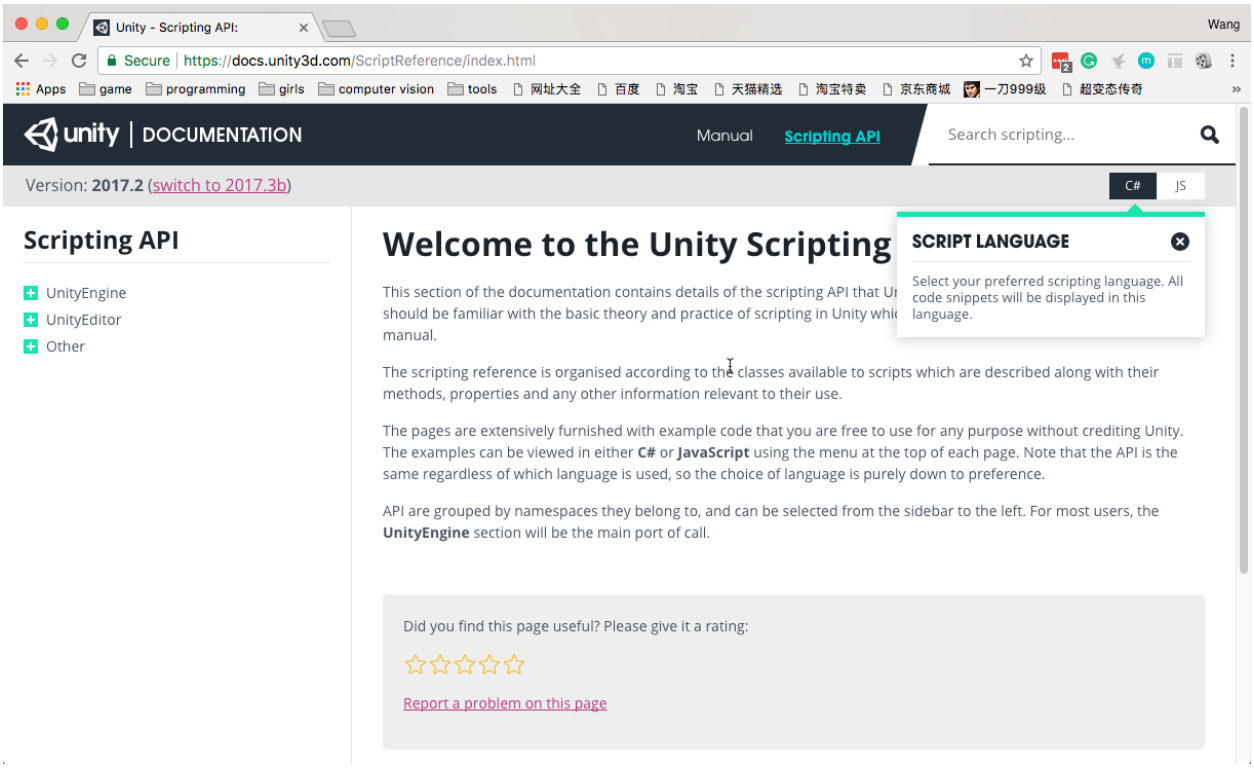


输入名称EnemyMovement，然后点击Create and Add即可。

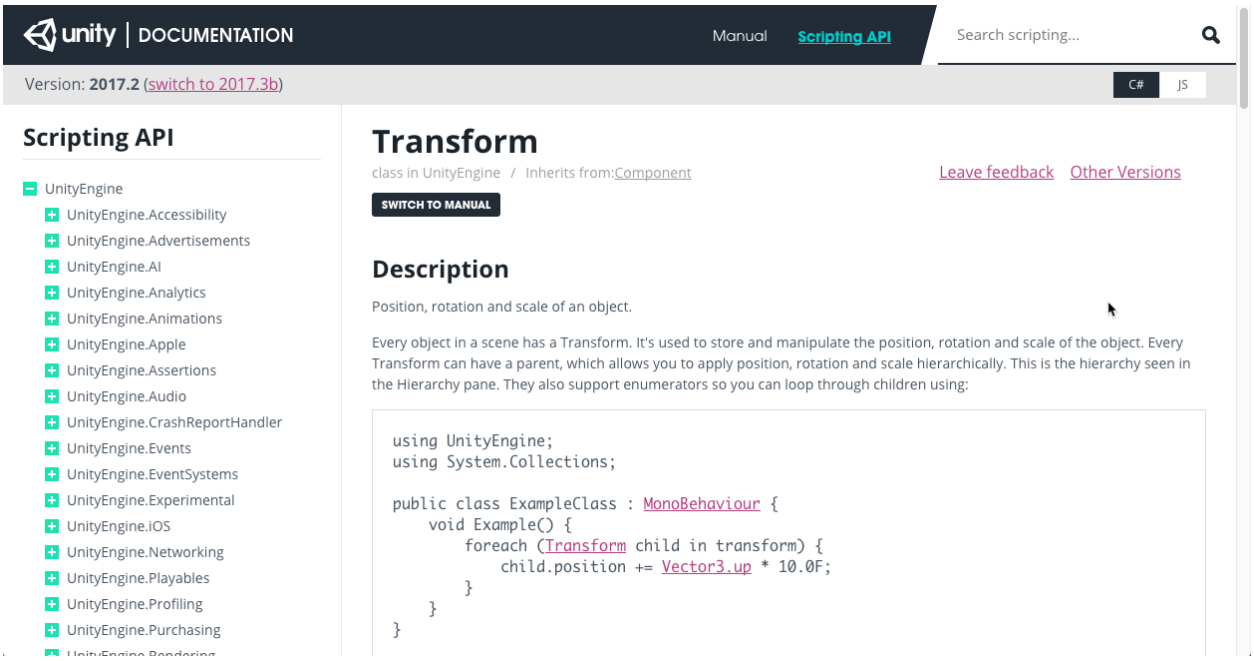
创建完成后右键单击该组件，选择Edit Script，从而在代码编辑器中打开该文件。当然，也可以双击打开。
默认情况下，我们使用的是MonoDevelop作为代码编辑器。



Unity提供了丰富的API可供调用，而事实上我们不太可能也没有必要记住所有的这些API。
一个常用的方法就是在浏览器中打开官方的文档备用：
https://docs.unity3d.com/ScriptReference/index.html

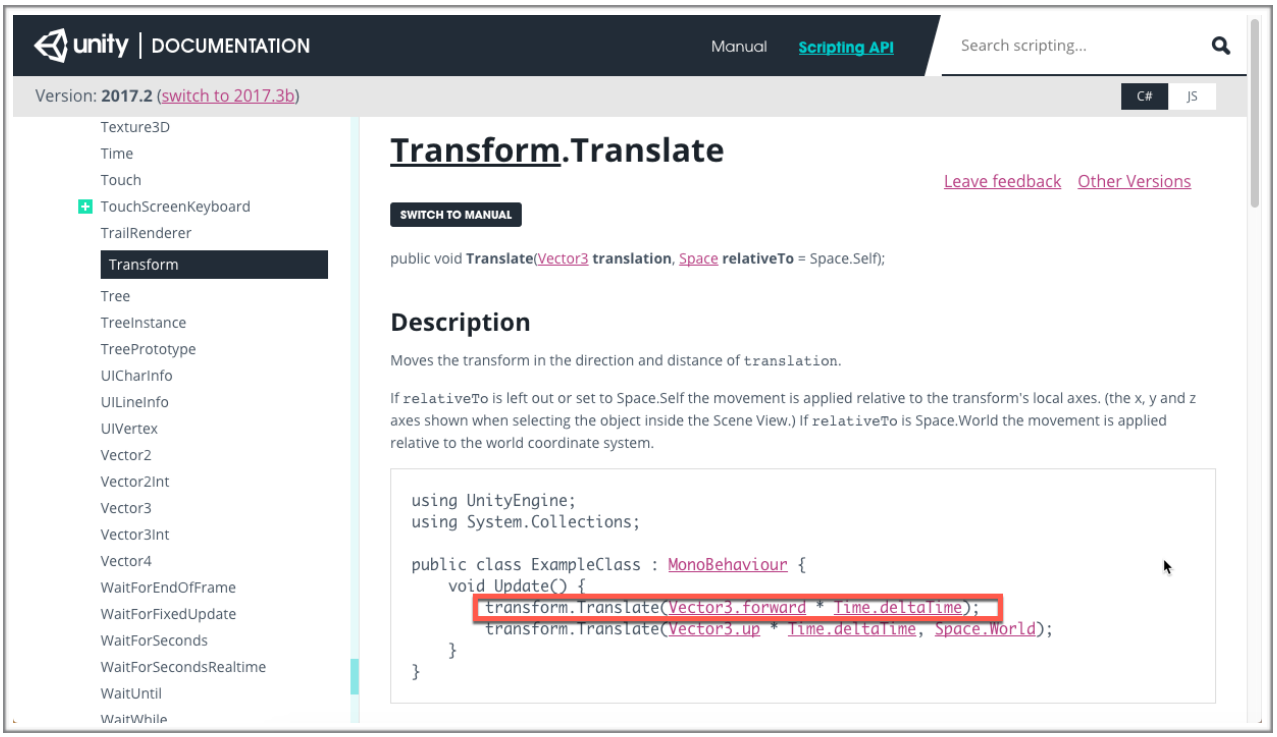


比如这里我们需要让敌人角色在场景中行走，那么显然要考虑更改其位置信息，而角色位置信息对应的其实就是Transform属性，那么我们需要在API文档中搜索Transform，



可以看到，API文档中不但给出了Transform的详细描述，还给出了示例代码。

在相关页面向下滚动，并找到Public Methods部分，里面有一个Translate方法，旁边的介绍是“让游戏对象的transform沿着translation的方向和距离移动”，显然这正是我们所需要的方法。点击Translate进入方法的详细介绍，会找到一段示例代码。



我们可以拷贝以上示例代码中用红色圈出的那一行代码，也就是
`transform.Translate(Vector3.forward * Time.deltaTime);`

然后把它粘贴到EnemyMovement.cs的Update方法中，此时其中的代码如下：

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EnemyMovement : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }
```

```
    // Update is called once per frame
    void Update () {

        transform.Translate(Vector3.forward * Time.deltaTime);
    }
}
```

保存后回到Unity编辑器，点击Play按钮预览效果，可以看到僵尸可以向前行走了。

但是此时它的移动速度有点太快了，让我们来做一下微调。

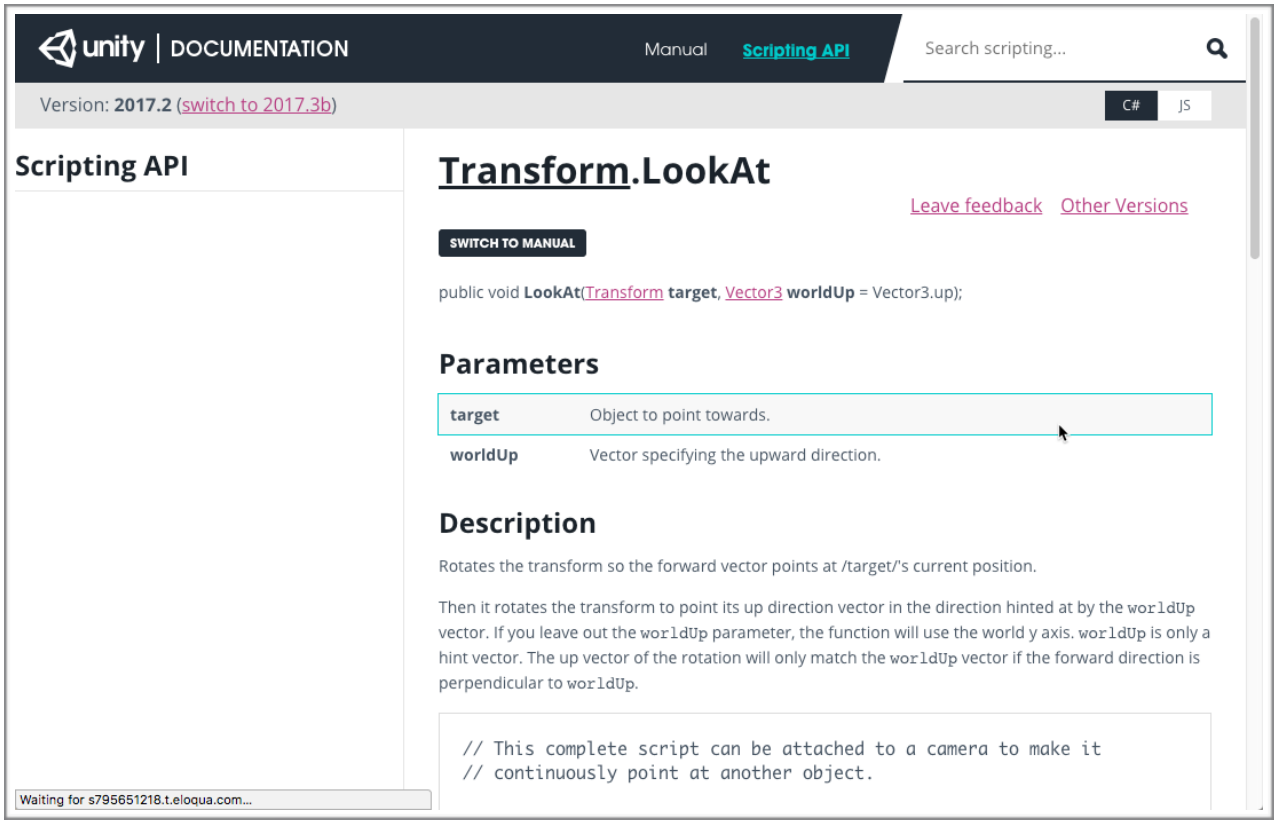
更改Update方法中刚才粘贴过去的那行代码，

```
transform.Translate(Vector3.forward * Time.deltaTime * 0.3f);
```

这里我们只是在最后乘以0.3f，从而降低僵尸的移动速度。

此外，我们还希望可以让僵尸始终朝着主摄像机的方向移动。

因此回到API在线文档，在Transform的Public Functions中有一个LookAt方法，通过链接进入该方法的详细描述和示例代码如下。



拷贝示例代码中Update方法中的这行代码，

```
 transform.LookAt(target);
```

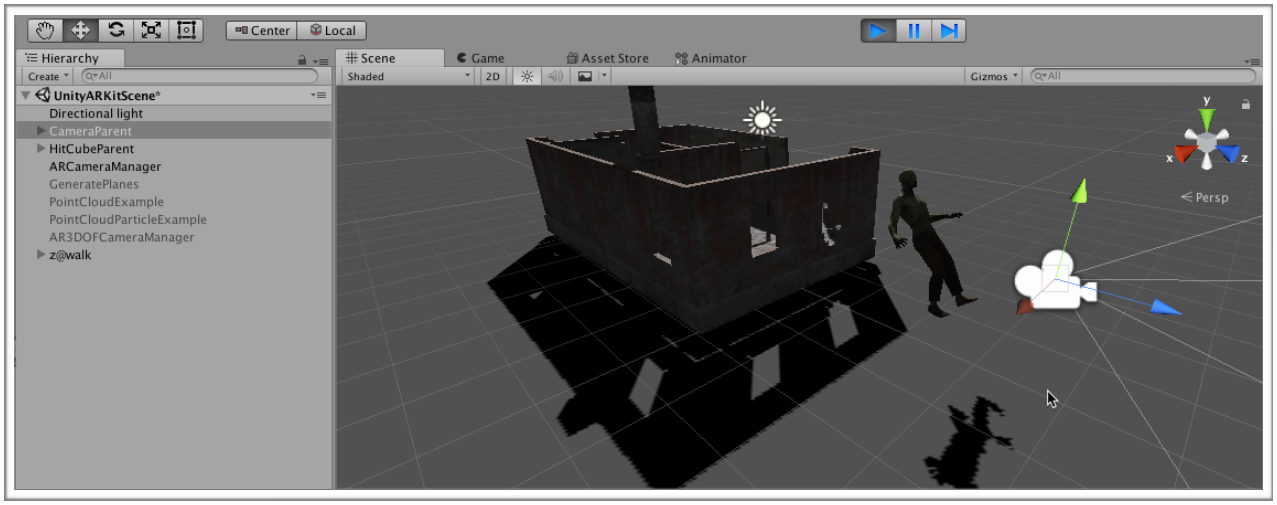
然后将其粘贴到EnemyMovement.cs中的Update方法中，

```
void Update () {

     transform.Translate(Vector3.forward * Time.deltaTime *
0.3f);
     transform.LookAt(target);
}
```

当然，可以看到这里的target使用红色标出，显然是有问题的。
更改这行代码，修改后的Update方法体代码如下：

```
void Update () {

     transform.Translate(Vector3.forward * Time.deltaTime *
0.3f);
     transform.LookAt(Camera.main.transform.position);
}
```

保存后回到Unity编辑器，点击Play按钮查看效果。可以看到代码起作用了，但是如果我们把主摄像机的位置移到半空中，那么僵尸也直接开始空中漫步了~



因此我们要继续完善代码。

回到API文档的Transform页面，在Variables中可以找到eulerAngles这个属性，它代表以degree为单位的Euler旋转角。关于什么是EulerAngles，这里暂时不做详细的解释，我们先了解下如何使用它来修复这个问题。
点击链接可以看到关于eulerangles的详细描述，

Transform.eulerAngles

Leave feedback Other Versions

SWITCH TO MANUAL

public Vector3 **eulerAngles**;

Description

The rotation as Euler angles in degrees.

The x, y, and z angles represent a rotation z degrees around the z axis, x degrees around the x axis, and y degrees around the y axis (in that order).

Only use this variable to read and set the angles to absolute values. Don't increment them, as it will fail when the angle exceeds 360 degrees. Use Transform.Rotate instead.

```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour {
    public float yRotation = 5.0F;
    void Update() {
        yRotation += Input.GetAxis("Horizontal");
        transform.eulerAngles = new Vector3(10, yRotation, 0);
    }
    void Example() {
        print(transform.eulerAngles.x);
        print(transform.eulerAngles.y);
        print(transform.eulerAngles.z);
    }
}
```

从中拷贝Update方法中的以下代码：

```
transform.eulerAngles = new Vector3(10, yRotation, 0);
```

将其粘贴到EnemyController.cs的Update方法中，

```
void Update () {

     transform.Translate(Vector3.forward * Time.deltaTime *
0.3f);
     transform.LookAt(Camera.main.transform.position);
     transform.eulerAngles = new Vector3(10, yRotation, 0);
     }
```

注意eulerAngles中Vector3的三个属性值其实就是Transform中Rotation的x,y,z值。我们需要限制僵尸在地面上移动，因此需要将Rotation中的x值和y值设置为0。
更改这行代码，更改后的Update方法体代码如下：

```
void Update () {

     transform.Translate(Vector3.forward * Time.deltaTime *
0.3f);
     transform.LookAt(Camera.main.transform.position);
     transform.eulerAngles = new Vector3(0,
transform.eulerAngles.y, 0);
     }
```

保存后回到Unity编辑器，点击Play按钮预览效果，发现无论怎么更改主摄像机在x轴和y轴的数值，都不会产生“僵尸上天”的怪异举动了~

好了，今天的学习到此为止。