在继续添加新的功能之前，我们需要对当前的游戏功能测试一下，确保一切工作正常。

首先我们要对场景做一些小的调整。

在Unity编辑器的Hierarchy视图中找到HItCubeParent下的ruined_house子对象，然后在Inspector视图中右键单击Unity AR Hit Test Example(Script)组件，选择Edit Script，并对其中的代码进行编辑。

当前的代码有一个小小的问题，特别是在Update方法中。当前的脚本会检测用户在屏幕上的触碰操作，并将其转换成一个新的坐标。但问题在于，当触碰游戏的UI按钮，比如射击按钮时，也会执行类似的操作，并将整个地图移动到新的坐标。这显然不是我们希望看到的。

因此我们将更改UnityARHitTestExample.cs的代码如下：

```
using System;
using System.Collections.Generic;
//1.导入UI事件系统
using UnityEngine.EventSystems;

namespace UnityEngine.XR.iOS
{
     public class UnityARHitTestExample : MonoBehaviour
     {
          public Transform m_HitTransform;

        bool HitTestWithResultType (ARPoint point,
ARHitTestResultType resultTypes)
        {
            List<ARHitTestResult> hitResults =
UnityARSessionNativeInterface.GetARSessionNativeInterface
().HitTest (point, resultTypes);
            if (hitResults.Count > 0) {
                foreach (var hitResult in hitResults) {
                    Debug.Log ("Got hit!");
                    m_HitTransform.position =
UnityARMatrixOps.GetPosition (hitResult.worldTransform);
                    m_HitTransform.rotation =
UnityARMatrixOps.GetRotation (hitResult.worldTransform);
                    Debug.Log (string.Format ("x:{ 0:0.######}
y:{ 1:0.######} z:{ 2:0.######}", m_HitTransform.position.x,
m_HitTransform.position.y, m_HitTransform.position.z));
                    return true;
                }
            }
            return false;
```

```
        }

          // Update is called once per frame
          void Update () {
               if (Input.touchCount > 0 && m_HitTransform !=
null)
               {
                    var touch = Input.GetTouch(0);
                    //2.对此行代码进行调整，添加另外一个逻辑判断条件，也
即UI系统没有进行交互
                    if ((touch.phase == TouchPhase.Began ||
touch.phase == TouchPhase.Moved) && !
EventSystem.current.IsPointerOverGameObject(0))
                    {
                         var screenPosition =
Camera.main.ScreenToViewportPoint(touch.position);
                         ARPoint point = new ARPoint {
                              x = screenPosition.x,
                              y = screenPosition.y
                         } ;

                    // prioritize reults types
                    ARHitTestResultType[] resultTypes = {

ARHitTestResultType.ARHitTestResultTypeExistingPlaneUsingExtent,
                        // if you want to use infinite planes
use this:
                        //
ARHitTestResultType.ARHitTestResultTypeExistingPlane,

ARHitTestResultType.ARHitTestResultTypeHorizontalPlane,

ARHitTestResultType.ARHitTestResultTypeFeaturePoint
                    };

                    foreach (ARHitTestResultType resultType in
resultTypes)
                    {
                        if (HitTestWithResultType (point,
resultType))
                        {
                            return;
                        }
                    }
                    }
```

```
                }
            }


        }
}
```

以上仅做了两处调整，按照数字编号解释一下：
1.导入了Unity的UI事件交互系统的
2.添加了一个判断条件，仅当UI事件交互系统没有响应时才会执行下面的操作。

通过这两处调整，就可以规避我们刚刚提到的问题。

回到Unity编辑器，接下来我们还希望实现当用户触碰Start Game按钮开启游戏后，可以删除UnityARHitTestExample脚本。因为当游戏正式开启后，我们不希望再实时更改ruined_house对象的位置，而希望它固定在某个位置。此外，在开启游戏后还需要做的就是禁用Start Game按钮，直到游戏重新开始。

为此，在Hierarchy视图中选择HItCubeParent下的ruined_house子对象，然后在Inspector视图中点击Add Component，创建一个新的脚本文件，将其命名为StartGame，并在MonoDevelop中将其打开。

更改其中的代码如下：

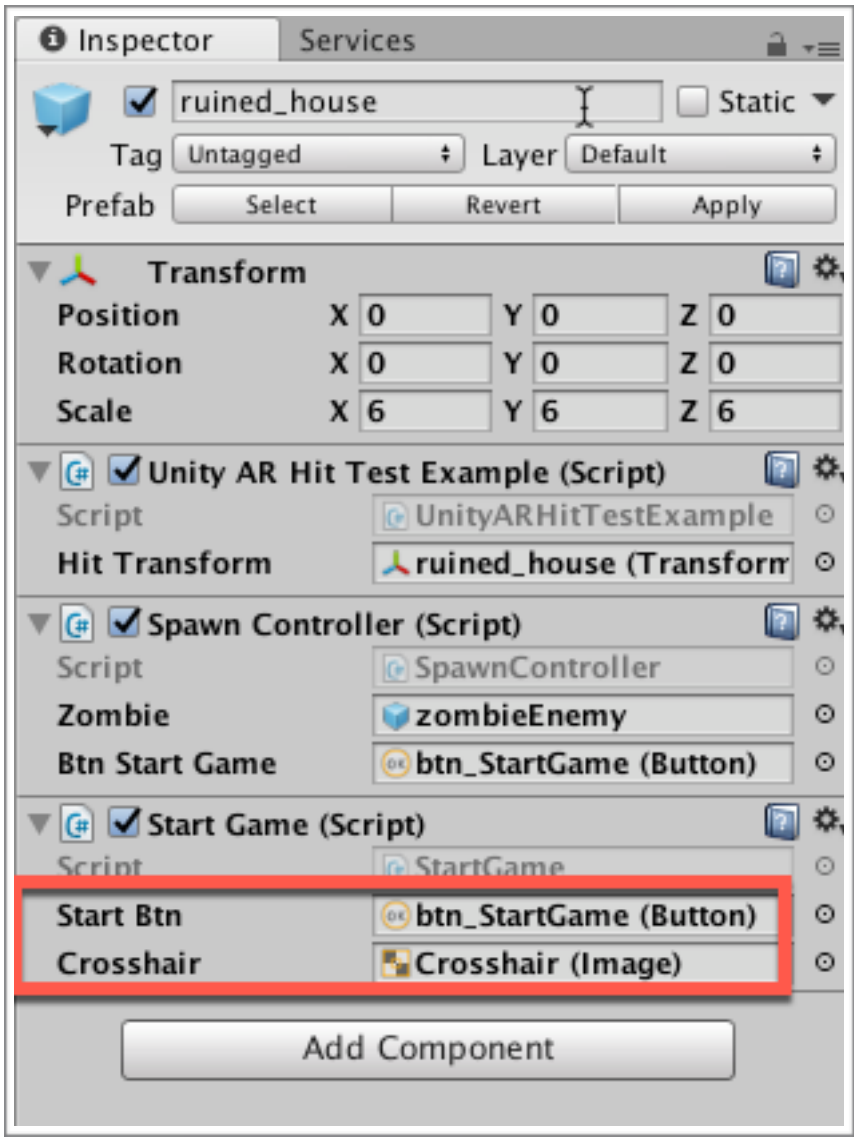
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

//1.导入UI相关的命名空间
using UnityEngine.UI;
//2.导入ARKit相关的命名空间
using UnityEngine.XR.iOS;

public class StartGame : MonoBehaviour {

    //3.开始按钮
    public Button startBtn;
    //4.创建到UnityARHitTestExample的引用
    private UnityARHitTestExample unityARHitTestExample;
    //5.crosshair
    public Image crosshair;
```

```
	// Use this for initialization
	void Start () {

		//6.添加开始游戏按钮的事件响应机制
		startBtn.onClick.AddListener (StartNewGame);

	}


	void StartNewGame(){

		//7.获取到UnityARHitTestExample的引用
		unityARHitTestExample =
GetComponent<UnityARHitTestExample> ();
		//8.删除到UnityARHitTestExample的引用
		Destroy (unityARHitTestExample);
		//9.禁用开始游戏按钮
		startBtn.gameObject.SetActive (false);
		//10.启用辅助瞄准
		crosshair.gameObject.SetActive (true);
	}
}
```

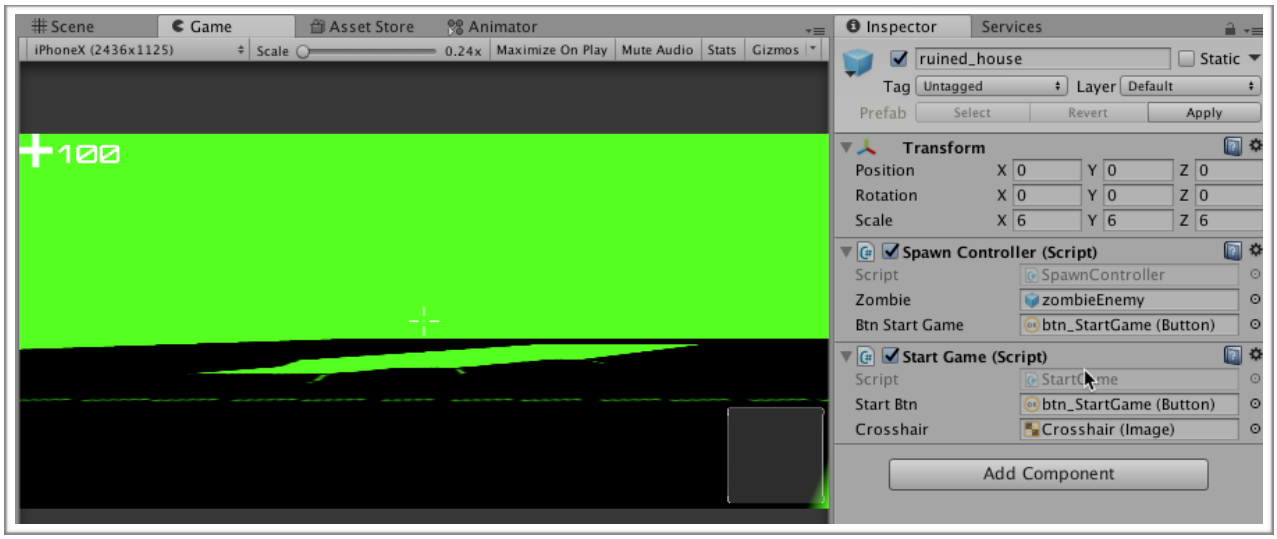
这里按照注释行的数字编号顺序来简单解释一下：
1.这里导入了和UI相关的命名空间
2.导入了和ARKit相关的命名空间
3.创建到开始按钮的引用
4.创建到UnityARHitTestExample的引用
5.创建到crosshair准星的引用
6.添加开始游戏按钮的事件响应机制
7.获取到UnityARHitTestExample的引用
8.删除到UnityARHitTestExample的引用
9.禁用开始游戏按钮
10.启用辅助瞄准

接下来回到Unity编辑器，做一些简单的设置。

首先在Hierarchy视图中的Canvas下找到Crosshair，并将其禁用，因为我们会在代码中将其启用。
接下来在Hierarchy视图中找到HitCubeParent下的ruined_house子对象，然后在Start Game(Script)组件处设置Start Btn和Crosshair如下。

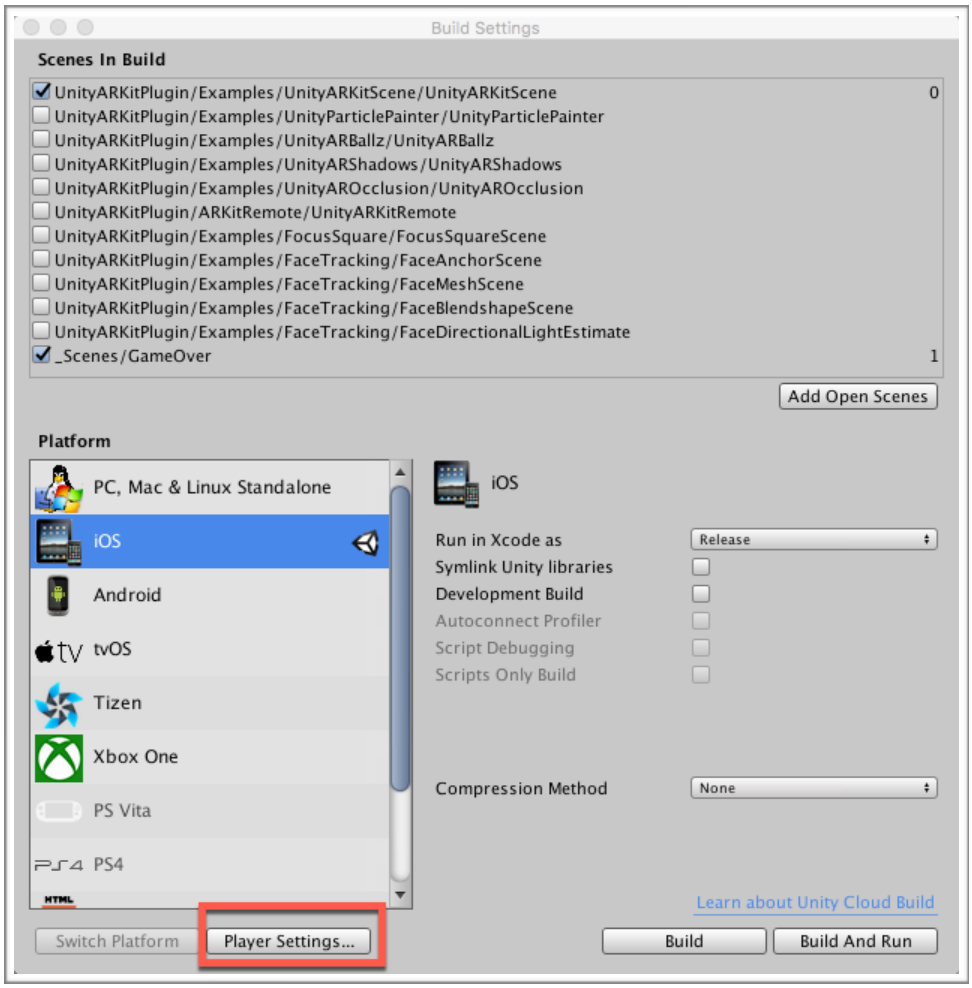


点击Unity编辑器工具栏上的Play按钮，然后切换到Game视图。当我们按下Start Game按钮的时候，可以看到Inspector视图处的UnityARHitTestExample脚本对象已经没有了，Start Game按钮也消失了，而准星则出现在界面中间。如下图所示。

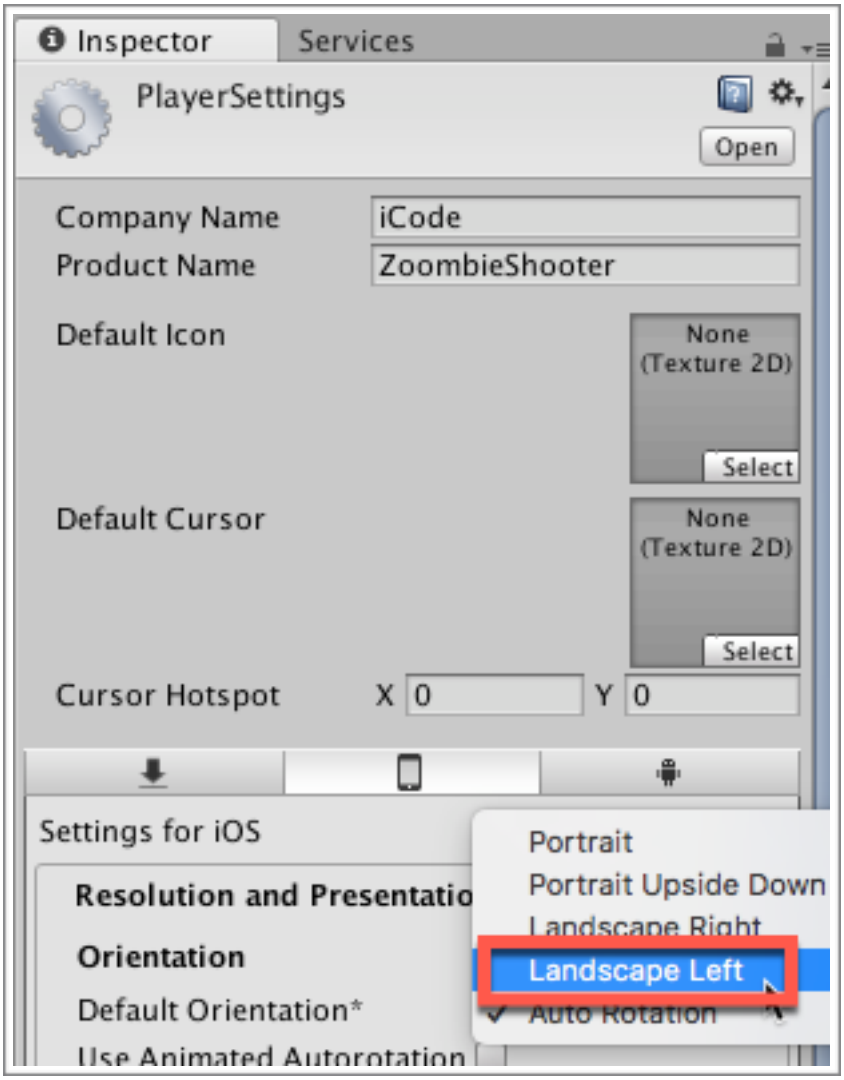


接下来我们要在iPhone设备上进行测试。

为此，从Unity菜单栏中选择File–Build Settings，
然后点击Player Settings，

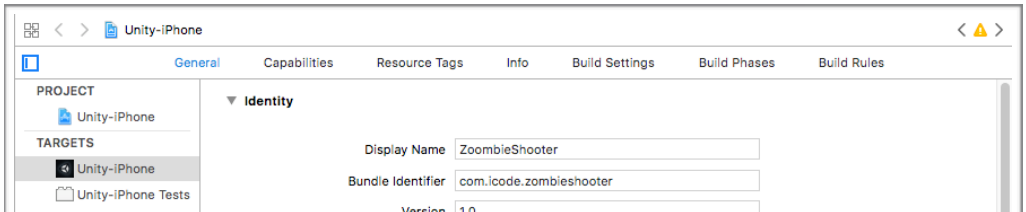


首先更改设备朝向，在Resolution and Presentation部分的Default Orientation属性处，从下拉列表中选择Landscape Left，如图所示。



然后点击Build and Run按钮编译运行。
此时可能会提示你之前有同名的项目，选择Replace即可。

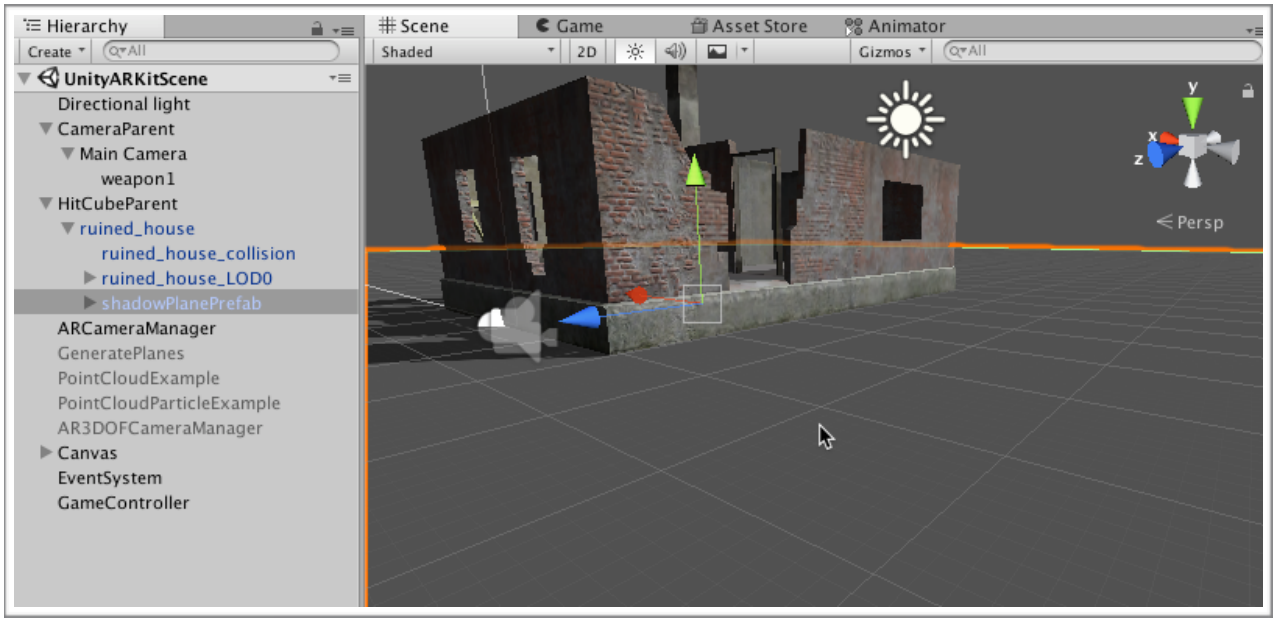
此时系统会自动在Xcode中打开生成的项目，当然首先会看到几个错误提示。
接下来要在Xcode的General- Signing中更改Team信息。
之前提到过，你需要注册一个苹果开发者账号，才能顺利进行下面的工作。
如果还没有注册过，需要在苹果官网注册：https://developer.apple.com/



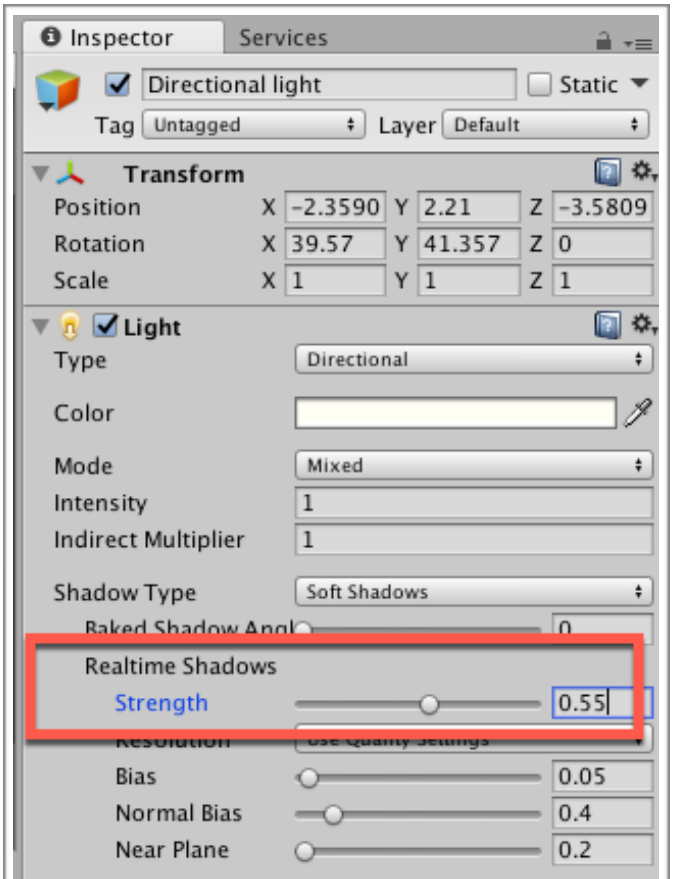
然后确保iPhone设备连接到电脑上，点击Xcode工具栏上的编译运行按钮（向右的三角）即可。
可以看到，在我们触碰Start Game之前，ruined_house模型在空间中的位置是会发生变化的。当触碰Start Game按钮后，房屋模型的空间位置就不变了，然后就会看到僵尸敌人从各个方向来袭。触碰屏幕右下角可以开始攻击。我们可以借助准星的力量进行瞄准。当敌人受到攻击时，会受到一个向后的冲力，同时会播放音效和粒子特效。在遭到敌人攻击时，也会有对应的音效和屏幕特效。

需要特别说明的是，在手机上实际测试游戏的时候，最好是在户外。如果是在室内测试，那么建议将房屋模型和敌人的Transform比例进行相应的缩写。

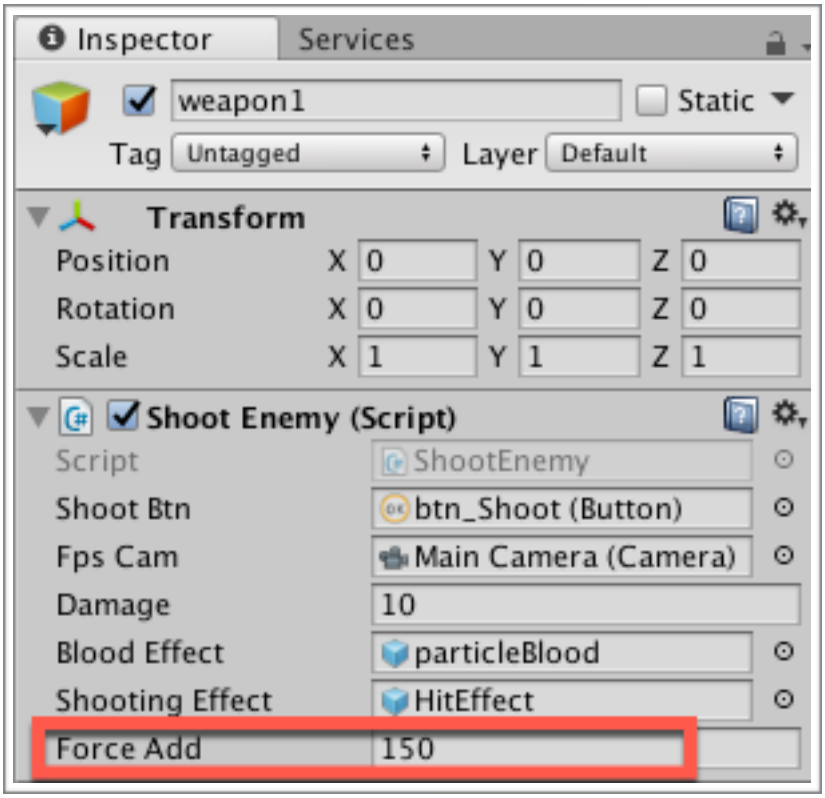
实际测试之后，我们发现有一些小的地方可以继续优化。
首先，我们不太希望看到房屋底部的阴影，因此需要把HitCubeParent下的shadowPlanePrefab的位置稍微往上提一下。



此外，现在房屋的阴影有点太生硬了，因此需要更改阴影的力度。
在Hierarchy视图中选择Directional Light，然后在Inspector视图中将Light组建的Shadow Type下的Realtime Shadows的Strength属性调整到0.55左右。



最后在Hierarchy视图中找到CameraParent下的Main Camera的子对象weapon1，然后在Inspector视图中将Shoot Enemy(Script)组建中的Force Add属性值降低到150，从而让敌人受到的冲击力显得更为自然。



OK，现在我们的调整就基本到位了。

本课内容到此结束，从下一课开始，我们将继续为游戏添加更多有意思的功能。