

【翻译】第四章节：世界空间中的着色器（关于 uniforms）

2014-11-25 08:40:00 2328 人阅读 [Unity3D](#) [cg](#) [世界空间](#)

A- A+

文章内容	例子源码	网友评论	最后编辑：2014-12-21 17:36:23
本文永久地址： http://www.omuying.com/article/92.aspx ，【 文章转载请注明出处！ 】			

原文链接：http://en.wikibooks.org/wiki/Cg_Programming/Unity/Shading_in_World_Space

本章节是《Unity Cg 编程》基础部分的最后一个章节，本教程主要介绍 uniform 参数。

在本教程中，我们将看到着色器片段颜色的改变会受世界位置的影响，其实这个概念并不复杂，但却十分重要。例如，阴影灯光和环境贴图，我们将看看真实世界中的着色器，还有为什么要使用着色器。

从对象空间到世界空间的转变

在《[在着色器中调试](#)》章节中提到，顶点输入参数使用语义词 POSITION 指定对象的坐标，即本地对象（模型）的网格空间坐标，对象空间（对象的坐标系统）是特定于每个游戏对象的，然而，所有的游戏对象会被转换到一个共同的坐标系中（世界空间）。

如果一个对象被直接放入世界空间中，游戏对象会通过转换组件直接将对象的坐标转成世界坐标。你可以在 Scene 视图或者 Hierarchy 视图中选中一个对象，然后在 Inspector 视图中查看 Transform 组件。在 Transform 组件中包括 Position、Rotation 和 Scale 属性，这个组件用来指定从本地坐标到世界坐标的顶点转换（如果一个游戏对象拥有父级对象，那么转换组件只转换父级对象的坐标）。在《[Vertex Transformations](#)》章节中，我们将讨论顶点的转换、旋转、缩放以及 4 x 4 矩阵组合变换的细节。

回到我们的例子中：从对象空间到世界空间的转换是通过 4 x 4 矩阵来转换的，也叫“模型矩阵”，这个矩阵在 Unity 中通过 uniform 参数 _Object2World 已经声明了：

```
1 uniform float4x4 _Object2World;
```

既然在 Unity 中已经声明，我们就不需要再次去声明它，所以我们可以着色器中直接使用：

```
01 Shader "Cg shading in world space"
02 {
03     SubShader
04     {
05         Pass
06         {
07             CGPROGRAM
08
09             #pragma vertex vert
10             #pragma fragment frag
11
12             // uniform float4x4 _Object2World;
13             // automatic definition of a Unity-specific uniform parameter
14             struct vertexInput
15             {
16                 float4 vertex : POSITION;
17             };
18             struct vertexOutput
19             {
20                 float4 pos : SV_POSITION;
21                 float4 position_in_world_space : TEXCOORD0;
22             };
23
24             vertexOutput vert(vertexInput input)
25             {
26                 vertexOutput output;
27
28                 output.pos = mul(UNITY_MATRIX_MVP, input.vertex);
29                 output.position_in_world_space = mul(_Object2World, input.vertex);
30                 // transformation of input.vertex from object
31                 // coordinates to world coordinates;
```

针孔摄像头



无线监控摄像机

针孔摄像头



伪装微型摄像机

ios开发培训

unity3d摄像机

外企猎头公司

微型摄像机


室内设计师

unity3d移动


最新文章



【原创】C# 基础之 Lambda表达式 - 907 次阅读




【原创】C#基础之 IEnumerable和 IEnumerator - 792 次阅读



【原创】C#基础之事件 - 886 次阅读



【原创】C#基础之委托 - 912 次阅读



【原创】C#基础之委托的使用 - 856 次阅读

针孔摄像头



无线监控摄像机

针孔摄像头



伪装微型摄像机

unity3d摄像机

ios开发培训

微型摄像机

外企猎头公司

unity3d移动

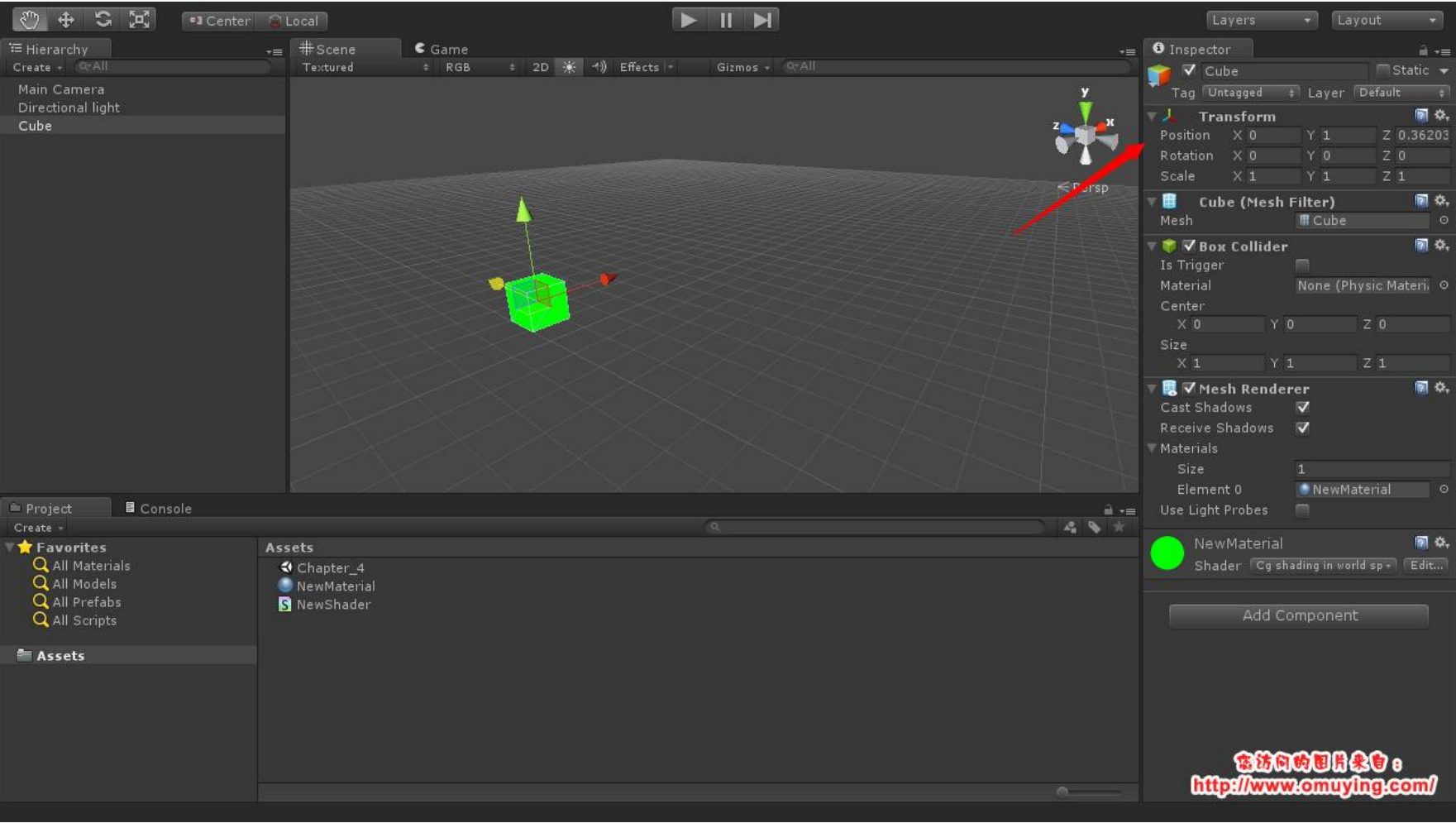
猎头公司排名

随机阅读

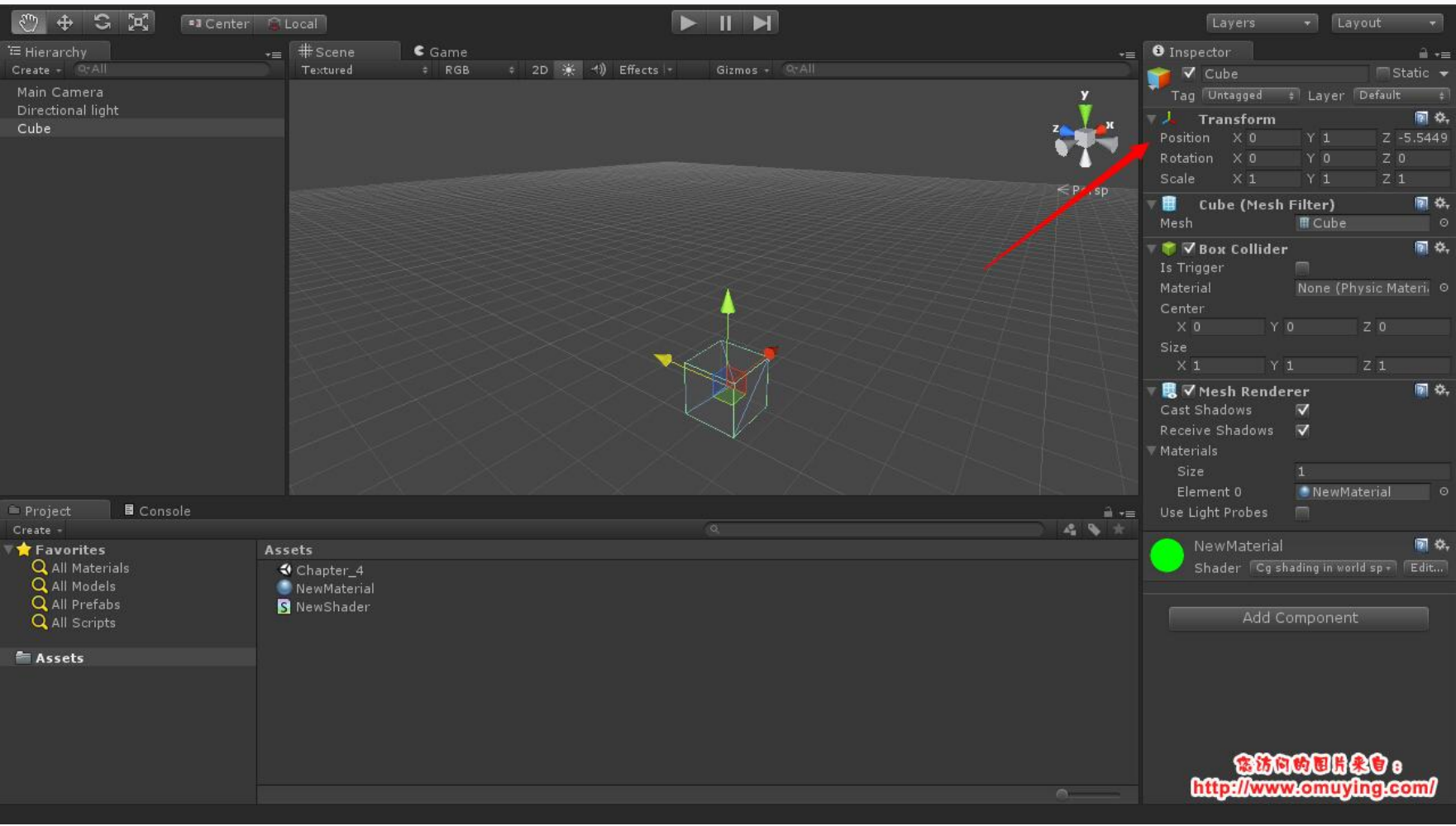
```
32     return output;
33 }
34
35 float4 frag(vertexOutput input) : COLOR
36 {
37     float dist = distance(input.position_in_world_space, float4(0.0,
0.0, 0.0, 1.0));
38     // computes the distance between the fragment position
39     // and the origin (the 4th coordinate should always be
40     // 1 for points).
41     if (dist < 5.0)
42     {
43         return float4(0.0, 1.0, 0.0, 1.0);
44         // color near origin
45     }
46     else
47     {
48         return float4(0.3, 0.3, 0.3, 1.0);
49         // color far from origin
50     }
51 }
52 ENDCG
53 }
54 }
55 }
```

通常情况下，应用程序必须设置 uniform 参数的值，然而在 Unity 中，我们不用担心，因为 Unity 可以正确设置预定义 uniform 参数的值，比如 _Object2World。

这个着色器把顶点位置转换到世界空间中，并作为输出结构体传给片段着色器。对于片段着色器，输出结构体中的参数包含片段在世界坐标中的插值位置，基于这个点到世界坐标系中原点的距离，有两种颜色被设置，因此，如果你在 Unity 中把这个着色器添加到一个对象上，然后你再移动这个对象，你会发现，在接近世界坐标的原点时它会变成绿色，如图：



而远离世界坐标的原点它则会变成暗灰色，如图：



- 【翻译】第二十章节：凹凸表面光照（关于法线贴图） - 1921 次阅读
- 【翻译】第九章节：漫反射（关于每顶点漫反射和多光源漫反射） - 1946 次阅读
- 【翻译】第十九章节：纹理层（关于多重纹理） - 1661 次阅读
- 【翻译】第二十三章节：投影（关于使用投影纹理贴图实现投影） - 2520 次阅读
- 【翻译】第二十二章节：Cookies（关于投影纹理贴图塑造光的形状） - 1392 次阅读

更多的 Unity Uniforms

在 Unity 中，定义了几个和 _Object2World 一样的 float4x4 矩阵，下面是在系列教程中使用的 uniforms 简短列表：

```
01 uniform float4 _Time, _SinTime, _CosTime; // time values
02 uniform float4 _ProjectionParams;
03 // x = 1 or -1 (-1 if projection is flipped)
04 // y = near plane; z = far plane; w = 1/far plane
05 uniform float4 _ScreenParams;
06 // x = width; y = height; z = 1 + 1/width; w = 1 + 1/height
07 uniform float4 unity_Scale; // w = 1/scale; see _World2Object
08 uniform float3 _WorldSpaceCameraPos;
09 uniform float4x4 _Object2World; // model matrix
10 uniform float4x4 _World2Object; // inverse model matrix
11 // (all but the bottom-right element have to be scaled
12 // with unity_Scale.w if scaling is important)
13 uniform float4 _LightPositionRange; // xyz = pos, w = 1/range
14 uniform float4 _WorldSpaceLightPos0;
15 // position or direction of light source
16
17 uniform float4x4 UNITY_MATRIX_MVP; // model view projection matrix
18 uniform float4x4 UNITY_MATRIX_MV; // model view matrix
19 uniform float4x4 UNITY_MATRIX_V; // view matrix
20 uniform float4x4 UNITY_MATRIX_P; // projection matrix
21 uniform float4x4 UNITY_MATRIX_VP; // view projection matrix
22 uniform float4x4 UNITY_MATRIX_T_MV;
23 // transpose of model view matrix
24 uniform float4x4 UNITY_MATRIX_IT_MV;
25 // transpose of the inverse model view matrix
26 uniform float4x4 UNITY_MATRIX_TEXTURE0; // texture matrix
27 uniform float4x4 UNITY_MATRIX_TEXTURE1; // texture matrix
28 uniform float4x4 UNITY_MATRIX_TEXTURE2; // texture matrix
29 uniform float4x4 UNITY_MATRIX_TEXTURE3; // texture matrix
30 uniform float4 UNITY_LIGHTMODEL_AMBIENT; // ambient color
```

对于 Unity 的内置 uniforms 官方列表，你可以查看《[ShaderLab builtin values](#)》。

一些 uniforms 实际上是定义在 UnityCG.cginc 代码引用的 UnityShaderVariables.cginc 文件中，在 Unity 4.0 版本之前，必须通过 #include "UnityCG.cginc" 来引用 UnityCG.cginc 文件，但在新的版本中，这是可选的。

还有一些 uniforms 在 Unity 中没有自动定义，比如 _LightColor0，所以必须显式地定义他们：

```
1 uniform float4 _LightColor0;
```

Unity 并不总是更新这些 uniforms，这儿需要特别指出的是 _WorldSpaceLightPos0 和 _LightColor0 只需要在 Shader passes 标记中正确的设置就可以了，例如：在着色器的 Pass{...} 块的第一行使用标记 {"LightMode" = "ForwardBase"}。详情可以阅读《[Diffuse Reflection](#)》。

用户指定 Uniforms：着色器属性

还有一个更重要的 uniform 参数：用户自定义的 uniforms。实际上，这是 Unity 的属性，你可以认为他们是着色器的用户自定义 uniform 参数。通常一个着色器不带参数只能由编写的程序员在一些特定的程序中使用，但是如果一个着色器拥有参数并且还带有描述性的说明，那么这个着色器就可以被其他人使用，另外，如果你打算出售你的着色器，为着色器的提供参数会大大增加它的价值。

因为在 Unity 的 ShaderLab 中使用《[description of shader properties](#)》非常不错，通过下面的例子我们来了解如何使用着色器属性，首先，我们声明一个属性，然后我们再定义一个与属性名称相同、类型相同的 uniforms。

```
01 Shader "Cg shading in world space"
02 {
03     Properties
04     {
05         _Point ("a point in world space", Vector) = (0., 0., 0., 1.0)
06         _DistanceNear ("threshold distance", Float) = 5.0
07         _ColorNear ("color near to point", Color) = (0.0, 1.0, 0.0, 1.0)
08         _ColorFar ("color far from point", Color) = (0.3, 0.3, 0.3, 1.0)
09     }
10
11     SubShader
12     {
13         Pass
```

```

14     {
15         CGPROGRAM
16
17         #pragma vertex vert
18         #pragma fragment frag
19
20         #include "UnityCG.cginc"
21         // defines _Object2World and _World2Object
22         // uniforms corresponding to properties
23         uniform float4 _Point;
24         uniform float _DistanceNear;
25         uniform float4 _ColorNear;
26         uniform float4 _ColorFar;
27
28         struct vertexInput
29         {
30             float4 vertex : POSITION;
31         };
32         struct vertexOutput
33         {
34             float4 pos : SV_POSITION;
35             float4 position_in_world_space : TEXCOORD0;
36         };
37
38         vertexOutput vert(vertexInput input)
39         {
40             vertexOutput output;
41
42             output.pos = mul(UNITY_MATRIX_MVP, input.vertex);
43             output.position_in_world_space = mul(_Object2World, input.vertex);
44             return output;
45         }
46
47         float4 frag(vertexOutput input) : COLOR
48         {
49             float dist = distance(input.position_in_world_space, _Point);
50             // computes the distance between the fragment position
51             // and the position _Point.
52             if (dist < _DistanceNear)
53             {
54                 return _ColorNear;
55             }
56             else
57             {
58                 return _ColorFar;
59             }
60         }
61     } ENDCG
62 }
63 }
64 }

```

使用这些参数，任何人都可以修改这个着色器并查看这个着色器的效果。而且我们还可以通过脚本来设置着色器的属性，例如，给一个对象挂载 C# 脚本，然后可以像下面代码那样去设置着色器的属性：

```

1  renderer.sharedMaterial.SetVector("_Point", Vector4(1.0, 0.0, 0.0, 1.0));
2  renderer.sharedMaterial.SetFloat("_DistanceNear", 10.0);
3  renderer.sharedMaterial.SetColor("_ColorNear", Color(1.0, 0.0, 0.0));
4  renderer.sharedMaterial.SetColor("_ColorFar", Color(1.0, 1.0, 1.0));

```

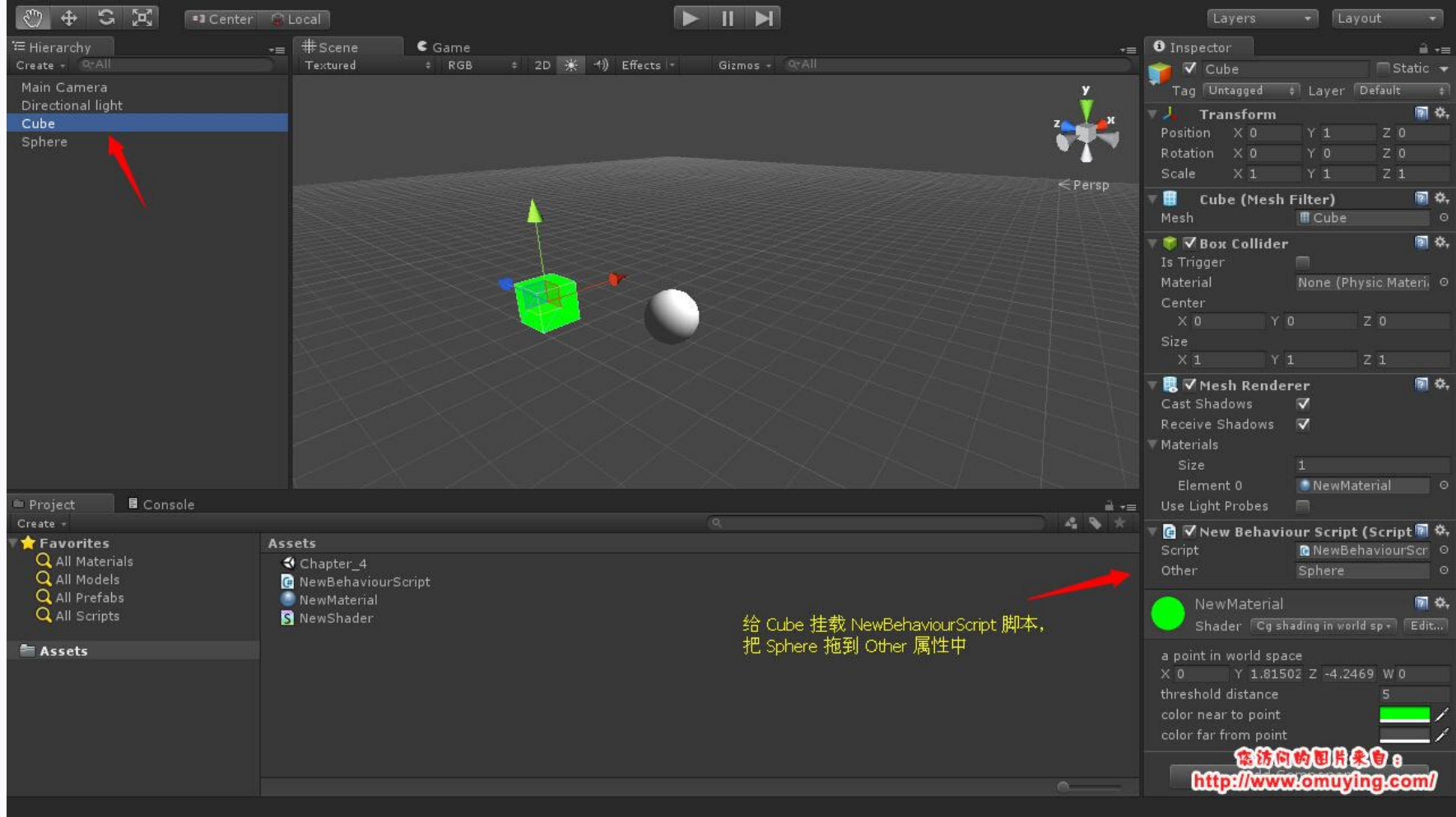
使用 sharedMaterial 我们可以改变所有使用了这个材质的对象的参数，如果你只希望改变一个并使用了这个材质的对象的参数，那么你应该使用 material。假如你设置 _Point 属性为另一个对象的位置信息，这样你只需要在 Unity 中移动这个对象就可以查看效果了，你可以复制/粘贴 下面的代码到一个 C# 脚本中：

```

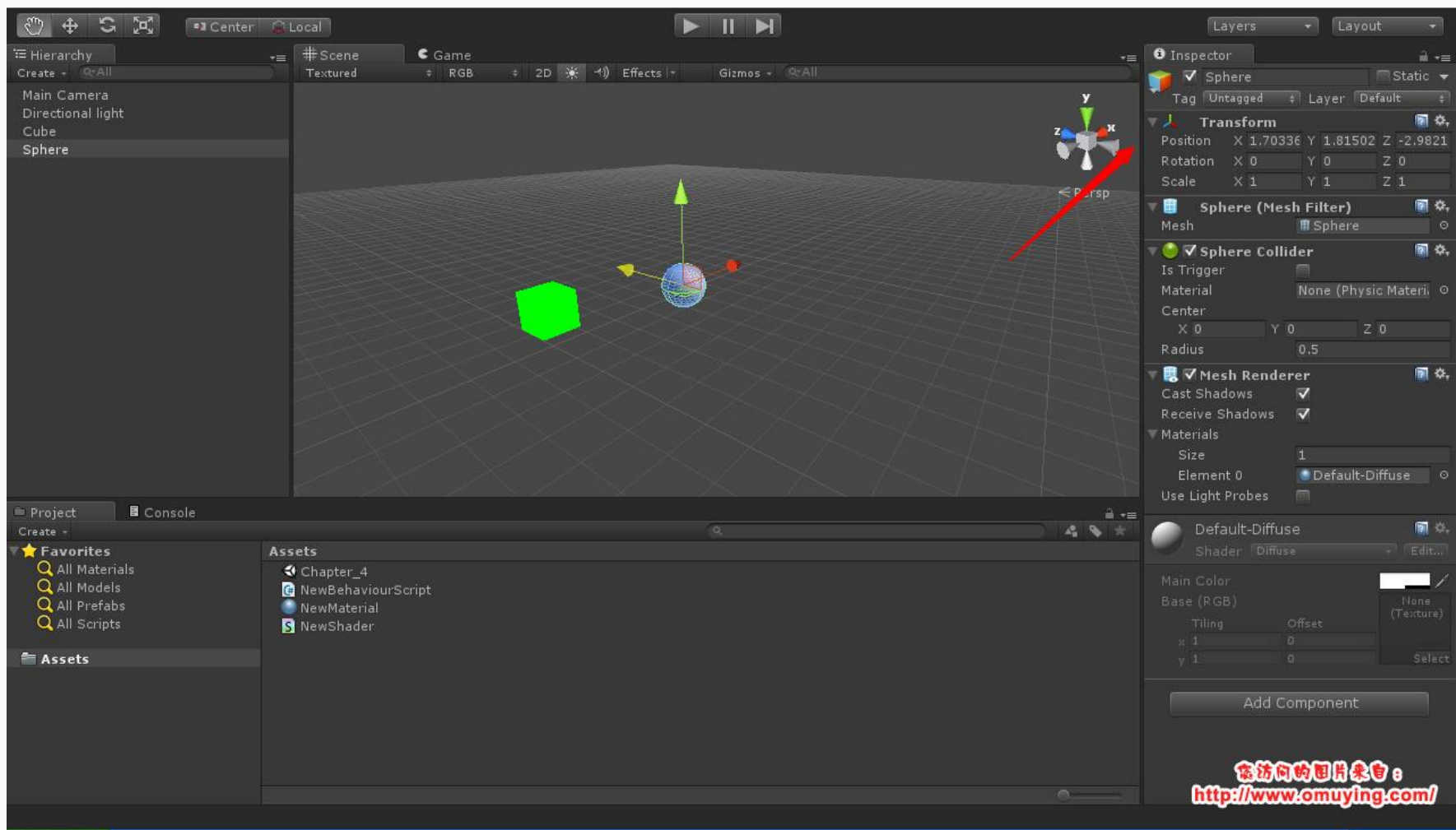
01  using UnityEngine;
02  using System.Collections;
03
04  [ExecuteInEditMode]
05  public class NewBehaviourScript : MonoBehaviour
06  {
07      public GameObject other;
08
09      void Update()
10      {
11          if(other != null)
12          {
13              this.renderer.sharedMaterial.SetVector("_Point",
14              other.transform.position);
15          }
16      }
17  }

```

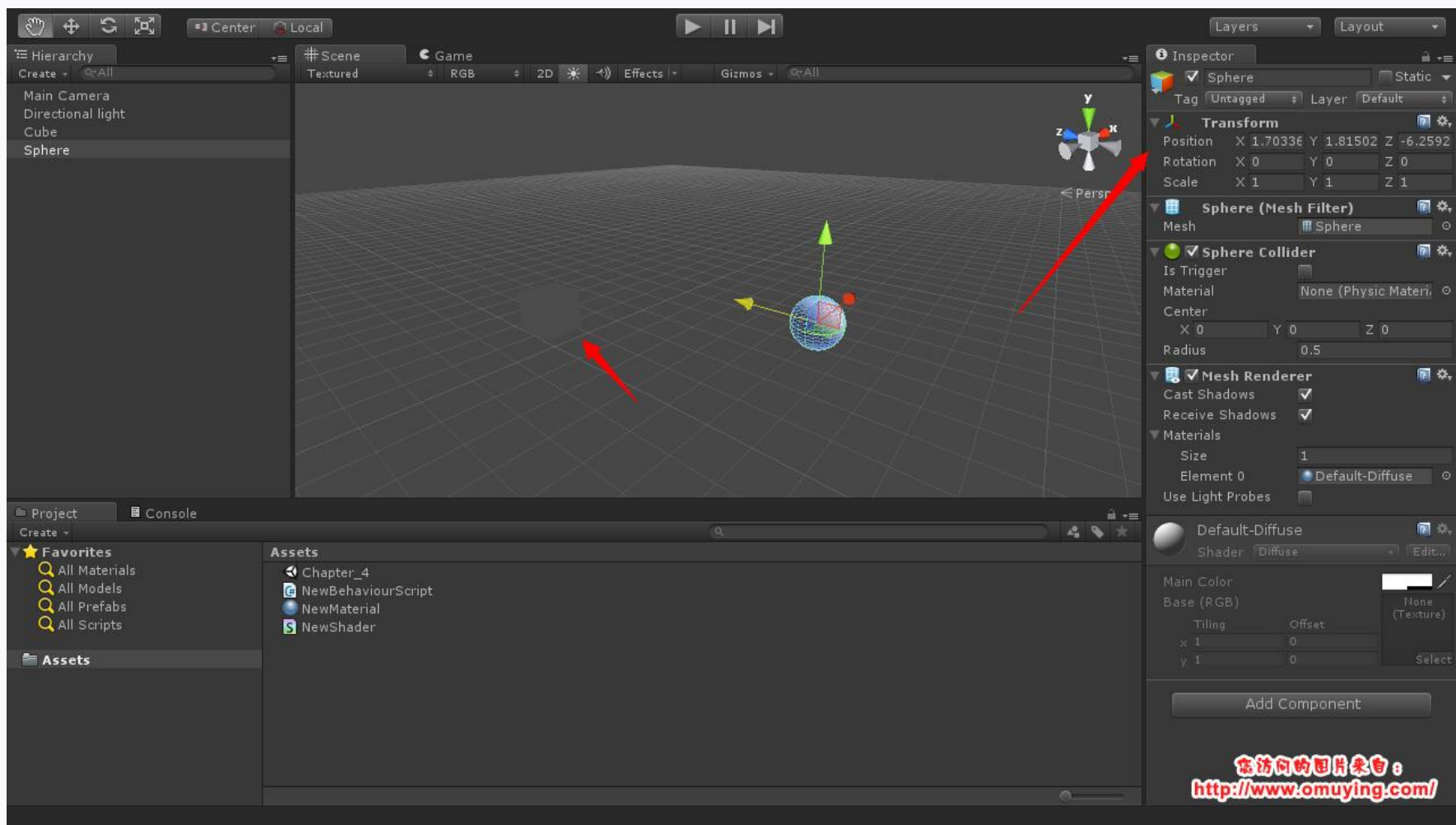
然后你可以把这个脚本挂载到一个对象上面，并且在 Inspector 视图中设置变量 other 的值为另一个对象，如图：



然后我们只需要移动另一个对象（ Sphere ）就可以看到（ Cube ）颜色的变化，当小球离原点近时，如图：



当小球离原点远时，如图：



恭喜你，本教程你应该学会了：

- 1、怎样转换一个顶点到世界坐标。
- 2、Unity 内置的 uniforms。

资源下载地址：[点击下载](#)，共下载 16 次。

前一篇：[第三章节：在着色器中调试（关于顶点输入参数）](#)

后一篇：[第五章节：剖面模型（关于片段擦除和三角形面剔除）](#)



赞

11 人



打酱油

0 人



呵呵

0 人



鄙视

0 人



正能量

0 人



0

3 条评论

最新 最早 最热



changok88

这是我查到POSITION与SV_POSITION的区别介绍，如果有和我一样不明白的可以进来看看，http://blog.csdn.net/zhao_92221/article/details/46797969

2015年10月14日

回复

顶

转发



Jasmine_Gui

本人在unity5.1.2中实验，this.renderer会报错，要用this.GetComponent<Renderer>()

2015年12月10日

回复

顶

转发



965

GetComponent<Renderer>().sharedMaterial

8月18日

回复

顶

转发

社交帐号登录:



微信



微博



QQ



人人

更多»



说点什么吧...



发布

最终幻想正在使用多说

3 条评论

最新 最早 最热



changok88

这是我查到POSITION与SV_POSITION的区别介绍，如果有和我一样不明白的可以进来看看，http://blog.csdn.net/zhao_92221/article/details/46797969

2015年10月14日

回复

顶

转发



Jasmine_Gui

本人在unity5.1.2中实验，this.renderer会报错，要用this.GetComponent<Renderer>()

2015年12月10日

回复

顶

转发



965

GetComponent<Renderer>().sharedMaterial

8月18日

回复

顶

转发

社交帐号登录:



微信



微博



QQ



人人

更多»



说点什么吧...



发布

