

【翻译】第三章节：在着色器中调试（关于顶点输入参数）

2014-11-24 08:41:00 2500 人阅读 [Unity3D](#) [cg](#) [顶点输出参数](#)

A⁻ A⁺

文章内容	例子源码	网友评论	最后编辑：2014-12-21 17:28:52
本文永久地址： http://www.omuying.com/article/91.aspx ，【 文章转载请注明出处！ 】			

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

基于《[最简单的着色器](#)》和《[RGB 立方体](#)》章节，本教程将讨论着色器中的顶点输入参数。

这个教程还介绍了如何在 Unity 中调试着色器，我们给可视化组件的片段颜色设置一个假色图像。然后根据组件中的生成图像的颜色强度来决定着色器中值的设定，这是一种非常原始的调试技巧，但是在 Unity 中我们别无选择。

顶点数据从哪里获得

在《[RGB 立方体](#)》章节中，你已经知道片段着色器如何从顶点着色器的顶点输出参数结构体中获得数据，但这儿有个问题就是我们应该从哪儿获得顶点着色器的数据？答案是在 Unity 中，一个游戏对象的 MeshRender 组件在每帧都会发送游戏对象的网格数据到 OpenGL（这被称为一次“draw call”，注意，每次 draw call 都有一些性能开销，因此，每次给 OpenGL 提交大网格数据（一次 draw call）比每次提交几个小的网格数据（多次 draw call）更有效率），这些数据通常包括一个三角形列表，并且每个三角形都包括三个顶点，而且每个顶点都有一定的属性，比如位置等。这些属性来自顶点着色器的顶点输入参数，在 Cg 中顶点输入参数的属性通常通过语义映射的方式来实现，即，每个顶点输入参数都有一个特定的语义，例如：POSITION, NORMAL, TEXCOORD0, TEXCOORD1, TANGENT, COLOR 等等，然而在 Unity 的 Cg 中有一些特定的实现方式，内置顶点输入参数都有一个特定的名称。

Unity 内置顶点输入参数以及如何使用

在 Unity 中，内置顶点输入参数（位置，表面的法线、纹理坐标、切线向量和顶点颜色）不仅要具备一定的语义，还要有一定的名称和类型，此外，他们还应该被包含在顶点输入参数的结构体中。例如：

```
01 struct vertexInput
02 {
03     float4 vertex : POSITION; // position (in object coordinates,
04     // i.e. local or model coordinates)
05     float4 tangent : TANGENT;
06     // vector orthogonal to the surface normal
07     float3 normal : NORMAL; // surface normal vector (in object
08     // coordinates; usually normalized to unit length)
09     float4 texcoord : TEXCOORD0; // 0th set of texture
10     // coordinates (a.k.a. "UV"; between 0 and 1)
11     float4 texcoord1 : TEXCOORD1; // 1st set of texture
12     // coordinates (a.k.a. "UV"; between 0 and 1)
13     fixed4 color : COLOR; // color (usually constant)
14 };
```

我们可以像下面代码中的那样来使用这个结构体：

```
01 Shader "Cg shader with all built-in vertex input parameters"
02 {
03     SubShader
04     {
05         Pass
06         {
07             CGPROGRAM
08
09             #pragma vertex vert
10             #pragma fragment frag
11
12             struct vertexInput
13             {
```



伪装微型摄像机



针孔摄像头



无线监控摄像机

外企猎头公司

unity3d移动

微型摄像机

unity3d摄像机

猎头公司排名


室内设计师




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

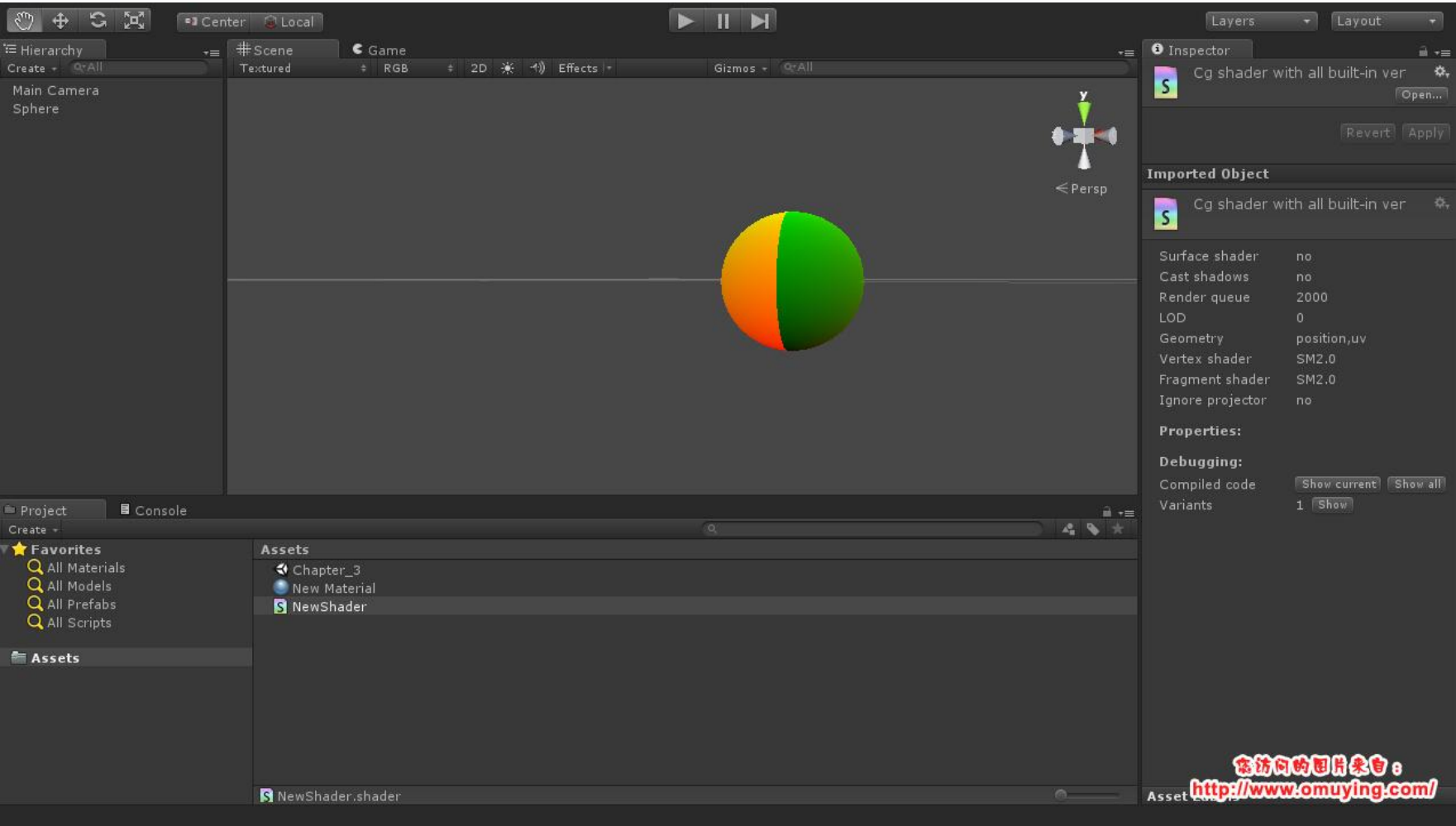


广告

随机阅读

```
14         float4 vertex : POSITION;
15         float4 tangent : TANGENT;
16         float3 normal : NORMAL;
17         float4 texcoord : TEXCOORD0;
18         float4 texcoord1 : TEXCOORD1;
19         fixed4 color : COLOR;
20     };
21     struct vertexOutput
22     {
23         float4 pos : SV_POSITION;
24         float4 col : TEXCOORD0;
25     };
26
27     vertexOutput vert(vertexInput input)
28     {
29         vertexOutput output;
30
31         output.pos = mul(UNITY_MATRIX_MVP, input.vertex);
32         output.col = input.texcoord; // set the output color
33
34         // other possibilities to play with:
35
36         // output.col = input.vertex;
37         // output.col = input.tangent;
38         // output.col = float4(input.normal, 1.0);
39         // output.col = input.texcoord;
40         // output.col = input.texcoord1;
41         // output.col = input.color;
42
43         return output;
44     }
45
46     float4 frag(vertexOutput input) : COLOR
47     {
48         return input.col;
49     }
50     ENDCG
51 }
52 }
53 }
```

这段代码效果如图：



在《[RGB 立方体](#)》章节中，我们已经知道如何在顶点坐标中设置片段颜色值，在这个例子中，片段颜色被设置成了纹理坐标，这样我们可以看到在 Unity 中提供了什么样的纹理坐标。

注意，切线只有前三个分量表示切线方向，第四个分量表示缩放，这在处理视差贴图（parallax mapping）时比较有用。详情可以查看《[Projection of Bumpy Surfaces](#)》。

预定义输入结构

通常情况下，你只须指定真正需要的顶点输入参数来获得更高的性能，例如：位置、法线和一组纹理坐标，有时候你可能还需要切线向量，Unity 在 UnityCG.cginc 文件中提供了最常见的预定义输入结构：
appdata_base、appdata_tan 和 appdata_full。

```
01 struct appdata_base
02 {
03     float4 vertex : POSITION;
```

- 【翻译】第四章：世界空间中的着色器（关于 uniforms） - 2327 次阅读
- 【翻译】第二十一章：凹凸表面投影（关于视差贴图） - 1462 次阅读
- 【翻译】第二章：RGB 立方体（关于顶点输出参数） - 2257 次阅读
- 【原创】Shader 内置 Shader 之 Vertex Lit 学习 - 3365 次阅读
- 【翻译】第二十五章：弧形玻璃（关于折射贴图） - 1722 次阅读

```
04 float3 normal : NORMAL;
05 float4 texcoord : TEXCOORD0;
06 };
07 struct appdata_tan
08 {
09     float4 vertex : POSITION;
10     float4 tangent : TANGENT;
11     float3 normal : NORMAL;
12     float4 texcoord : TEXCOORD0;
13 };
14 struct appdata_full
15 {
16     float4 vertex : POSITION;
17     float4 tangent : TANGENT;
18     float3 normal : NORMAL;
19     float4 texcoord : TEXCOORD0;
20     float4 texcoord1 : TEXCOORD1;
21     fixed4 color : COLOR;
22     // and additional texture coordinates only on XBOX360
23 };
```

因此，我们可以改写上面的着色器文件，在上面的着色器代码文件中添加 #include "UnityCG.cginc"。

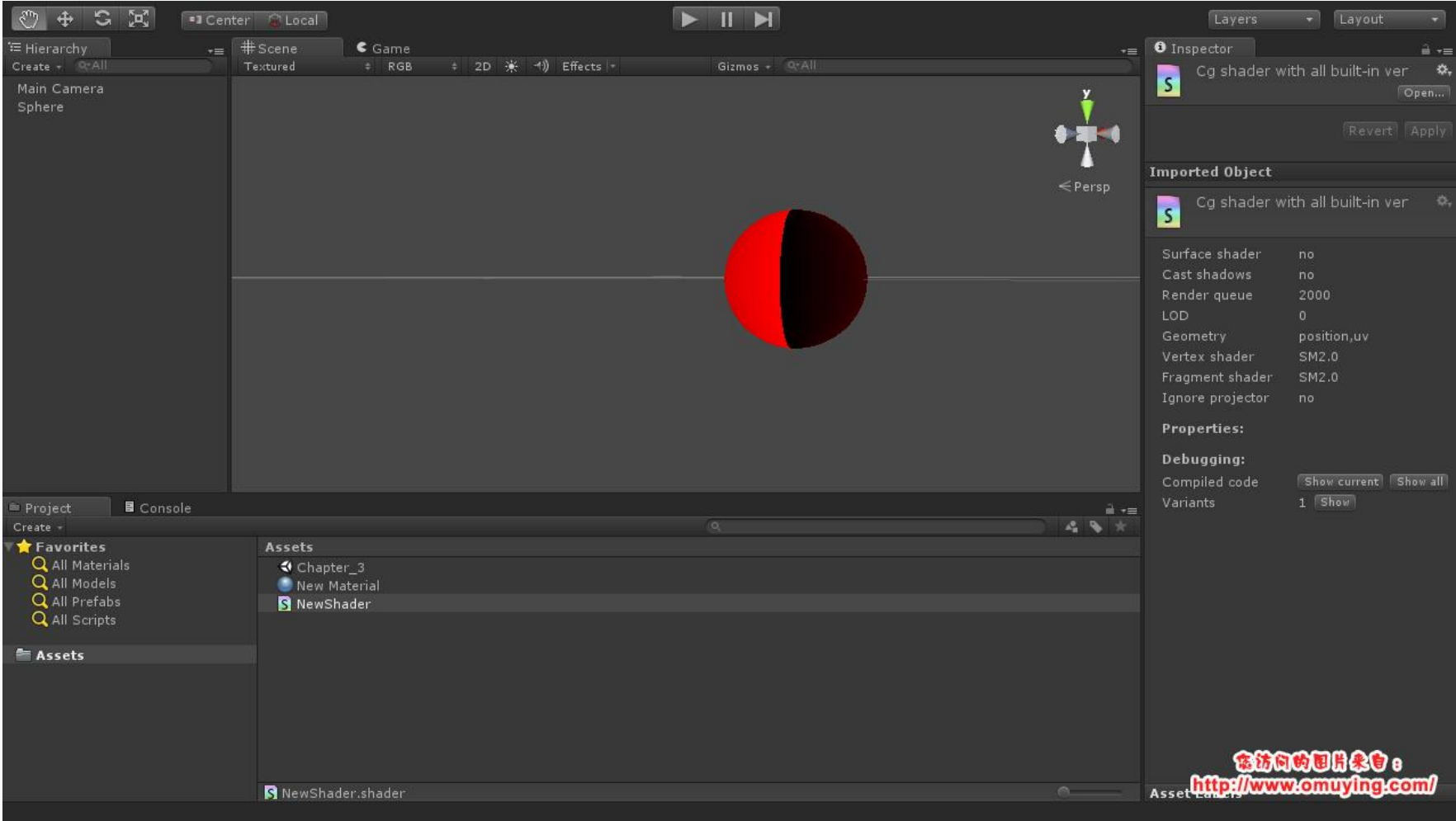
```
01 Shader "Cg shader with all built-in vertex input parameters"
02 {
03     SubShader
04     {
05         Pass
06         {
07             CGPROGRAM
08
09             #pragma vertex vert
10             #pragma fragment frag
11             #include "UnityCG.cginc"
12
13             struct vertexOutput
14             {
15                 float4 pos : SV_POSITION;
16                 float4 col : TEXCOORD0;
17             };
18
19             vertexOutput vert(appdata_full input)
20             {
21                 vertexOutput output;
22
23                 output.pos = mul(UNITY_MATRIX_MVP, input.vertex);
24                 output.col = input.texcoord;
25
26                 return output;
27             }
28
29             float4 frag(vertexOutput input) : COLOR
30             {
31                 return input.col;
32             }
33             ENDCG
34         }
35     }
36 }
```

什么是假色图像

我们可以通过只关注一个颜色分量，来理解假色图像中的信息，例如，如果顶点输入参数用 TEXCOORD0 来为一个球体写入片段颜色并用 texcoord 的 x 坐标来作为片段的红色分量，那么，不管输出的颜色值是纯红色、黄色还是品红色，他们的红色分量都是 1。相反的，不论是蓝色、绿色还是任何强度青绿色中的红色分量，他们的红色分量都是 0。如果你以前不知道只关注一个颜色分量，这对你可能有些挑战，因此，你可以通过下面的例子来学习如何只关注一个颜色分量，在顶点着色器中使用这行代码去设置输出参数：

```
1 output.col = float4(input.texcoord.x, 0.0, 0.0, 1.0);
```

这行代码使用 texcoord 的 x 分量来设置输出参数的红色分量，并且把绿色和蓝色的分量设置为 0，效果如图：

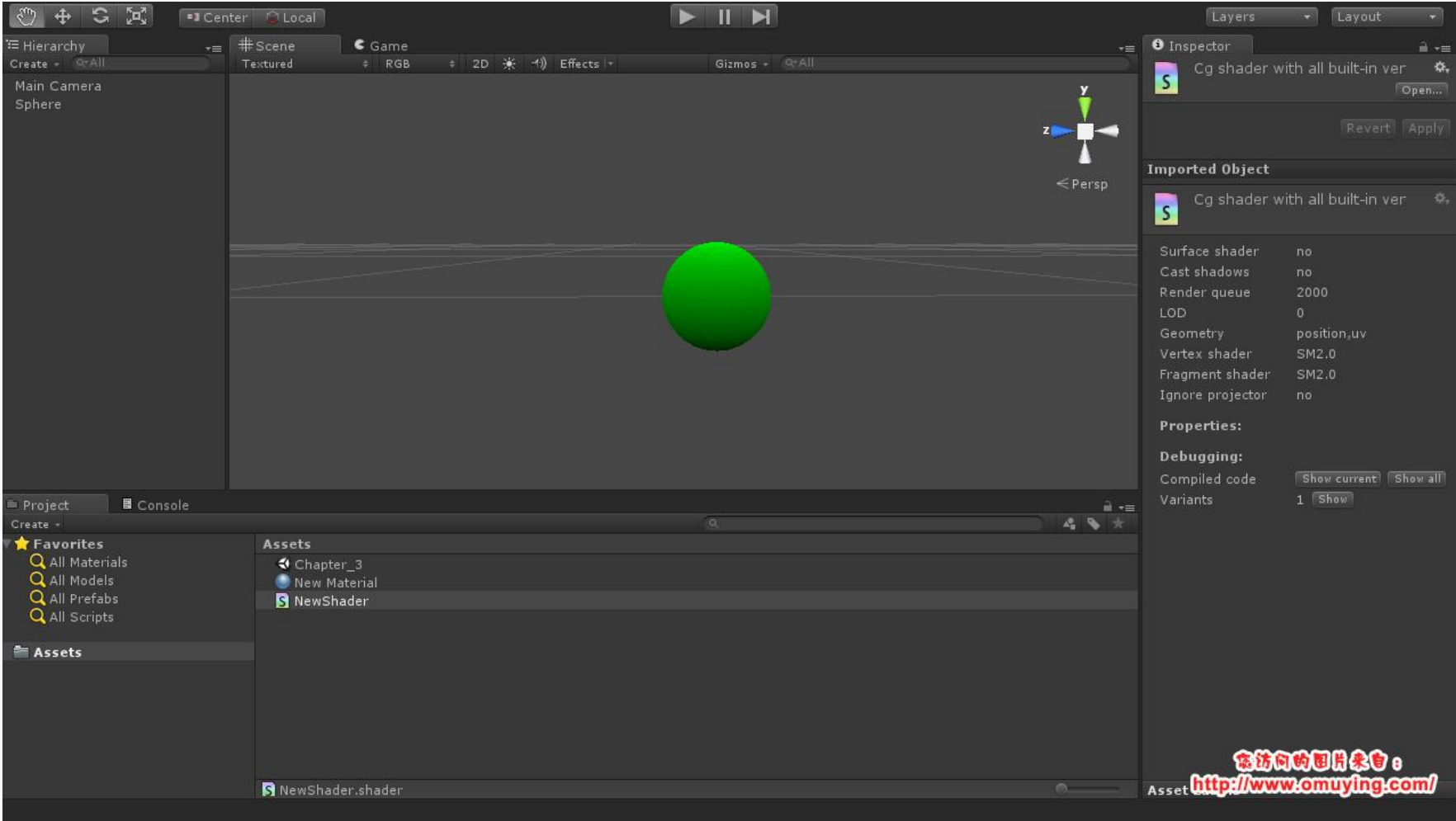


如果你只关注红色分量或者只显示红色分量，你可以看到球从 360 度下降到 0 度之后它的值也从 0 升到 1。它实际上非常类似于行星表面的经纬度坐标（在球面坐标中，它对应方位）。

如果 texcoord 的 x 分量对应经度，那么人们往往会期望 y 分量可能对应纬度，但是需要注意的是，纹理坐标值的范围始终是 0 至 1。因此它的值从下到上是 0 至 1，你可以想像 y 分量为绿色时的情况：

```
1 | output.col = float4(0.0, input.texcoord.y, 0.0, 1.0);
```

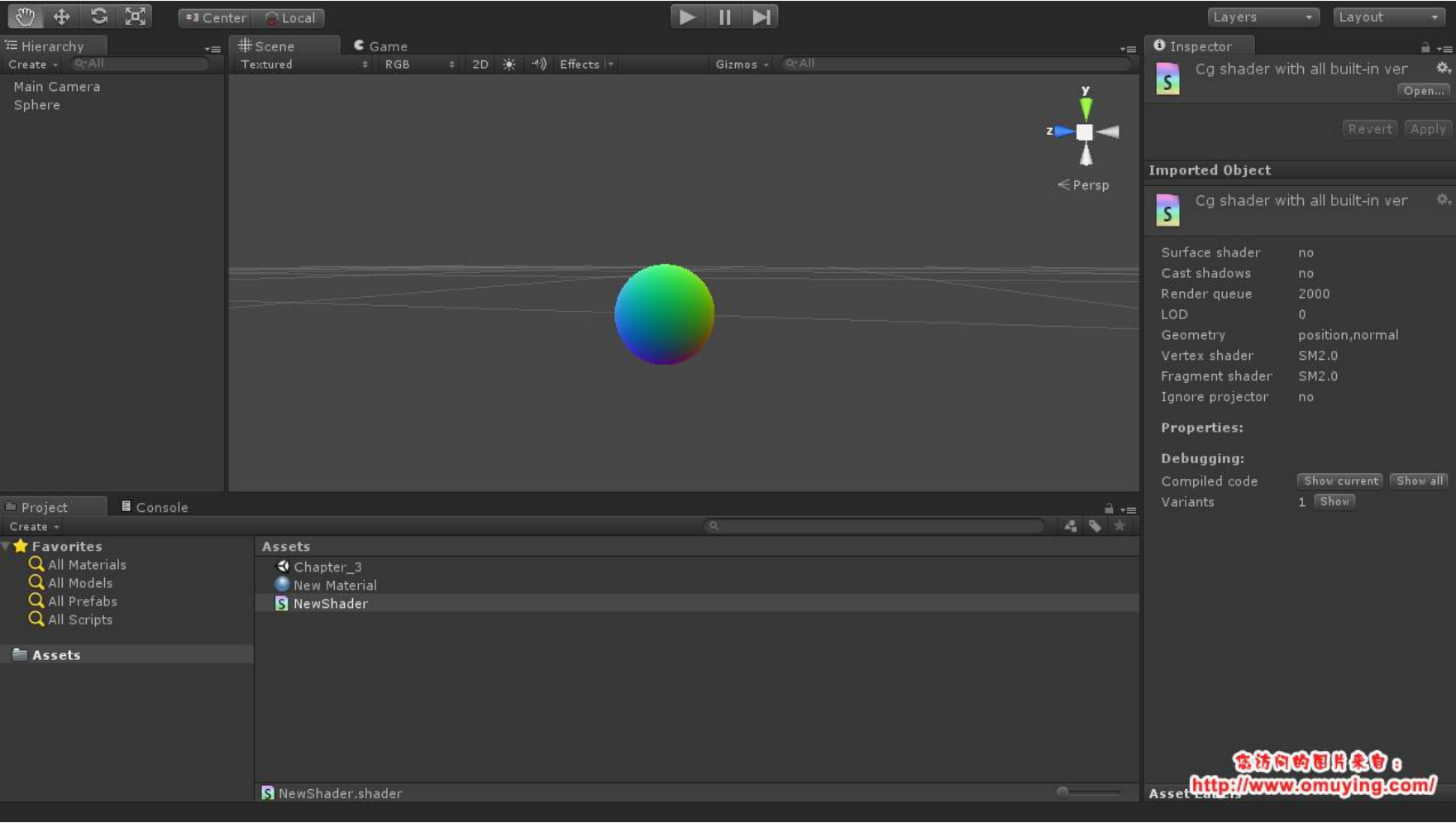
效果如图：



正如颜色分量，纹理坐标的值介于 0 至 1 之间。因为被规范化的向量坐标的值是介绍 -1 至 1 之间，所以您可以通过在每个分量上面加上 1 然后再除以 2 来把他们的值映射到 0 至 1。例如：

```
1 | output.col = float4((input.normal + float3(1.0, 1.0, 1.0)) / 2.0, 1.0);
```

效果如图：



注意，法线（normal）是一个三维向量。每个分量的坐标范围是黑色 -1 至全亮 +1。

如果你所使用数值的范围超出了 0 至 1 或者 -1 至 +1，你必须把它的颜色分量值的范围映射到 0 至 1，其实在着色器中如果你指定的颜色分量值的范围超出了 0 至 1，程序会自动把范围限制在 0 至 1，即：如果值小于 0，则为 0，如果值大于 1，则为 1，但是你至少应该知道如果颜色分量的取值范围不在 0 至 1 之间，那么你应该把这个值映射到 0 至 1。

调试练习

为了练习在着色器中进行调试，下面包含了几行不同的代码，你可以把下面的代码依次替换到上面的顶点着色器代码中并查看结果，然后思考为什么结果是黑色，你甚至可以把值的范围不设置在 0 至 1 之间，并查看结果。更多函数和操作符可以查看《[Vector and Matrix Operations](#)》。

```
1 output.col = input.texcoord - float4(1.5, 2.3, 1.1, 0.0);
1 output.col = input.texcoord.zzzz;
1 output.col = input.texcoord / tan(0.0);
```

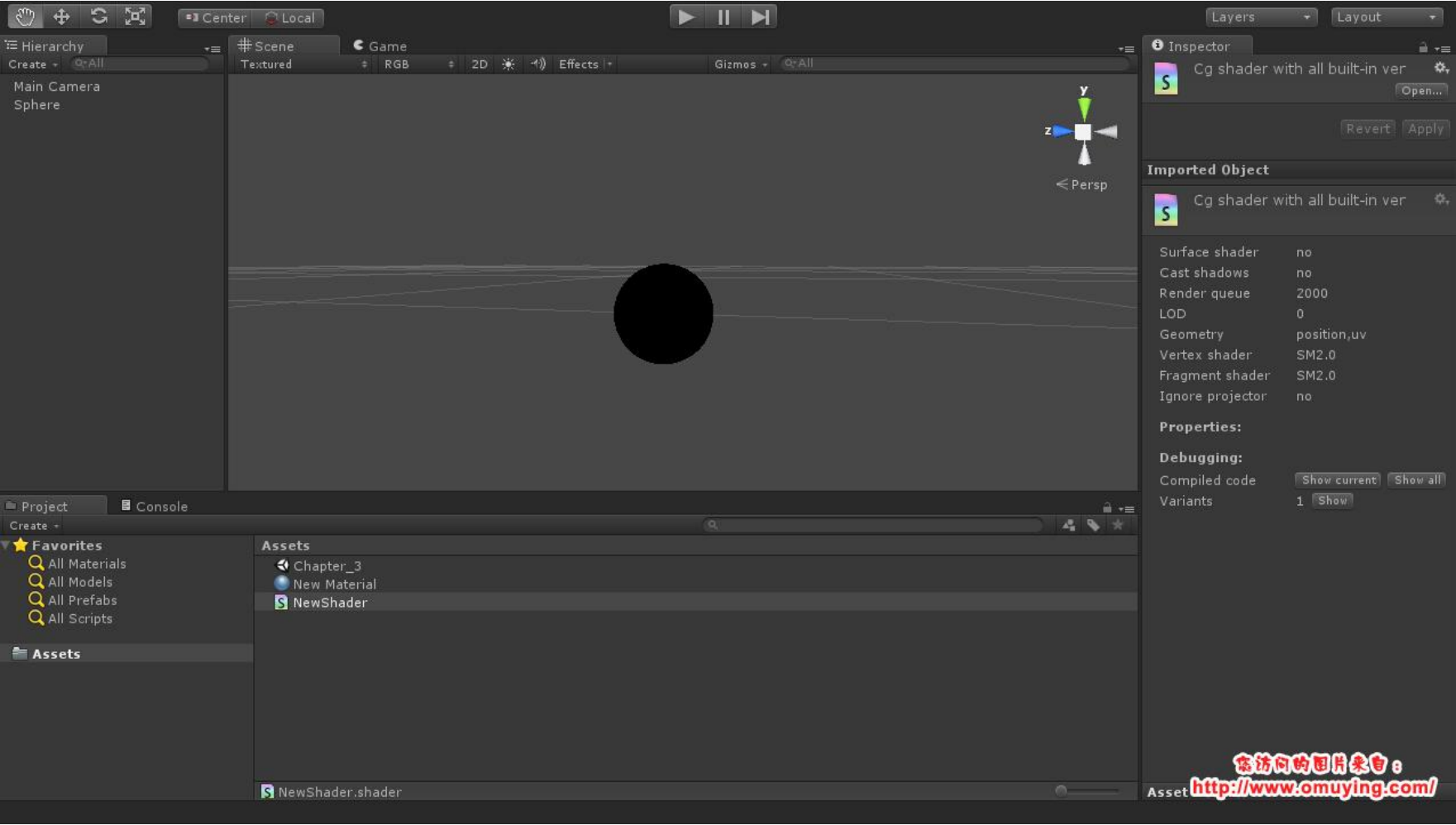
下面几行代码使用了 dot 和 cross 函数：

```
1 output.col = dot(input.normal, input.tangent.xyz) * input.texcoord;
1 output.col = dot(cross(input.normal, input.tangent.xyz), input.normal) * input.texcoord;
1 output.col = float4(cross(input.normal, input.normal), 1.0);
1 output.col = float4(cross(input.normal, input.vertex.xyz), 1.0);
2 // only for a sphere!
```

使用 radians() 还是返回黑色吗？为什么呢？

```
1 output.col = radians(input.texcoord);
```

上面几行的代码效果如图：



恭喜你，学完本节，你应该了解：

- 1、Unity 内置的顶点输入参数列表。
- 2、根据参数如何设置片段输出的颜色分量。

资源下载地址：[点击下载](#)，共下载 31 次。

前一篇：[Shader 内置 Shader 之 Parallax Diffuse 学习](#)

后一篇：[第四章节：世界空间中的着色器（关于 uniforms）](#)



赞

9 人



打酱油

0 人



呵呵

0 人



鄙视

0 人



正能量

0 人



0条评论

最新 最早 最热

还没有评论，沙发等你来抢

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

0条评论

最新 最早 最热

还没有评论，沙发等你来抢

社交帐号登录: 微信 微博 QQ 人人 [更多»](#)



说点什么吧...



发布

