首页 Unity3D Shader

.Net(C#)

英语

其他 源码

用户登录

# 【翻译】第二十三章节:投影(关于使用投影纹理贴图实现投影)

2014-12-21 08:34:00 2520 人阅读 Unity3D cg 投影

 $A^ A^+$ 

文章内容

例子源码

网友评论

最后编辑: 2014-12-21 10:12:22

本文永久地址:http://www.omuying.com/article/114.aspx , 【文章转载请注明出处!】

原文链接:http://en.wikibooks.org/wiki/Cg\_Programming/Unity/Projectors

本教程介绍投影,它是 Unity 的特定渲染组件,本篇教程基于《Cookies》章节,如果你没有阅读过,你应 该先阅读它。

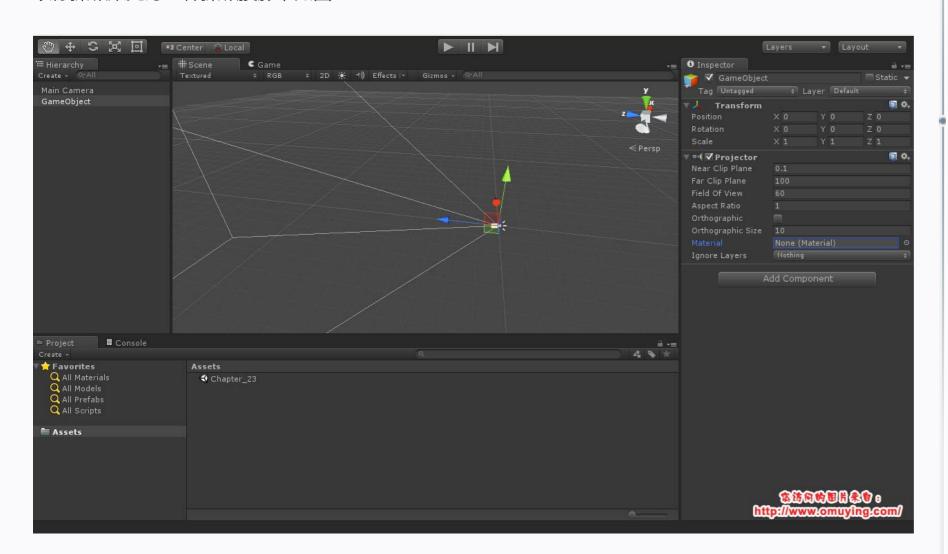
### Unity 中的投影

Unity 中的投影有点类似于聚光灯,事实上,它们可以被用于类似的应用,但还是有一个重要的区别:对于 聚光灯,在《Cookies》章节中我们知道,所有被聚光灯照亮的对象着色器,我们必须计算聚光灯的光照, 如果一个对象的着色器忽略聚光灯,它只是不会被聚光灯照亮,这于投影不同:每个投影都与一个着色器和 一个材质关系,并且这个材质被应用到投影范围内的任何对象上,因此一个对象的着色器不需要处理投影, 相反,投影用其着色器在投影范围内的所有对象上作为附加的渲染 pass 来获得一定的效果,它通过添加投 影图像光照或者衰减物体颜色来产生假的阴影,事实上,投影着色器通过使用不同的混合公式来实现这些效 果。

有时人们甚至考虑把投影作为更加"自然"的方式来实现光照,但是光与材质之间的相互作用通常是特定于 每个材质的,而单个投影着色器是不能处理所有这些差异的。投影的三个基本实现方式为:给对象添加光、 调节对象的颜色、或者同时给对象添加光以及调节对象的颜色,在本例中我们通过给对象添加光和衰减对象 的颜色来实现投影。

## 添加光实现投影

为了创建投影,我们可以选择 GameObject -> Create Empty 菜单来创建一个空对象,然后选中这个新创 建的对象,接着我们在主菜单栏选择 Component -> Effects -> Projector 来给对象添加投影,现在你可 以像操纵聚光灯一样操纵投影,如图:



在 Inspector 视图中设置投影的详细介绍可以查看《Unity's reference manual》章节,在本章节中,我们 着重考虑的是投影的材质,因为它将被应用在投影范围内的所有对象上,所以我们需要创建一个材质并给它



【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



【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



猎头公司排名

企业名录

西安猎头公司



找保姆照顾老人

房地产猎头

室内设计师

随机阅读

分配一个合适的着色器,通常这个着色器不需要访问游戏对象的材质,因为它把材质应用到其他对象上,因此这个着色器不需要访问游戏对象的纹理,同时这个着色器也不需要获得光源的任何信息,然而它可以访问到游戏对象的顶点属性以及它自身的着色器属性。

着色器给对象添加光可以用来在这个对象上投影任何的影像,这有点类似于投影仪或者电影放映机,因此我们可以使用类似于《Cookies》章节中的聚光灯 cookies 纹理图像,但有所不同的是纹理图像的 RGB 颜色也可以被添加,这样做可以用来实现彩色投影,为此我们可以通过将片段颜色设置为纹理图像的 RGBA 颜色并使用混合公式

#### 1 | Blend One One

这个公式只在帧缓冲的颜色中添加了片段颜色(根据纹理图像的不同,我们可以使用 Blend SrcAlpha One 来移除透明度为 0 的颜色)。

还有一点与聚光灯的 Cookie 不同,我们应该使用 Unity 指定的 uniform \_Projector 矩阵代替 \_LightMatrix0 矩阵来把位置从对象空间转换到投影空间,然而投影空间中的坐标与光空间的坐标非常相似 ——除了把 x 和 y 坐标限制在正确的范围内,因此我们不需要再给 x 和 y 坐标加上 0.5 了,但是还需要 除以 w 坐标(对于投影纹理贴图总是如此),因此我们需要把 x 和 y 坐标除以 w 坐标或者使用 tex2Dproj。

行 ZWrite Off 确保我们不改变深度缓存,因为我们只是给已栅格化的网格添加光,Offset -1, -1 用来改变在网格前面添加光的深度,这样确保我们现在进行栅格化没有网格阻塞,着色器的代码如下:

```
Shader "Cg projector shader for adding light"
  02
      {
  03
         Properties
  04
            _ShadowTex ("Projected Image", 2D) = "white" {}
  05
  06
  07
         SubShader
08
         {
  09
            Pass
10
  11
               Blend One One
               // add color of _ShadowTex to the color in the framebuffer
12
  13
               ZWrite Off // don't change depths
14
               Offset -1, -1 // avoid depth fighting
  15
16
               CGPROGRAM
  17
  18
               #pragma vertex vert
  19
               #pragma fragment frag
  20
  21
               // User-specified properties
               uniform sampler2D _ShadowTex;
  22
  23
               // Projector-specific uniforms
  25
               uniform float4x4 _Projector; // transformation matrix
  26
               // from object space to projector space
  27
  28
               struct vertexInput
  29
  30
                  float4 vertex : POSITION;
  31
                  float3 normal : NORMAL;
  32
  33
               struct vertexOutput
  34
  35
                  float4 pos : SV_POSITION;
  36
                  float4 posProj : TEXCOORD0;
  37
                  // position in projector space
  38
               };
  39
  40
               vertexOutput vert(vertexInput input)
  41
  42
                  vertexOutput output;
  43
  44
                  output.posProj = mul(_Projector, input.vertex);
                  output.pos = mul(UNITY_MATRIX_MVP, input.vertex);
  45
                  return output;
  46
  47
               }
  48
               float4 frag(vertexOutput input) : COLOR
  49
  50
  51
                  if (input.posProj.w > 0.0) // in front of projector?
  52
  53
                     return tex2D(_ShadowTex, input.posProj.xy / input.posProj.w);
  54
                     // alternatively: return tex2Dproj(
  55
                     // _ShadowTex, input.posProj);
  56
```

新无图片

【原创】Shader 内置 Shader 之 Bumped Diffuse 学习 - 1707 次 阅读

Wo)

【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】第二十章节:凹凸表面光照(关于法线贴图) - 1920次阅读

暂无图片

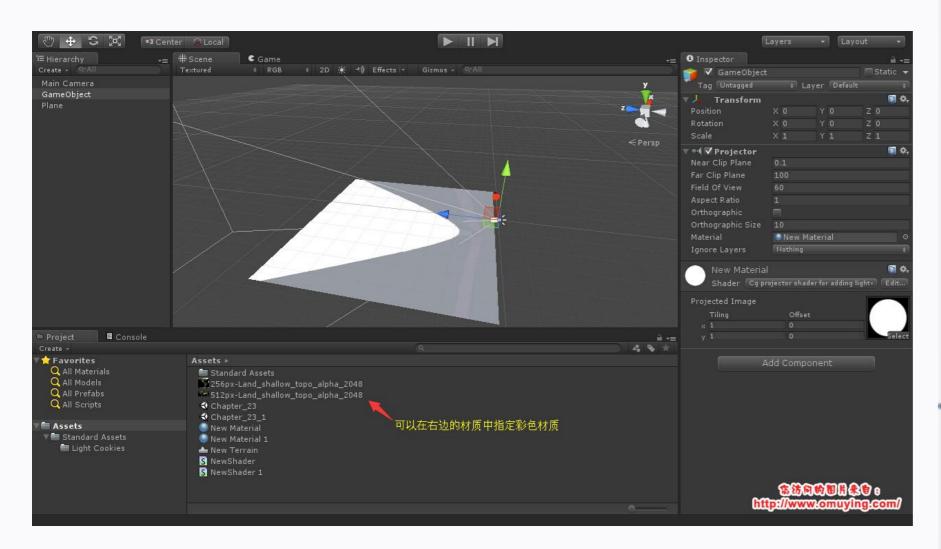
【翻译】第二十四章节:表 面反射(关于反射贴 图) - 1441次阅读

暂无图片

【翻译】第十三章节:双面平滑表面(关于双面每像素光照) - 1100次阅读

```
57
                else // behind projector
58
59
                    return float4(0.0, 0.0, 0.0, 0.0);
60
61
62
             ENDCG
63
64
65
       // The definition of a fallback shader should be commented out
66
       // during development:
       // Fallback "Projector/Light"
68
69 }
```

注意,我们必须检测 w 是正值(即在片段在投影的前面不是后面),如果不检测,投影会在物体的背后添加光,此外对于换行模式纹理图像必须是正方形,另外着色器纹理图像命名为 \_ShadowTex 是为了与内置投影着色器相兼容,这个着色器的效果如图:



#### 衰减颜色实现投影

通过调节颜色来创建投影的步骤与上面基本相同,唯一的区别是着色器的代码,下面的例子通过衰减颜色来添加阴影,特别是地板的颜色,注意在实际应用中,投影对象的颜色不应该被衰减,这可以通过为投影对象指定一个层(在 Inspector 视图中)然后在 Inspector 视图投影的选项中设置忽略这个层来实现。

为了给影子一定的形状,我们使用纹理图像的 alpha 分量来确定阴影的黑暗程度(我们可以使用标准资源库中的光 Cookie 纹理),为了衰减帧缓冲中的颜色,我们需与 1 减 alpha 相乘(即因子 alpha 为 0 时它等于 1),因此混合的公式为:

#### 1 | Blend Zero OneMinusSrcAlpha

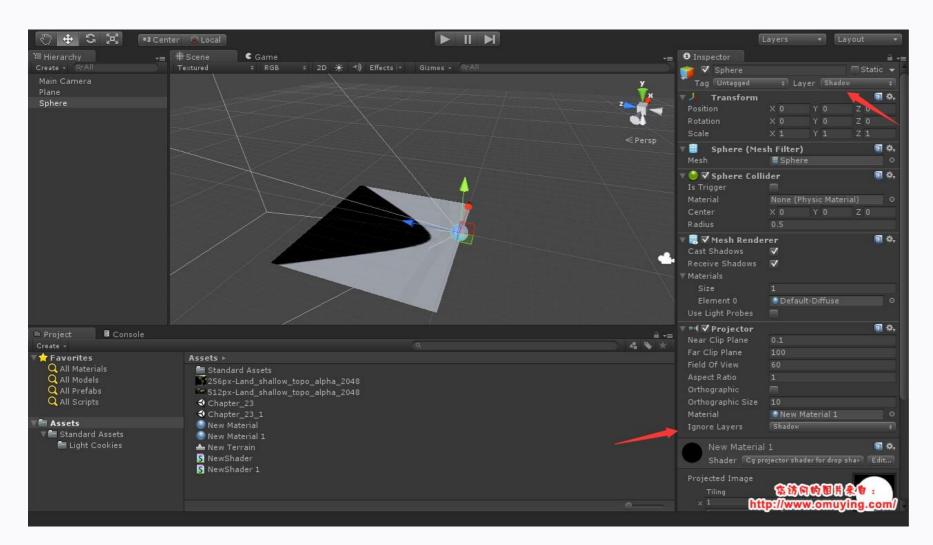
零表明我们不添加任何光线,即使阴影很深也没有光线被添加,相反在片段着色器中 alpha 分量应该被减小,这个可以通过与一个小于 1 的数相乘来实现,在帧缓冲中独立调节颜色分量,我们需要使用混合 Blend Zero SrcColor 或者 Blend Zero OneMinusSrcColor。

对于添加光使用不同的混合公式实际上只是为了对着色器效果进行比较,着色器的代码如下:

```
Shader "Cg projector shader for drop shadows"
02
       Properties
03
04
          _ShadowTex ("Projected Image", 2D) = "white" {}
05
06
       SubShader
07
98
          Pass
09
10
             Blend Zero OneMinusSrcColor // attenuate color in framebuffer
11
             // by 1 minus alpha of _ShadowTex
12
13
             ZWrite Off // don't change depths
14
             Offset -1, -1 // avoid depth fighting
15
             CGPROGRAM
16
17
18
             #pragma vertex vert
```

```
19
             #pragma fragment frag
20
21
             // User-specified properties
             uniform sampler2D _ShadowTex;
23
24
             // Projector-specific uniforms
25
             uniform float4x4 _Projector; // transformation matrix
26
             // from object space to projector space
27
28
             struct vertexInput
29
             {
30
                float4 vertex : POSITION;
31
                float3 normal : NORMAL;
32
33
             struct vertexOutput
34
35
                float4 pos : SV_POSITION;
36
                float4 posProj : TEXCOORD0;
37
                // position in projector space
38
             };
39
40
             vertexOutput vert(vertexInput input)
41
42
                vertexOutput output;
43
44
                output.posProj = mul(_Projector, input.vertex);
45
                output.pos = mul(UNITY_MATRIX_MVP, input.vertex);
46
                return output;
47
48
49
             float4 frag(vertexOutput input) : COLOR
50
51
                if (input.posProj.w > 0.0) // in front of projector?
52
53
                   return tex2D(_ShadowTex, input.posProj.xy / input.posProj.w);
54
                   // alternatively: return tex2Dproj(
55
                   // _ShadowTex, input.posProj);
56
57
                else // behind projector
58
59
                   return float4(0.0, 0.0, 0.0, 0.0);
60
61
62
             ENDCG
63
64
65
66
       // The definition of a fallback shader should be commented out
67
       // during development:
68
       // Fallback "Projector/Light"
69 }
```

### 这个着色器的效果如图:



# 恭喜你,在本章节中,你应该了解:

- 1、Unity 投影的工作方式。
- 2、如何通过添加光实现投影。
- 3、如果通过衰减对象颜色实现投影。

