用户登录

首页 Unity3D

Shader

.Net(C#)

英语

其他

源码

【翻译】第十六章节:表面纹理光照(关于纹理漫射光照)

2014-12-08 08:15:00 1307 人阅读 Unity3D cg 表面纹理光照

A- A+

文章内容

例子源码

网友评论

最后编辑: 2014-12-21 18:29:31

本文永久地址:http://www.omuying.com/article/105.aspx,【文章转载请注明出处!】

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

本教程介绍 per-vertex lighting of textured surfaces。

本教程结合了《纹理球》和《镜面高光》章节中的着色器代码,然后通过纹理的材质颜色来计算光照,如果你还没有读过这两个章节,请先阅读他们。

纹理和每顶点漫反射光照

在《纹理球》章节中我们知道纹理颜色最后通过片段着色器中输出,因此在光照计算的参数中也有可能会使用纹理颜色,尤其是在漫反射中,经常会使用纹理颜色来作为材质常数 $k_{\rm diffuse}$, Phong reflection model 中漫反射计算公式为:

 $I_{\text{diffuse}} = I_{\text{incoming}} k_{\text{diffuse}} \max(0, \mathbf{N} \cdot \mathbf{L})$

在这个公式中,红、绿、蓝颜色分量上的材质常数可以由纹理来确定。

着色器代码

027

与《镜面高光》章节中的每顶点光照相比,这个例子的顶点着色器还需要计算两个额外的颜色:diffuseColor 和 specularColor,它们使用语义词 TEXCOORD1 和 TEXCOORD2。

参数 diffuseColor 在片段着色器中乘以纹理颜色,但是 specularColor 仅仅是镜面术语,它不应该和纹理颜色相乘,其实这样做是比较合适的,这有时候也被称为"separate specular color",事实上 Unity 的 ShaderLab 也有一个叫做"SeparateSpecular"的选项来激活或者关闭它。

需要注意的是 _Color 属性用来与 diffuseColor 的各个分量相乘,因此它的作用是作为一个颜色过滤器来调整纹理的颜色,此外,我们还需要一个与着色器属性相同的 fallback 着色器 (fallback的讨论 可以查看《漫反射》章节),着色器的代码如下:

```
Shader "Cg per-vertex lighting with texture"
001
002
003
        Properties
004
           MainTex ("Texture For Diffuse Material Color", 2D) = "white" {}
005
006
           _Color ("Overall Diffuse Color Filter", Color) = (1,1,1,1)
           _SpecColor ("Specular Material Color", Color) = (1,1,1,1)
007
           Shininess ("Shininess", Float) = 10
800
009
010
        SubShader
011
012
           Pass
013
              Tags { "LightMode" = "ForwardBase" }
014
              // pass for ambient light and first light source
015
016
017
              CGPROGRAM
018
019
              #pragma vertex vert
              #pragma fragment frag
020
021
              #include "UnityCG.cginc"
022
              uniform float4 LightColor0;
023
              // color of light source (from "Lighting.cginc")
024
025
026
              // User-specified properties
```

uniform sampler2D MainTex;



暂无图片

【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

暂无图片

【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



随机阅读

```
uniform float4 _Color;
028
              uniform float4 _SpecColor;
029
              uniform float _Shininess;
030
031
032
              struct vertexInput
033
              {
                 float4 vertex : POSITION;
034
035
                 float3 normal : NORMAL;
                 float4 texcoord : TEXCOORDO;
036
037
              };
038
              struct vertexOutput
039
040
                 float4 pos : SV_POSITION;
                 float4 tex : TEXCOORD0;
041
042
                 float3 diffuseColor : TEXCOORD1;
043
                 float3 specularColor : TEXCOORD2;
044
              };
045
046
              vertexOutput vert(vertexInput input)
047
048
                 vertexOutput output;
049
050
                 float4x4 modelMatrix = _Object2World;
                 float4x4 modelMatrixInverse = _World2Object;
051
052
                 // multiplication with unity_Scale.w is unnecessary
053
                 // because we normalize transformed vectors
054
055
                 float3 normalDirection = normalize(mul(float4(input.normal, 0.0),
     modelMatrixInverse).xyz);
                 float3 viewDirection = normalize(_WorldSpaceCameraPos -
056
     mul(modelMatrix, input.vertex).xyz);
                 float3 lightDirection;
057
                 float attenuation;
058
059
                 if (0.0 == _WorldSpaceLightPos0.w) // directional light?
060
061
                    attenuation = 1.0; // no attenuation
062
063
                    lightDirection = normalize(_WorldSpaceLightPos0.xyz);
064
065
                 else // point or spot light
066
067
                    float3 vertexToLightSource = _WorldSpaceLightPos0.xyz -
     mul(modelMatrix, input.vertex).xyz;
                    float distance = length(vertexToLightSource);
068
069
                    attenuation = 1.0 / distance; // linear attenuation
070
                    lightDirection = normalize(vertexToLightSource);
071
072
073
                 float3 ambientLighting = UNITY_LIGHTMODEL_AMBIENT.rgb *
     _Color.rgb;
074
075
                 float3 diffuseReflection = attenuation * _LightColor0.rgb *
     _Color.rgb * max(0.0, dot(normalDirection, lightDirection));
076
077
                 float3 specularReflection;
078
                 if (dot(normalDirection, lightDirection) < 0.0) // light source on</pre>
     the wrong side?
079
080
                    specularReflection = float3(0.0, 0.0, 0.0);
                    // no specular reflection
081
082
                 else // light source on the right side
083
084
                    specularReflection = attenuation * _LightColor0.rgb *
085
     _SpecColor.rgb * pow(max(0.0, dot(reflect(-lightDirection, normalDirection),
     viewDirection)), _Shininess);
086
087
088
                 output.diffuseColor = ambientLighting + diffuseReflection;
                 output.specularColor = specularReflection;
089
                 output.tex = input.texcoord;
090
091
                 output.pos = mul(UNITY_MATRIX_MVP, input.vertex);
092
                 return output;
093
              }
094
095
              float4 frag(vertexOutput input) : COLOR
096
097
                 return float4(input.specularColor + input.diffuseColor *
     tex2D(_MainTex, input.tex.xy), 1.0);
098
099
100
              ENDCG
101
           }
102
103
           Pass
104
105
              Tags { "LightMode" = "ForwardAdd" }
106
              // pass for additional light sources
107
              Blend One One // additive blending
108
109
              CGPROGRAM
110
111
              #pragma vertex vert
```

【翻译】第二十六章节:天

空盒(关于用环境贴图渲 染背景) - 2194 次阅

读

【翻译】第十四章节:多个 灯(关于在一个 pass 中 遍历处理多个光

【原创】Shader 内置 Shader 之 Vertex Lit 学 习 - 3365 次阅读

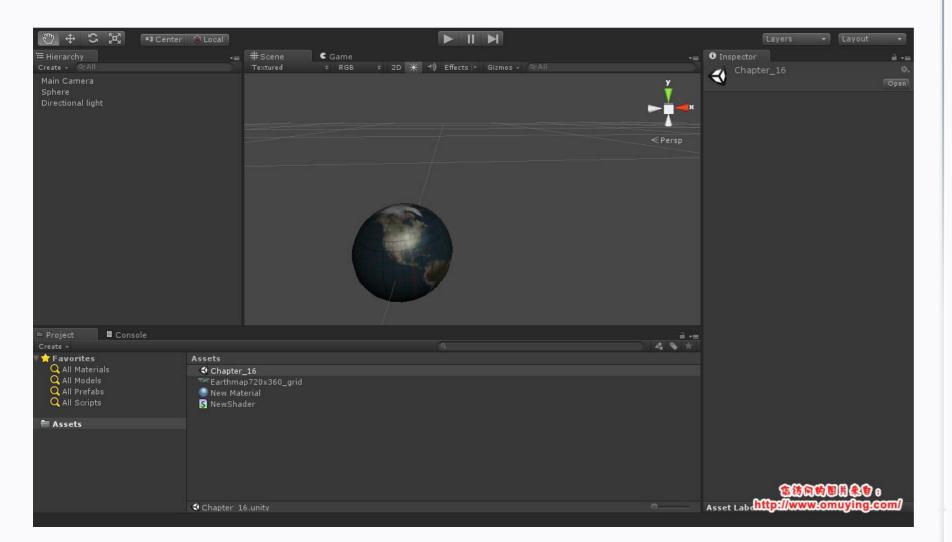
源) - 1932 次阅读

【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

【翻译】第十七章节:光泽 纹理 (关于光泽贴 图) - 1201 次阅读

```
#pragma fragment frag
112
113
              #include "UnityCG.cginc"
114
115
              uniform float4 _LightColor0;
              // color of light source (from "Lighting.cginc")
116
117
118
              // User-specified properties
              uniform sampler2D MainTex;
119
120
              uniform float4 _Color;
              uniform float4 _SpecColor;
121
122
              uniform float _Shininess;
123
124
              struct vertexInput
125
              {
126
                 float4 vertex : POSITION;
127
                 float3 normal : NORMAL;
128
                 float4 texcoord : TEXCOORD0;
129
              };
130
              struct vertexOutput
131
132
                 float4 pos : SV_POSITION;
133
                 float4 tex : TEXCOORD0;
134
                 float3 diffuseColor : TEXCOORD1;
135
                 float3 specularColor : TEXCOORD2;
136
              };
137
138
              vertexOutput vert(vertexInput input)
139
140
                 vertexOutput output;
141
142
                 float4x4 modelMatrix = _Object2World;
                 float4x4 modelMatrixInverse = _World2Object;
143
                 // multiplication with unity_Scale.w is unnecessary
144
145
                 // because we normalize transformed vectors
146
147
                 float3 normalDirection = normalize(mul(float4(input.normal, 0.0),
     modelMatrixInverse).xyz);
                 float3 viewDirection = normalize(_WorldSpaceCameraPos -
148
     mul(modelMatrix, input.vertex).xyz);
                 float3 lightDirection;
149
                 float attenuation;
150
151
152
                 if (0.0 == _WorldSpaceLightPos0.w) // directional light?
153
154
                    attenuation = 1.0; // no attenuation
155
                    lightDirection = normalize(_WorldSpaceLightPos0.xyz);
156
157
                 else // point or spot light
158
159
                    float3 vertexToLightSource = _WorldSpaceLightPos0.xyz -
     mul(modelMatrix, input.vertex).xyz;
                    float distance = length(vertexToLightSource);
160
                    attenuation = 1.0 / distance; // linear attenuation
161
162
                    lightDirection = normalize(vertexToLightSource);
163
164
165
                 float3 diffuseReflection = attenuation * _LightColor0.rgb *
     _Color.rgb * max(0.0, dot(normalDirection, lightDirection));
166
                 float3 specularReflection;
167
                 if (dot(normalDirection, lightDirection) < 0.0) // light source on</pre>
168
     the wrong side?
169
                    specularReflection = float3(0.0, 0.0, 0.0);
170
                    // no specular reflection
171
172
173
                 else // light source on the right side
174
                    specularReflection = attenuation * LightColor0.rgb *
175
     _SpecColor.rgb * pow(max(0.0, dot(reflect(-lightDirection, normalDirection),
     viewDirection)), _Shininess);
176
177
                 output.diffuseColor = diffuseReflection; // no ambient
178
                 output.specularColor = specularReflection;
179
                 output.tex = input.texcoord;
180
181
                 output.pos = mul(UNITY_MATRIX_MVP, input.vertex);
182
                 return output;
183
184
185
              float4 frag(vertexOutput input) : COLOR
186
187
                 return float4(input.specularColor + input.diffuseColor *
     tex2D(_MainTex, input.tex.xy), 1.0);
188
189
190
              ENDCG
191
           }
192
        // The definition of a fallback shader should be commented out
193
        // during development:
194
        // Fallback "Specular"
195
196 }
```

如何把纹理图像应用到这个着色器,你可以参考《纹理球》章节,效果如图:



恭喜你,在本章节中你应该了解:

- 1、纹理与每顶点光照如何结合。
- 2、什么是 "separate specular color" 。

资源下载地址:点击下载,共下载14次。

前一篇:第十五章节:纹理球(关于纹理球面) 后一篇:第十七章节:光泽纹理(关于光泽贴图)



3人



打酱油 0人



呵呵 0人



鄙视 0人



正能量

0人

最新 最早 最热



























3条评论



博主,请问你知道纹理贴图的UV坐标是怎么映射的吗?当UV值增大不知道贴图如何移动,不 知道怎么映射的,

2015年1月23日 回复 顶 转发



歪妖内涵网

好东西 谢谢分享

2015年9月21日 回复 顶 转发



CTQDn

这个更刺j激,准备好手纸哦A片。。hTTp://uVU.cc/igSW



24小时前 回复 顶 参转发

社交帐号登录: 🙄 微信 🚮 微博 🚨 QQ 👢 人人 更多»













最终幻想 - 个人博客 | 关于网站 | 联系我们 | 友情链接 | 网站声明

Copyright © 2012-2016 最终幻想 - 个人博客 苏ICP备09017906号-5