

随机阅读



【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 中遍历处理多个光源） - 1932 次阅读



【原创】Shader 内置 Shader 之 Vertex Lit 学习 - 3365 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

```
028     uniform float4 _Color;
029     uniform float4 _SpecColor;
030     uniform float _Shininess;
031
032     struct vertexInput
033     {
034         float4 vertex : POSITION;
035         float3 normal : NORMAL;
036         float4 texcoord : TEXCOORD0;
037     };
038     struct vertexOutput
039     {
040         float4 pos : SV_POSITION;
041         float4 tex : TEXCOORD0;
042         float3 diffuseColor : TEXCOORD1;
043         float3 specularColor : TEXCOORD2;
044     };
045
046     vertexOutput vert(vertexInput input)
047     {
048         vertexOutput output;
049
050         float4x4 modelMatrix = _Object2World;
051         float4x4 modelMatrixInverse = _World2Object;
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);
056         float3 viewDirection = normalize(_WorldSpaceCameraPos -
mul(modelMatrix, input.vertex).xyz);
057         float3 lightDirection;
058         float attenuation;
059
060         if (0.0 == _WorldSpaceLightPos0.w) // directional light?
061         {
062             attenuation = 1.0; // no attenuation
063             lightDirection = normalize(_WorldSpaceLightPos0.xyz);
064         }
065         else // point or spot light
066         {
067             float3 vertexToLightSource = _WorldSpaceLightPos0.xyz -
mul(modelMatrix, input.vertex).xyz;
068             float distance = length(vertexToLightSource);
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
the wrong side?
079         {
080             specularReflection = float3(0.0, 0.0, 0.0);
081             // no specular reflection
082         }
083         else // light source on the right side
084         {
085             specularReflection = attenuation * _LightColor0.rgb *
_SpecColor.rgb * pow(max(0.0, dot(reflect(-lightDirection, normalDirection),
viewDirection)), _Shininess);
086         }
087
088         output.diffuseColor = ambientLighting + diffuseReflection;
089         output.specularColor = specularReflection;
090         output.tex = input.texcoord;
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
```

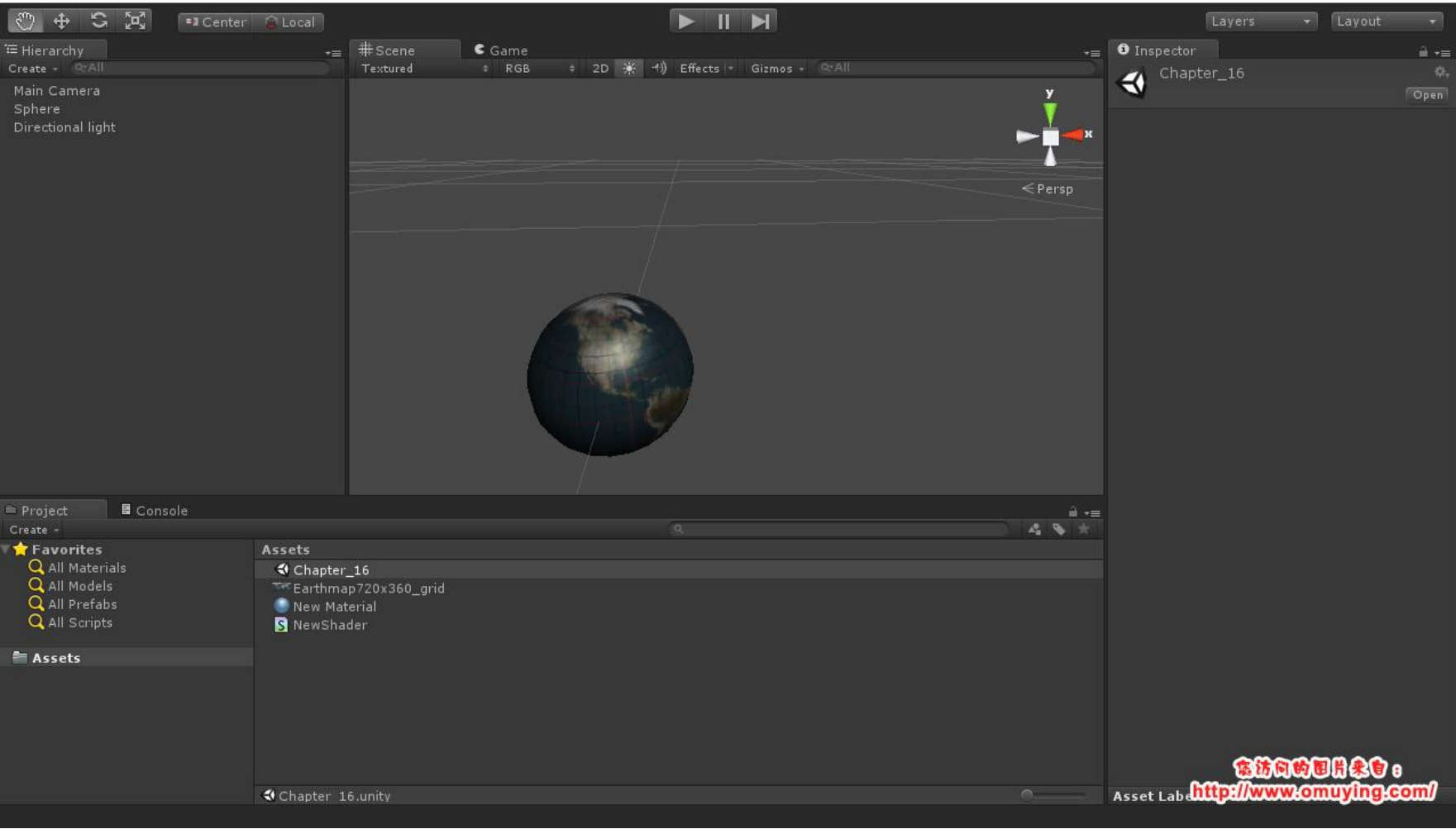


```

112 #pragma fragment frag
113
114 #include "UnityCG.cginc"
115 uniform float4 _LightColor0;
116 // color of light source (from "Lighting.cginc")
117
118 // User-specified properties
119 uniform sampler2D _MainTex;
120 uniform float4 _Color;
121 uniform float4 _SpecColor;
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;
143     float4x4 modelMatrixInverse = _World2Object;
144     // multiplication with unity_Scale.w is unnecessary
145     // because we normalize transformed vectors
146
147     float3 normalDirection = normalize(mul(float4(input.normal, 0.0),
modelMatrixInverse).xyz);
148     float3 viewDirection = normalize(_WorldSpaceCameraPos -
mul(modelMatrix, input.vertex).xyz);
149     float3 lightDirection;
150     float attenuation;
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;
160         float distance = length(vertexToLightSource);
161         attenuation = 1.0 / distance; // linear attenuation
162         lightDirection = normalize(vertexToLightSource);
163     }
164
165     float3 diffuseReflection = attenuation * _LightColor0.rgb *
_Color.rgb * max(0.0, dot(normalDirection, lightDirection));
166
167     float3 specularReflection;
168     if (dot(normalDirection, lightDirection) < 0.0) // light source on
the wrong side?
169     {
170         specularReflection = float3(0.0, 0.0, 0.0);
171         // no specular reflection
172     }
173     else // light source on the right side
174     {
175         specularReflection = attenuation * _LightColor0.rgb *
_SpecColor.rgb * pow(max(0.0, dot(reflect(-lightDirection, normalDirection),
viewDirection)), _Shininess);
176     }
177
178     output.diffuseColor = diffuseReflection; // no ambient
179     output.specularColor = specularReflection;
180     output.tex = input.texcoord;
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 }
193 // The definition of a fallback shader should be commented out
194 // during development:
195 // Fallback "Specular"
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）](#)



赞

3 人



打酱油

0 人



呵呵

0 人



鄙视

0 人



正能量

0 人



3 条评论

最新 最早 最热



薛儒麟

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

这个更刺激，准备好手纸哦 A 片。。 hTTp://uVU.cc/igSW




24小时前 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



薛儒麟

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

这个更刺激，准备好手纸哦 A 片。。 hTTp://uVU.cc/igSW




24小时前  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