

随机阅读

```

001 Shader "Cg per-pixel lighting with texture"
002 {
003     Properties
004     {
005         _MainTex ("RGBA Texture For Material Color", 2D) = "white" {}
006         _Color ("Diffuse Material Color", Color) = (1,1,1,1)
007         _SpecColor ("Specular Material Color", Color) = (1,1,1,1)
008         _Shininess ("Shininess", Float) = 10
009     }
010     SubShader
011     {
012         Pass
013         {
014             Tags { "LightMode" = "ForwardBase" }
015             // pass for ambient light and first light source
016
017             CGPROGRAM
018
019             #pragma vertex vert
020             #pragma fragment frag
021
022             #include "UnityCG.cginc"
023             uniform float4 _LightColor0;
024             // color of light source (from "Lighting.cginc")
025
026             // User-specified properties
027             uniform sampler2D _MainTex;
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 posWorld : TEXCOORD0;
042                 float3 normalDir : TEXCOORD1;
043                 float4 tex : 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                 output.posWorld = mul(modelMatrix, input.vertex);
056                 output.normalDir = normalize(mul(float4(input.normal, 0.0),
057 modelMatrixInverse).xyz);
058                 output.tex = input.texcoord;
059                 output.pos = mul(UNITY_MATRIX_MVP, input.vertex);
060                 return output;
061             }
062
063             float4 frag(vertexOutput input) : COLOR
064             {
065                 float3 normalDirection = normalize(input.normalDir);
066
067                 float3 viewDirection = normalize(_WorldSpaceCameraPos -
068 input.posWorld.xyz);
069                 float3 lightDirection;
070                 float attenuation;
071
072                 float4 textureColor = tex2D(_MainTex, input.tex.xy);
073
074                 if (0.0 == _WorldSpaceLightPos0.w) // directional light?
075                 {
076                     attenuation = 1.0; // no attenuation
077                     lightDirection = normalize(_WorldSpaceLightPos0.xyz);
078                 }
079                 else // point or spot light
080                 {
081                     float3 vertexToLightSource = _WorldSpaceLightPos0.xyz -
082 input.posWorld.xyz;
083                     float distance = length(vertexToLightSource);
084                     attenuation = 1.0 / distance; // linear attenuation
085                     lightDirection = normalize(vertexToLightSource);
086                 }
087
088                 float3 ambientLighting = textureColor.rgb *
089 UNITY_LIGHTMODEL_AMBIENT.rgb * _Color.rgb;
090
091                 float3 diffuseReflection = textureColor.rgb * attenuation *
092 _LightColor0.rgb * _Color.rgb * max(0.0, dot(normalDirection,
093 lightDirection));

```



暂无图片

【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



暂无图片

【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



暂无图片

【翻译】第四章节：世界空间中的着色器（关于 uniforms） - 2327 次阅读



暂无图片

【翻译】第七章节：顺序无关的透明度（关于顺序无关的混合） - 1978 次阅读



暂无图片

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


```

088         float3 specularReflection;
089         if (dot(normalDirection, lightDirection) < 0.0) // light source on
the wrong side?
091         {
092             specularReflection = float3(0.0, 0.0, 0.0);
093             // no specular reflection
094         }
095         else // light source on the right side
096         {
097             // for usual gloss maps: "... * textureColor.a"
098             specularReflection = attenuation * _LightColor0.rgb *
_SpecColor.rgb * (1.0 - textureColor.a) * pow(max(0.0, dot( reflect(-
lightDirection, normalDirection), viewDirection)), _Shininess);
099         }
100
101         return float4(ambientLighting + diffuseReflection +
specularReflection, 1.0);
102     }
103
104     ENDCG
105 }
106
107 Pass
108 {
109     Tags { "LightMode" = "ForwardAdd" }
110     // pass for additional light sources
111     Blend One One // additive blending
112
113     CGPROGRAM
114
115     #pragma vertex vert
116     #pragma fragment frag
117
118     #include "UnityCG.cginc"
119     uniform float4 _LightColor0;
120     // color of light source (from "Lighting.cginc")
121
122     // User-specified properties
123     uniform sampler2D _MainTex;
124     uniform float4 _Color;
125     uniform float4 _SpecColor;
126     uniform float _Shininess;
127
128     struct vertexInput
129     {
130         float4 vertex : POSITION;
131         float3 normal : NORMAL;
132         float4 texcoord : TEXCOORD0;
133     };
134     struct vertexOutput
135     {
136         float4 pos : SV_POSITION;
137         float4 posWorld : TEXCOORD0;
138         float3 normalDir : TEXCOORD1;
139         float4 tex : TEXCOORD2;
140     };
141
142     vertexOutput vert(vertexInput input)
143     {
144         vertexOutput output;
145
146         float4x4 modelMatrix = _Object2World;
147         float4x4 modelMatrixInverse = _World2Object;
148         // multiplication with unity_Scale.w is unnecessary
149         // because we normalize transformed vectors
150
151         output.posWorld = mul(modelMatrix, input.vertex);
152         output.normalDir = normalize(mul(float4(input.normal, 0.0),
modelMatrixInverse).xyz);
153         output.tex = input.texcoord;
154         output.pos = mul(UNITY_MATRIX_MVP, input.vertex);
155         return output;
156     }
157
158     float4 frag(vertexOutput input) : COLOR
159     {
160         float3 normalDirection = normalize(input.normalDir);
161
162         float3 viewDirection = normalize(_WorldSpaceCameraPos -
input.posWorld.xyz);
163         float3 lightDirection;
164         float attenuation;
165
166         float4 textureColor = tex2D(_MainTex, input.tex.xy);
167
168         if (0.0 == _WorldSpaceLightPos0.w) // directional light?
169         {
170             attenuation = 1.0; // no attenuation
171             lightDirection = normalize(_WorldSpaceLightPos0.xyz);
172         }
173         else // point or spot light
174         {

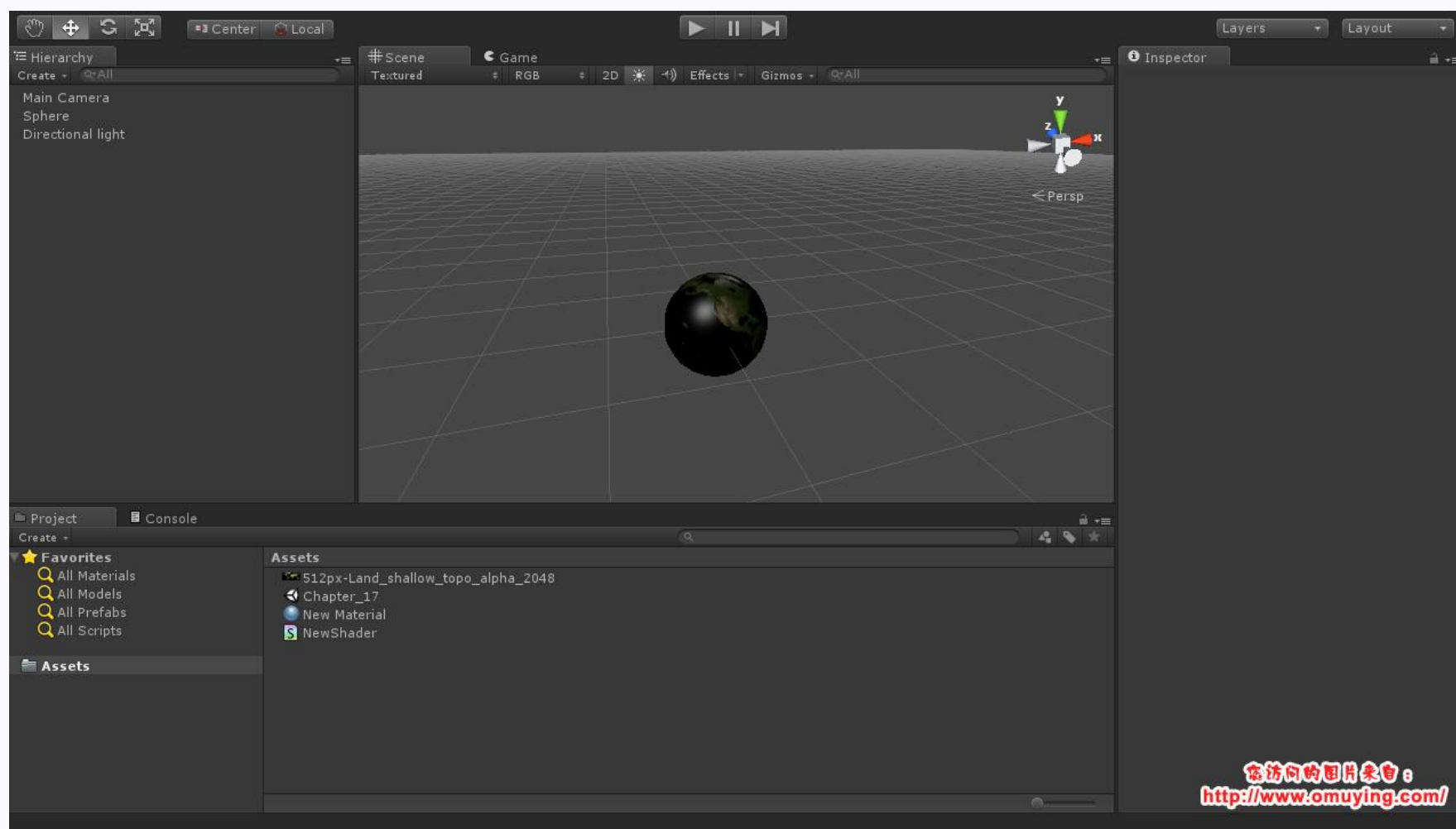
```

```

175     float3 vertexToLightSource = _WorldSpaceLightPos0.xyz -
input.posWorld.xyz;
176     float distance = length(vertexToLightSource);
177     attenuation = 1.0 / distance; // linear attenuation
178     lightDirection = normalize(vertexToLightSource);
179 }
180
181     float3 diffuseReflection = textureColor.rgb * attenuation *
_LightColor0.rgb * _Color.rgb * max(0.0, dot(normalDirection,
lightDirection));
182
183     float3 specularReflection;
184     if (dot(normalDirection, lightDirection) < 0.0) // light source on
the wrong side?
185     {
186         specularReflection = float3(0.0, 0.0, 0.0);
187         // no specular reflection
188     }
189     else // light source on the right side
190     {
191         // for usual gloss maps: "... * textureColor.a"
192         specularReflection = attenuation * _LightColor0.rgb *
_SpecColor.rgb * (1.0 - textureColor.a) * pow(max(0.0, dot(reflect(-
lightDirection, normalDirection), viewDirection)), _Shininess);
193     }
194
195     return float4(diffuseReflection + specularReflection, 1.0);
196     // no ambient lighting in this pass
197 }
198
199     ENDCG
200 }
201 }
202 // The definition of a fallback shader should be commented out
203 // during development:
204 // Fallback "Specular"
205 }

```

在上面着色器中，当 alpha 分量为 0 时，我们设置材质颜色为深蓝色，效果如图：



在《平滑的镜面高光》章节中我们已经知道，镜面高光使用每顶点光照来渲染通常效果不是很好，但这有时候也会因为性能问题而别无选择，为了在《表面纹理光照》章节的着色器中使用光泽贴图（gloss mapping），我们需要用下面的代码来修改所有 pass 中片段着色器的代码：

```

1 float4 frag(vertexOutput input) : COLOR
2 {
3     float4 textureColor = tex2D(_MainTex, input.tex.xy);
4     return float4(input.specularColor * (1.0 - textureColor.a) +
input.diffuseColor * textureColor.rgb, 1.0);
5 }

```

这儿需要注意的是，通常情况下，我们不需要使用 (1.0 - textureColor.a) 与材质颜色相乘，只用 textureColor.a 来与材质颜色相乘就可以了，完整的每顶点光照光泽纹理的着色器代码如下：

```

001 Shader "Cg per-vertex lighting with texture"
002 {
003     Properties
004     {
005         _MainTex ("Texture For Diffuse Material Color", 2D) = "white" {}
006         _Color ("Overall Diffuse Color Filter", Color) = (1,1,1,1)

```

```

007     _SpecColor ("Specular Material Color", Color) = (1,1,1,1)
008     _Shininess ("Shininess", Float) = 10
009 }
010 SubShader
011 {
012     Pass
013     {
014         Tags { "LightMode" = "ForwardBase" }
015         // pass for ambient light and first light source
016
017         CGPROGRAM
018
019         #pragma vertex vert
020         #pragma fragment frag
021
022         #include "UnityCG.cginc"
023         uniform float4 _LightColor0;
024         // color of light source (from "Lighting.cginc")
025
026         // User-specified properties
027         uniform sampler2D _MainTex;
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);

```



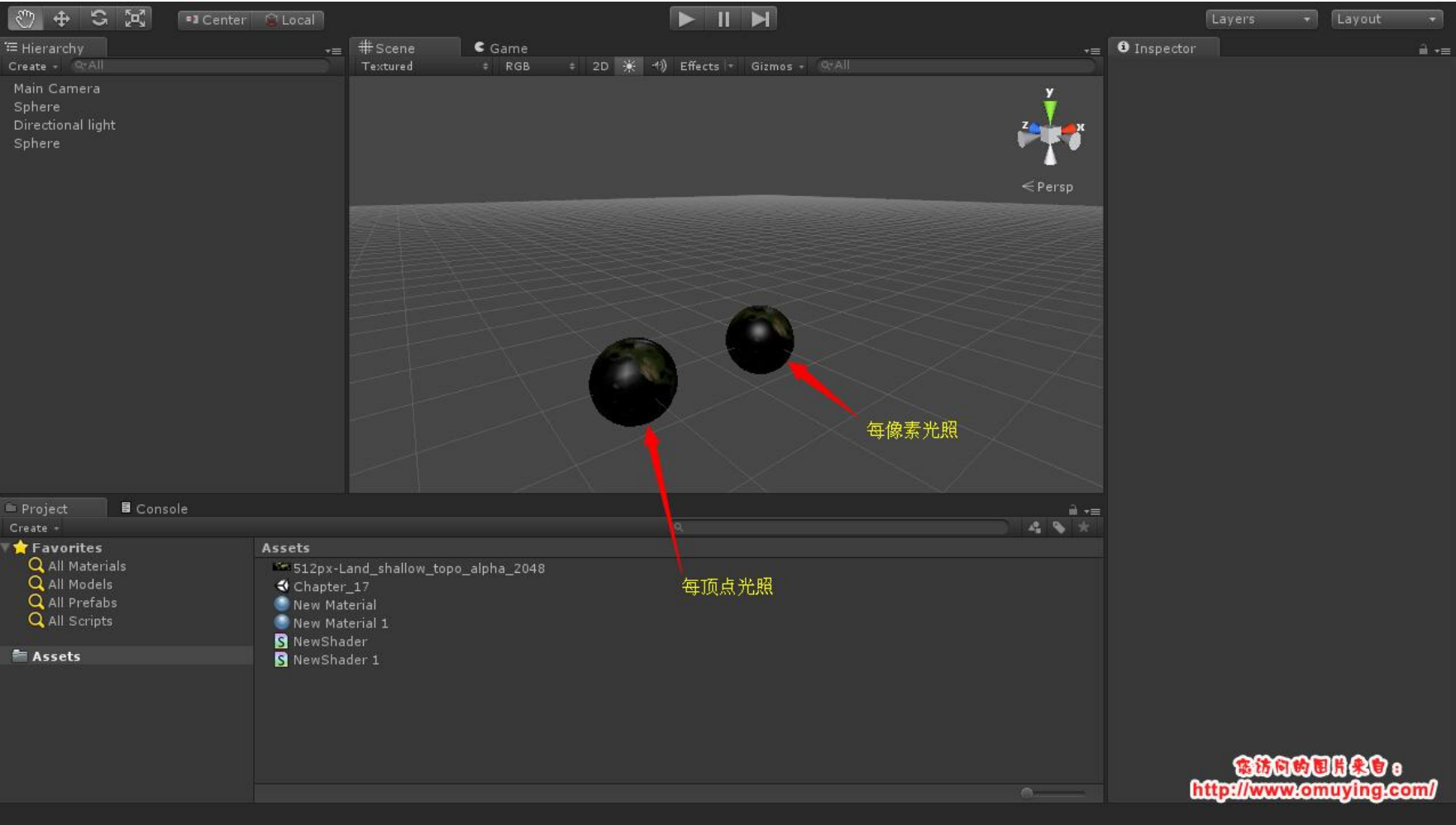
```

092         return output;
093     }
094
095     float4 frag(vertexOutput input) : COLOR
096     {
097         float4 textureColor = tex2D(_MainTex, input.tex.xy);
098         return float4(input.specularColor * (1.0 - textureColor.a) +
input.diffuseColor * textureColor.rgb, 1.0);
099     }
100
101     ENDCG
102 }
103
104 Pass
105 {
106     Tags { "LightMode" = "ForwardAdd" }
107     // pass for additional light sources
108     Blend One One // additive blending
109
110     CGPROGRAM
111
112     #pragma vertex vert
113     #pragma fragment frag
114
115     #include "UnityCG.cginc"
116     uniform float4 _LightColor0;
117     // color of light source (from "Lighting.cginc")
118
119     // User-specified properties
120     uniform sampler2D _MainTex;
121     uniform float4 _Color;
122     uniform float4 _SpecColor;
123     uniform float _Shininess;
124
125     struct vertexInput
126     {
127         float4 vertex : POSITION;
128         float3 normal : NORMAL;
129         float4 texcoord : TEXCOORD0;
130     };
131     struct vertexOutput
132     {
133         float4 pos : SV_POSITION;
134         float4 tex : TEXCOORD0;
135         float3 diffuseColor : TEXCOORD1;
136         float3 specularColor : TEXCOORD2;
137     };
138
139     vertexOutput vert(vertexInput input)
140     {
141         vertexOutput output;
142
143         float4x4 modelMatrix = _Object2World;
144         float4x4 modelMatrixInverse = _World2Object;
145         // multiplication with unity_Scale.w is unnecessary
146         // because we normalize transformed vectors
147
148         float3 normalDirection = normalize(mul(float4(input.normal, 0.0),
modelMatrixInverse).xyz);
149         float3 viewDirection = normalize(_WorldSpaceCameraPos -
mul(modelMatrix, input.vertex).xyz);
150         float3 lightDirection;
151         float attenuation;
152
153         if (0.0 == _WorldSpaceLightPos0.w) // directional light?
154         {
155             attenuation = 1.0; // no attenuation
156             lightDirection = normalize(_WorldSpaceLightPos0.xyz);
157         }
158         else // point or spot light
159         {
160             float3 vertexToLightSource = _WorldSpaceLightPos0.xyz -
mul(modelMatrix, input.vertex).xyz;
161             float distance = length(vertexToLightSource);
162             attenuation = 1.0 / distance; // linear attenuation
163             lightDirection = normalize(vertexToLightSource);
164         }
165
166         float3 diffuseReflection = attenuation * _LightColor0.rgb *
_Color.rgb * max(0.0, dot(normalDirection, lightDirection));
167
168         float3 specularReflection;
169         if (dot(normalDirection, lightDirection) < 0.0) // light source on
the wrong side?
170         {
171             specularReflection = float3(0.0, 0.0, 0.0);
172             // no specular reflection
173         }
174         else // light source on the right side
175         {
176             specularReflection = attenuation * _LightColor0.rgb *
_SpecColor.rgb * pow(max(0.0, dot(reflect(-lightDirection, normalDirection),
viewDirection)), _Shininess);

```

```
177     }
178
179     output.diffuseColor = diffuseReflection; // no ambient
180     output.specularColor = specularReflection;
181     output.tex = input.texcoord;
182     output.pos = mul(UNITY_MATRIX_MVP, input.vertex);
183     return output;
184 }
185
186 float4 frag(vertexOutput input) : COLOR
187 {
188     float4 textureColor = tex2D(_MainTex, input.tex.xy);
189     return float4(input.specularColor * (1.0 - textureColor.a) +
input.diffuseColor * textureColor.rgb, 1.0);
190 }
191
192 ENDCG
193 }
194 }
195 // The definition of a fallback shader should be commented out
196 // during development:
197 // Fallback "Specular"
198 }
```

使用每顶点光照来表现光泽纹理的效果如图：



恭喜你，在本章节中你应该了解：

- 1、什么是光泽纹理。
- 2、使用每像素光照来实现光泽纹理。
- 3、使用每顶点光照来实现光泽纹理。

资源下载地址：[点击下载](#)，共下载 24 次。

前一篇：[第十六章节：表面纹理光照（关于纹理漫射光照）](#)
后一篇：[第十八章节：透明纹理（关于在片段擦除、混合中使用 alpha 纹理）](#)



赞
2 人



打酱油
0 人



呵呵
0 人







鄙视
0 人




正能量
0 人



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

最终幻想正在使用多说