

随机阅读



暂无图片

【翻译】第九章：漫反射
（关于每顶点漫反射和多
光源漫反
射） - 1946 次阅读



暂无图片

【翻译】第十九章：纹理
层（关于多重纹
理） - 1661 次阅读



暂无图片

【原创】Shader 内置
Shader 之 Parallax
Diffuse 学习 - 1257 次
阅读



暂无图片

【翻译】第十七章：光泽
纹理（关于光泽贴
图） - 1201 次阅读



暂无图片

【翻译】第二十一章：凹
凸表面投影（关于视差贴
图） - 1462 次阅读

```

054 float4x4 modelMatrixInverse = _World2Object;
055 // multiplication with unity_Scale.w is unnecessary
056 // because we normalize transformed vectors
057
058 output.posWorld = mul(modelMatrix, input.vertex);
059 output.normalDir = normalize(mul(float4(input.normal, 0.0),
modelMatrixInverse).xyz);
060 output.pos = mul(UNITY_MATRIX_MVP, input.vertex);
061 output.posInObjectCoords = input.vertex; //测试用, 可以删除
062 return output;
063 }
064
065 float4 frag(vertexOutput input) : COLOR
066 {
067     if (input.posInObjectCoords.y > 0.0) //测试用, 可以删除
068     {
069         discard; // drop the fragment if y coordinate > 0
070     }
071     float3 normalDirection = normalize(input.normalDir);
072
073     float3 viewDirection = normalize(_WorldSpaceCameraPos -
input.posWorld.xyz);
074     float3 lightDirection;
075     float attenuation;
076
077     if (0.0 == _WorldSpaceLightPos0.w) // directional light?
078     {
079         attenuation = 1.0; // no attenuation
080         lightDirection = normalize(_WorldSpaceLightPos0.xyz);
081     }
082     else // point or spot light
083     {
084         float3 vertexToLightSource = _WorldSpaceLightPos0.xyz -
input.posWorld.xyz;
085         float distance = length(vertexToLightSource);
086         attenuation = 1.0 / distance; // linear attenuation
087         lightDirection = normalize(vertexToLightSource);
088     }
089
090     float3 ambientLighting = UNITY_LIGHTMODEL_AMBIENT.rgb *
_Color.rgb;
091
092     float3 diffuseReflection = attenuation * _LightColor0.rgb *
_Color.rgb * max(0.0, dot(normalDirection, lightDirection));
093
094     float3 specularReflection;
095     if (dot(normalDirection, lightDirection) < 0.0) // light source on
the wrong side?
096     {
097         specularReflection = float3(0.0, 0.0, 0.0);
098         // no specular reflection
099     }
100     else // light source on the right side
101     {
102         specularReflection = attenuation * _LightColor0.rgb *
_SpecColor.rgb * pow(max(0.0, dot(reflect(-lightDirection, normalDirection),
viewDirection)), _Shininess);
103     }
104
105     return float4(ambientLighting + diffuseReflection +
specularReflection, 1.0);
106 }
107 ENDCG
108 }
109
110 Pass
111 {
112     Tags { "LightMode" = "ForwardAdd" }
113     // pass for additional light sources
114     Blend One One // additive blending
115     Cull Back // render only front faces
116
117     CGPROGRAM
118
119     #pragma vertex vert
120     #pragma fragment frag
121
122     #include "UnityCG.cginc"
123     uniform float4 _LightColor0;
124     // color of light source (from "Lighting.cginc")
125
126     // User-specified properties
127     uniform float4 _Color;
128     uniform float4 _SpecColor;
129     uniform float _Shininess;
130     uniform float4 _BackColor;
131     uniform float4 _BackSpecColor;
132     uniform float _BackShininess;
133
134     struct vertexInput
135     {
136         float4 vertex : POSITION;
137         float3 normal : NORMAL;

```

```

138     };
139     struct vertexOutput
140     {
141         float4 pos : SV_POSITION;
142         float4 posWorld : TEXCOORD0;
143         float3 normalDir : TEXCOORD1;
144         float4 posInObjectCoords : TEXCOORD2; //测试用, 可以删除
145     };
146
147     vertexOutput vert(vertexInput input)
148     {
149         vertexOutput output;
150
151         float4x4 modelMatrix = _Object2World;
152         float4x4 modelMatrixInverse = _World2Object;
153         // multiplication with unity_Scale.w is unnecessary
154         // because we normalize transformed vectors
155
156         output.posWorld = mul(modelMatrix, input.vertex);
157         output.normalDir = normalize(mul(float4(input.normal, 0.0),
modelMatrixInverse).xyz);
158         output.pos = mul(UNITY_MATRIX_MVP, input.vertex);
159         output.posInObjectCoords = input.vertex; //测试用, 可以删除
160         return output;
161     }
162
163     float4 frag(vertexOutput input) : COLOR
164     {
165         if (input.posInObjectCoords.y > 0.0) //测试用, 可以删除
166         {
167             discard; // drop the fragment if y coordinate > 0
168         }
169         float3 normalDirection = normalize(input.normalDir);
170
171         float3 viewDirection = normalize(_WorldSpaceCameraPos -
input.posWorld.xyz);
172         float3 lightDirection;
173         float attenuation;
174
175         if (0.0 == _WorldSpaceLightPos0.w) // directional light?
176         {
177             attenuation = 1.0; // no attenuation
178             lightDirection = normalize(_WorldSpaceLightPos0.xyz);
179         }
180         else // point or spot light
181         {
182             float3 vertexToLightSource = _WorldSpaceLightPos0.xyz -
input.posWorld.xyz;
183             float distance = length(vertexToLightSource);
184             attenuation = 1.0 / distance; // linear attenuation
185             lightDirection = normalize(vertexToLightSource);
186         }
187
188         float3 diffuseReflection = attenuation * _LightColor0.rgb *
_Color.rgb * max(0.0, dot(normalDirection, lightDirection));
189
190         float3 specularReflection;
191         if (dot(normalDirection, lightDirection) < 0.0) // light source on
the wrong side?
192         {
193             specularReflection = float3(0.0, 0.0, 0.0);
194             // no specular reflection
195         }
196         else // light source on the right side
197         {
198             specularReflection = attenuation * _LightColor0.rgb *
_SpecColor.rgb * pow(max(0.0, dot(reflect(-lightDirection, normalDirection),
viewDirection)), _Shininess);
199         }
200
201         return float4(diffuseReflection + specularReflection, 1.0);
202         // no ambient lighting in this pass
203     }
204     ENDCG
205 }
206
207 Pass
208 {
209     Tags { "LightMode" = "ForwardBase" }
210     // pass for ambient light and first light source
211     Cull Front // render only back faces
212
213     CGPROGRAM
214
215     #pragma vertex vert
216     #pragma fragment frag
217
218     #include "UnityCG.cginc"
219     uniform float4 _LightColor0;
220     // color of light source (from "Lighting.cginc")
221
222     // User-specified properties
223     uniform float4 _Color;

```



```

224 uniform float4 _SpecColor;
225 uniform float _Shininess;
226 uniform float4 _BackColor;
227 uniform float4 _BackSpecColor;
228 uniform float _BackShininess;
229
230 struct vertexInput
231 {
232     float4 vertex : POSITION;
233     float3 normal : NORMAL;
234 };
235 struct vertexOutput
236 {
237     float4 pos : SV_POSITION;
238     float4 posWorld : TEXCOORD0;
239     float3 normalDir : TEXCOORD1;
240     float4 posInObjectCoords : TEXCOORD2; //测试用, 可以删除
241 };
242
243 vertexOutput vert(vertexInput input)
244 {
245     vertexOutput output;
246
247     float4x4 modelMatrix = _Object2World;
248     float4x4 modelMatrixInverse = _World2Object;
249     // multiplication with unity_Scale.w is unnecessary
250     // because we normalize transformed vectors
251
252     output.posWorld = mul(modelMatrix, input.vertex);
253     output.normalDir = normalize(mul(float4(-input.normal, 0.0),
modelMatrixInverse).xyz);
254     output.pos = mul(UNITY_MATRIX_MVP, input.vertex);
255     output.posInObjectCoords = input.vertex; //测试用, 可以删除
256     return output;
257 }
258
259 float4 frag(vertexOutput input) : COLOR
260 {
261     if (input.posInObjectCoords.y > 0.0) //测试用, 可以删除
262     {
263         discard; // drop the fragment if y coordinate > 0
264     }
265     float3 normalDirection = normalize(input.normalDir);
266
267     float3 viewDirection = normalize(_WorldSpaceCameraPos -
input.posWorld.xyz);
268     float3 lightDirection;
269     float attenuation;
270
271     if (0.0 == _WorldSpaceLightPos0.w) // directional light?
272     {
273         attenuation = 1.0; // no attenuation
274         lightDirection = normalize(_WorldSpaceLightPos0.xyz);
275     }
276     else // point or spot light
277     {
278         float3 vertexToLightSource = _WorldSpaceLightPos0.xyz -
input.posWorld.xyz;
279         float distance = length(vertexToLightSource);
280         attenuation = 1.0 / distance; // linear attenuation
281         lightDirection = normalize(vertexToLightSource);
282     }
283
284     float3 ambientLighting = UNITY_LIGHTMODEL_AMBIENT.rgb *
_BackColor.rgb;
285
286     float3 diffuseReflection = attenuation * _LightColor0.rgb *
_BackColor.rgb * max(0.0, dot(normalDirection, lightDirection));
287
288     float3 specularReflection;
289     if (dot(normalDirection, lightDirection) < 0.0) // light source on
the wrong side?
290     {
291         specularReflection = float3(0.0, 0.0, 0.0);
292         // no specular reflection
293     }
294     else // light source on the right side
295     {
296         specularReflection = attenuation * _LightColor0.rgb *
_BackSpecColor.rgb * pow(max(0.0, dot(reflect(-lightDirection,
normalDirection), viewDirection)), _BackShininess);
297     }
298
299     return float4(ambientLighting + diffuseReflection +
specularReflection, 1.0);
300 }
301 ENDCG
302 }
303
304 Pass
305 {
306     Tags { "LightMode" = "ForwardAdd" }
307     // pass for additional light sources

```

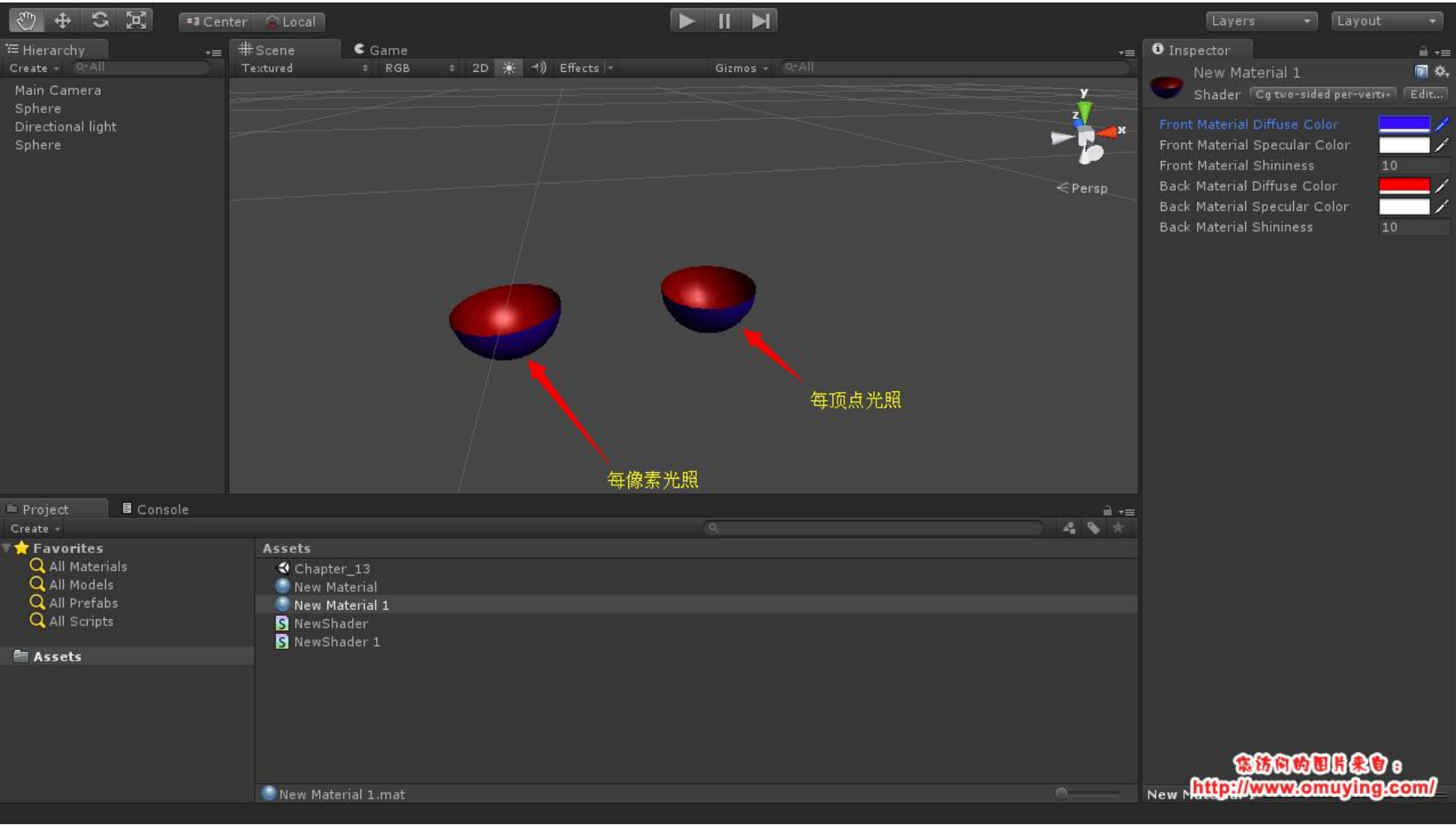
```

308 Blend One One // additive blending
309 Cull Front // render only back faces
310
311 CGPROGRAM
312
313 #pragma vertex vert
314 #pragma fragment frag
315
316 #include "UnityCG.cginc"
317 uniform float4 _LightColor0;
318 // color of light source (from "Lighting.cginc")
319
320 // User-specified properties
321 uniform float4 _Color;
322 uniform float4 _SpecColor;
323 uniform float _Shininess;
324 uniform float4 _BackColor;
325 uniform float4 _BackSpecColor;
326 uniform float _BackShininess;
327
328 struct vertexInput
329 {
330     float4 vertex : POSITION;
331     float3 normal : NORMAL;
332 };
333 struct vertexOutput
334 {
335     float4 pos : SV_POSITION;
336     float4 posWorld : TEXCOORD0;
337     float3 normalDir : TEXCOORD1;
338     float4 posInObjectCoords : TEXCOORD2; //测试用, 可以删除
339 };
340
341 vertexOutput vert(vertexInput input)
342 {
343     vertexOutput output;
344
345     float4x4 modelMatrix = _Object2World;
346     float4x4 modelMatrixInverse = _World2Object;
347     // multiplication with unity_Scale.w is unnecessary
348     // because we normalize transformed vectors
349
350     output.posWorld = mul(modelMatrix, input.vertex);
351     output.normalDir = normalize(mul(float4(-input.normal, 0.0),
modelMatrixInverse).xyz);
352     output.pos = mul(UNITY_MATRIX_MVP, input.vertex);
353     output.posInObjectCoords = input.vertex; //测试用, 可以删除
354     return output;
355 }
356
357 float4 frag(vertexOutput input) : COLOR
358 {
359     if (input.posInObjectCoords.y > 0.0) //测试用, 可以删除
360     {
361         discard; // drop the fragment if y coordinate > 0
362     }
363     float3 normalDirection = normalize(input.normalDir);
364
365     float3 viewDirection = normalize(_WorldSpaceCameraPos -
input.posWorld.xyz);
366     float3 lightDirection;
367     float attenuation;
368
369     if (0.0 == _WorldSpaceLightPos0.w) // directional light?
370     {
371         attenuation = 1.0; // no attenuation
372         lightDirection = normalize(_WorldSpaceLightPos0.xyz);
373     }
374     else // point or spot light
375     {
376         float3 vertexToLightSource = _WorldSpaceLightPos0.xyz -
input.posWorld.xyz;
377         float distance = length(vertexToLightSource);
378         attenuation = 1.0 / distance; // linear attenuation
379         lightDirection = normalize(vertexToLightSource);
380     }
381
382     float3 diffuseReflection = attenuation * _LightColor0.rgb *
_BackColor.rgb * max(0.0, dot(normalDirection, lightDirection));
383
384     float3 specularReflection;
385     if (dot(normalDirection, lightDirection) < 0.0) // light source on
the wrong side?
386     {
387         specularReflection = float3(0.0, 0.0, 0.0);
388         // no specular reflection
389     }
390     else // light source on the right side
391     {
392         specularReflection = attenuation * _LightColor0.rgb *
_BackSpecColor.rgb * pow(max(0.0, dot(reflect(-lightDirection,
normalDirection), viewDirection)), _BackShininess);
393     }

```

```
394         return float4(diffuseReflection + specularReflection, 1.0);
395         // no ambient lighting in this pass
396     }
397 }
398 ENDCG
399 }
400 }
401 // The definition of a fallback shader should be commented out
402 // during development:
403 // Fallback "Specular"
404 }
```

在场景中添加两个球体，一个使用 two-sided per-vertex lighting，另一个使用 two-sided per-pixel lighting，并观察他们的不同，效果如下：



恭喜你，在本章节中你应该了解：

- 1、如何用 per-pixel lighting 渲染 two-sided surfaces。

资源下载地址：[点击下载](#)，共下载 12 次。

前一篇：[第十二章：光滑的镜面高光（关于每像素光照）](#)

后一篇：[第十四章节：多个灯（关于在一个 pass 中遍历处理多个光源）](#)



赞
2 人



打酱油
0 人



呵呵
0 人



鄙视
0 人



正能量
0 人



0

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

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