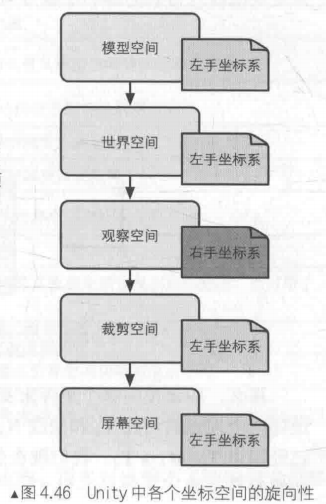
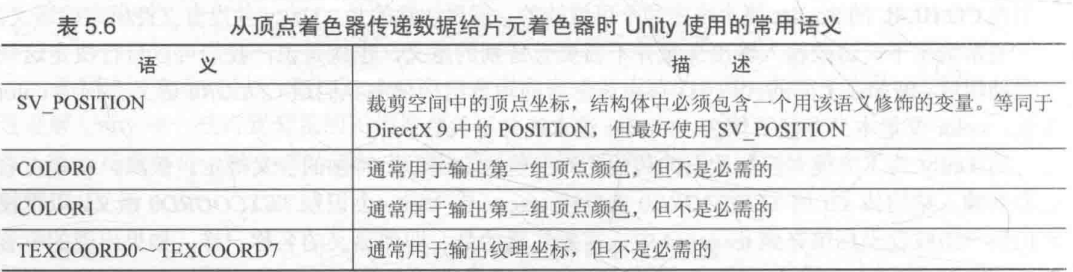
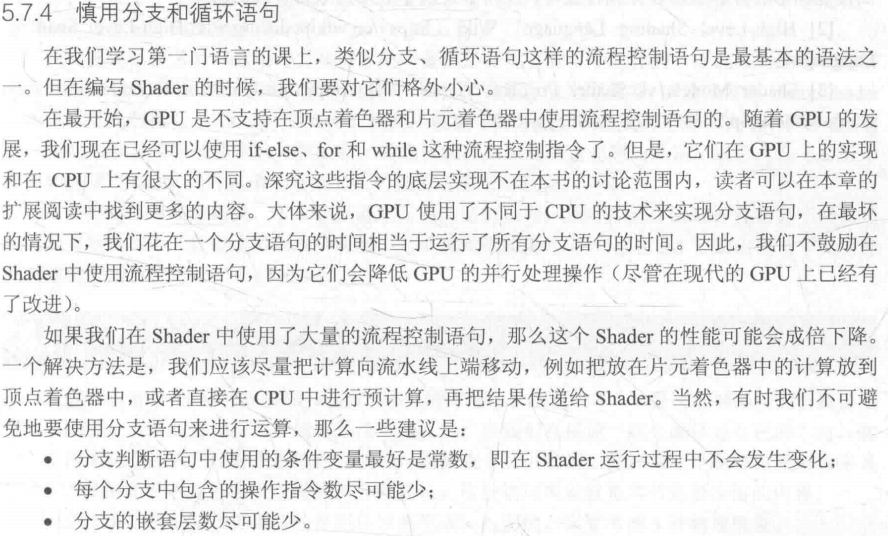
Unity在模型空间和世界空间是左手坐标系，在眼坐标是右手坐标系









**逐顶点环境光照**

Shader "Unlit/DiffuseShader"

{

Properties

{

\_Diffuse ("Diffuse", Color) = (1.0, 1.0, 1.0, 1.0)

\_MainTex ("Texture", 2D) = "white" {}

}

SubShader

{

Tags { "RenderType"="Opaque" }

LOD 100

Pass

{

Tags { "LightMode" = "ForwardBase"}

CGPROGRAM

#pragma vertex vert

#pragma fragment frag

#include "Lighting.cginc"

fixed4 \_Diffuse;

struct a2v

{

float4 vertex : POSITION;

float3 normal : NORMAL;

};

struct v2f

{

float4 pos: SV\_POSITION;

fixed3 color: COLOR;

};

v2f vert(a2v v)

{

v2f o;

o.pos = mul(UNITY\_MATRIX\_MVP, v.vertex);

fixed3 ambient = UNITY\_LIGHTMODEL\_AMBIENT.xyz;

fixed3 worldNormal = normalize(mul(v.normal, (float3x3)\_World2Object));

//fixed3 worldNormal = normalize(mul((float3x3)transpose(\_World2Object), v.normal));

fixed3 worldLight = normalize(\_WorldSpaceLightPos0.xyz);

fixed cos = saturate(dot(worldNormal, worldLight));

fixed3 diffuse = \_LightColor0.rgb \* \_Diffuse.rgb \* cos;

o.color = ambient + diffuse;

return o;

}

fixed4 frag(v2f i) : SV\_Target

{

return fixed4(i.color, 1.0);

}

ENDCG

}

}

FALLback "Diffuse"

}

对于上面的情况重点注意mul(v.normal, (float3x3)\_World2Object)，实际上mul可以适用矩阵\*列向量，行向量\*矩阵，矩阵\*矩阵



**逐像素镜面光照**

Shader "Unlit/DiffuseShader"

{

Properties

{

\_Diffuse ("Diffuse", Color) = (1.0, 1.0, 1.0, 1.0)

\_Specular ("Specular", Color) = (1.0, 1.0, 1.0, 1.0)

\_Gloss ("Gloss", Range(8.0, 256.0)) = 20.0

\_MainTex ("Texture", 2D) = "white" {}

}

SubShader

{

Tags { "RenderType" = "Opaque" }

LOD 100

Pass

{

Tags { "LightMode" = "ForwardBase"}

CGPROGRAM

#pragma vertex vert

#pragma fragment frag

#include "Lighting.cginc"

fixed4 \_Diffuse;

fixed4 \_Specular;

float \_Gloss;

struct a2v

{

float4 vertex : POSITION;

float3 normal : NORMAL;

};

struct v2f

{

float4 pos : SV\_POSITION;

float3 worldNormal : TEXCOORD;

float3 worldPos : TEXCOORD1;

};

v2f vert(a2v v)

{

v2f o;

o.pos = mul(UNITY\_MATRIX\_MVP, v.vertex);

o.worldNormal = normalize(mul(v.normal, (float3x3)\_World2Object));

o.worldPos = mul(\_Object2World, v.vertex).xyz;

return o;

}

fixed4 frag(v2f i) : SV\_Target

{

fixed3 ambient = UNITY\_LIGHTMODEL\_AMBIENT.xyz;

fixed3 worldNormal = normalize(i.worldNormal);

**fixed3 worldLightDir = normalize(\_WorldSpaceLightPos0.xyz);// 这里是平行光照才能得到方向**

fixed3 diffuse = \_LightColor0.rgb \* \_Diffuse.rgb \* saturate(dot(worldNormal, worldLightDir));

fixed3 reflectDir = normalize(reflect(-worldLightDir, worldNormal));

fixed3 viewDir = normalize(\_WorldSpaceCameraPos.xyz - i.worldPos.xyz);

fixed3 specular = \_LightColor0.rgb \* \_Specular.rgb \* pow(saturate(dot(reflectDir, viewDir)), \_Gloss);

return fixed4(ambient + diffuse + specular, 1.0);

}

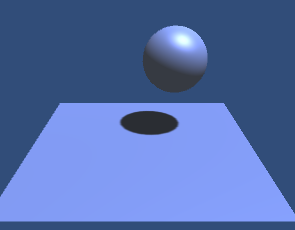
ENDCG

}

}

FALLback "Specular"

}





修改为内置函数：

//o.worldNormal = normalize(mul(v.normal, (float3x3)\_World2Object));

o.worldNormal = UnityObjectToWorldNormal(v.normal);

//fixed3 worldLightDir = normalize(\_WorldSpaceLightPos0.xyz);

fixed3 worldLightDir = normalize(UnityWorldSpaceLightDir(i.worldPos));

//fixed3 viewDir = normalize(\_WorldSpaceCameraPos.xyz - i.worldPos.xyz);

fixed3 viewDir = normalize(UniyWorldSpaceViewDir(i.worldPos));

光照+纹理

Shader "Unlit/DiffuseShader"

{

Properties

{

\_Color ("Color Tint", Color) = (1.0, 1.0, 1.0, 1.0)

\_Diffuse ("Diffuse", Color) = (1.0, 1.0, 1.0, 1.0)

\_Specular ("Specular", Color) = (1.0, 1.0, 1.0, 1.0)

\_Gloss ("Gloss", Range(8.0, 256.0)) = 20.0

\_MainTex ("Texture", 2D) = "white" {}

}

SubShader

{

Tags { "RenderType" = "Opaque" }

LOD 100

Pass

{

Tags { "LightMode" = "ForwardBase"}

CGPROGRAM

#pragma vertex vert

#pragma fragment frag

#include "Lighting.cginc"

fixed4 \_Color;

sampler2D \_MainTex;

**float4 \_MainTex\_ST;**

fixed4 \_Diffuse;

fixed4 \_Specular;

float \_Gloss;

struct a2v

{

float4 vertex : POSITION;

float3 normal : NORMAL;

float4 texcoord : TEXCOORD0;

};

struct v2f

{

float4 pos : SV\_POSITION;

float3 worldNormal : TEXCOORD;

float3 worldPos : TEXCOORD1;

float2 uv : TEXCOORD2;

};

v2f vert(a2v v)

{

v2f o;

o.pos = mul(UNITY\_MATRIX\_MVP, v.vertex);

o.worldNormal = UnityObjectToWorldNormal(v.normal);

o.worldPos = mul(unity\_ObjectToWorld, v.vertex).xyz;

**o.uv = v.texcoord.xy \* \_MainTex\_ST.xy + \_MainTex\_ST.zw;**

return o;

}

fixed4 frag(v2f i) : SV\_Target

{

fixed3 albedo = tex2D(\_MainTex, i.uv).rgb \* \_Color.rgb;

fixed3 ambient = UNITY\_LIGHTMODEL\_AMBIENT.xyz \* albedo;

fixed3 worldNormal = normalize(i.worldNormal);

fixed3 worldLightDir = normalize(UnityWorldSpaceLightDir(i.worldPos));

fixed3 diffuse = \_LightColor0.rgb \* \_Diffuse.rgb \* albedo \* saturate(dot(worldNormal, worldLightDir));

fixed3 reflectDir = normalize(reflect(-worldLightDir, worldNormal));

fixed3 viewDir = normalize(UnityWorldSpaceViewDir(i.worldPos));

fixed3 specular = \_LightColor0.rgb \* \_Specular.rgb \* pow(saturate(dot(reflectDir, viewDir)), \_Gloss);

return fixed4(ambient + diffuse + specular, 1.0);

}

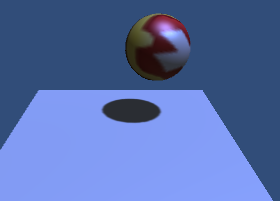
ENDCG

}

}

FALLback "Specular"

}



**注意：上面的\_MainTex\_ST并不随便起的**

在unity中，我们需要使用纹理名\_ST的方式来生命某个纹理的属性，其中ST是缩放和平移的缩写。\_MainTex\_ST可以让我们得到该纹理的缩放和平移值，\_MainTex\_ST.xy存储的是缩放值，而\_MainTex\_ST.zw存放的是偏移值。

上面的**o.uv = v.texcoord.xy \* \_MainTex\_ST.xy + \_MainTex\_ST.zw;可以哦那个下面公式代替：**

**o.uv = TRANSFORM\_TEX(v.texcoord, \_MainTex);**

**纹理属性**

