# 关于点的坐标系变换

1 将坐标转换为投影空间

mul(**UNITY\_MATRIX\_MVP**, 局部顶点坐标);

**UnityObjectToClipPos**(顶点坐标);

2 局部空间和世界空间的转换

mul(**unity\_ObjectToWorld**, 局部顶点坐标);

mul(**unity\_WorldToObject**, 世界顶点坐标);

# 关于向量的坐标系变换

1 观察向量 点->摄像机

float3 **WorldSpaceViewDir**(float4 局部坐标);

float3 **UnityWorldSpaceViewDir**(float4 世界坐标);

float3 ObjSpaceViewDir(float4 局部坐标)

2 光照方向 点->光源 **仅可用于前向渲染,因为前向渲染\_WorldSpaceLisghtPos0 才会正确赋值**

float3 **WorldSpaceLightDir**(float4 局部坐标);

float3 **UnityWorldSpaceLightDir**(float4 世界坐标);

float3 ObjSpaceLightDir(float4 局部坐标)

3 法线 从局部到世界

float3 **UnityObjectToWorldNormal**(float3 局部法线向量);

4 普通向量 局部与世界的转换

float3 **UnityObjectToWorldDir**(float3 局部向量);

float3 **UnityWorldToObjectDir**(float3 世界向量)

# 光照

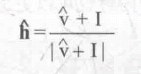
### 漫反射

兰伯特定律：

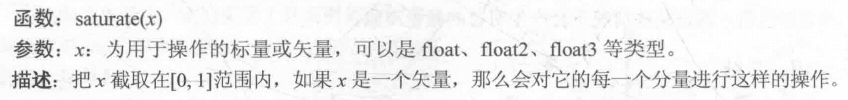
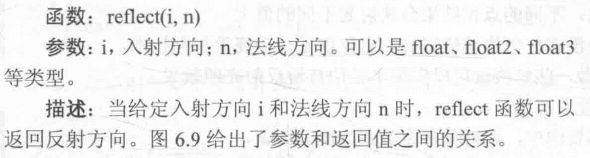
半兰伯特定律：因为max(0, n·I)的最小值截断到0,0以下的都是0，无法正确表示。因此使用半兰伯特定律

### 高光反射

Phong模型：

Blinn模型：

### 光照函数



### 光照变量

1 环境光

**UNITY\_LIGHTMODEL\_AMBIENT**

**例如：fixed3 ambient = UNITY\_LIGHTMODEL\_AMBIENT.xyz;**

2 光源颜色

**\_LightColor0**

**例如：\_LightColor0.rgb**

3 光源方向

**\_WorldSpaceLightPos0**

**例如：\_WorldSpaceLightPos0.xyz**

**4 摄像机位置**

**\_WorldSpaceCameraPos**

### 添加光照的过程

1 获取环境光 fixed3 ambient = UNITY\_LIGHTMODEL\_AMBIENT.rgb;

2 获取漫反射

1. 获取世界法线向量 float3 worldNormal = UnityObjectToWorldNormal(normal);
2. 获取世界坐标 float3 worldPos = mul(unity\_ObjectToWorld, position).xyz;
3. 获取光照方向 float3 worldLightDir = UnityWorldSpaceLightDir(worldPos);
4. fixed3 diffuse = \_LightColor0.rgb \* \_Diffuse \* saturate(dot(worldNormal, worldLightDir));

3 获取高光反射

1. 获取视角方向 float3 worldViewDir = UnityWorldSpaceViewDir(worldPos);
2. 获取中间值 float3 halfDir = normalize(worldLightDir+ worldViewDir);
3. fixed3 specular = \_LightColor0.rgb \* \_Specular \* pow(saturate(dot(halfDir, worldNormal)), \_Gloss);

4 最终结果

fixed4 color = fixed4(ambient+diffuse+specular, 1.0);

# 纹理

### 纹理函数

1 获取正确的纹理坐标

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

2 纹理采样

fixed4 texColor = **tex2D**(\_MainTex, i.uv);

3 获取法线纹理中的法线

## 添加纹理的过程

1 求反光率

1. fixed4 texColor = tex2D(\_MainTex, uv);
2. fixed3 albedo = texColor.rgb \* \_Color.rgb;

2 求环境光

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

3 求漫反射

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

4 求高光反射(同上，不受影响)

fixed3 specular

5 最终结果

fixed4 color = fixed4(ambient+diffuse+specular, 1.0);

#### 法线纹理

法线纹理 有两种 模型空间的法线纹理， 切线空间的法线纹理。

法线纹理 处理的位置也有两种 在切线坐标系处理，在世界坐标系处理。

一 在切线坐标系处理

1. 获取 局部坐标系到切线坐标系的变换矩阵

TANGENT\_SPACE\_ROTATION;

// 获取获取 局部坐标系下的 tangent, normal,

// binormal = cross(normalize(tangent), normalize(normal))\*tangent.w;

//float3x3 rotation = float3x3(tangent.xyz, binormal, v.normal);

1. 将光照和视觉 向量从局部坐标系变换到 切线坐标系

fixed3 lightDir = mul(rotation, ObjSpaceLightDir(vertex)).xyz;

fixed3 viewDir = mul(rotation, ObjSpaceViewDir(vertex)).xyz;

1. 通过法线纹理获取法线向量

fixed4 packedNormal = tex2D(\_BumpMap, uv);

fixed3 tangentNormal = UnpackNormal(packedNormal);

tangentNormal.xy \*= \_BumpScale; //这里需要乘以系数

tangentNormal.z = sqrt(1.0 – saturate(dot(tangentNormal.xy, tangentNormal.xy)));

1. 求反光率 同上
2. 求环境光 同上
3. 求漫反射

fixed3 diffuse = \_LightColor0.rgb \* albedo \* saturate(dot(tangentNormal, worldLightDir))

1. 求高光反射

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

1. 求和

二 在世界坐标系处理

1. 获取从 切线坐标系 到 世界坐标系 的矩阵

float3 worldNormal = UnityObjectToWorldNormal(v.normal);

float3 worldTangent = UnityObjectToWroldDir(v.tangent);

float3 worldBiNormal = cross(worldNoraml, worldTangent) \* v.tangent.w;

将三个向量以列展开

float3 line1 = float3(worldTangent.x, worldBiNormal.x, worldNormal.x);

float3 line2 = float3(worldTangent.y, worldBiNormal.y, worldNormal.y);

float3 line3 = float3(worldTangent.z, worldBiNormal.z, worldNormal.z);

1. 获取法线

fixed3 tangentNormal = Unpacknormal(tex2D(\_BumpMap, uv));

tangentNormal.xy \*= \_BumpScale; //这里需要乘以系数

tangentNormal.z = sqrt(1.0 – saturate(dot(tangentNormal.xy, tangentNormal.xy)));

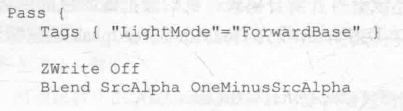
将该法线从切线坐标系 转为 世界坐标系

tangentNormal = normalize(half3(dot(line1, tangentNormal), dot(line2, tangentNormal), dot(line3, tangentNormal)))

1. 以后步骤同上

## 透明纹理

一 透明度混合

 1 透明度标签设置

2 片元着色器最后返回颜色的时候，需要将透明度带上

