LitSphere

重点解释如下两句代码

TANGENT\_SPACE\_ROTATION;

o.tan1 = mul(rotation, UNITY\_MATRIX\_IT\_MV[0].xyz);

**TANGENT\_SPACE\_ROTATION**

// Declares 3x3 matrix 'rotation', filled with tangent space basis

#define TANGENT\_SPACE\_ROTATION \

    float3 binormal = cross( normalize(v.normal), normalize(v.tangent.xyz) ) \* v.tangent.w; \

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

\是宏换行连接符,

v.normal就是切向坐标的z轴,

v.tangent.xyz是切向坐标的x轴，

v.tangent.w齐次坐标相关,值为1或-1，代表uv正向或是反向

第一句实际是计算切向坐标的y轴

第二句使用切向坐标x,y,z轴向量构建了从模型坐标转到切向坐标系的矩阵

**o.tan1 = mul(rotation, UNITY\_MATRIX\_IT\_MV[0].xyz);**

这句作用是把观察坐标中的x轴转换到切向坐标，为了了解这句，需要趟几个坑

**1 mul函数**

(unity shader中的mul，gl和dx没确认过)

第一种写法，参数1为矩阵，参数2为向量:  mul把向量当列向量

第二种写法，参数1为向量，参数2为矩阵:  mul向量当为行向量

比如最常见的写法,把点从模型空间转到世界空间，这里\_Object2World就是针对列向量的矩阵，而v.vertex就是列向量

mul(\_Object2World,v.vertex)

如果换成行向量就是

mul(v.vertex,transpose(\_Object2World))

注意Unity默认一般用上面的写法，所以unity内部用的应该是列向量，\_Object2World,\_World2Object这类矩阵也都是跟列向量乘的

**2 从观察坐标转换到模型坐标**

unity没提供从观察系直接转换到切向坐标的矩阵，这里需要先从观察系转换到模型坐标  
具体方法就是  
mul(UNITY\_MATRIX\_VM, float3(1, 0, 0)) 等同于  
mul(float3(1,0,0),transpose(UNITY\_MATRIX\_VM))     
而unity中并没有UNITY\_MATRIX\_VM(观察到模型的变换矩阵)，但是有UNITY\_MATRIX\_IT\_MV(模型到观察的逆转置)，进一步变化  
mul(float3(1,0,0), (float3x3)UNITY\_MATRIX\_IT\_MV)) 等同于  
UNITY\_MATRIX\_IT\_MV[0].xyz  
其中1,0,0是观察系中的x轴,UNITY\_MATRIX\_IT\_MV[0].xyz代表观察系中x轴在模型坐标系中的向量表示

3 从模型坐标转换到切向坐标

rotation \* UNITY\_MATRIX\_IT\_MV[0].xyz 就是 把模型坐标系中的一个向量(观察系的x轴)转换到切向坐标，结果就是切向坐标中的一个向量(观察系中的x轴)

4 说下比较容易跟这个混淆的法线空间转换

上面的3部是没有用到法线空间转换的  
法线的空间转换比较特殊，正常向量转换直接乘相应矩阵，但法向量无论向哪个空间转换

除非这个转换矩阵不包含缩放，只要包含缩放的转换矩阵，直接乘法线会使法线不再垂直于原来的向量或平面

解决方法就是法线向量要乘转换矩阵的逆转置矩阵，具体原因[http://blog.csdn.net/onafioo/article/details/51889541](http://blog.csdn.net/onafioo/article/details/51889541" \t "https://blog.csdn.net/onafioo/article/details/_blank)，

乘逆转置矩阵是在保证法线垂直性不变的情况下推倒出来的，记住结论就可以。

举例

把普通向量从模型空间转换到世界空间

float4 worldV = mul(\_Object2World,v.vertex);

把法向量从模型空间转换到世界空间，可以用如下两种方法，效果相同

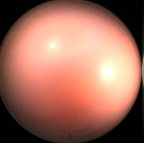
float4 worldNorm = mul(v.normal, \_World2Object);

float4 worldNorm = mul(transpose(\_World2Object), v.normal);

这里注意\_World2Object 就是\_Object2World的逆矩阵

surf中  
litSphereUV.x = dot(IN.tan1, o.Normal);  这个计算方式 跟贴图相关

使用的贴图是已经烘培好灯光效果的2d贴图(缺点是光照方向固定了不能再动态更改)



这里对应关系可以这么理解，随便假定下观察系坐标，比如自己右是观察系x正向，前是z正向，上是y正向，那么右就是IN.tan1,随便找个物体放在面前，物体的法线方向就是

o.Normal,当观察者不发生变化，而物体某个点转动，法线变化时，像素位置就会在贴图上变化