渲染管线中的法线变换矩阵

The Normal Matrix

在渲染管线中，模型的坐标会从局部空间(Local space)经Model matrix(简记为**M**)变换到世界空间(World space)，从世界空间经View matrix(简记为**V**)变换到观察空间(View space，也称为eye space)，然后再经Projection matrix变换到裁剪空间(clip space) (vertex shader要计算出裁剪空间的坐标)，最后经视口变换(viewport transform)变换到屏幕空间。

(更多关于渲染管线的介绍见这篇文章，关于坐标变换见这篇文章)

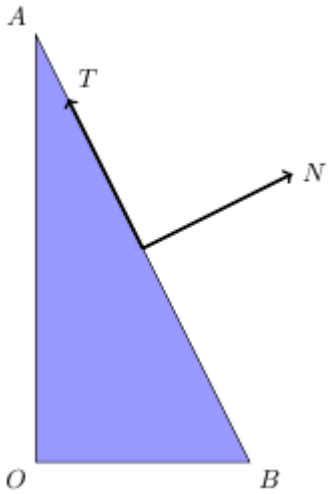
在进行光照计算时，为了得到逼真的效果，一般要使用到模型的顶点的法线。可以在观察空间(View space)或世界空间(World space)中进行光照计算。中View space中进行光照计算的好处是观察者(即Camera)的坐标永远是(0, 0)。

假设在View space中进行光照计算。Local space到View space的变换矩阵为**M**乘以**V**，简记为**MV**。

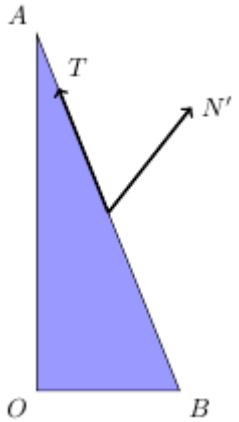
Vertex Shader中输入数据有顶点位置aVertex及法线N，它们都是在局部空间中的向量。则View space空间中的顶点位置可由以下公式计算:

$MV \cdot aVertex$。

而View space中顶点的法线N一般不能由$MV \cdot N$计算得到。比如，当模型发生non-uniform缩放时，经$MV \cdot N$变换后的所谓“法线”已不与模型表面垂直了，根本不是法线了。如图1是未发生变换时三角形的法线，而图2是使用$MV \cdot N$变换后的三角形“法线”。



(图1. 变换前三角形的法线)



(图2. 使用$MV \cdot N$变换后的“法线”)

那么用于法线变换的矩阵应该长什么样呢?答案是：

把法线从local space变换到view space的变换矩阵为顶点位置变换矩阵**MV**的逆矩阵的转置矩阵，即$(MV^{-1})^T$。

(如果在world space中进行光照计算，则normal matrix为$(M^{-1})^T$)。

推导

上面图1中的T和图2中的T'表示模型顶点的切线向量。切线向量可以由三角形边上的某两个点(P_2, P_1)计算得到。比如：

$$T = P_2 - P_1$$

三维空间中切线向量T可以写成一个四维向量(把最后一个分量设置为0)，则:

$$T * MV = (P_2 - P_1) * MV$$

可以得到:

$$T * MV = P_2 * MV - P_1 * MV$$

$$T' = P_2' - P_1'$$

因为P_2', P_1'是三角形变换后的顶点，所以T'为变换后三角形变换后的边的切线。因此可以使用MV来变换切向量。

上面已说过了MV是不能用来变换法线向量的。但我们知道在一个点处，**法线向量永远与切线向量垂直**，即$T \cdot N = 0$。

因为法线向量是一个仅是一个方向向量，其没有齐次坐标(即第四个分量为0)，则平移对法线无作用。设我们要求的把法线从局部空间变换到观察空间的矩阵为一个3×3矩阵G，则变换后的法线向量N'和变换后的切线向量T'也满足$N' \cdot T' = 0$。

所以：
(由于切线也只是一个方向向量，不受平移变换影响，以下推导中我们用MV表示真正的MV矩阵的左上角的3×3子矩阵)

$$N' \cdot T' = (GN) \cdot (MVT) = 0$$

注意，这里GN和MVT其实都是列向量，所以它们的点积可以这样计算:

$$(GN)\cdot(MVT)=(GN)^T*(MVT)=N^TG^TMVT$$

注意: N^TG^TMVT中首尾两个符号，假如:$G^TMV=I$，其中I是单位矩阵，则$N'\cdot T'=N^T*T=N\cdot T=0$，即新的法线向量和新的切线向量垂直。

则$G^TMV=I\iff G=((MV)^{-1})^T$

因此正确的**normal transform matrix**为$((MV)^{-1})^T$

当然，如果是在世界空间中进行光照计算，则法线变换的矩阵为$(M^{-1})^T$。

有些情形下(如，仅有旋转和平移时)，使用MV对法线进行变换也会得到正确的结果，这是为什么呢？因为这时候，MV左上角的3×3矩阵是正交矩阵，即$(MV)^{-1}=(MV)^T\implies G=MV$。

注意计算逆矩阵的过程性能代价很大，不适合在vertex shader或pixel shader(或fragment shader)中对每一个顶点甚至像素都计算一遍法线变换矩阵。一般是在CPU上计算一次，然后把它放到shader中的一个uniform变量中。

另外这篇文章指出在大部分正常情形下，甚至不用计算法线变换矩阵，就能得到变换后的法线。

References:

- The Normal Matrix
- How to calculate the normal matrix?
- Normal Matrix in plain English
- Stop Using Normal Matrix
- Basic-Lighting