# 【OpenGL】矩阵的理解

矩阵真的是一个很神奇的数学工具， 虽然单纯从数学上看， 它并没有什么特别的意义， 但一旦用到空间中的坐标变换，它就“一遇风云便成龙”， 大显神威了。简单的工具实现了复杂的功能，便预示着要理解它我们还是要花上点功夫的。下面就简单介绍一下OpenGL中的转换矩阵。

**1 转换矩阵的原理**OpenGL中的转换矩阵是这样定义的:  
              [Xx, Yx, Zx, Tx]  
             [Xy, Yy, Zy, Ty]  
M   =      [Xz, Yz, Zz, Tz]  
              [0, 0, 0, 1 ]

其实我们可以这么理解这个变换矩阵， 它表示了一个局部坐标系， 这个局部坐标系，是把世界坐标系的原点移到(Tx, Ty, Tz)，把X轴转到(Xx, Xy, Xz)， Y轴转到(Yx, Yy, Yz)，Z轴转到(Zx, Zy, Zz)而形成的。用它来变换一个世界坐标系中的点V， 就是得到这个局部坐标系中的点。  
要证明这一点很容易， 我们从可以从更通用的方面来考虑，假设我们用矩阵Ma来表示坐标系a, Mb来表示坐标系b, Mt表示从a到b的转换， 那么：  
Mt \* Ma = Mb  
Mt \* Ma \* (Ma)^-1 = Mb \* (Ma)^-1  
矩阵虽然不符合乘法交换律，但其符合乘法结合律， 于是：  
Mt\* (Ma \* (Ma)^-1) = Mb \* (Ma)^-1  
Mt = Mb \* (Ma)^-1  
这就是a到b转换矩阵的表达式，现在我们从世界坐标系转换到局部坐标系，a表示的世界坐标系是个单位矩阵，所以:  
Mt = Mb  
即局部坐标系的矩阵表示就是从世界坐标系到局部坐标系的转换矩阵。

我们再进一步分析，如果我们用这个矩阵来变换一个点V(Vx, Vy, Vz, 1),需要把这个点右乘变换矩阵

                        [Xx, Yx, Zx, Tx]   [Vx]  
                       [Xy, Yy, Zy, Ty]   [Vy]  
V' = M\*T =       [Xz, Yz, Zz, Tz] \* [Vz]  
                       [0, 0, 0, 1   ]   [1 ]

对于V变换后的x分量，Vx' = Xx\*Vx + Yx\*Vy + Zx\*Vz + Tx,我们可以发现影响V的x分量的只有X,Y,Z轴旋转的x分量和平移的x分量，对于V的y, z分量也是同样道理。

**2 行主序， 列主序**  
OpenGL中推荐用一维数组来表示此转换矩阵 ： typedef GLfloat Matrix16[16];  
为了能快速的访问X轴， Y轴， Z轴， 该数组是按列主序来表示这个矩阵的：  
[m0, m4, m8, m12]  
[m1, m5, m9, m13]  
[m2, m6, m10,m14]  
[m3, m7, m11,m15]  
这样， 为了访问X轴， 即访问m0, m1, m2,因为他们是连续的存储空间，所以速度比较快， 相反， 如果我们数组按行主序来表示这个矩阵：  
[m0, m1, m2, m3 ]  
[m4, m5, m6, m7 ]  
[m8, m9, m10, m11]  
[m12, m13, m14, m15]  
我们发现为了访问X轴， 即m0, m4, m8， 是不连续的地址， 因此速度就慢了下来。  
所以我们可以知道， OpenGL为什么采用列主序的矩阵， 那是因为其所定义的转换矩阵如果按列主序存入数组， 我们对X,Y,Z轴就可以有较快的访问速度。也就是说， 如果我非要把这个矩阵按列主序的方式存入数组也可以， 只不过速度慢了点而已。（当然， 我们要告诉OpenGL我们是按行主序表示的）。

其实， 如果我们换一种方式来表示转换矩阵：  
                [Xx, Xy, Xz, 0]  
                [Yx, Yy, Yz, 0]  
M' =         [Zx, Zy, Zz, 0]  
                [Tx, Ty, Tz, 1]

这个矩阵是是前一个转换矩阵的转置，我们把这个矩阵按行主序存入数组就比较划算了。原因很明显， 为了快速访问X轴，我们希望Xx, Xy, Xz是连续存储的， 那么自然要按行存储了。

其实， 如果让我设计OpenGL,我会选择用第二种方式来表示转换矩阵，原因如下：  
如果我要转换一个点V, 依次经过三个转换矩阵L, M, N的转换， 那么对于第一种方式：  
V' = N\*(M\*(L\*V)) = (N\*M\*L) \* V  
我们的组合转换矩阵是N\*M\*L， 与我们定义的转换过程刚好相反， 但是， 如果我们是第二种方式表示的话，我转换一个点是左乘转换矩阵而不是右乘了：  
V' = ((V\*L)\*M)\*N = V \* (L\*M\*N)  
组合转换矩阵是按我们变换的顺序组合起来的， 就比较直观了， 然后我们按行主序存储此矩阵， 速度依然。

**3 二维数组存储矩阵**  
很多人有这样错误的认识， 就是在OpenGL中如果用二维数组来表示转换矩阵， 速度就比较慢， 而这种认识或多或少源于<<OpenGL超级宝典>>中的阐述。但是， 事实是这样吗？  
二维数组如下：  
typedef GLfloat Matrix44[4][4];

按我们理解的，逻辑上的二维数组， 其表示为：  
[m00, m01, m02, m03]  
[m10, m11, m12, m13]  
[m20, m21, m22, m23]  
[m30, m31, m32, m33]  
因为这个逻辑模型， 导致我们产生那种错误的认识：  
X轴是用m00, m10, m20表示的， 而他们是不连续的， 所以比较慢， 但是， 这只是其逻辑模型， 如果按逻辑模型去理解的话， 一维数组的逻辑模型是：  
[m0, m1, m2, m3, m4, m5, m6, m7, m8, m9, m10, m11, m12, m13, m14, m15]  
那我们是不是可以说， 一维数组根本不能用来表示矩阵？ 当然不是。  
其实， 不论是一维数组还是二维数组， 其在内存中的物理模型都是连续的16个float型的内存单元：  
一维数组：[m0, m1, m2, m3, m4, m5, m6, m7, m8, m9, m10, m11, m12, m13, m14, m15]  
二维数组：[m00, m01, m02, m03, m10, m11, m12, m13, m20, m21, m22, m23, m30, m31, m32, m33]  
看到这里， 既然一维数组可以用列主序表示并很快， 为什么二维数组就不快了呢？他们除了访问时的名字不一样， 本质上并没有区别啊：  
[m00, m10, m20, m30]  
[m01, m11, m21, m31]  
[m02, m12, m22, m32]  
[m03, m13, m23, m33]  
我们可以看到，二维数组按列主序表示的转换矩阵是这样的， 访问X轴即访问m00, m01, m02， 连续的， 一样快。  
只不过， 这种表示方式和我们所理解的二维数组的逻辑模型不太统一， 有些不直观罢了。这一点在OpenGL红宝书的说的比较正确：二维数组的元素m[i][j]将位于OpenGL变换矩阵的第i列， 第j行， 因此容易产生行列混淆，为了避免行列混淆， 推荐用一维数组表示。 真正的原因是为了避免行列混淆， 而不是速度。