十一、使用矩阵进行顶点变换

1. 使用矩阵进行变换

回到前一章的简单仿射变换，假设现在有一点 P,我们对其进行平移 D，可以知道平移之后的点 P'的计算方法为$P_x = P_x + D_x$、$P_y = P_y + D_y$、$P_z = P_z + D_z$。这时候我们可以使用点 P 构造一个齐次坐标$(P_x, P_y, P_z, 1)$,同时我们使用 P 的齐次坐标构建一个矩阵

$$A = [P_x \quad P_y \quad P_z \quad 1]$$

这时候我们再使用平移量 S 构造另一个矩阵：

$$B = \begin{bmatrix} 1 & 0 & 0 & D_x \\ 0 & 1 & 0 & D_y \\ 0 & 0 & 1 & D_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

我们使用矩阵

$$C = A \times B$$

可以得到一个新矩阵：

$$C = [P_x + D_x \quad P_y + D_y \quad P_z + D_z \quad 1]$$

可以发现新得到的矩阵就是平移结果的齐次坐标，所以我们可以采用矩阵方式表达平移，为什么这样表达呢？下一节我会说明。

同时注意，我们可以用列排序的方式构造 P 的矩阵为：

$$A = \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix}$$

这时候我们就应该用

$$C = B \times A$$

的方式来计算平移结果，并且平移结果为

$$C = \begin{bmatrix} P_x + D_x \\ P_y + D_y \\ P_z + D_z \\ 1 \end{bmatrix}$$

至于到底用那种方式，完全看个人喜好，不过 OpenGL 是采用列排序的方式计算的，也就是后一种。在本章，我将会使用前一种方式，这也是说明方法、原理和结果一样的话，到底是行表达还是列表达我们不必拘泥。

对于缩放和平移，我们同样可以构造如下的计算矩阵：

缩放矩阵：

$$\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

其中S_x、S_y、S_Z分别表示在 X、Y、Z 轴上面的缩放比例。

绕 X 轴旋转α的旋转矩阵：

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha & 0 \\ 0 & sin\alpha & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

绕 Y 轴旋转β的旋转矩阵:

$$\begin{bmatrix} cos\beta & 0 & sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\beta & 0 & cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

绕 Z 轴旋转γ的旋转矩阵:

$$\begin{bmatrix} cos\gamma & -sin\gamma & 0 & 0 \\ sin\gamma & cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

透视投影矩阵:

$$\begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{f+n}{f-n} & \frac{2fn}{n-f} \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

一般 OpenGL 使用的透视投影矩阵

$$\begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{2fn}{n-f} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

如果读者有学过 OpenGL 的透视投影矩阵，可以发现我们上面的透视矩阵中的第 3 行第 3 列和第 4 行第 3 列的符号和某些人给出的不一样。这是因为我们的相机坐标系是把屏幕向里定义为 Z 轴正方向，我们的 CVV 也是屏幕向里为 Z 轴正方形。而取负号这种的相机坐标系是屏幕向外为正，而他们的 CVV 和我们一样，所以第三和第四行略有不同(需要改变符号)，有兴趣的同学可以自己按照本系列文章前面的内容自行推导(非常简单)。这里也是看个人喜好来处理了，但是自己要记得到底使用哪一种，如果我们的模型是采用右手坐标系的化，需要使用 OpenGL 的这种矩阵(从 3ds Max 导出的模型是右手坐标系的模型)。

2. 多个变换的合并

   假设现在有一点 P，经过多个矩阵变换(假设按顺序经过 ABC 三个变换)。我们应该先计算 P×A，得到一个新顶点 P'，然后使用 P'×B 得到 P''，然后继续使用 P''×C 得到最终点 P_e，并且我们可以写成一个式子: $P_e = P \times A \times B \times C$，对于每个顶点，我们需要计算三次矩阵乘法。根据矩阵乘法的结合律，我们可以写成$P_e = P \times (A \times B \times C)$，我们提前计算好矩阵$M = (A \times B \times C)$，后续每个顶点只需要计算一次$P_e = P \times M$，这样可以节省运算量，并且我们可以发现，每个矩阵表示了一个变换，放在前面的矩阵表示这个变换先执行，我们可以把几次变换合并成一个变换。