坐标系与矩阵(6): 模型视图投影矩阵

模型视图投影矩阵，也就是常说的 MVP，有很多的书和资料，参考资料中会列出我推荐的相关资料，会详细介绍推导过程。之所以还要写这一篇，是因为它比较重要，也为了保证‘坐标系与矩阵’系列文章的完整性。所以本篇主要是我对这块的理解，具体的公式推导尽可能不提。

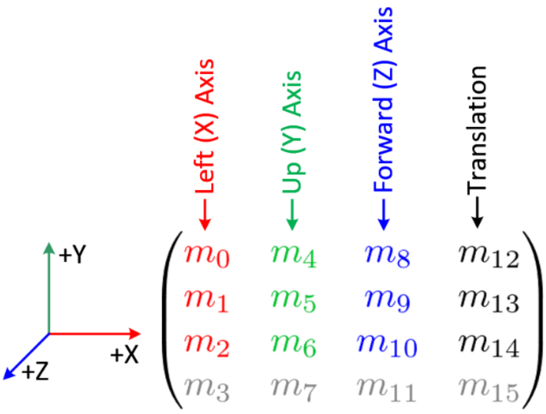
首先，假设我们需要装饰一间屋子，我们会把家具放在合适的位置，这个位置都是相对于房间中某一个原点的坐标系而言，类似第四篇中提到的 ECEF 和 ENU 之间的转换，存在一个矩阵，实现家具在房间坐标系（相对）的位置 p_{model} 转换到地球坐标系（绝对）下的位置 p_{world}，我们称为模型矩阵，记为 M_{model}：

$$p_{world} = M_{model} \cdot p_{model}$$

不难理解，p_{model} 和 p_{world} 在不同场景下都有意义和不同的优势。装饰后我们拍一张家居图，就要选一个合适的角度来拍摄了，所谓的横看成岭侧成峰。同样需要一个矩阵，实现家具在相机坐标系（相对）的位置 p_{view} 转换到地球坐标系（绝对）下的位置 p_{world}，我们称为视图矩阵，记为 M_{view}：

$$p_{world} = M_{view} \cdot p_{view}$$

基于之前的介绍，通常全球坐标系 F: X(1,0,0),Y(0,1,0),Z(0,0,1)确定，局部坐标系下三个轴的方向也确定的话，我们可以很容易的计算 M_{model} 和 M_{view}：



图片来自 http://www.songho.ca/

我们需要获取相机坐标系下对应的位置：

$$\begin{aligned} p_{camera} &= {M_{view}}^{-1} \cdot M_{model} \cdot p_{model} \\ &= M_{mv} \cdot p_{model} \end{aligned}$$

这样的好处是，就好比我们拍照片时，如果模型要变化，相机也要变，问题就比较复杂，但在这种情况下，等同于保持相机不变，而让人做调整，最终找到一个好的角度，通过减少变量的方式简化问题。

至此，我们介绍了模型视图矩阵，这里，多插一句，就是法线的转换。已知：

$$n \cdot p = (nx, ny, nz) \begin{pmatrix} x \\ y \\ z \end{pmatrix} = 0$$

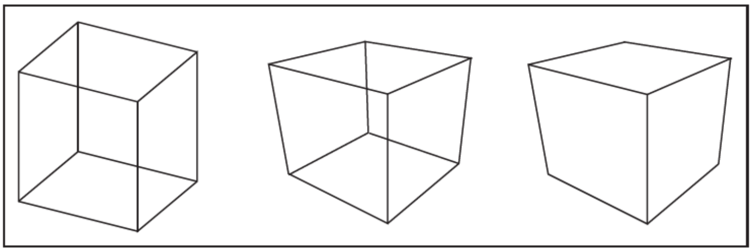
此时，已知一点$p(x,y,z)$，对应的法线$n(nx,ny,nz)$。该点经过矩阵M转换到新的坐标系下，对应的法线n'：

$$n' \cdot p' = n' \cdot M \cdot p = 0$$

两个公式可得，法线变化对应的矩阵是逆矩阵：

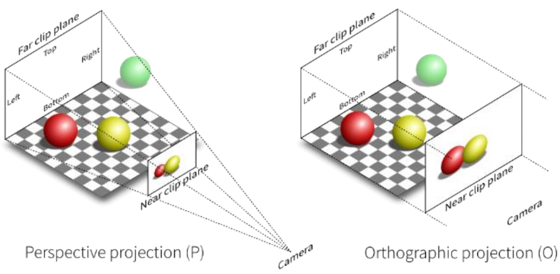
$$n' = (nx, ny, nz) \cdot M^{-1}$$

下面进入投影部分，既然是投影，就是一种降维求近似解的过程，我们可以理解为洗照片，把 3D 空间降维到 2D，最主要的有两种方式：正交投影和透视投影。



如上图显示了两者的主要区别。图中如下依次为正交投影，透视投影，没有 wireframe 的透视投影。可见，正交投影符合欧几里得的平行线不相交特性，更符合几何体在空间中的客观存在方式，比如乐高积木；而在透视投影下平行线则会相交，更符合人眼‘近大远小’的特点，比如‘鸽子为什么这么大’。

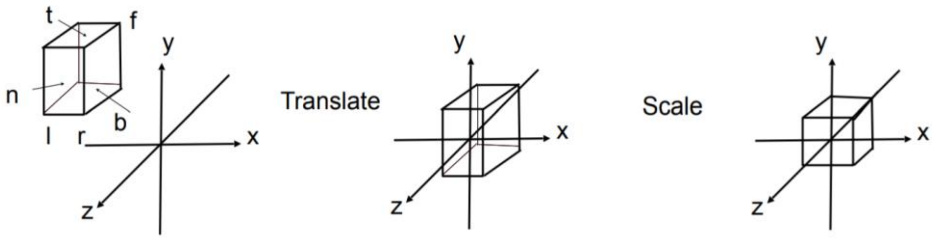
正交投影



如上图左侧，相机下形成一个四棱锥，我们会把影像投影到近裁剪面上，这也是各类设备和眼睛成像的基本原理，对应常见的透视投影。而正交投影类似于你把相机放到无限远，则此时远

近裁剪面之间的差别小到可以忽略不计，如右图，这就好比阳光，理论上都是从太阳这个点发射的，但在地球上，我们会觉得太阳光是平行的。

正交投影的过程非常简单，首先，我们定义一个$[-1,1]^3$之间的立方体，然后对成像场景构建一个包围盒，先做一个平移，将包围盒的原点平移到立方体的原点，再做缩放，则包围盒的三个方向都拉伸到相同长度的立方体，自然，包围盒中的几何对象映射到该立方体对应的范围，过程如下：

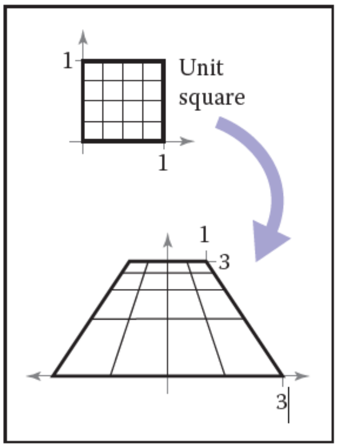


这里，对一个 top, bottom, left, right, near and far. 该过程对应的矩阵为：

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

这里要强调的是，此时我们采用的是右手坐标系，z 轴射向我们，所以$n > f$。

透视投影



上图，正交投影和透视投影下的区别体现了两者本质的区别，欧氏几何体现了是同一个平面内的关系，正交投影直接丢弃掉 Z 值形成了一个平面，因此保留了欧氏几何的规则。而透视投影则考虑了多平面，多视角下的区别。

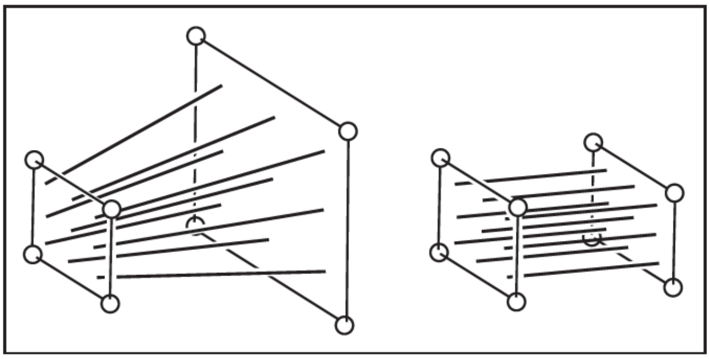
那么，如何让两条平行线相交呢？在第三篇介绍平移时，讲到了齐次坐标实现了仿射变换，这里，齐次坐标以增加一个维度的代价，实现了相同点在多平面下的表达方式。

$$\begin{cases} Ax + By + C = 0 \\ Ax + By + D = 0 \end{cases}$$

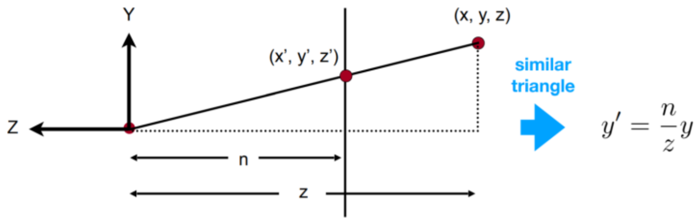
如上的两条平行线，本来是无解的，但在齐次坐标下，当$w = 0$时，也就是无穷远时有解：

$$\begin{cases} Ax + By + Cw = 0 \\ Ax + By + Dw = 0 \end{cases}$$

如何获取透视投影对应的矩阵呢，下图提供了一种直观思路，先把左侧的视锥体挤压成右侧，再基于右侧的正交投影就能解决该问题。



这样，只要我们掌握了挤压的算法，该问题就可以解决。我们定义两个挤压过程要遵守的规则，远近裁剪面对应的 z 值不变，远裁剪面的中心点挤压前后保持不变。而挤压对应相似三级凹形的映射关系：



基于相似三角形和 z 值的特点（近裁剪面所有点不变，远裁剪面的中心点不变），可得如果三个结论：

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} nx \\ ny \\ ? \\ z \end{pmatrix} \quad \begin{pmatrix} x \\ y \\ n \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} nx \\ ny \\ n^2 \\ n \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ f \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 \\ 0 \\ f^2 \\ f \end{pmatrix}$$

可得：

$$M_{persp \to ortho} = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -nf \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

这样，最终的透视投影矩阵以及投影矩阵有两种情况：

$$M_{persp} = M_{ortho} M_{persp \to ortho}$$

$$M_{projection} = \begin{cases} M_{persp} \\ M_{ortho} \end{cases}$$

这样，我们可以得到最终的模型视图投影矩阵，实现将 3D 空间下的p_{model}映射到 2D 平面：

$$\begin{aligned} p_{projection} &= M_{projection} \cdot M_{mv} \cdot p_{model} \\ &= M_{mvp} \cdot p_{model} \end{aligned}$$

下一篇和本篇在原理上没有区别，但主要专注于视觉中相机本身的范畴。

参考资料：OpenGL Transformation http://www.songho.ca/opengl/gl_transform.html

GAMES101 https://sites.cs.ucsb.edu/~lingqi/teaching/games101.html

Fundamentals of Computer Graphics 第七章