Note for lectures 3 and 4

图像变换是图形学的基础，变换时的先后顺序是有影响的

旋转有两种表示
1. 欧拉角（row,pitch,yaw)
2. 四元数

若绕静坐标系（世界坐标系）旋转，则左乘，也是变换矩阵坐标矩阵；若是绕动坐标系旋转（自身建立一个坐标系），则右乘，也就是坐标矩阵变换矩阵。 即，左乘是相对于坐标值所在的坐标系（世界坐标系）下的三个坐标轴进行旋转变换。而右乘则是以当前点为旋转中心，进行旋转变换。

How to implement a basic transformation

- 变换可以合成（把所有变换的矩阵先乘到一起）也可以分解（把一个复杂矩阵分解成多个简单矩阵），要注意矩阵乘法的顺序是从右到左
- **Transform Ordering Matters!**
  Since Matrix multiplication is not commutative and applied right to left, so
  $R_{45} * T_{(1,0)} / ne T_{(1,0)} * R_{45}$

Using matrices

由于齐次方程就是在基础的变换阔一圈，所以不给出

- Rotation(旋转),scale(缩放), shear(拉伸),这三位又称为线性变换
  Linear Transforms

$$x' = a\,x + b\,y$$
$$y' = c\,x + d\,y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{x}' = \mathbf{M}\,\mathbf{x}$$

Homogeneous coordinates

WHY?

为什么我们舍弃基础变换，就是因为平移(不是线性变换)需要另外一套矩阵表示平移的位置

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

We don't want translation to be a special case
但是由以上表达可知，就算换到齐次坐标，也是先线性变换再平移

SOLUTION: HOMOGENOUS COORDINATES

- 多加一个纬度表示点或者是向量，目的是为了得到一个正确的结果(比如点-点是向量)
  - 如果是向量,后面多加一个0 $2D\ vector\ \hat{v} = (x, y, 0)^T$
  - 如果是点，后面多加一个1 $2D\ point\ = (x, y, 1)^T$

有时候点不是标准的表示法

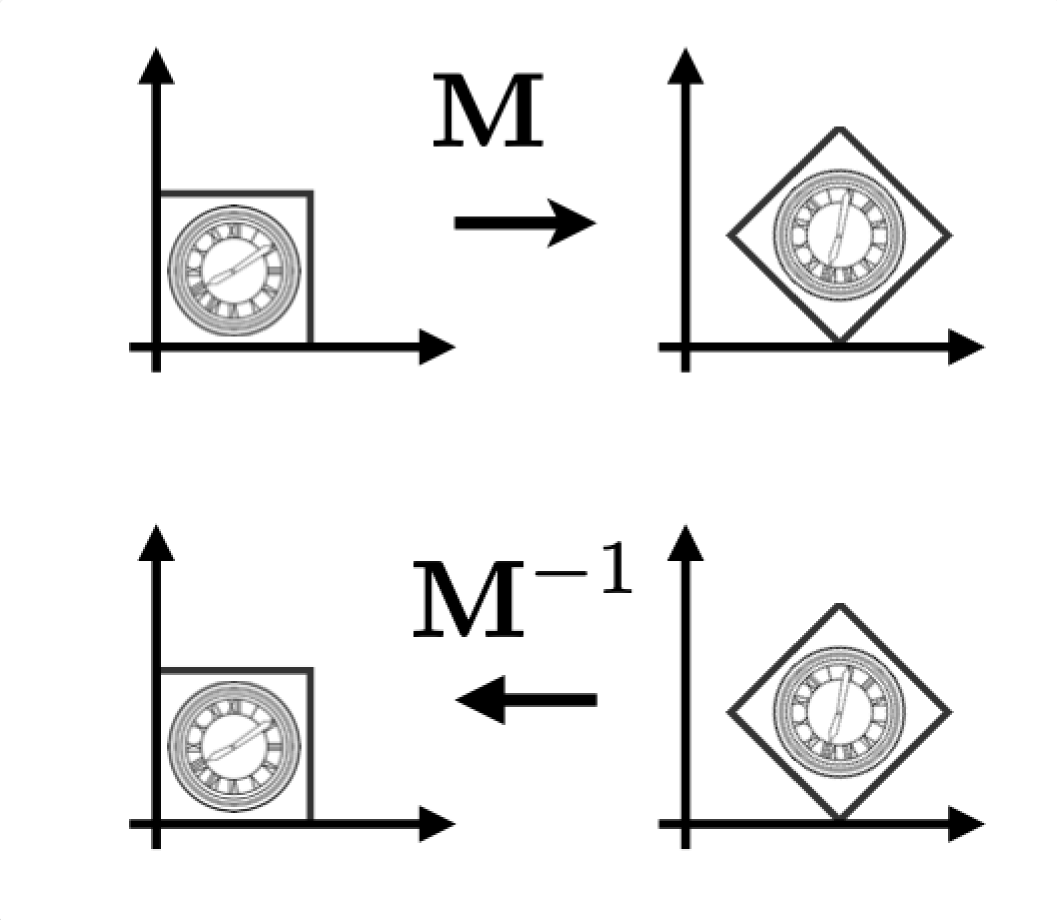
$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} \text{ is the 2D point } \begin{pmatrix} x/w \\ y/w \\ 1 \end{pmatrix}, w \neq 0$$

Affine Transformations(仿射变换)

仿射变换=线性变换+平移
用齐次坐标表示

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- Inverse Transform逆变换
  对变换的矩阵取反，相应的几何变换也会复原



2D CASE

- Scale

$$S(s_x, s_y) = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Rotation

$$R(\alpha) = \begin{pmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Translation

$$T(t_x, t_y) = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

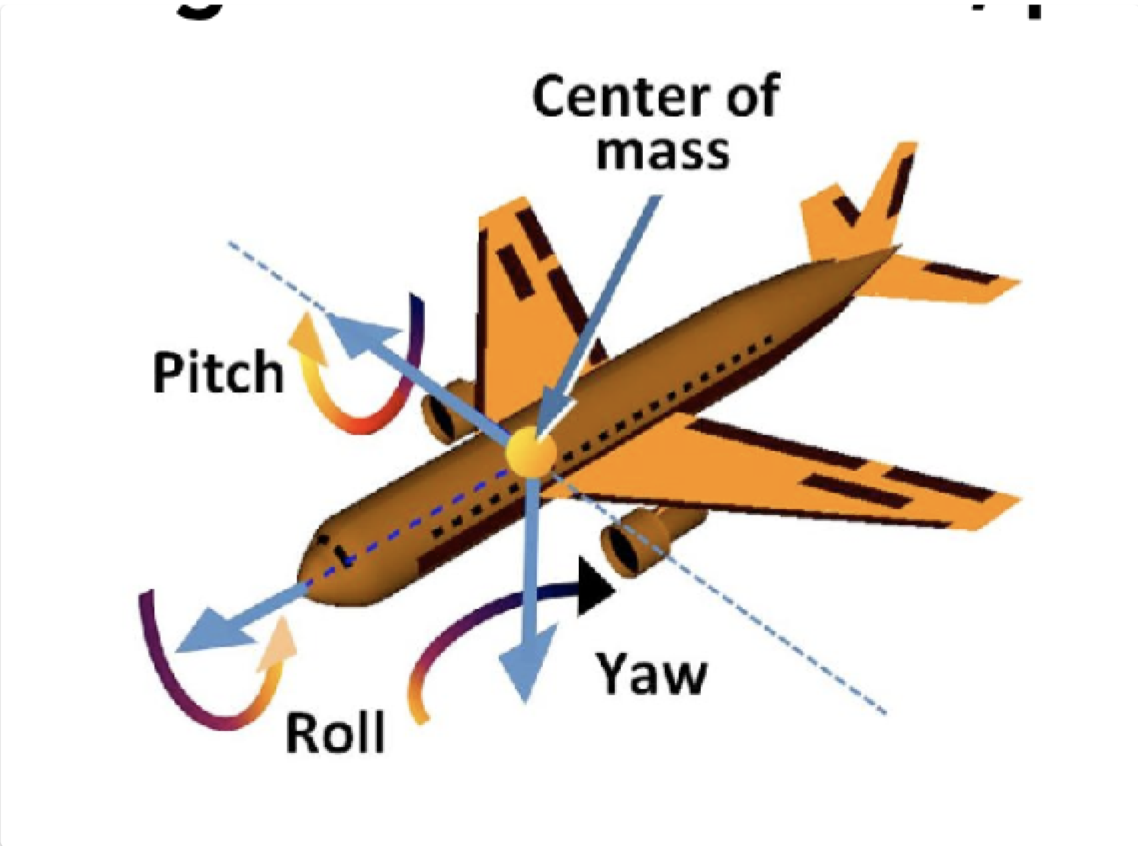
3D CASE

scale translation就是把z的那圈加上

Rotation

物体绕任何轴旋转都可以分解到xyz三个方向上三维空间绕任意轴旋转矩阵的推导

Euler angles


Center of mass
Pitch
Roll
Yaw

Rotation around x-, y-, or z-axis

$$\mathbf{R}_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R}_y(\alpha) = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R}_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rodrigues' Rotation Formula

绕过原点轴n转α角度
证明留个坑以后写Appendix 1 罗格斯证明

$$\mathbf{R}(\mathbf{n}, \alpha) = \cos(\alpha)\mathbf{I} + (1 - \cos(\alpha))\mathbf{n}\mathbf{n}^T + \sin(\alpha) \begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}$$

MVP (not the most valuable player)

View(视图) transformation和Projection(投影)transformation合称为Viewing(观测) transformation

Model transformation(placing objects)

Just like finding a good place and arrange people

View/Camera/ModelView transformation (placing camera)

Just like finding a good "angle" to put the camera
把相机整到原点，对齐xyz看向-z方向
但是一个特殊矩阵不好转到规范坐标系，但是转规范坐标系是好转到特殊矩阵的，所以我们反着绕地球一圈，再取逆就得到了我们想要的 逆变换原理

- Rotate g to -Z, t to Y, (g x t) To X
- Consider its inverse rotation: X to (g x t), Y to t, Z to -g

$$R_{view}^{-1} = \begin{bmatrix} x_{\hat{g}\times\hat{t}} & x_t & x_{-g} & 0 \\ y_{\hat{g}\times\hat{t}} & y_t & y_{-g} & 0 \\ z_{\hat{g}\times\hat{t}} & z_t & z_{-g} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \xrightarrow{\text{WHY?}} \quad R_{view} = \begin{bmatrix} x_{\hat{g}\times\hat{t}} & y_{\hat{g}\times\hat{t}} & z_{\hat{g}\times\hat{t}} & 0 \\ x_t & y_t & z_t & 0 \\ x_{-g} & y_{-g} & z_{-g} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Projection transformation(从摄影机眼中看)

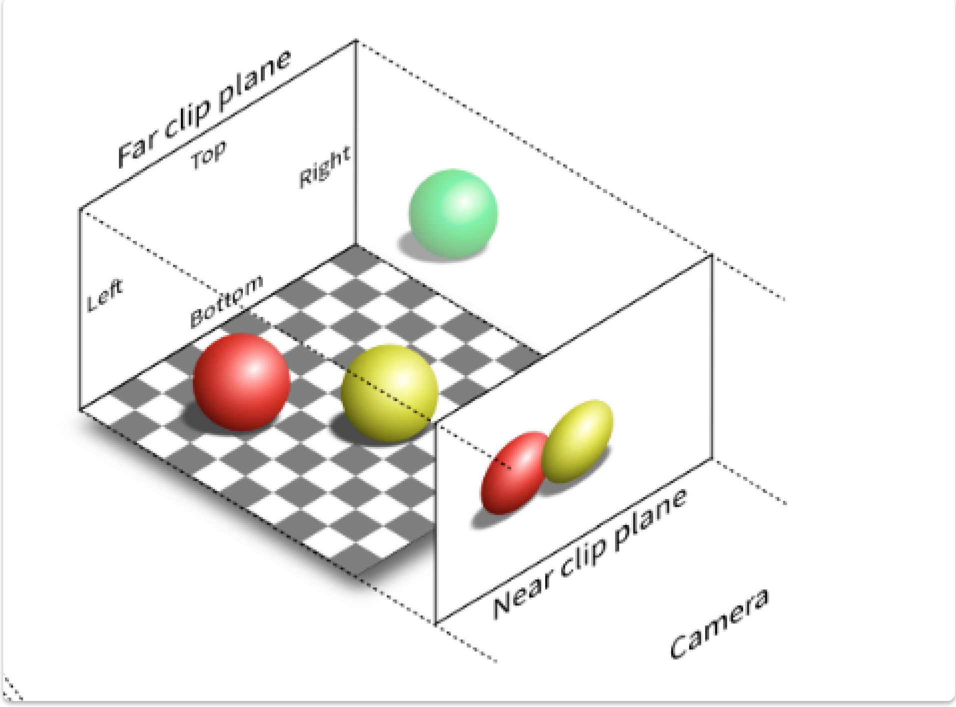
Take a photo, cheese!

- 3D to 2D

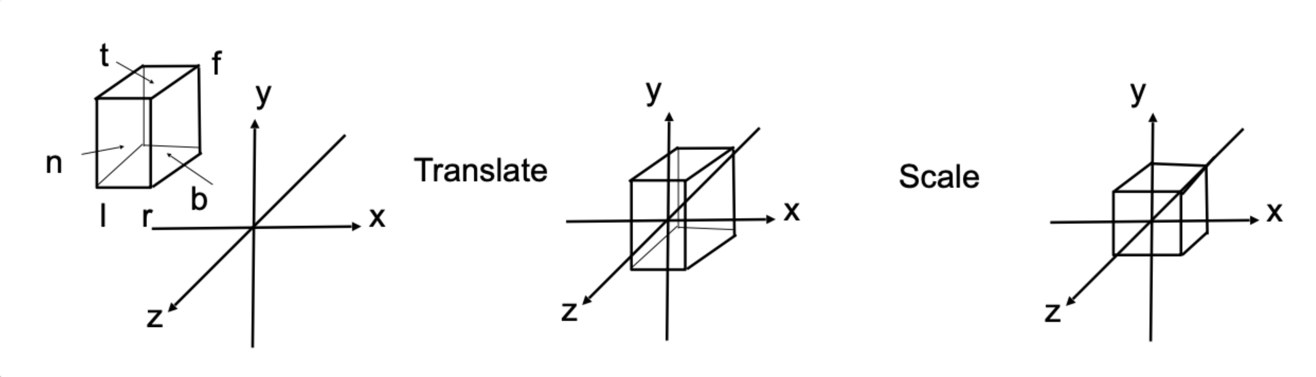
“canonical” cube是一个放置在原点的从-1到1的正立方体

ORTHOGRAPHIC PROJECTION 正交投影(CUBOID TO “CANONICAL” CUBE)

会对物体造成一定的拉伸，怎么解决之后会说
相机无限远，打过来的平行光



- Step
  1. Center cuboid by translating 也就是把立方体整到原点
  2. Scale into “canonical” cube

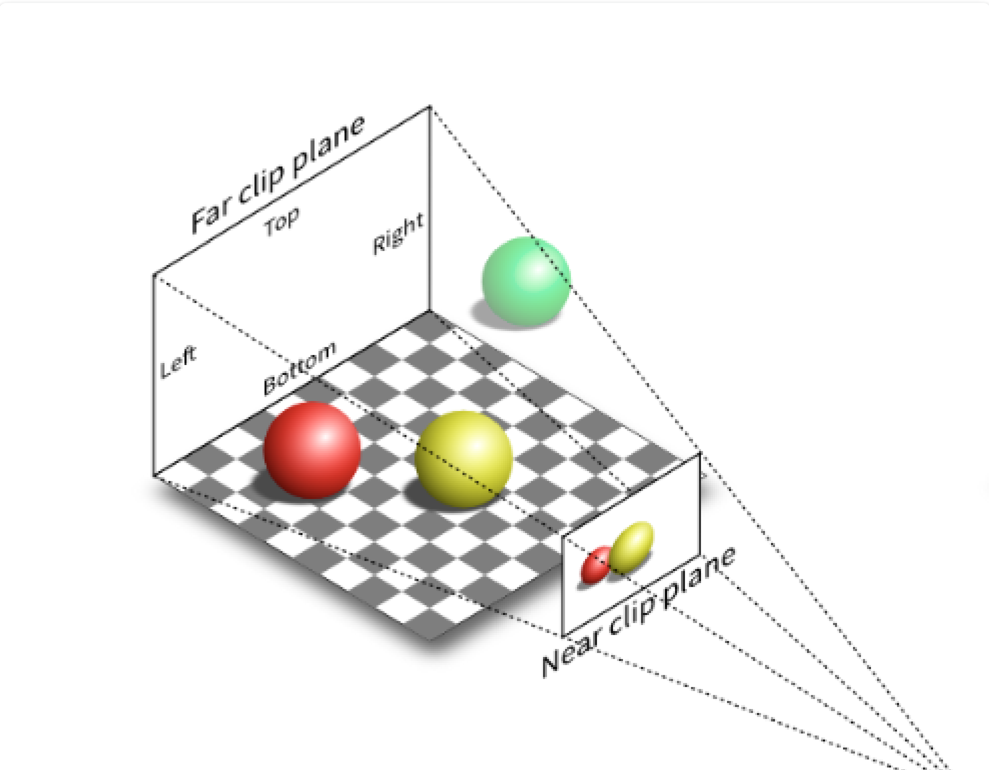


- Transformation matrix

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

PERSPECTIVE PROJECTION 透视投影(FRUSTUM TO “CANONICAL” CUBE)

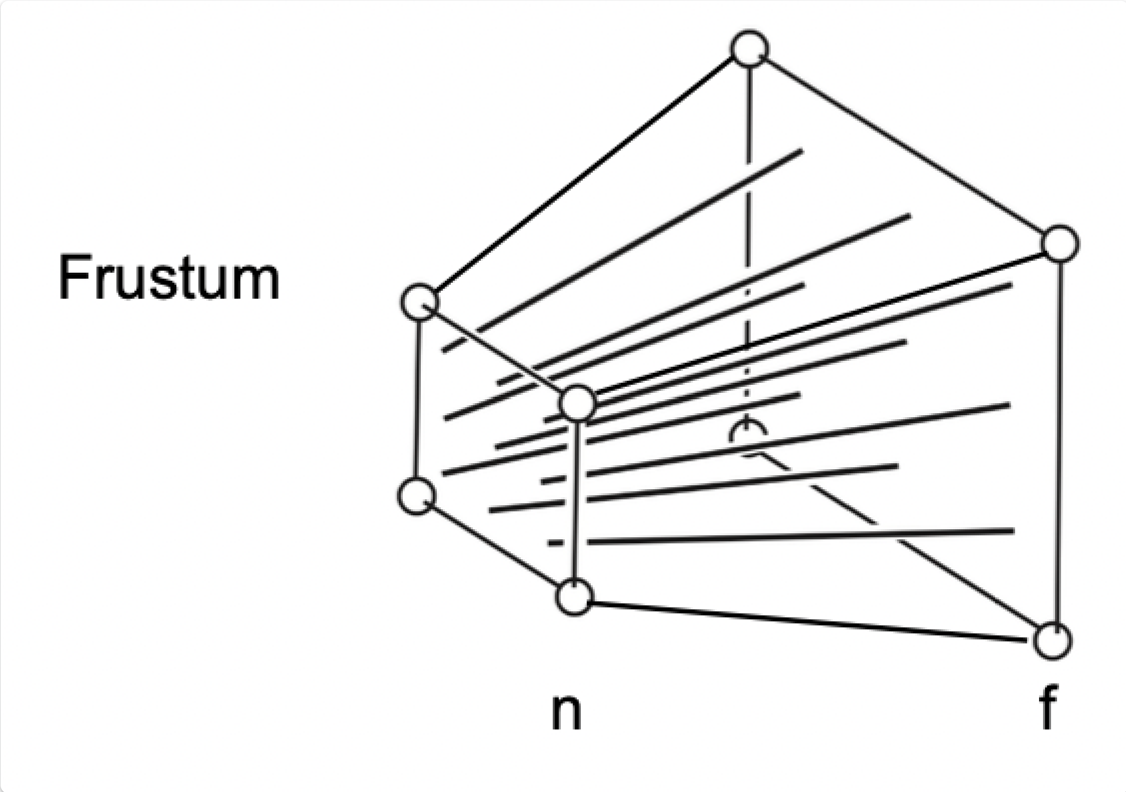
也就是画画的透视法



Perspective projection (P)

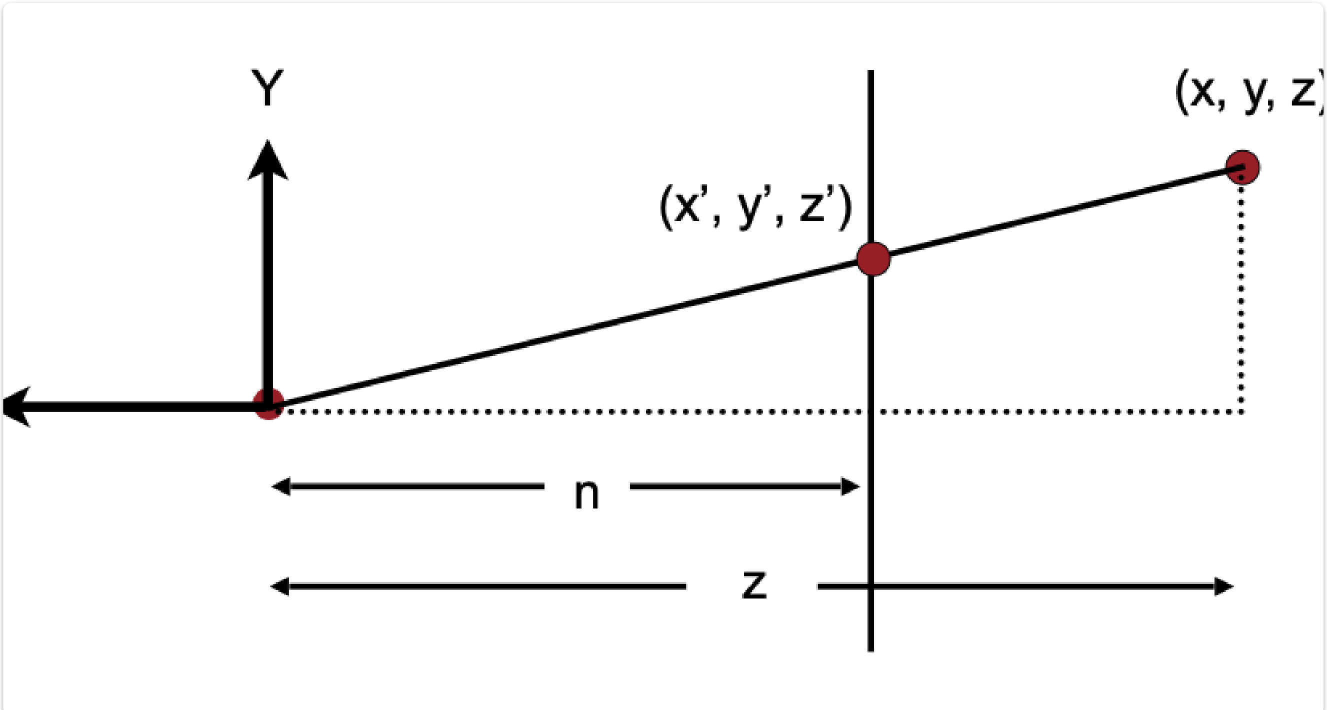
Step

1. First “squish” the frustum into a cuboid



transformation

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} -> \begin{pmatrix} nx \\ ny \\ unknown \\ z \end{pmatrix}$$



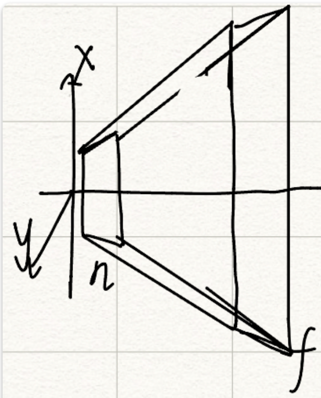
we get $y' = \frac{n}{z} y$

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

2. Then Do orthographic projection

We know how to do

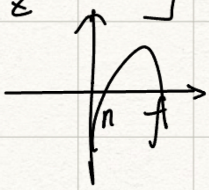
问nf中间处是向后移还是向前移了？



设点 $\begin{bmatrix} A \\ B \\ z \\ 1 \end{bmatrix}$, $n \le z \le f$

$$M_{prep \to ortho} \begin{bmatrix} A \\ B \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} nA \\ nB \\ (n+f)z - nf \\ z \end{bmatrix}$$

设 $h(x) = (n+f)z - nf - z^2$



$h(n) = h(f) = 0$

$h(x)$ 在 (n, f) 恒大于0.

得证 (n, f) 之间的点，变换后向后移动