Lecture 04

本节课的内容：

- 3D 变换
- 观测变换
  - 视图变换 / 摄像机变换
  - 投影变换
    - 正交投影
    - 透视投影

1.3D 变换

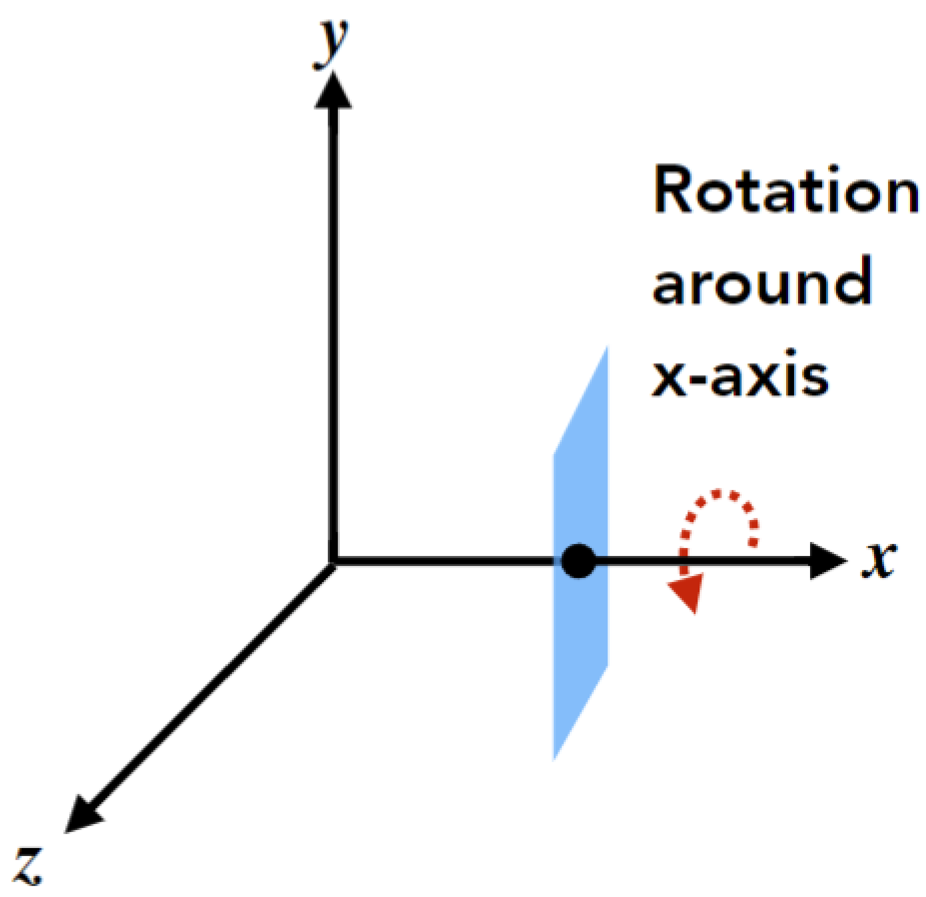
1.定义3D的点与向量

- 点： $(x, y, z, 1)^T$
- 向量：$(x, y, z, 0)^T$

2.3D变换表示

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

- 对于 缩放/平移，其与 2D 形式无异
- 对于 旋转，在 3D 中，其基础的旋转是 **绕轴旋转**



1.绕X轴旋转

对于绕 X 轴旋转，需要做的是保证点的 X 轴上坐标不发生变化，而其他进行角度为 α 的旋转，即

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

2.绕Z轴旋转

与 X 轴类似

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

3.绕Y轴旋转

由于叉乘的性质，即$z \times x = y$，而绕 Y 轴旋转的矩阵与前两者有一些不同

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

4.分解复杂 3D 旋转

所有复杂的旋转都能分解到 X, Y, Z 三个轴上的不同角度的旋转，即

$$\mathbf{R}_{xyz}(\alpha, \beta, \gamma) = \mathbf{R}_x(\alpha)\mathbf{R}_y(\beta)\mathbf{R}_z(\gamma)$$

- 三个角度被称为 欧拉角

而更近一步，对于绕某一个过原点的轴，其方向为 n 旋转 α 时，其 **Rodrigues‘Rotation Formula** 可以写作

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

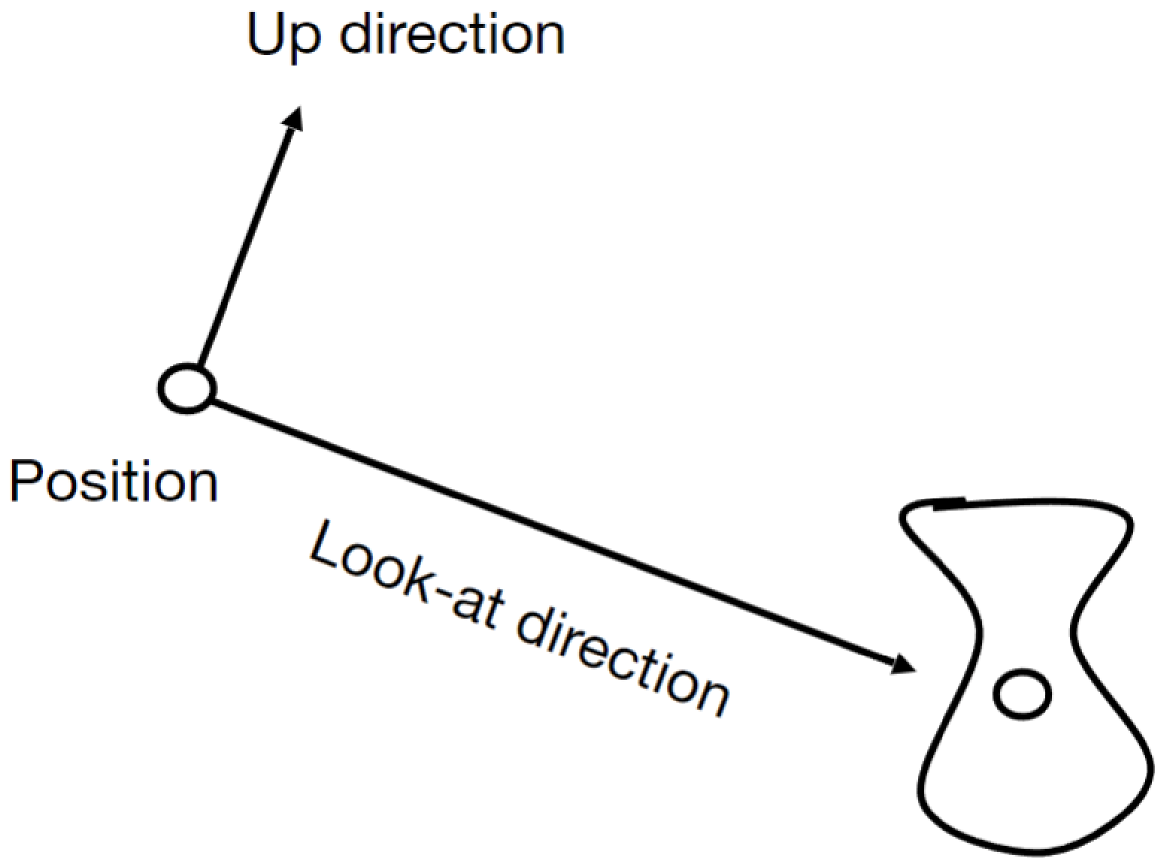
2.观测变换

由三部分组成

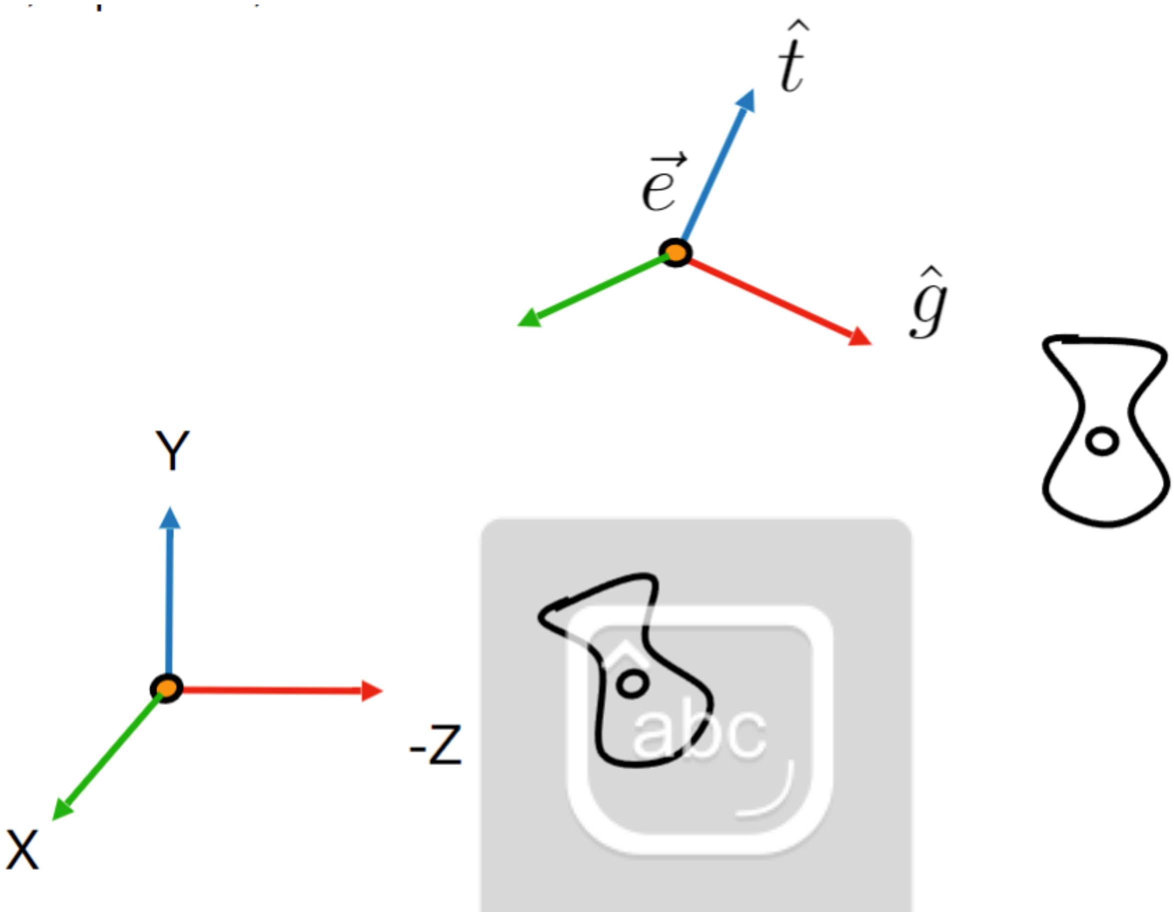
- 模型变换
- 视图变换
- 投影变换
- 简称为 MVP

1.模型视图变换

- 相机/视图的定义由三部分组成
  - 位置 $\vec{e}$
  - 视角方向 $\hat{g}$
  - 向上方向 $\hat{t}$



- 由于相机和物体一起移动后，成像结果相同，所以将相机移动至原点更好，即
- 将相机/视图变换至 M_{view}
  - 移动 e 至原点
  - 旋转 g 朝向 $-Z$
  - 旋转 t 朝向 Y
  - 旋转 $g \times t$ 朝向 X



- 而 $M_{view} = R_{view}T_{view}$ 即等于先进行平移再进行旋转
- 对于 T_{view} ，其作用在于移动相机/视图位置到原点

$$T_{view} = \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

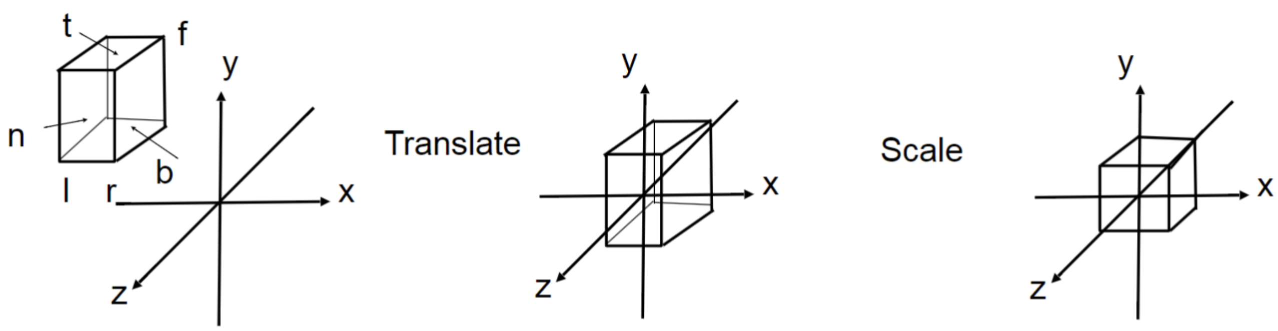
- 对于 R_{view} 将相机线性变换到原点并不容易实现，但是可以利用 **旋转矩阵是正定矩阵** 的性质，先求出 R_{view}^{-1} 再通过 **转置** 获得 R_{view}

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

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

2.投影变换

1.正交投影(orthographic projection)



- 选取一个立方体 $[l,r]\times[b,t]\times[f,n]$ (其中 l,r 为 X 轴左右范围，b,t 为 Y 轴上下范围，f,n 为 Z 轴远近范围)
  - 由于视图向 $-Z$ 轴看去，所以在数值大小上，$f<n$ ，在后面得计算会有体现。
- **平移** 立方体中心到原点
- **缩放** 立方体成为一个 **正则(canonical)立方体** ，其坐标范围为 $[-1，1]^3$

对于计算，我们先进行平移再进行缩放

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

- $f<n$ 所以应当是后者减去前者。
- 在 OpenGL 中使用 **左手坐标系**，其视图向 Z 轴看去，所以此时为 $[l,r]\times[b,t]\times[n,f]$ ，有不同之处。

2.透视投影(perspective projection)

- 符合远小近大的性质，比如平行线投影到某个平面后不再平行。
- 透视投影的步骤
  - 将 frustum “挤压”成 cuboid，即将远平面挤压成符合正交投影的平面
  - 再做一次正交投影

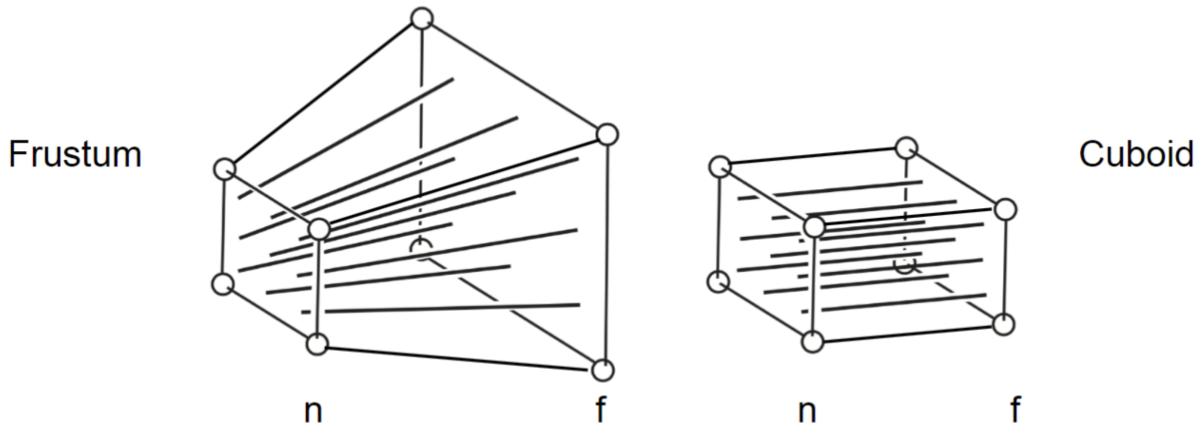
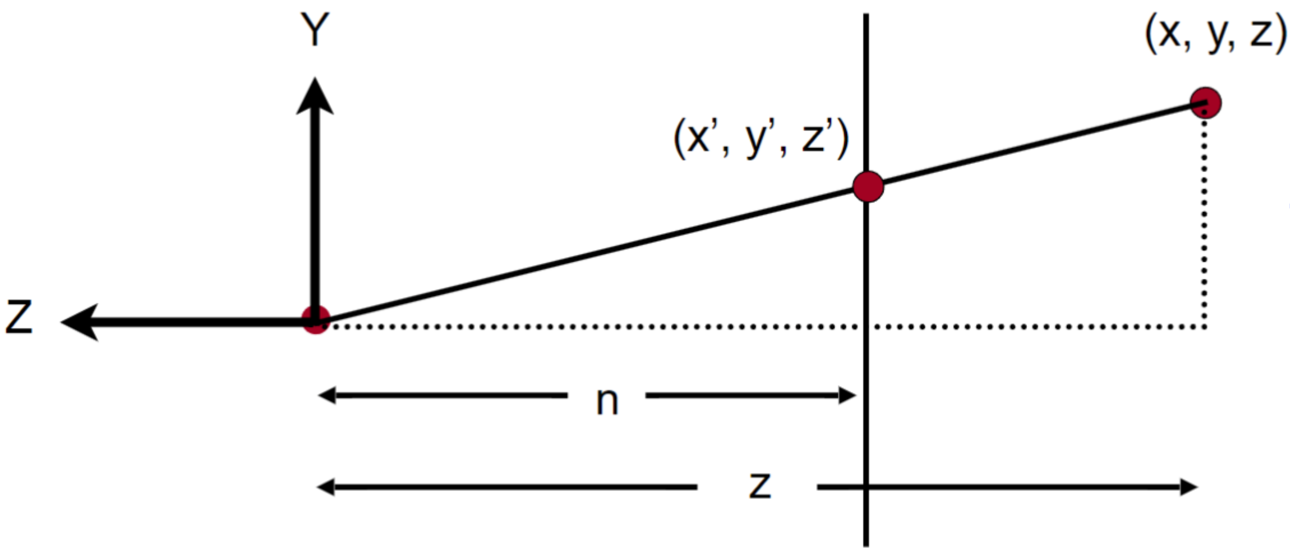


Fig. 7.13 from *Fundamentals of Computer Graphics, 4th Edition*

1.挤压

由于在近平面和远平面上的 Z 轴坐标不发生变化，我们可以从 Z, Y 轴构成的平面观看 “挤压过程”



得到 x, y 轴坐标的缩放公式

$$y' = \frac{n}{z}y$$
$$x' = \frac{n}{z}x$$

在齐次坐标系上可以写作

$$M_{presp \to ortho}^{4 \times 4} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} nx \\ ny \\ unkown \\ z \end{pmatrix}$$

由于近平面和远平面上的 Z 坐标在挤压过程中不发生改变，得到 z 坐标的表示形式

$$An + B = n^2$$
$$Af + B = f^2$$

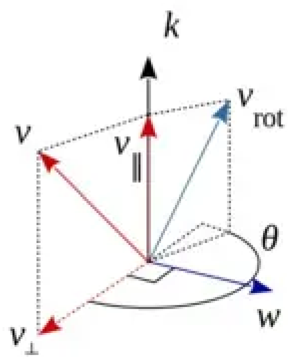
最终得到

$$M_{presp \to ortho}^{4 \times 4} = \begin{pmatrix} nx \\ ny \\ nfz + (-nf) \\ z \end{pmatrix}$$

在此之上再进行一次正交投影即可得到透视投影的结果

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

附 **Rodrigues' Rotation Formula** 的简易证明：



此处可得获取一些已知的条件

$$\begin{aligned}
&\mathbf{I} = v \\
&proj(\vec{s}, \hat{n}) = \hat{n}(\vec{s} \cdot \hat{n}) = nn^T\vec{s} \\
&N = k \times v
\end{aligned}$$

$$\begin{aligned}
&v = v_\perp + v\| \\
&v\| = (v \cdot k)k \\
&v_\perp = v - (v \cdot k)k
\end{aligned}$$

可得 v_{rot} 的组成

$$\begin{aligned}
&v_{rot} = v\| + v_{rot\perp} \\
&v_{rot\perp} = cos\theta v_\perp + sin\theta\omega \\
&\omega = k \times v
\end{aligned}$$

于是可得 v_{rot}

$$\begin{aligned}
&v_{rot} \\
&\quad = (v \cdot k)k + [v - (v \cdot k)k]cos\theta + sin\theta(k \times v) \\
&\quad = cos\theta v + (1 - cos\theta)(v \cdot k)k + sin\theta(k \times v) \\
&\quad = cos(\alpha)\mathbf{I} + (1 - cos(\alpha))\mathbf{nn}^T + sin(\alpha)\mathbf{N}
\end{aligned}$$