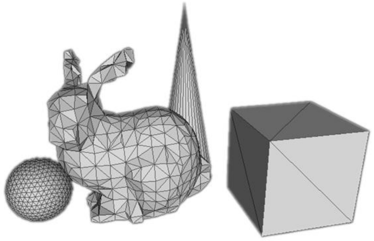
Introduction to Computer Graphics



Transformation

本节内容

- 向量
  - 向量概念、向量加减法
  - 向量坐标表示
  - 向量归一化
  - 向量运算
  - 点乘与叉乘
- 矩阵
  - 矩阵与向量的乘法
  - 矩阵与向量乘法的意义
  - 矩阵与矩阵的乘法
  - 正交基的构建、正交矩阵

本节内容

- 顶点变换流程
- 模型视图变换
  - 平移
  - 旋转
  - 缩放
  - 相机变换gluLookAt
- 投影变换
  - 正交投影
  - 透视投影

向量

- 向量
  - $\boldsymbol{a} = (a_1,\ a_2,\ a_3)^T$
  - $\boldsymbol{a}$的模长：　$|\boldsymbol{a}|=\sqrt{a_1^2 + a_2^2 + a_3^2}$
  - 归一化：　$$\frac{\boldsymbol{a}}{|\boldsymbol{a}|} = \frac{(a_1, a_2, a_3)^T}{\sqrt{a_1^2 + a_2^2 + a_3^2}}$$

向量

- 向量点乘（点积）
  - $\boldsymbol{a} = (a_1,\ a_2,\ a_3)^T$
  - $\boldsymbol{b} = (b_1,\ b_2,\ b_3)^T$
  - $\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{a}^{\boldsymbol{T}}\boldsymbol{b} = a_1b_1 + a_2b_2 + a_3b_3$

    $= |\boldsymbol{a}||\boldsymbol{b}|\boldsymbol{cos} < \boldsymbol{a}, \boldsymbol{b} >$
  - $\boldsymbol{a} \cdot \frac{\boldsymbol{b}}{|\boldsymbol{b}|}$ 几何意义：向量$\boldsymbol{a}$在$\boldsymbol{b}$方向上的投影

向量

- 向量叉乘（叉积）
  - $\boldsymbol{a} = (a_1,\ a_2,\ a_3)^T$
  - $\boldsymbol{b} = (b_1,\ b_2,\ b_3)^T$
  - $\boldsymbol{a} \times \boldsymbol{b} = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$

    $= (a_2b_3 - a_3b_2)\ \boldsymbol{i} + (a_3b_1 - a_1b_3)\boldsymbol{j} + (a_1b_2 - a_2b_1)\boldsymbol{k}$
  - $\boldsymbol{i} = (1,\ 0,\ 0)^T, \boldsymbol{j} = (0,\ 1,\ 0)^T, \boldsymbol{k} = (0,\ 0,\ 1)^T,$

矩阵

- 矩阵
  - 矩阵与向量的乘法
  - 投影观点

$$\begin{bmatrix} | \\ \mathbf{y} \\ | \end{bmatrix} = \begin{bmatrix} - & \mathbf{r_1} & - \\ - & \mathbf{r_2} & - \\ - & \mathbf{r_3} & - \end{bmatrix} \begin{bmatrix} | \\ \mathbf{x} \\ | \end{bmatrix}$$

  - 加权观点

$$\begin{bmatrix} | \\ \mathbf{y} \\ | \end{bmatrix} = \begin{bmatrix} | & | & | \\ \mathbf{c}_1 & \mathbf{c}_2 & \mathbf{c}_3 \\ | & | & | \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\mathbf{y} = x_1\mathbf{c}_1 + x_2\mathbf{c}_2 + x_3\mathbf{c}_3.$$

矩阵

- 矩阵
  - 矩阵与矩阵的乘法
  - AB=P

$$\begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & & \vdots \\ a_{i1} & \cdots & a_{im} \\ \vdots & & \vdots \\ a_{r1} & \cdots & a_{rm} \end{bmatrix} \begin{bmatrix} b_{11} & \cdots & b_{1j} & \cdots & b_{1c} \\ \vdots & & \vdots & & \vdots \\ b_{m1} & \cdots & b_{mj} & \cdots & b_{mc} \end{bmatrix} = \begin{bmatrix} p_{11} & \cdots & p_{1j} & \cdots & p_{1c} \\ \vdots & & \vdots & & \vdots \\ p_{i1} & \cdots & \boxed{p_{ij}} & \cdots & p_{ic} \\ \vdots & & \vdots & & \vdots \\ p_{r1} & \cdots & p_{rj} & \cdots & p_{rc} \end{bmatrix}$$

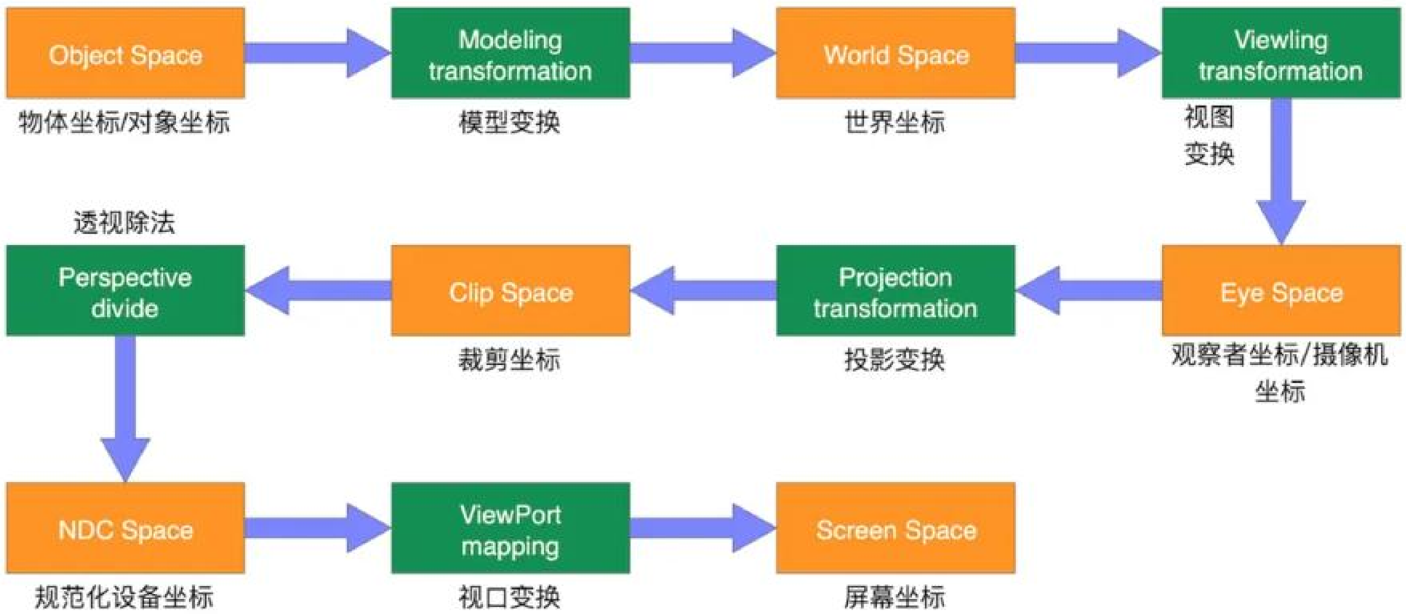
  - P=AB
  - $P^T = B^T A^T$
  - AB=AB (一般不成立!)

矩阵

- 正交基的构建、正交矩阵
  - 单位正交矩阵 $A^T A = A A^T = I$
  - 正交：各向量点积为0
  - 单位：各向量模长为1

  - 给定三个不共面的向量，如何构建正交基？
    - 叉乘

顶点变换的流程



Object Space
物体坐标/对象坐标
Modeling transformation
模型变换
World Space
世界坐标
Viewling transformation
视图变换
Eye Space
观察者坐标/摄像机坐标
Projection transformation
投影变换
Clip Space
裁剪坐标
透视除法
Perspective divide
NDC Space
规范化设备坐标
ViewPort mapping
视口变换
Screen Space
屏幕坐标

顶点变换的流程

模视变换

$$\begin{pmatrix} X_{eye} \\ Y_{eye} \\ Z_{eye} \\ W_{eye} \end{pmatrix} = M_{modelView} * \begin{pmatrix} X_{obj} \\ Y_{obj} \\ Z_{obj} \\ W_{obj} \end{pmatrix} = M_{view} * M_{model} * \begin{pmatrix} X_{obj} \\ Y_{obj} \\ Z_{obj} \\ W_{obj} \end{pmatrix}$$

投影变换

$$\begin{pmatrix} x_{clip} \\ y_{clip} \\ z_{clip} \\ w_{clip} \end{pmatrix} = M_{projection} * \begin{pmatrix} x_{eye} \\ y_{eye} \\ z_{eye} \\ w_{eye} \end{pmatrix}$$

透视除法

$$\begin{pmatrix} x_{ndc} \\ y_{ndc} \\ z_{ndc} \end{pmatrix} = \begin{pmatrix} x_{clip}/w_{clip} \\ y_{clip}/w_{clip} \\ z_{clip}/w_{clip} \end{pmatrix}$$

视口变换

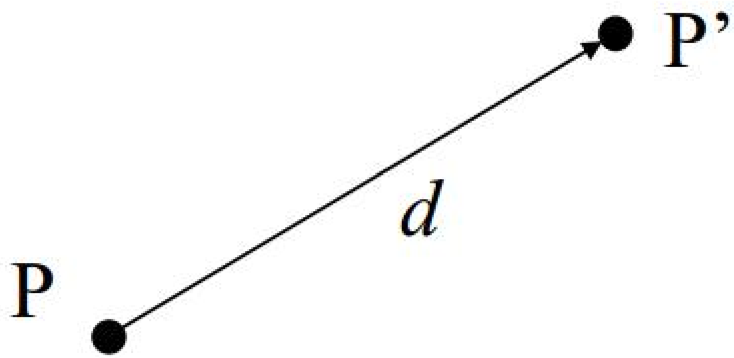
$$\begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} = \begin{pmatrix} \frac{w}{2} x_{ndc} + (x + \frac{w}{2}) \\ \frac{h}{2} y_{ndc} + (y + \frac{h}{2}) \\ \frac{f-n}{2} z_{ndc} + \frac{f+n}{2} \end{pmatrix}$$

模型视图变换

- 模型变换
  - 平移、旋转、缩放
  - 绕任意轴旋转
  - 复杂变换
- 相机变换（视点变换）

平移（translation)

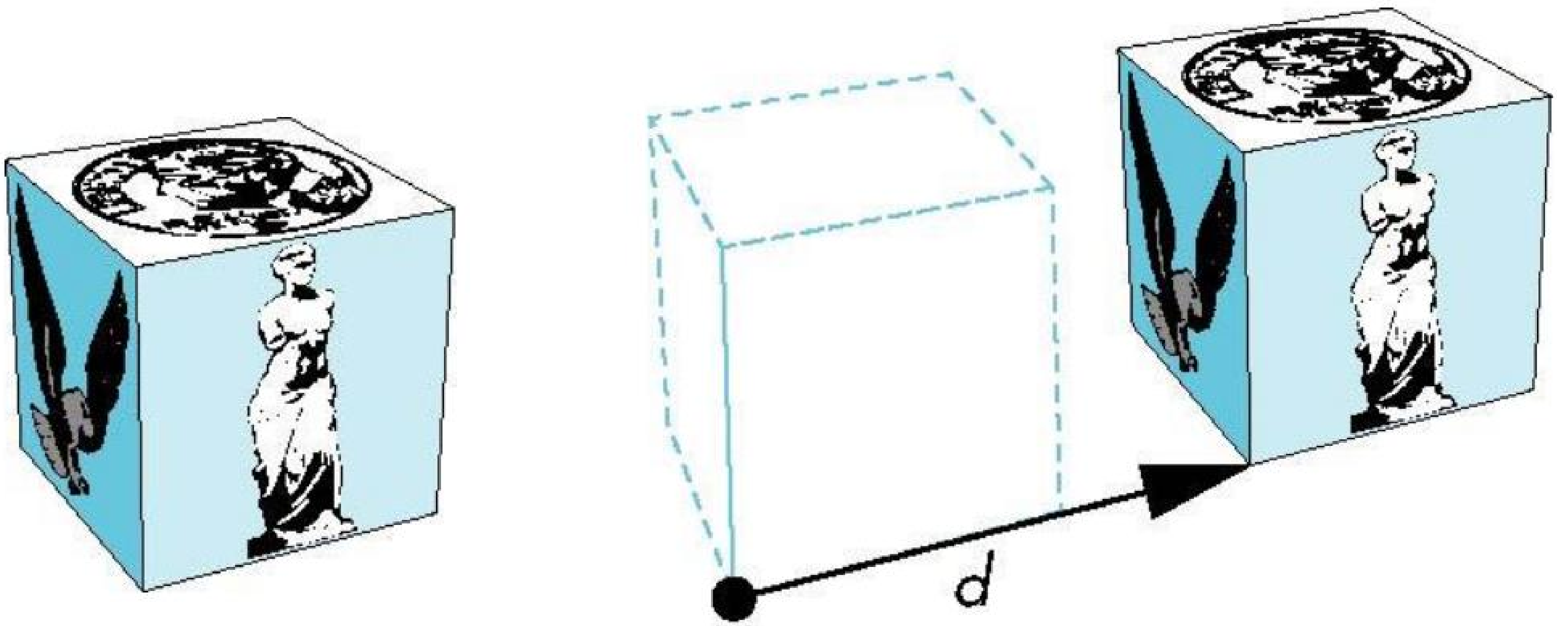
- 改变物体的位置



- 平移由平移向量*d*确定
  - 三个自由度 (degrees of freedom, DOF)
  - P' = P + *d*

对象的平移

- 把一个对象上的所有点沿同一方向移动相同距离



原始对象　　　　　　　　　　　　　　平移后的对象

平移的表示

- 应用在某个标架中的齐次坐标表示

$\mathbf{p}=[\ x\ y\ z\ 1]^T$

$\mathbf{p}'=[x'\ y'\ z'\ 1]^T$

$\mathbf{d}=[d_x\ d_y\ d_z\ 0]^T$

注意：这里是四维的齐次坐标，用的是点=点+向量

第四维度为1表示点，为0表示方向（向量）

因此 $\mathbf{p}' = \mathbf{p} + \mathbf{d}$ 或者

$x'=x+d_x$

$y'=y+d_y$

$z'=z+d_z$

平移矩阵

- 可以用在齐次坐标中一个4x4的矩阵**T**表示平移：**p'=Tp** 其中

$$\mathbf{T} = \mathbf{T}(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

请自行动手乘一下。

图形学中大量采用这种形式，这种形式更容易实现，因为所有的仿射变换（旋转和平移）都可以用这种形式统一表示，矩阵乘法可以复合在一起

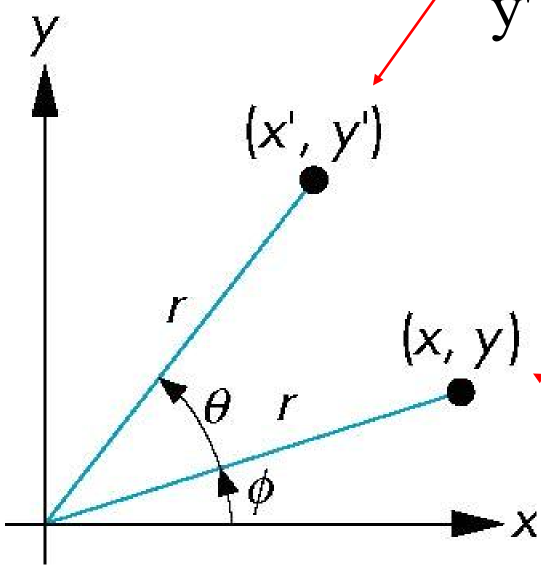
平移矩阵

- 请实现函数替代OpenGL的平移函数
  - myTranslate(float dx, float dy, float dz)
  - $$\begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

二维旋转（2-dimensional Rotation)

- 考虑绕原点旋转θ度
  - 半径保持不变，角度增加了θ

$x' = r \cos(\phi + \theta)$
$y' = r \sin(\phi + \theta)$

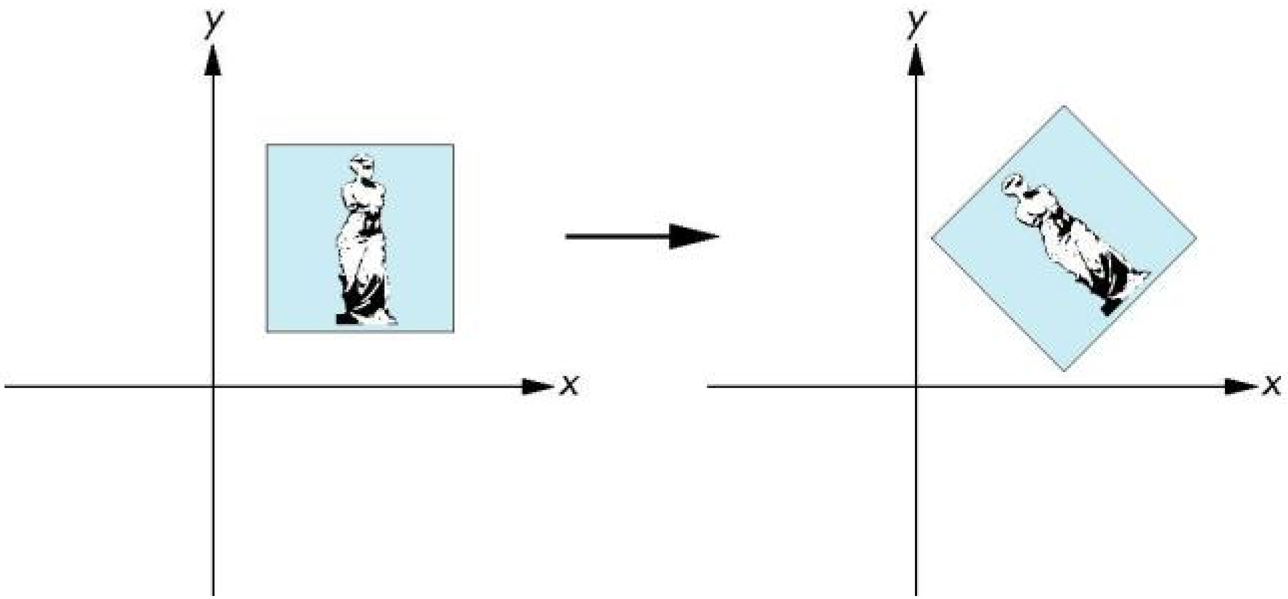


$x' = x \cos\theta - y \sin\theta$
$y' = x \sin\theta + y \cos\theta$

$x = r \cos\phi$
$y = r \sin\phi$

二维旋转

- 旋转轴：等效于三维空间绕z轴旋转
- 旋转角：（从z轴正方向看）逆时针方向为正向



绕z轴的旋转

- 在三维空间中绕z轴旋转，点的z坐标不变
  - 等价于在z=常数的平面上进行二维旋转

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$
$$z' = z$$

  - 其齐次坐标表示为

$$\mathbf{p}' = \mathbf{R}_{\mathbf{z}}(\theta)\mathbf{p}$$

绕z轴的旋转矩阵

x’=x cos θ –y sin θ
y’ = x sin θ + y cos θ
z’ =z

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

绕X轴和绕Y轴的旋转矩阵

- 与绕z轴的旋转完全类似
  - 对于绕x轴的旋转， x坐标不变

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

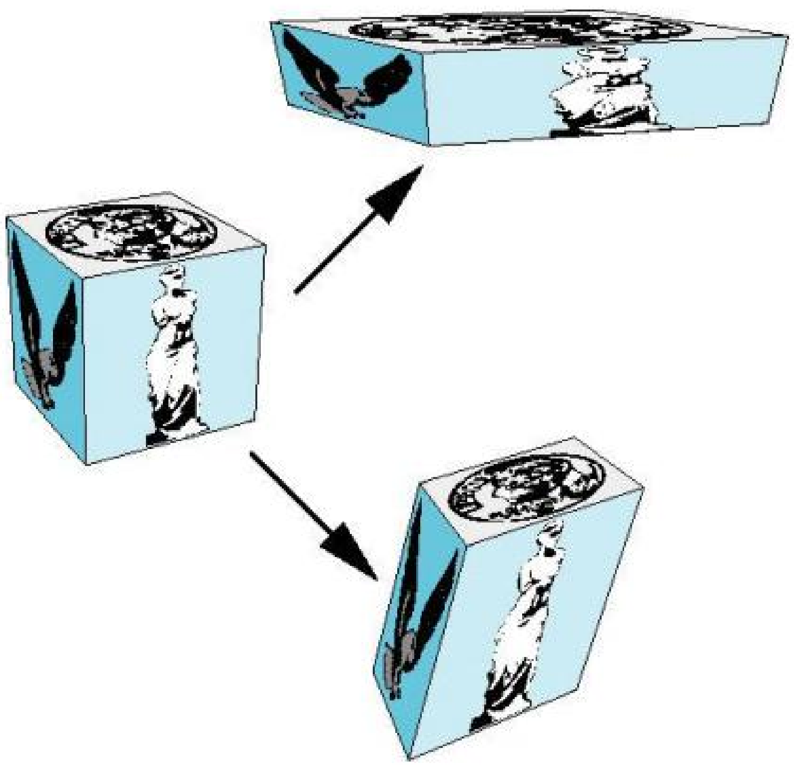
  - 对于绕y轴的旋转， y坐标不变

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

刚体变换（Rigid Transformation)

- 旋转与平移是两种刚体变换
  - 这两种变换的复合只能改变对象的位置与定向
  - 保角度和长度
- 其它的仿射变换会改变对象的形状和体积

非刚体变换



缩放（scaling）

- 沿每个坐标轴伸展或收缩(原点为不动点)

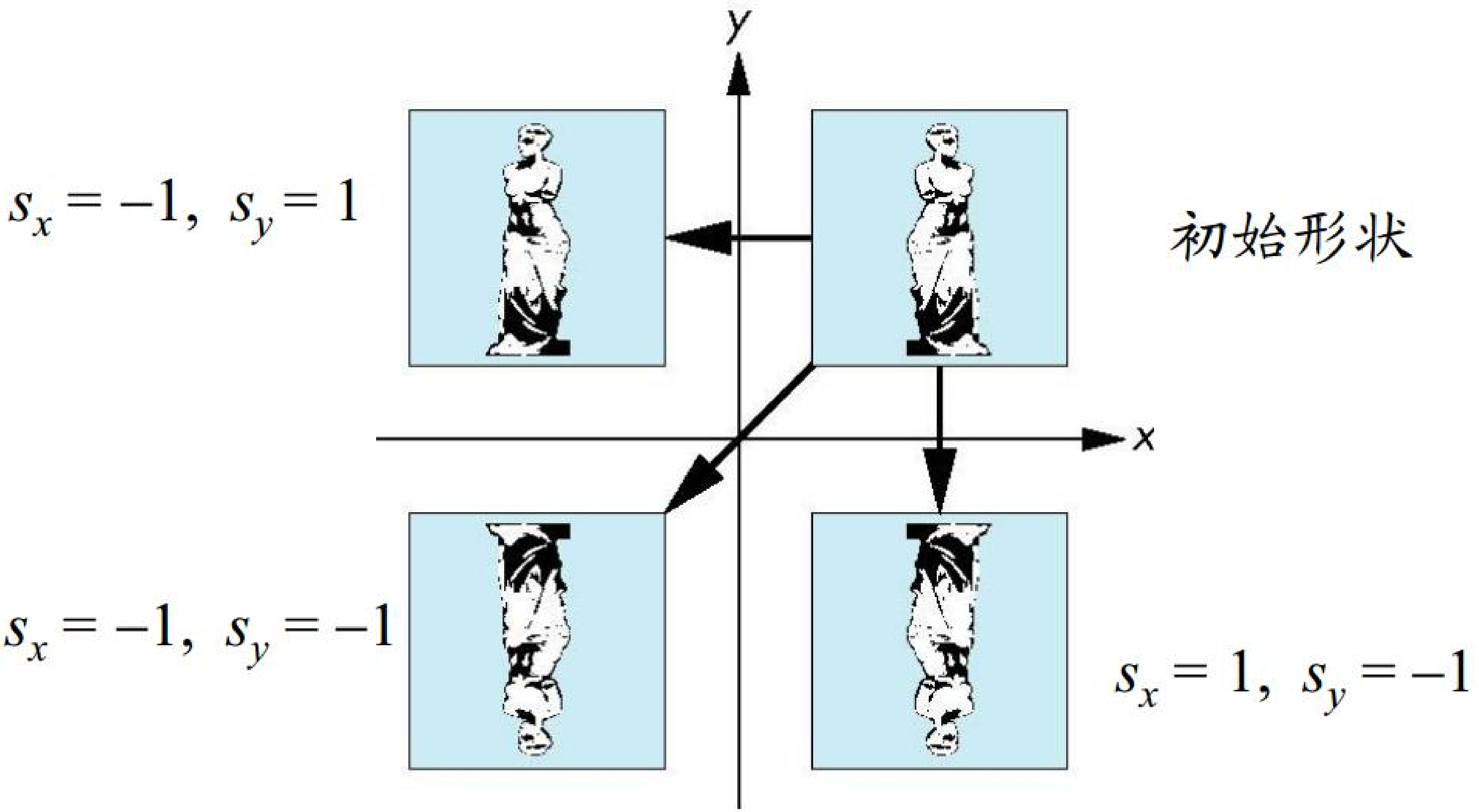
$$x' = s_x x$$
$$y' = s_y y$$
$$z' = s_z z$$

$$\mathbf{p}' = \mathbf{S}\mathbf{p}$$

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

反射（reflection)

- 特殊的缩放
  - 缩放系数为负数



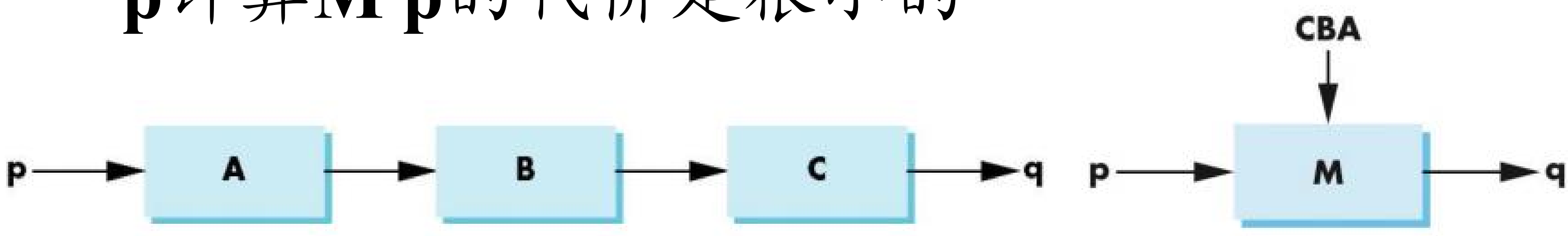
逆变换 (Inverse Transformation)

- 虽然可以直接计算矩阵的逆，但根据几何意义可以给出各种变换的逆
  - 平移: $\mathbf{T}^{-1}(d_x, d_y, d_z) = \mathbf{T}(-d_x, -d_y, -d_z)$
  - 旋转: $\mathbf{R}^{-1}(q) = \mathbf{R}(-q)$
    - 对所有旋转矩阵成立
    - 注意 $\cos(-q) = \cos(q)$, $\sin(-q) = -\sin(q)$

      $$\mathbf{R}^{-1}(q) = \mathbf{R}^{T}(q)$$
  - 缩放: $\mathbf{S}^{-1}(s_x, s_y, s_z) = \mathbf{S}(1/s_x, 1/s_y, 1/s_z)$

变换的复合 (Transformation Concatenation)

- 可以通过把旋转、平移与缩放矩阵相乘从而形成任意的仿射变换
- 由于对许多顶点应用同样的变换，因此构造矩阵**M = C B A**的代价相比于对许多顶点**p**计算**M p**的代价是很小的



- 难点在于如何根据应用程序的要求构造出满足要求的变换矩阵

变换的顺序(Order of Transformations)

- 注意在右边的矩阵是首先被应用的矩阵
- 从数学的角度来说，下述表示是等价的

$$p' = ABCp = A(B(Cp))$$

- 变换的顺序是不可交换的
  - 矩阵乘法不满足交换律

绕原点的一般旋转

- 绕过原点任一轴旋转θ角可以分解为绕x, y, z轴旋转的复合

$$\mathbf{R}(\theta) = \mathbf{R}_z(\theta_z)\ \mathbf{R}_y(\theta_y)\ \mathbf{R}_x(\theta_x)$$

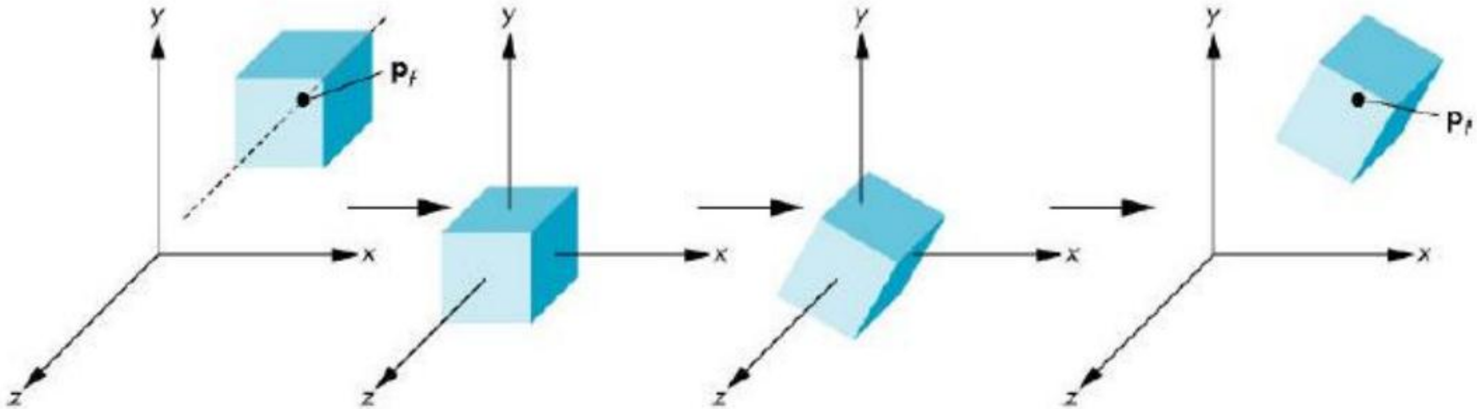
$\theta_x\ \theta_y\ \theta_z$ 被称为欧拉角（ Euler angles ）

- 注意：因为矩阵乘法不具有交换性，因此调换z,y,x的顺序将导致不同的旋转效果。调换顺序之后，如果为了得到原来的旋转效果，则旋转角度应相应改变。

不动点为p_f的旋转

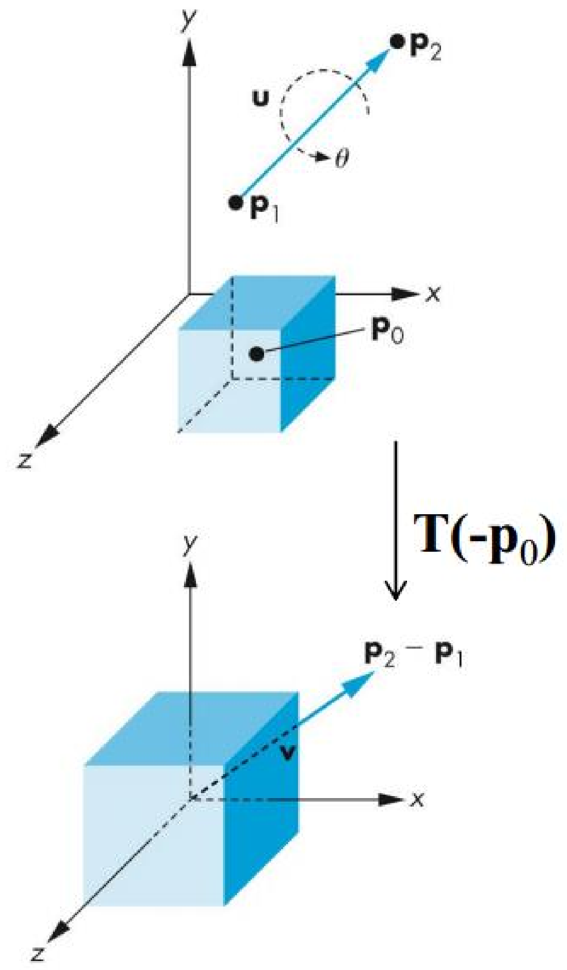
- 把不动点移到原点 $T(-p_f)$
- 旋转 $R(\theta)$
- 把不动点移回到原来位置 $T(p_f)$

$$M = T(p_f)\, R(\theta)\, T(-p_f)$$

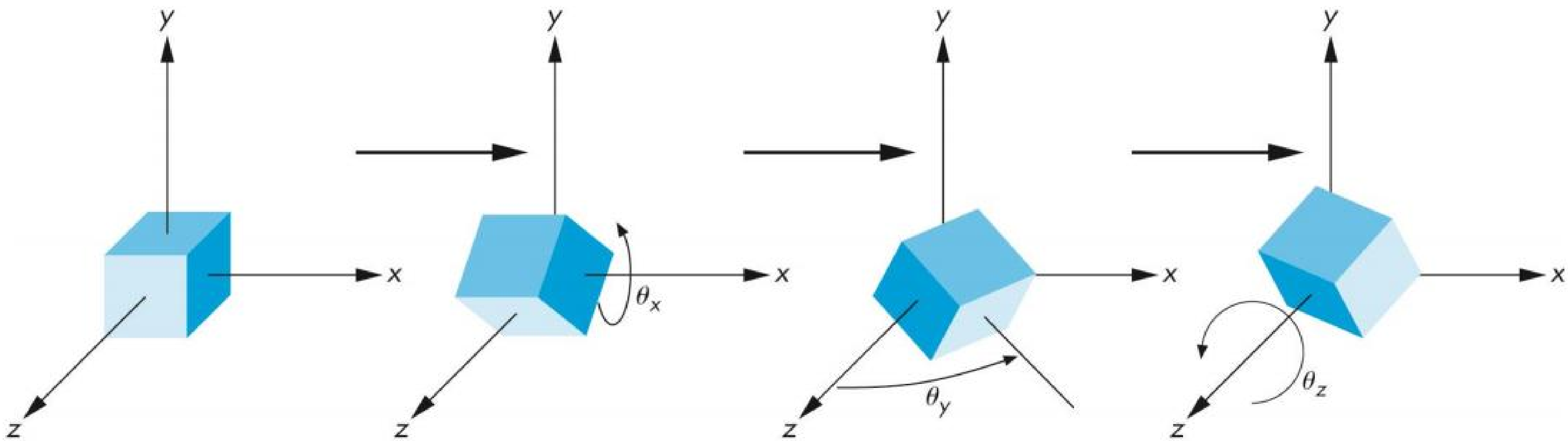


绕任意轴的旋转
(处理旋转中心)

- 不动点：立方体中心$\mathbf{p}_0$
- 旋转轴方向向量：
  $\mathbf{r} = \mathbf{p}2 - \mathbf{p}1$
- $\mathbf{T}(-\mathbf{p}_0)$平移不动点到原点



绕任意轴的旋转
(处理旋转)



- 策略：先经过两次旋转使旋转轴**r**与z轴对齐，然后绕z轴旋转角度θ

$$\mathbf{R} = \mathbf{R}_x(-\theta_x)\mathbf{R}_y(-\theta_y)\mathbf{R}_z(\theta)\ \mathbf{R}_y(\theta_y)\mathbf{R}_x(\theta_x)$$

绕任意轴旋转公式

- 绕任意轴

$$\boldsymbol{r} = \left(r_x, r_y, r_z\right)^T$$

$\boldsymbol{r}$为归一化的旋转轴

$$\mathbf{R} =$$

$$\begin{pmatrix} \cos\phi + (1-\cos\phi)r_x^2 & (1-\cos\phi)r_x r_y - r_z\sin\phi & (1-\cos\phi)r_x r_z + r_y\sin\phi \\ (1-\cos\phi)r_x r_y + r_z\sin\phi & \cos\phi + (1-\cos\phi)r_y^2 & (1-\cos\phi)r_y r_z - r_x\sin\phi \\ (1-\cos\phi)r_x r_z - r_y\sin\phi & (1-\cos\phi)r_y r_z + r_x\sin\phi & \cos\phi + (1-\cos\phi)r_z^2 \end{pmatrix}$$

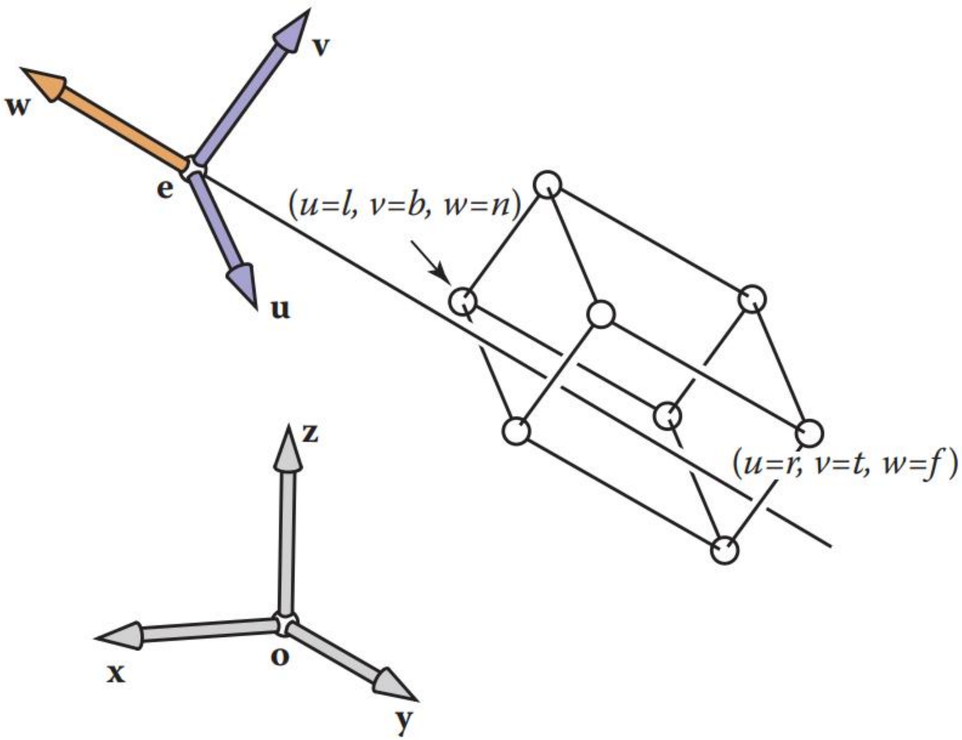
扩展到齐次坐标？

$\mathbf{M} = \mathbf{T}(\mathbf{p}_0)\,\mathbf{R}\mathbf{T}(-\mathbf{p}_0)$

模型视图变换

- 相机变换（视点变换）

- the eye position **e**,
- the gaze direction **g**,
- the view-up vector **t**.



模型视图变换

- 相机变换（视点变换）

  - the eye position $\mathbf{e}$,
  - the gaze direction $\mathbf{g}$,
  - the view-up vector $\mathbf{t}$.

$$\mathbf{w} = -\frac{\mathbf{g}}{\|\mathbf{g}\|},$$

$$\mathbf{u} = \frac{\mathbf{t} \times \mathbf{w}}{\|\mathbf{t} \times \mathbf{w}\|},$$

$$\mathbf{v} = \mathbf{w} \times \mathbf{u}.$$

$$\mathbf{M}_{\text{cam}} = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{w} & \mathbf{e} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} x_u & y_u & z_u & 0 \\ x_v & y_v & z_v & 0 \\ x_w & y_w & z_w & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix}$$