几何变换

计算机图形学——原理、方法及应用（第**4**版），潘云鹤、童若锋、耿卫东、唐敏、童欣，高等教育出版社，**2022**。

内容

- 变换
- 齐次坐标
- 变换的矩阵表达

变换

- 几何变换：计算物体新的位置
- 坐标变换：已知两个坐标系及物体在某一坐标系中的坐标，计算其在另一坐标系中的坐标。

由于物体最终由点表示，我们只需研究点的变换.

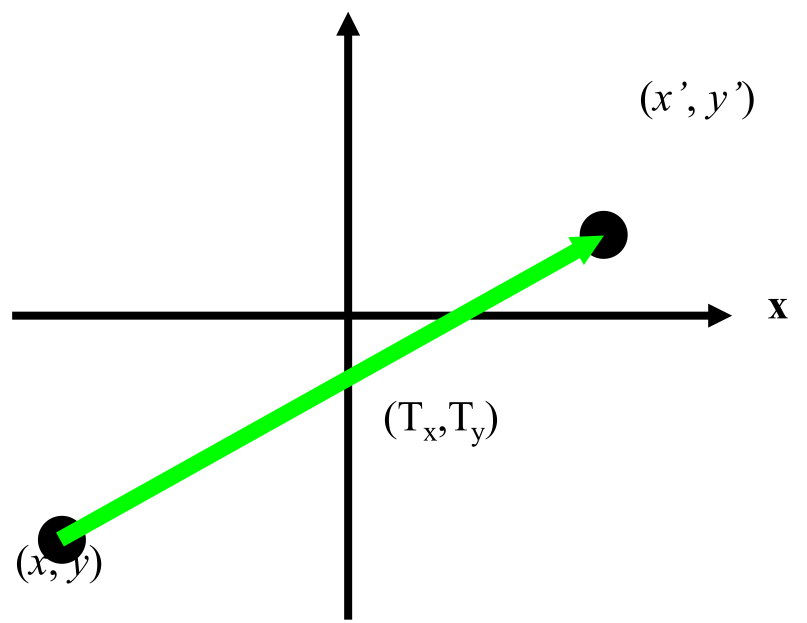
图形的几何变换

一、二维基本变换

1. 平移(Translation)

将图形对象从一个位置(x, y)移到另一个位置(x′，y′)的变换。

$$x' = x + T_x$$
$$y' = y + T_y$$



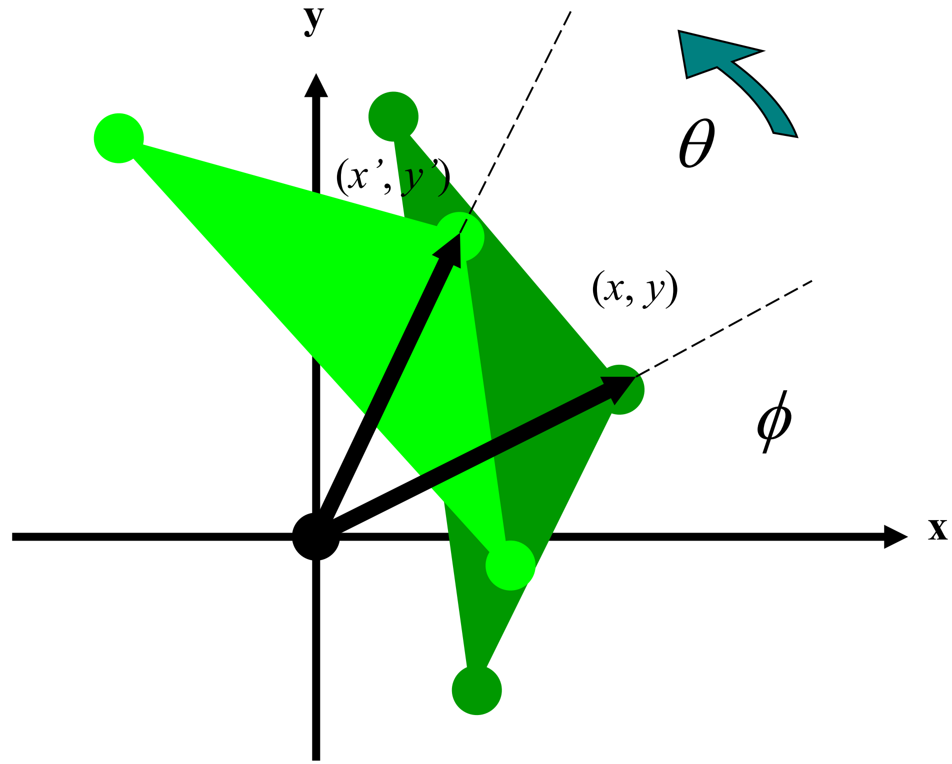
图形的几何变换

一、二维基本变换

2. 旋转(Rotation)

点 (x, y) 围绕原点逆时针转动一个角度 θ,

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



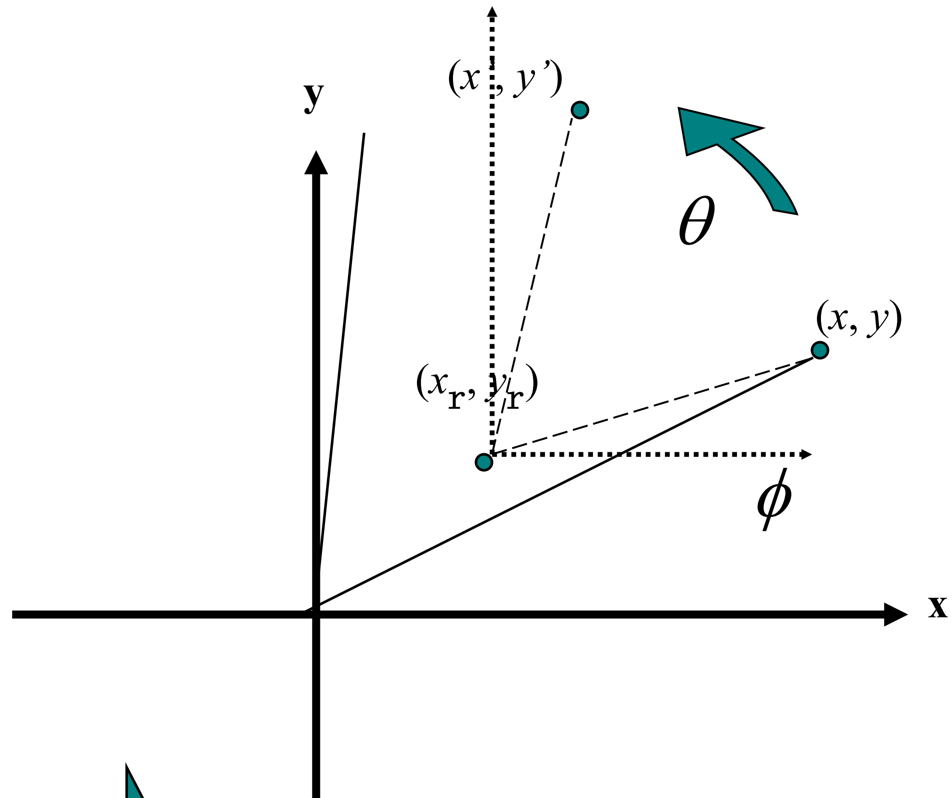
图形的几何变换

一、二维基本变换

2. 旋转(Rotation)

将以某个参考点(x_r, y_r)为圆心，将对象上的各点(x, y)围绕圆心转动一个逆时针角度θ .

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



$$newx = x - x_r$$
$$newy = y - y_r$$

$$newx' = newx\cos\theta - newy\sin\theta$$
$$newy' = newy\cos\theta + newx\sin\theta$$

$$x' = newx' + x_r$$
$$y' = newy' + y_r$$

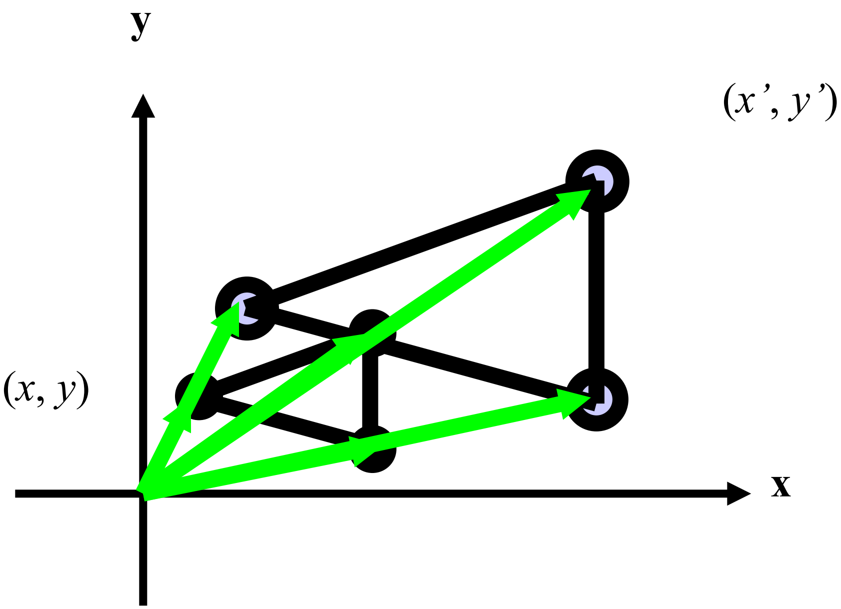
$$x' = x_r + (x - x_r)\cos\theta - (y - y_r)\sin\theta$$
$$y' = y_r + (y - y_r)\cos\theta + (x - x_r)\sin\theta$$

图形的几何变换

一、二维基本变换

3. 变比(Scaling)

使对象按比例因子(S_x, S_y)放大或缩小的变换



$$x' = x \cdot S_x$$
$$y' = y \cdot S_y \qquad (3.1)$$

固定点变比(scaling relative to a fixed point)。以a为固定点

1((1)作平移$T_x=-x_a$，$T_y=-y_a$；

2((2)按式(3.1)作变比；

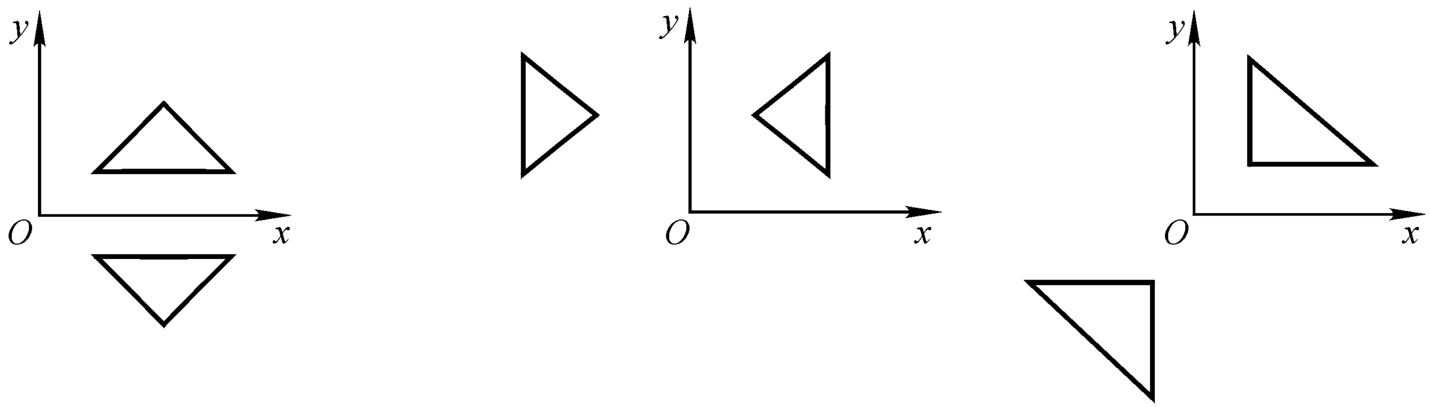
3((3)作1)的逆变换，即作平移$T_x=x_a$，$T_y=y_a$。

图形的几何变换

一、二维基本变换

3. 变比(Scaling)

当比例因子S_x或S_y小于0时，对象不仅变化大小，而且分别按x轴或y轴被反射



线性变换一般形式

$$
\begin{aligned}
x &\to a*x + b*y + c*z \\
y &\to d*x + e*y + f*z \\
z &\to g*x + h*y + i*z
\end{aligned}
\quad or \quad
\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ z \end{bmatrix}
$$

- 下述哪些变换能表示为这一形式?

  平移　　变比　　旋转

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

$\longrightarrow$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$x' = x\ \mathrm{Sx}$$
$$y' = y\ \mathrm{Sy}$$

$\longrightarrow$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$x' = x + T_x$$
$$y' = y + T_y$$

$\longrightarrow$

???

齐次坐标

$$[x, y] \rightarrow [x^*, y^*, a]$$

$$[x, y] \rightarrow [x, y, 1]$$

$$x = \frac{x^*}{a}, y = \frac{y^*}{a}$$

- Cartesian co-ordinates中的点(x,y,z) 的齐次坐标

  $$(x * w, y * w, z * w, w), w \neq 0$$

- 反过来，齐次(x,y,z,w) 在笛卡尔坐标系中的坐标为 $\left(x/w, y/w, z/w\right)$ $w \neq 0$

问题：w = 0 表示什么?

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

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$x' = x\,Sx$$
$$y' = y\,Sy$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$x' = x + T_x$$
$$y' = y + T_y$$

???

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

变换的矩阵表示

- 点P (x,y,z)用齐次坐标写为列向量 P_h
- 一个变换则可表示为4x4 的矩阵 M
- 施加一个变换则表示为矩阵乘法

$$Q_h = M * P_h$$

Matrix Representations and Homogeneous Co-ordinates

- Each of the transformations defined above can be represented by a 4x4 matrix
- Composition of transformations is represented by product of matrices
- So composition of transformations is also represented by 4x4 matrix

三维几何变换

- Translation

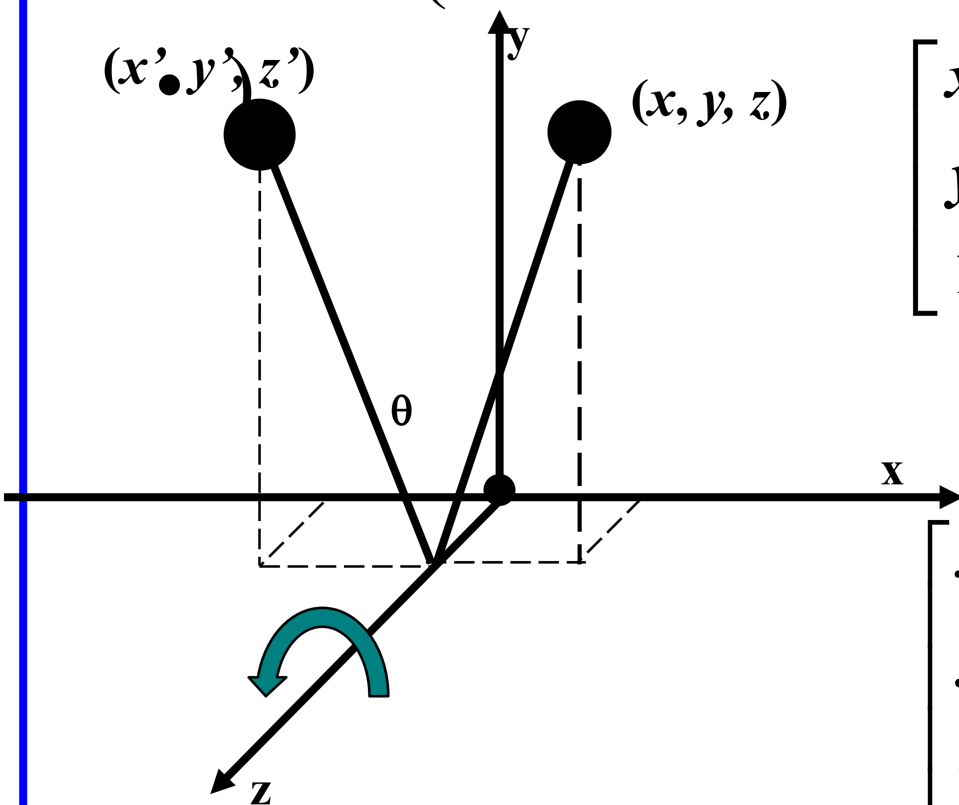
$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \boldsymbol{T_x} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \boldsymbol{T_y} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \boldsymbol{T_z} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \bullet \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \\ \mathbf{1} \end{bmatrix}$$

- Scaling

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

三维几何变换

- 旋转 (绕 Z axis 右手系)

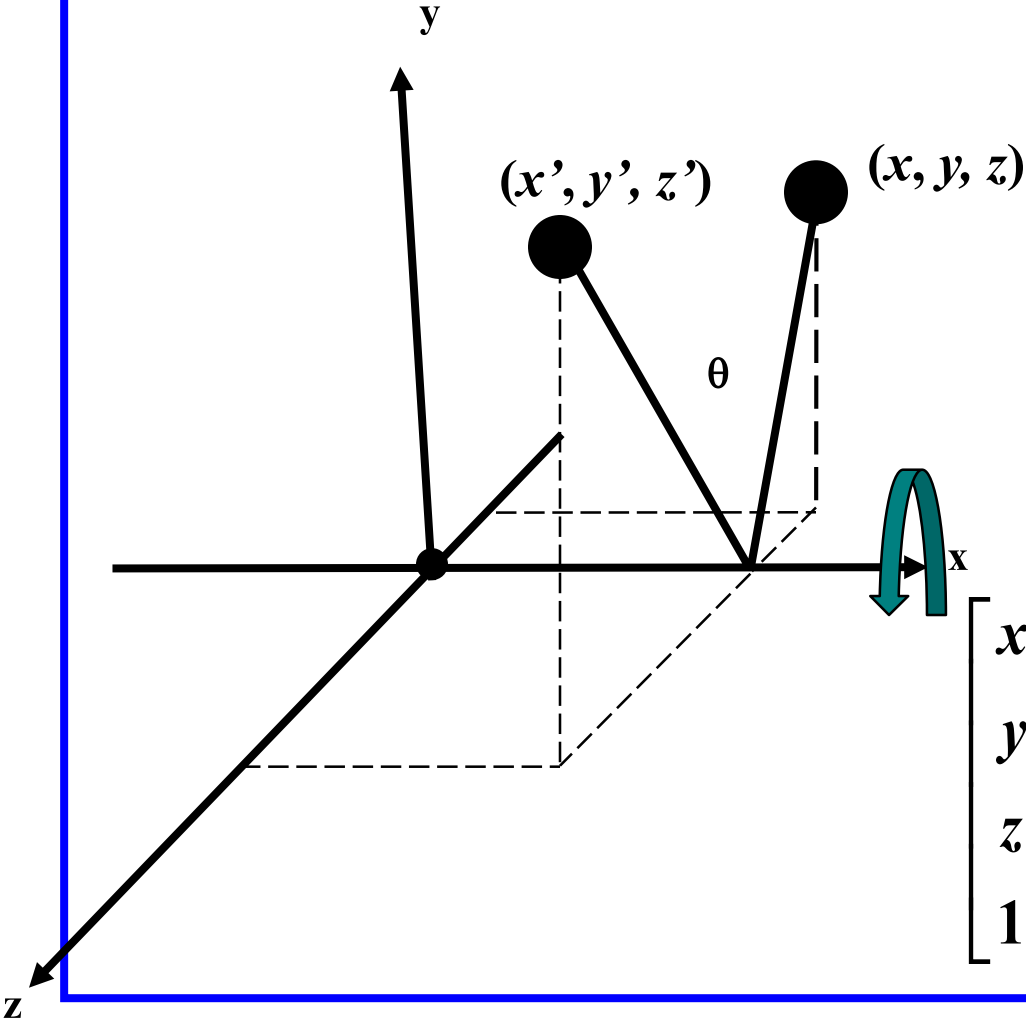


$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

三维几何变换

- 旋转 (x axis 右手系)

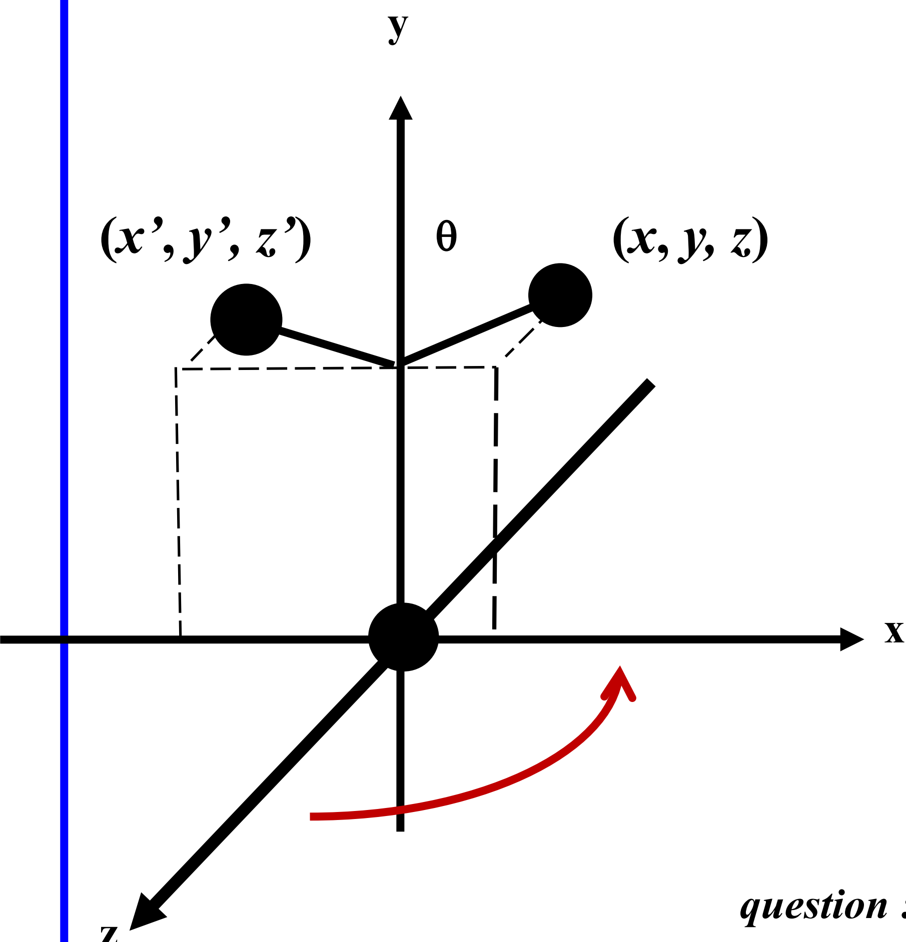


$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

三维几何变换

- 旋转 (绕 Y axis 右手系)



$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

question : why?

复合变换

对点p先施加变换A再施加变换B, 得到点q:
根据矩阵乘法的结合律，相当于施加变换BA

$$q = B\,(A\,p) = (B\,A)\,p$$

施加一系列变换则可表达如下：

$$Q_h = M_n * \dots * M_2 * M_1 * P_h$$
$$= M * P_h$$

- 问题：为什么费大功夫引入齐次坐标进行矩阵表达?

以点P为参考旋转的矩阵 (二维)

旋转中心 $P = \begin{bmatrix} T_x \\ T_y \\ 1 \end{bmatrix}$

变换到原点

$$M_1 = \begin{bmatrix} 1 & 0 & -T_x \\ 0 & 1 & -T_y \\ 0 & 0 & 1 \end{bmatrix}$$

旋转

$$M_2 = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

变换回去

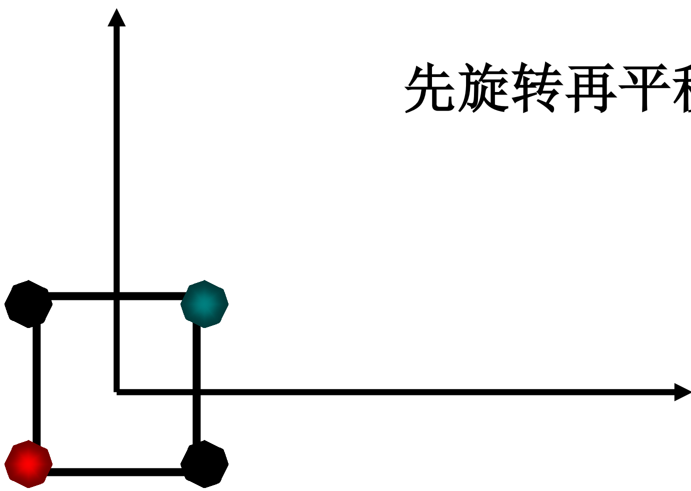
$$M_3 = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix}$$

整个变换. $B := M_4 A$

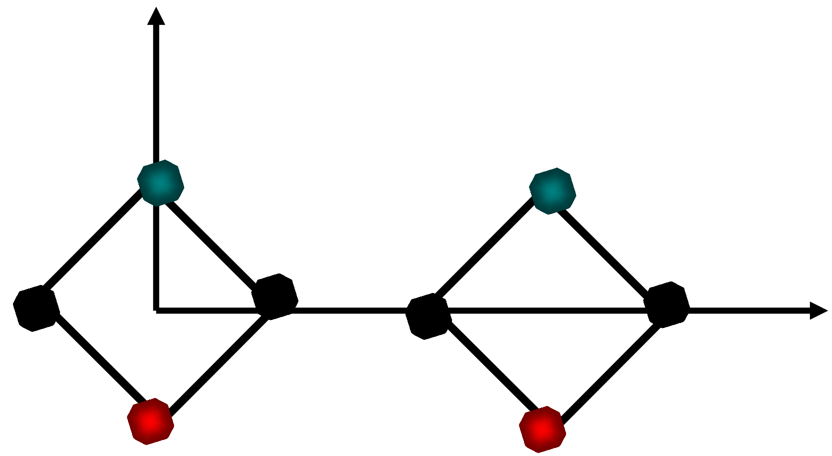
$$M_4 = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -T_x \\ 0 & 1 & -T_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$M_4 = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & -\cos(\theta)\, T_x + \sin(\theta)\, T_y + T_x \\ \sin(\theta) & \cos(\theta) & -\sin(\theta)\, T_x - \cos(\theta)\, T_y + T_y \\ 0 & 0 & 1 \end{bmatrix}$$

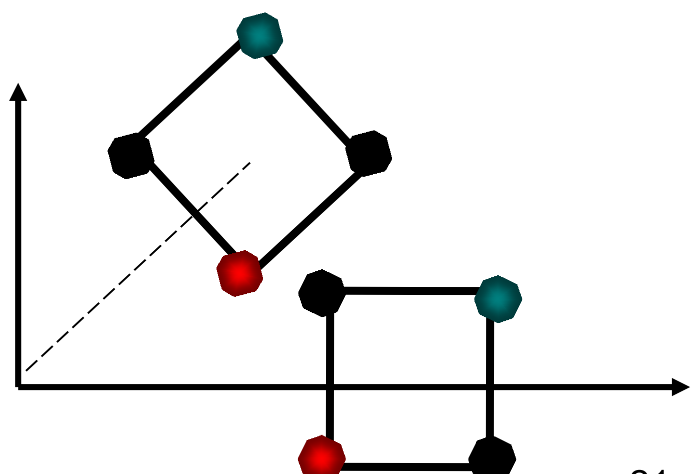
矩阵乘法不可交换



先旋转再平移



先平移再旋转



绕任意轴旋转

- 已知:

  Axis: (x_1,y_1,z_1) to (x_2,y_2,z_2)

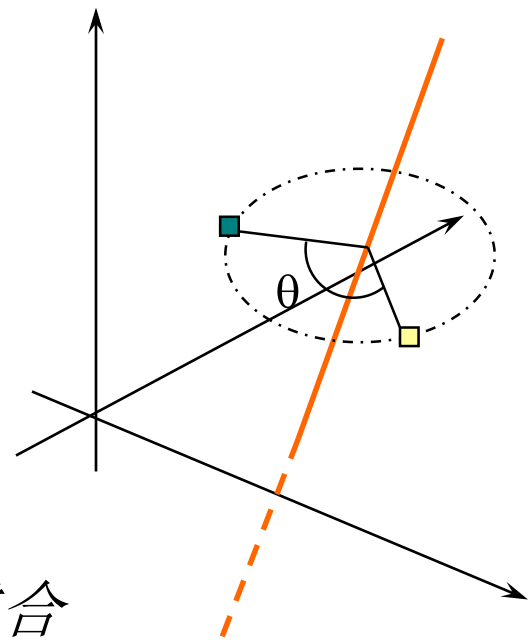
  旋转角度: θ

- 过程

  step1. 施加变换使得旋转轴与Z axis重合

  step2. 调用旋转θ

  step3. 施加step 1的逆变换

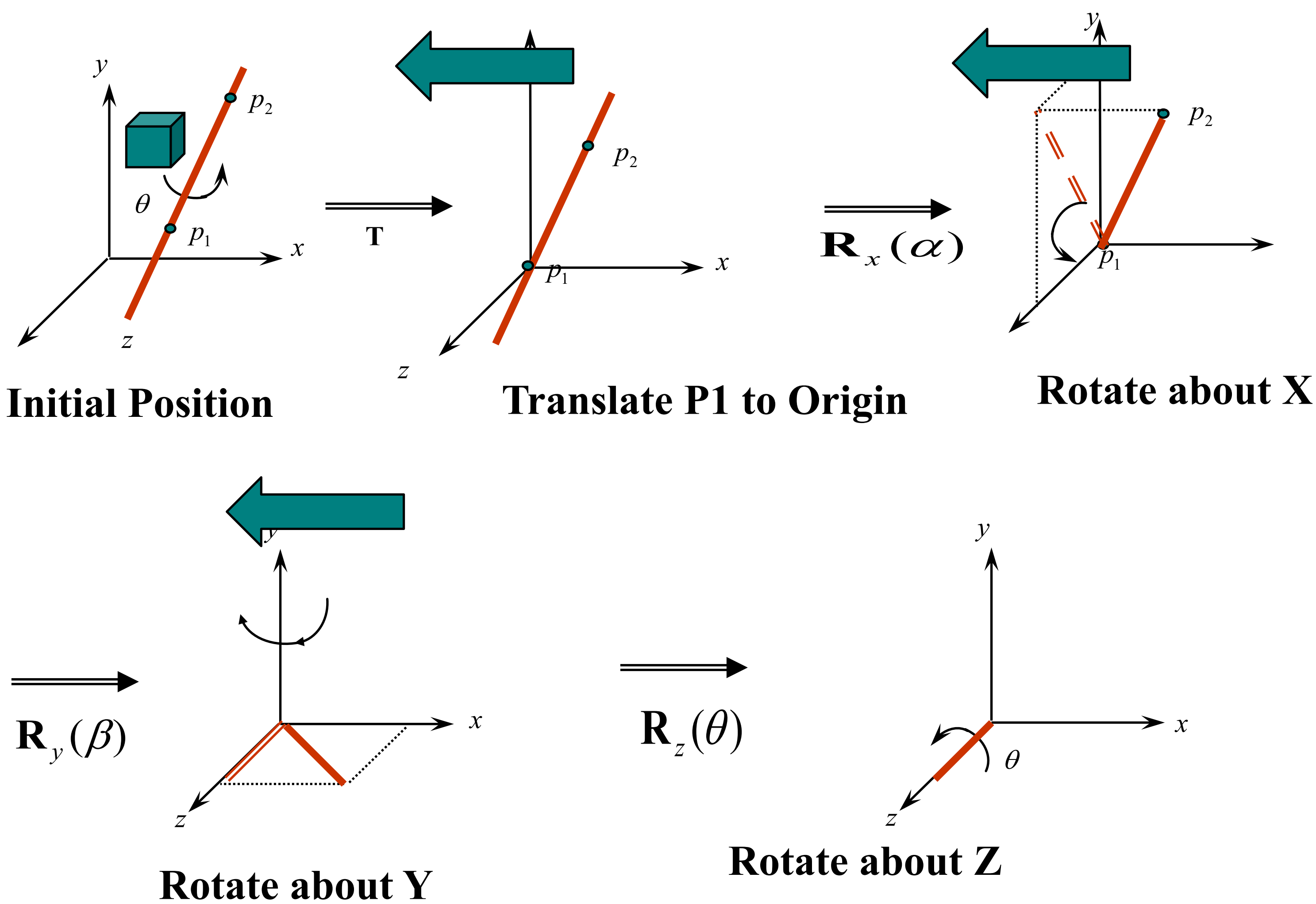


绕任意轴旋转

- Steps

$T_{-(x_1,y_1,z_1)}$	平移旋转轴使其过远点
$R_{(x,\alpha)}$	绕X轴旋转使其落于ZX 平面
$R_{(y,\beta)}$	绕Y轴旋转使其与Z 轴重合
$R_{(z,\theta)}$	绕Z轴转θ角
$R_{(y,-\beta)}$	$R_{(y,\beta)}$ 的逆变换
$R_{(x,-\alpha)}$	$R_{(x,\alpha)}$ 的逆变换
$T_{(x_1,y_1,z_1)}$	$T_{-(x_1,y_1,z_1)}$ 的逆变换

绕任意轴旋转



y
p_2
θ
p_1
x
z
Initial Position
T
p_2
p_1
x
z
Translate P1 to Origin
$\mathbf{R}_x(\alpha)$
y
p_2
p_1
Rotate about X
$\mathbf{R}_y(\beta)$
y
x
z
Rotate about Y
$\mathbf{R}_z(\theta)$
y
x
θ
z
Rotate about Z

OpenGL中的变换

OpenGL中开了三个堆栈用于存放三种变换

- **Model-view matrix**
- **Projection matrix**
- **Texture matrix**

glPushMatrix (void);
glPopMatrix (void);

当前变换矩阵

- 当前变换矩阵 (CTM) 是作用于顶点进行变换的矩阵.
- CTM 用4 x 4 矩阵表示，放于堆栈的顶部.

改变 CTM的函数

- Specify CTM mode : **glMatrixMode (*mode*);**

  mode* = (*GL_MODELVIEW | GL_PROJECTION | GL_TEXTURE)
- Load CTM : **glLoadIdentity (void); glLoadMatrix{fd} (*m);**

  m = 1D array of 16 elements arranged by the columns
- Multiply CTM : **glMultMatrix{fd} (*m);**
- Modify CTM : (multiplies CTM with appropriate transformation matrix)

  glTranslate {fd} (*x*, *y*, *z*);

  glScale {fd} (*x*, *y*, *z*);

  glRotate {fd} (*angle*, *x*, *y*, *z*);

  rotate counterclockwise around ray (0,0,0) to (x, y, z)

旋转实例

任务:

绕从 (4.0, 5.0, 6.0) 到 (5.0, 7.0, 9.0)的轴旋转45度

$(T_{-p1}, R_{45}, T_{+p1})$

```
glMatrixMode (GL_MODEVIEW);
glLoadIdentity ();
glTranslatef (4.0, 5.0, 6.0);
glRotatef (45.0, 1.0, 2.0, 3.0);
glTranslatef (-4.0, -5.0, -6.0);
```

- 注意：执行变换的次序是从后往前

矩阵堆栈

- OpenGL uses matrix stacks mechanism to manage modelling transformation hierarchy.

  glPushMatrix (void);

  glPopMatrix (void);

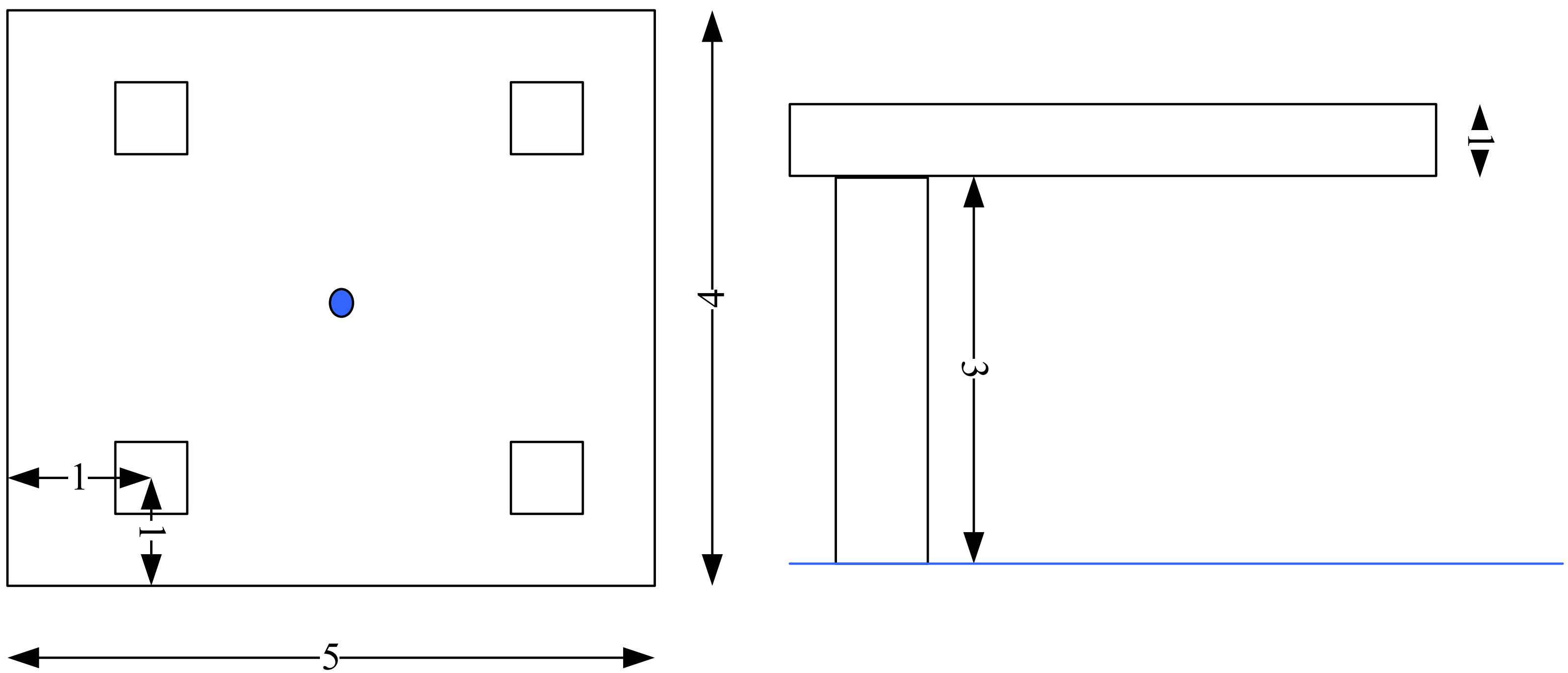
- OpenGL provides matrix stacks for each type of supported matrix to store matrices.

  Model-view matrix stack

  Projection matrix stack

  Texture matrix stack

实验 – 构建四条腿的桌子



glutSolidCube() …

Thank You