Computer Graphics 2019

6. Geometric Transformations

Hongxin Zhang
State Key Lab of CAD&CG, Zhejiang University

2019-10-23

Contents

- Transformations
- Homogeneous Co-ordinates
- Matrix Representations of Transformations

Transformations

- Procedures to compute new positions of objects
- Used to modify objects or to transform (map) from one co-ordinate system to another co-ordinate system

As all objects are eventually represented using points, it is enough to know how to transform points.

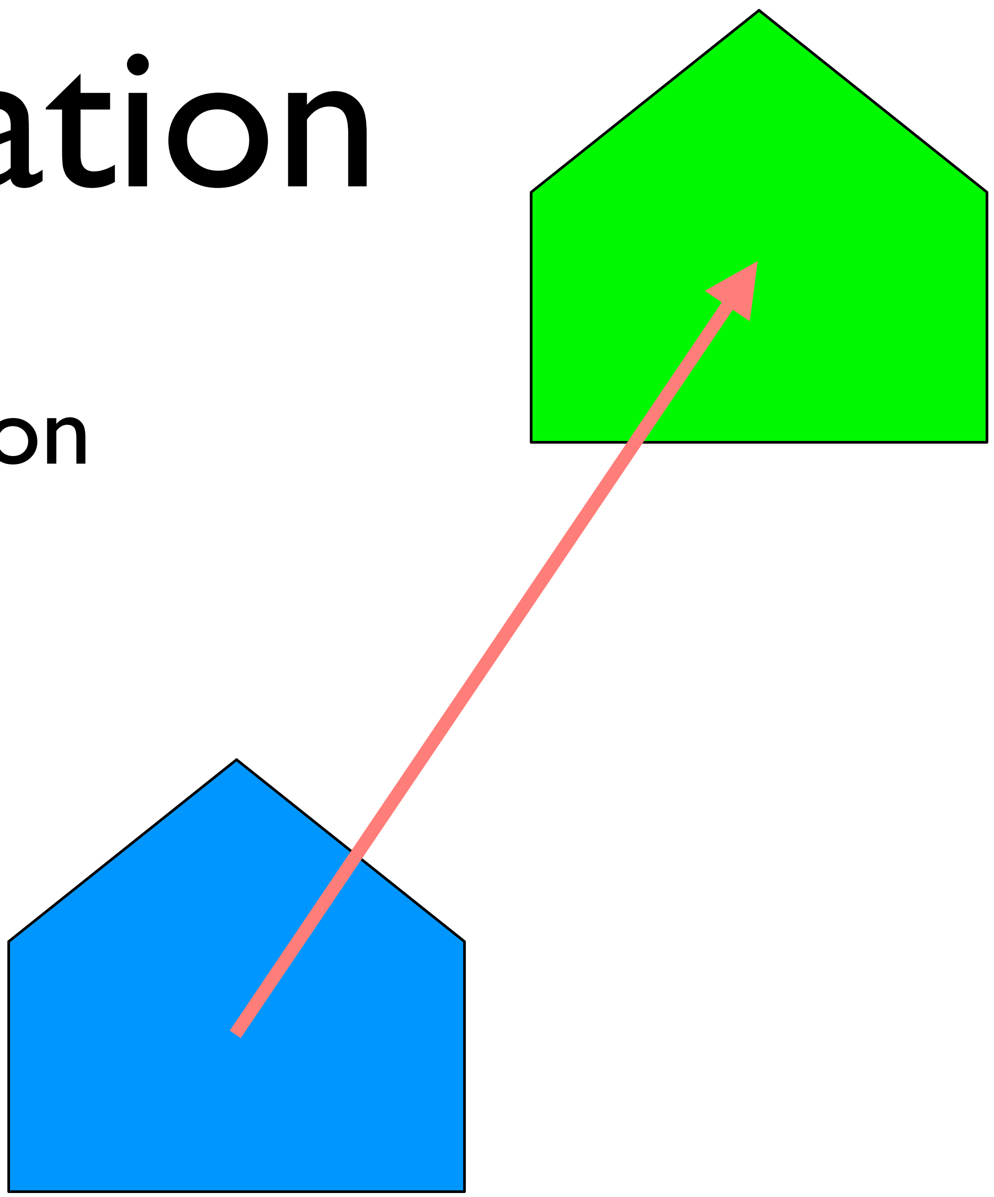
Translation

- Is a Rigid Body Transformation

$$x \Rightarrow x + T_x$$

$$y \Rightarrow y + T_y$$

$$z \Rightarrow z + T_z$$



- Translation vector (T_x, T_y, T_z) or shift vector

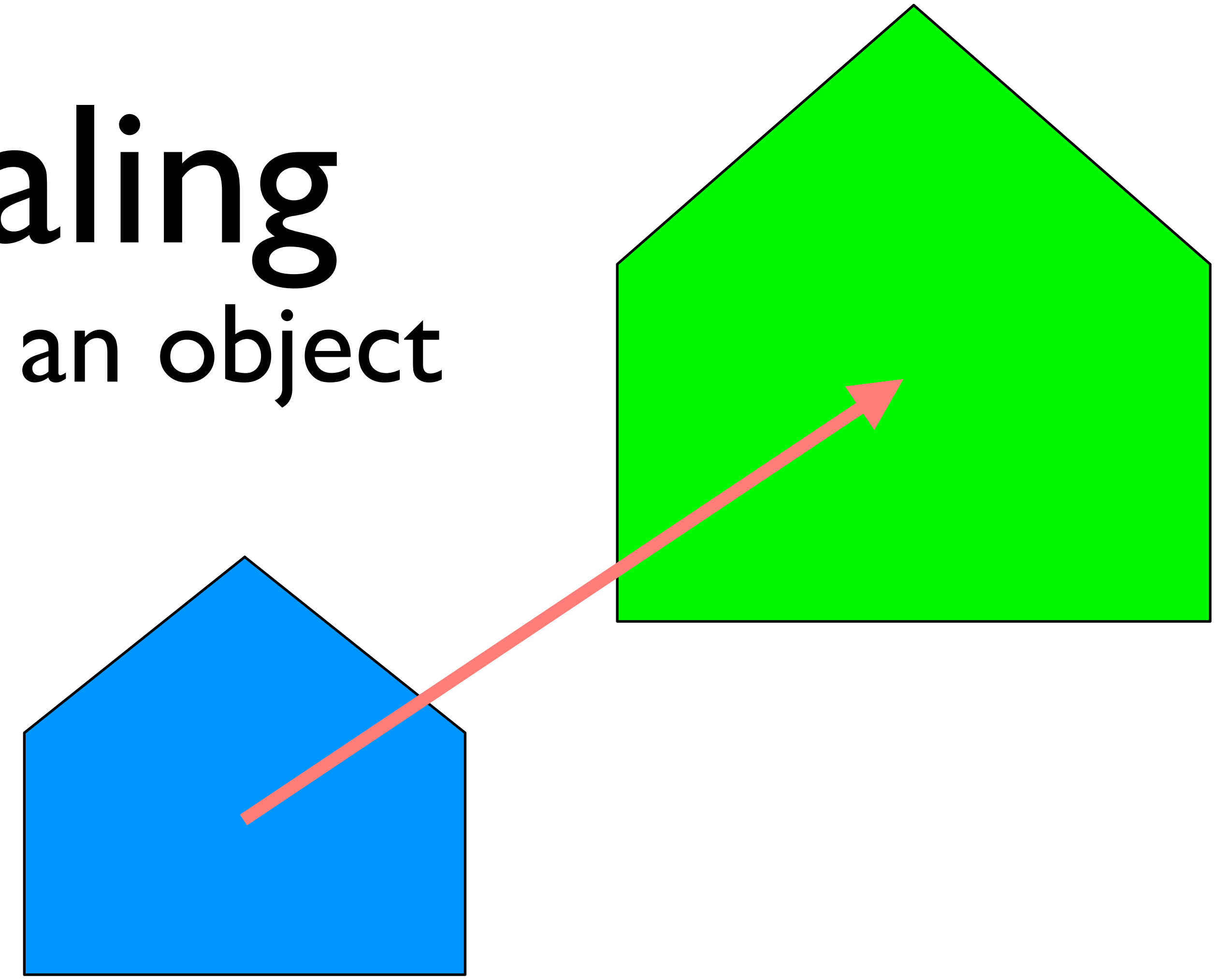
Scaling

- Changing the size of an object

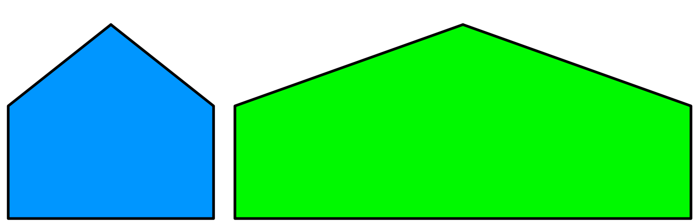
$$x \Rightarrow x * S_x$$

$$y \Rightarrow y * S_y$$

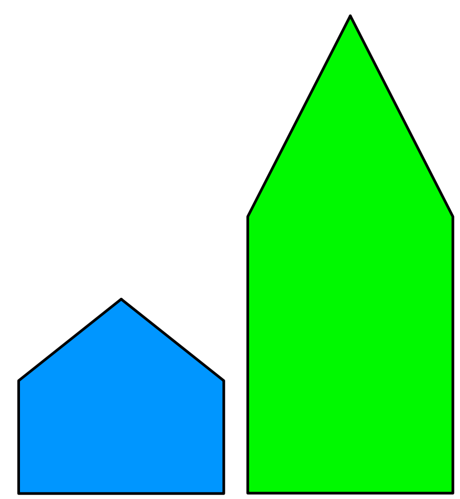
$$z \Rightarrow z * S_z$$



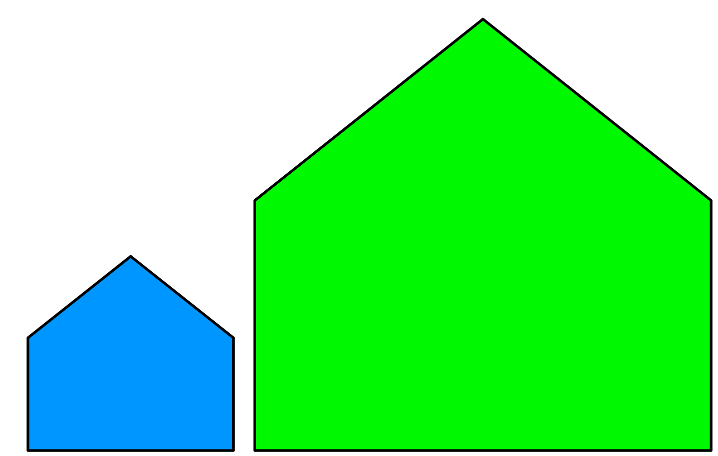
- Scale factor (S_x, S_y, S_z)



$S_y = 1$



$S_x = 1$



$S_x = S_y$

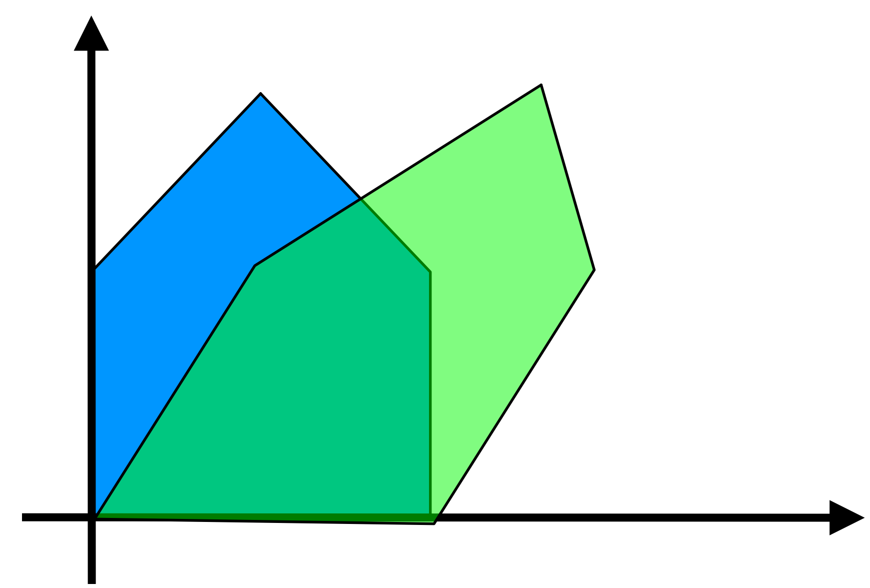
Shearing

- Produces shape distortions
- Shearing in x-direction

$x \Rightarrow x + a^* y$

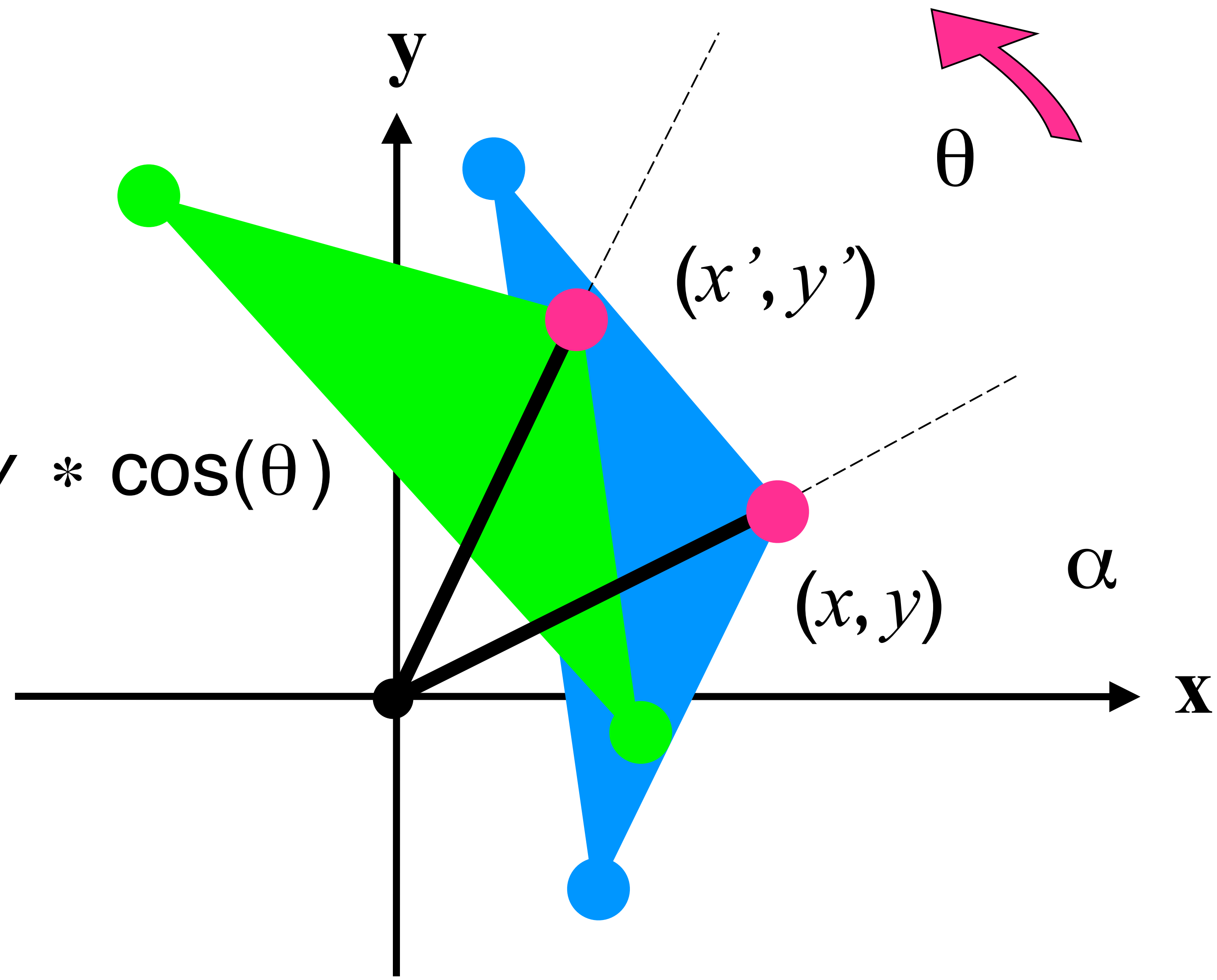
$y \Rightarrow y$

$z \Rightarrow z$

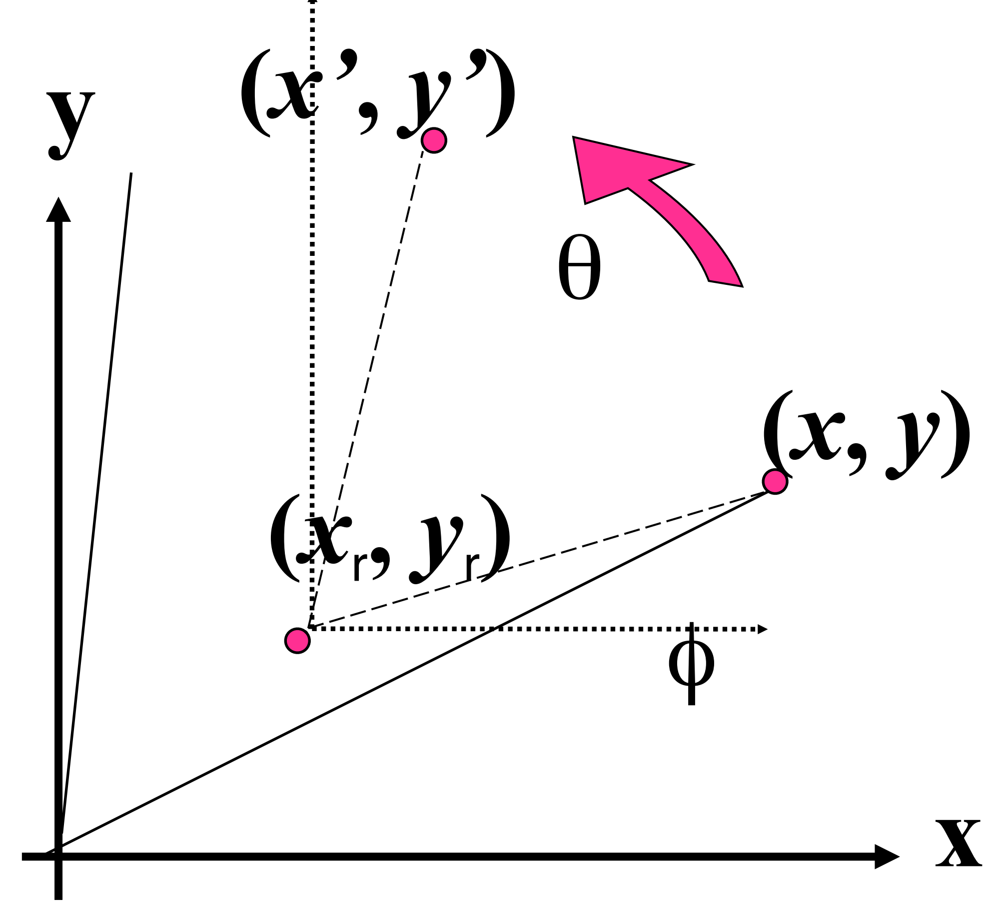


Rotation

$$y = x * \sin(\theta) + y * \cos(\theta)$$



Rotate around (x_r, y_r)



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

$$newx' = \mathbf{new}x\ \mathbf{cos}\theta - \mathbf{new}y\ \mathbf{sin}\theta$$
$$newy' = \mathbf{new}y\ \mathbf{cos}\theta + \mathbf{new}x\ \mathbf{sin}\theta$$

$$x' = newx' + x_r$$

$$y' = newy' + y_r$$

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

General Linear Transformation

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

- Which of the following can be represented in this form?
    - Translation
    - Scaling
    - Rotation

General Linear Transformation

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

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

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

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

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

$\longrightarrow$???

Homogeneous Co-ordinates

$$(x, y) \rightarrow (x, y, a)$$

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

$$(x, y) \rightarrow (x, y, 1)$$

- Any point (x, y, z) in Cartesian co-ordinates is written as

$$(xw, yw, zw, w), w \neq 0$$

in Homogeneous Co-ordinates

- The point (x, y, z, w) represents in Cartesian co-ordinates

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

What happens when w=0 ?

the point represented is a point at infinity

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

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

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

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

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

???

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

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

Matrix Notations for Transformations

- Point P (x,y,z) is written as the column vector P_h
- A transformation is represented by a 4x4 matrix M
- The transformation is performed by matrix multiplication

$$Q_h = M * P_h$$

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

Matrix Representations of Various Transformations

- Translation

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

- Scaling

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

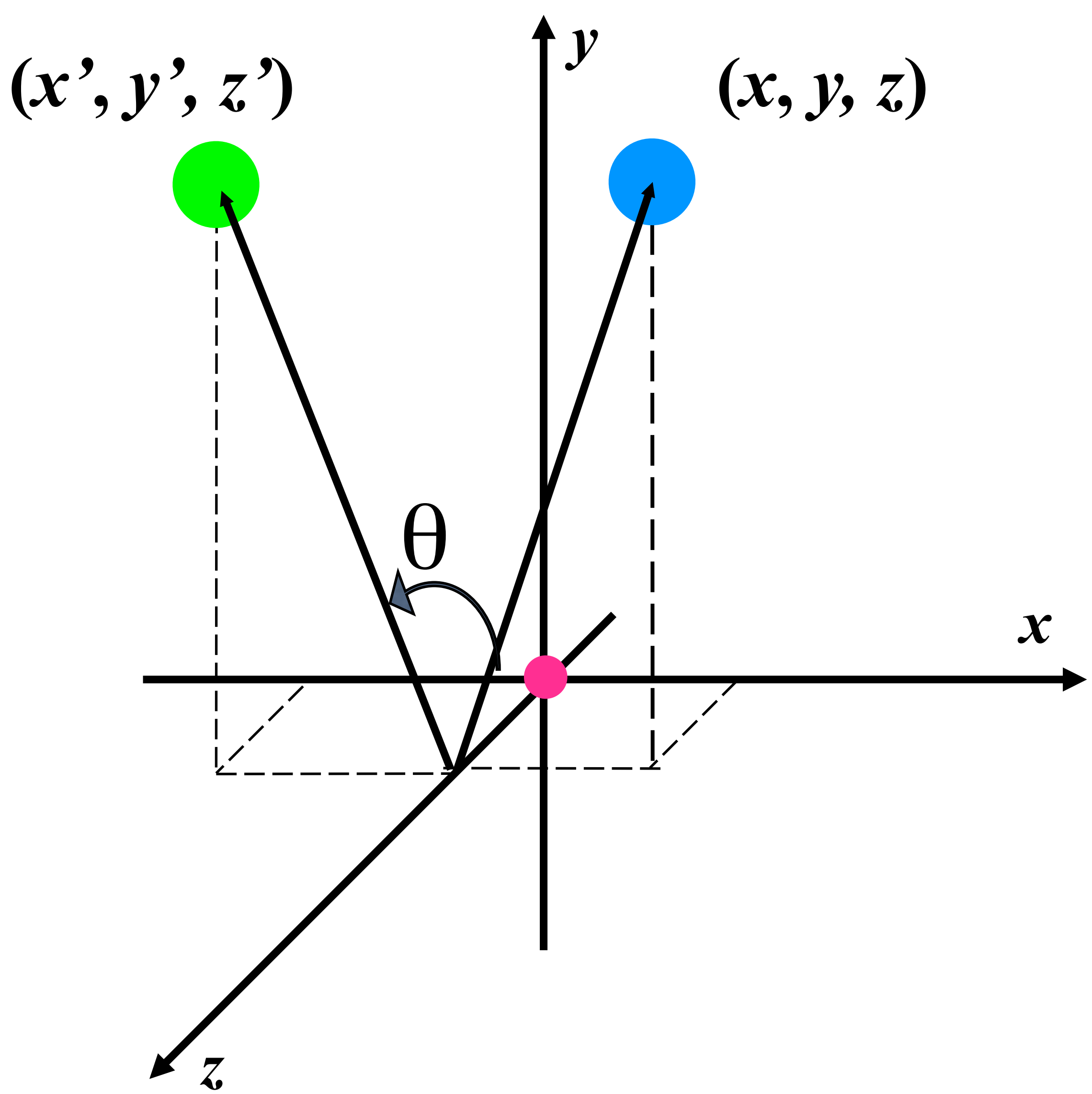
Matrix Representations of Various Transformations (contd.)

- Shearing (in X direction)

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & a & b & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Matrix Representations of Various Transformations (contd.)

Rotation (around Z axis)

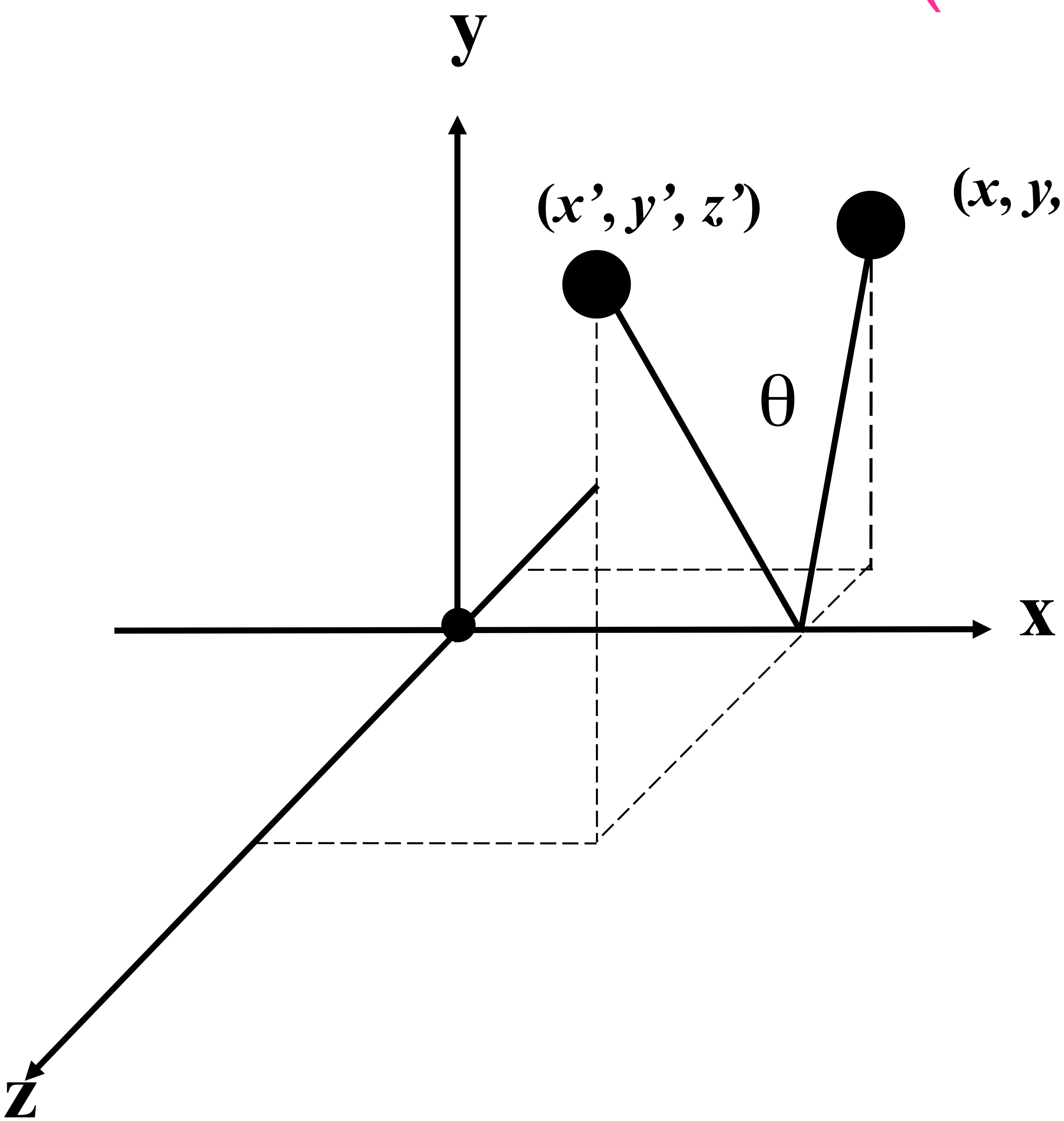


$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

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

Matrix Representations of Various Transformations (contd.)

Rotation (around X axis)

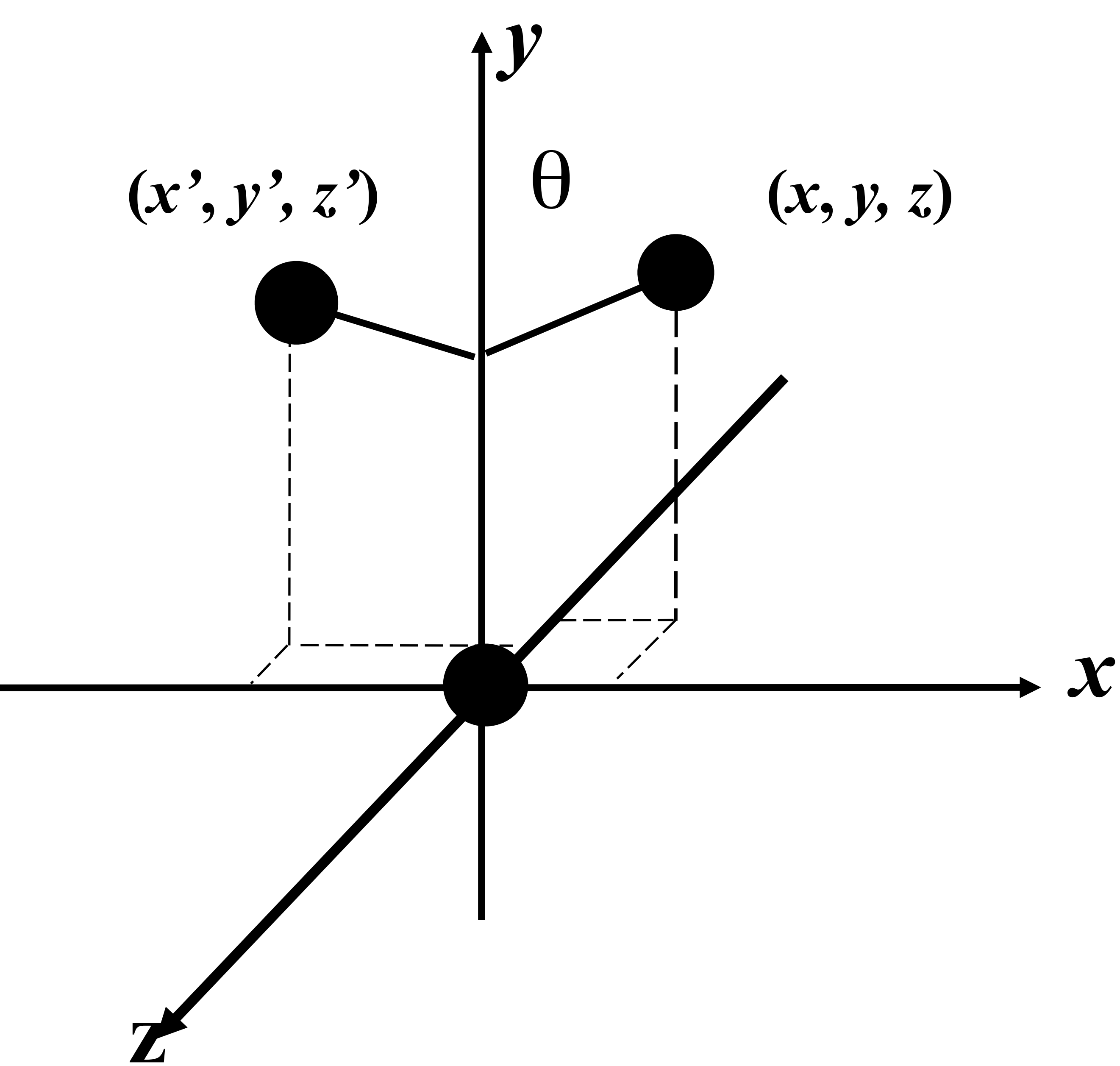


$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

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

Matrix Representations of Various Transformations (contd.)

- Rotation (around Y axis)



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

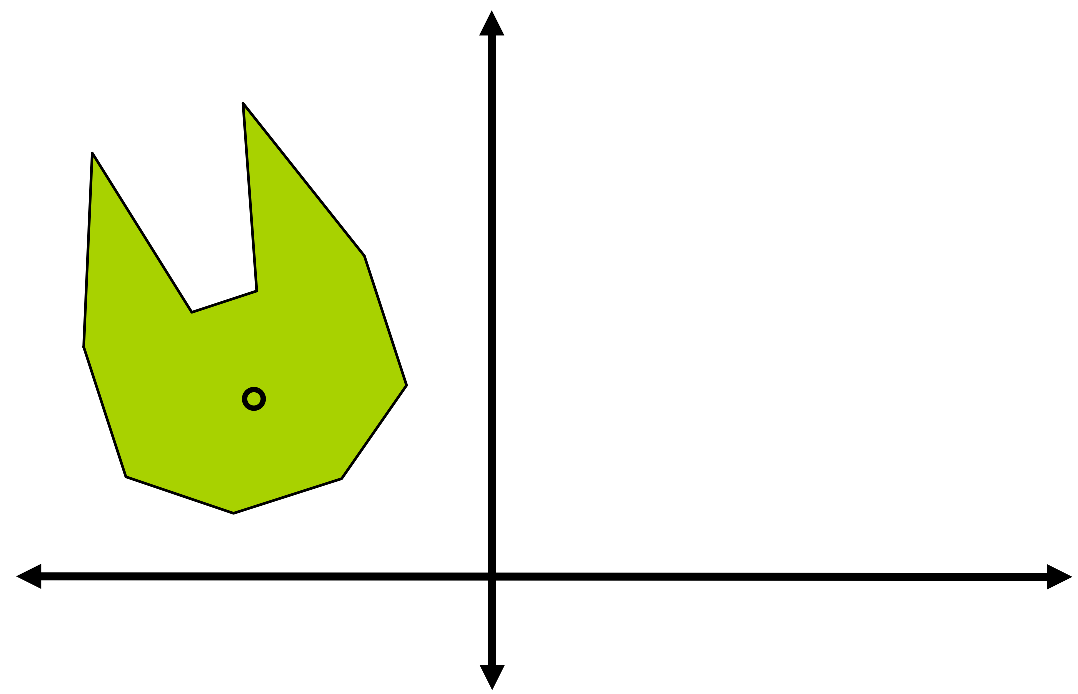
$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

question : why?

Properties of Transformations

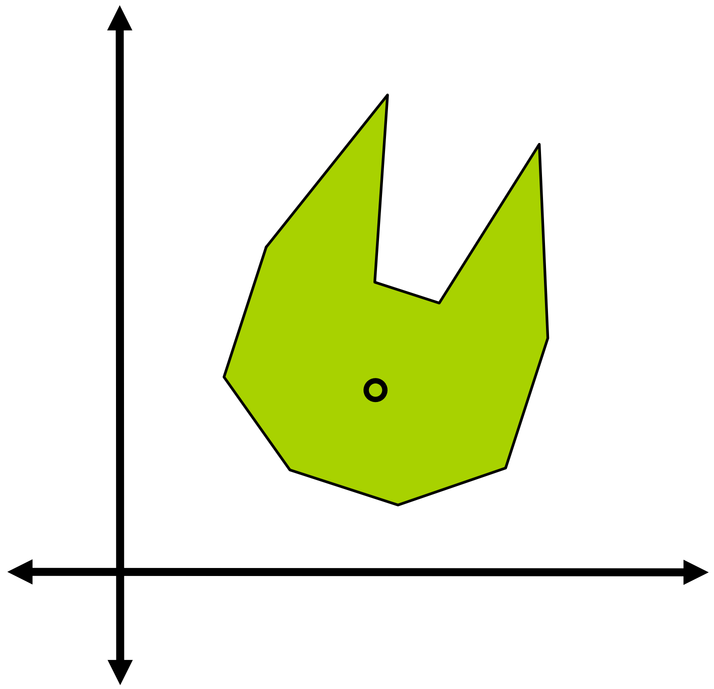
Type Preserves	Rigid Body: Rotation & translation	Linear General 3x3 matrix	Affine Linear + translation	Projective 4x4 matrix with last row $\neq(0,0,0,1)$
Lengths	Yes	No	No	No
Angles	Yes	No	No	No
Parallelness	Yes	Yes	Yes	No
Straight lines	Yes	Yes	Yes	Yes

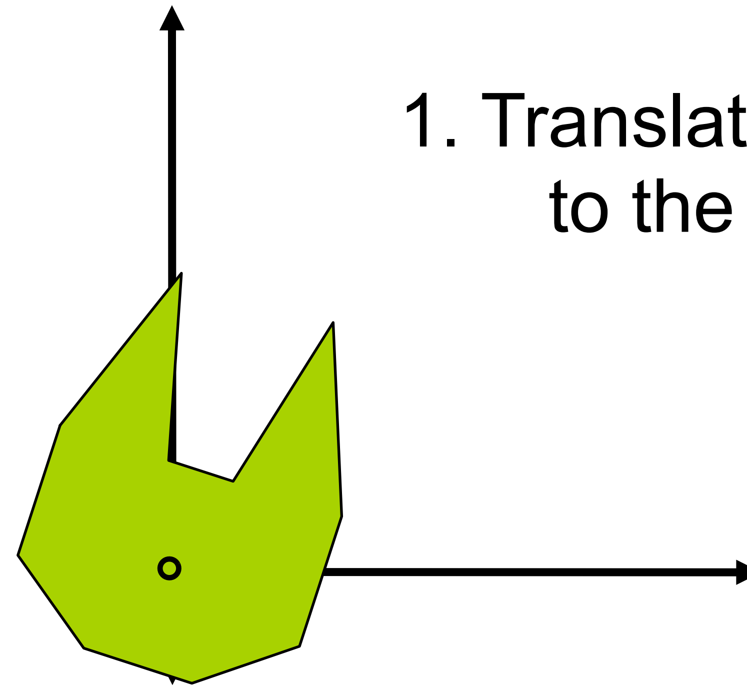
Simple Rotation



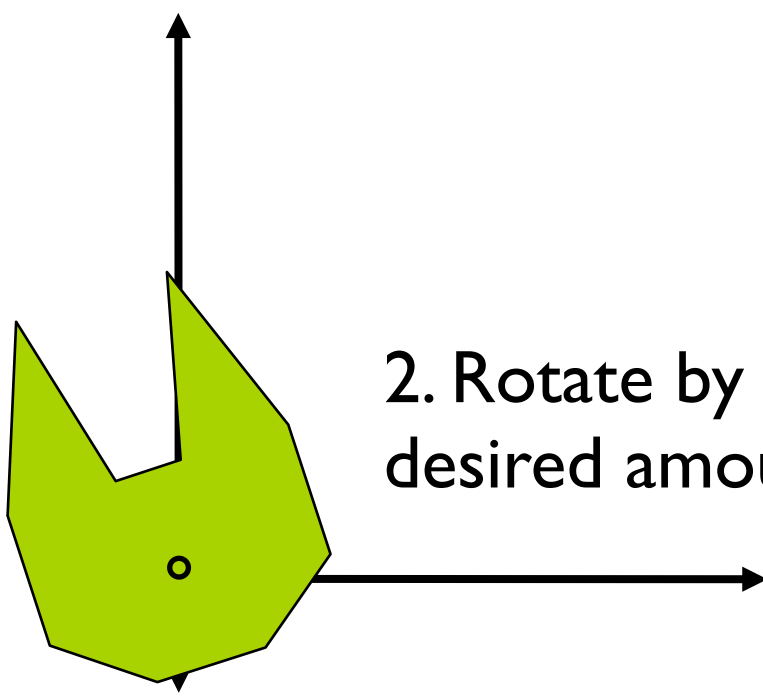
Suppose we wish to rotate the cat's head about its nose!

To rotate the cat's head about its nose



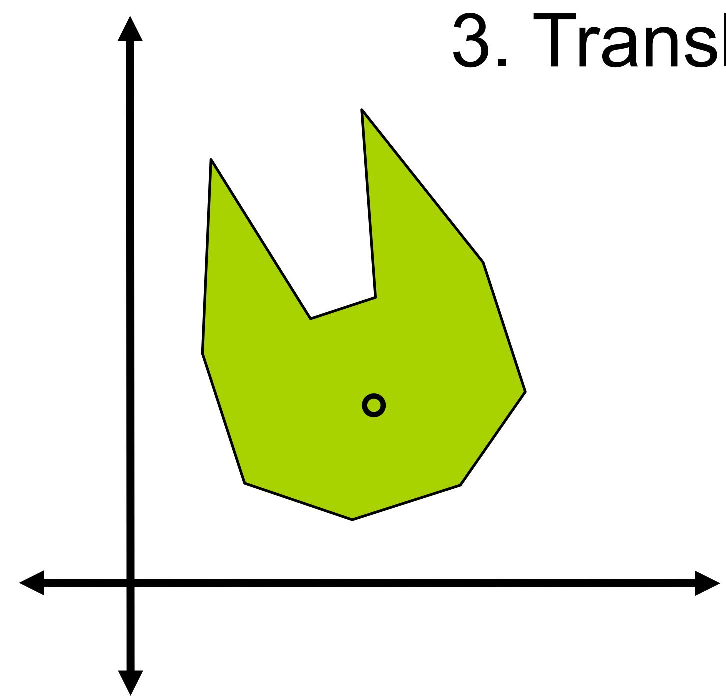


1. Translate the Nose to the Origin



2. Rotate by the desired amount

3. Translate back



Composition...

This is an instance of a general rule:
to apply transformation A to point p, and the transform result by transformation B, to obtain, say, q:

$$\boxed{q = (B\,A)\,p = B\,(A\,p)}$$

Composite Transformation

- Resultant of a sequence of transformations
- Composite transformation matrix is equal to the product of the sequence of the given transformation matrices

$$
\begin{aligned}
Q_h &= M_n * \ldots * M_2 * M_1 * P_h \\
&= M * P_h
\end{aligned}
$$

Rotation About Point P (Math)

Point about which to rotate $P = \begin{bmatrix} T_x \\ T_y \\ 1 \end{bmatrix}$

Translate to Origin

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

Rotate

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

Translate Back

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

Composition Maps a Point A to new Point B. $B := M_4\, A$

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

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

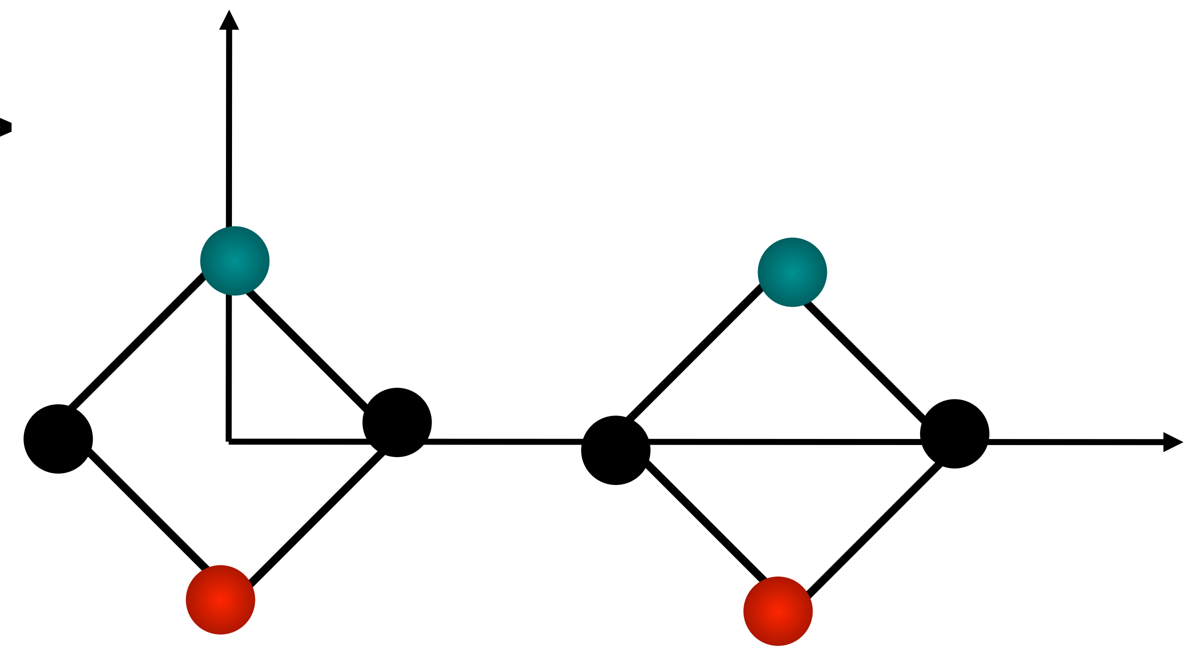
Scaling About Point P

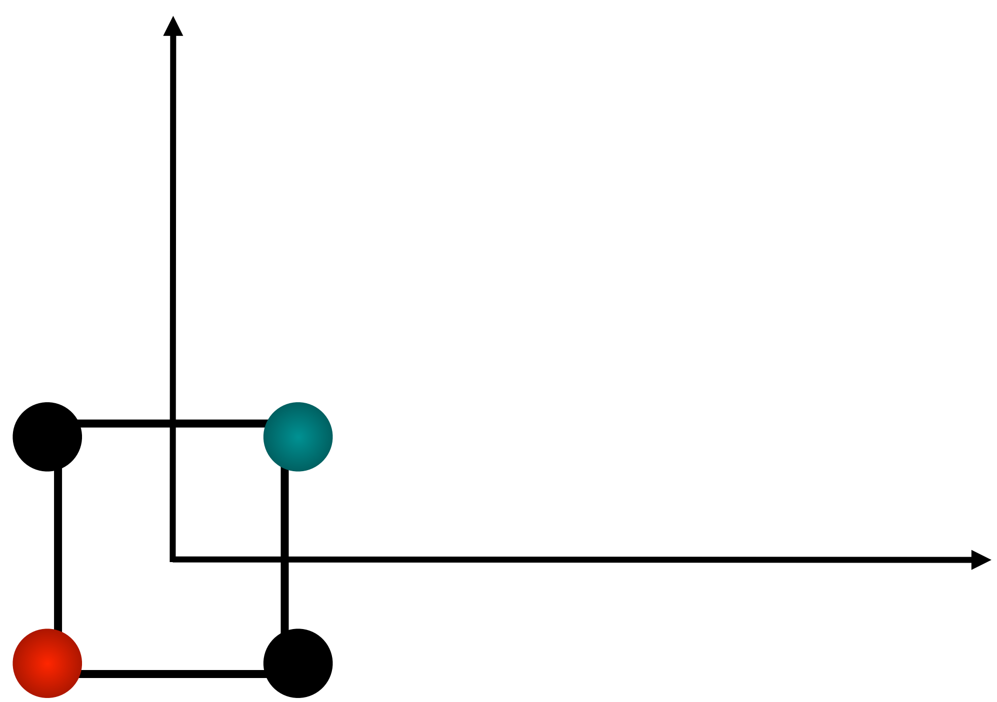
- Scaling also operates relative to the Origin.
- To make an object bigger without moving it
  - Translate P to origin.
  - Apply scaling.
  - Inverse translation.

$$M_4 = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -T_x \\ 0 & 1 & -T_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & -S_x T_x + T_x \\ 0 & S_y & -S_y T_y + T_y \\ 0 & 0 & 1 \end{bmatrix}$$

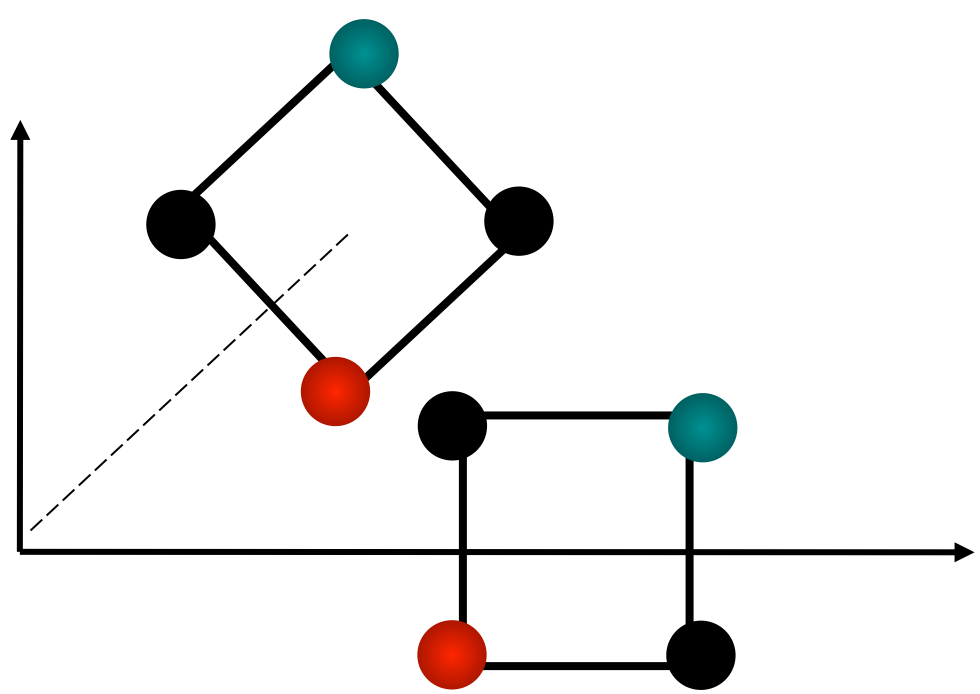
Matrix Multiplication is Not Commutative

First rotate, then translate =>





First translate, then rotate =>



Composite of basic transformations

- Order of multiplication of the matrices is important because matrix multiplication is not commutative
- Most of the transformations that we normally deal with can be obtained as composite of the 3 basic transformations, i.e., translation, scaling, and rotation

Rotation about arbitrary axis

- Given:

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

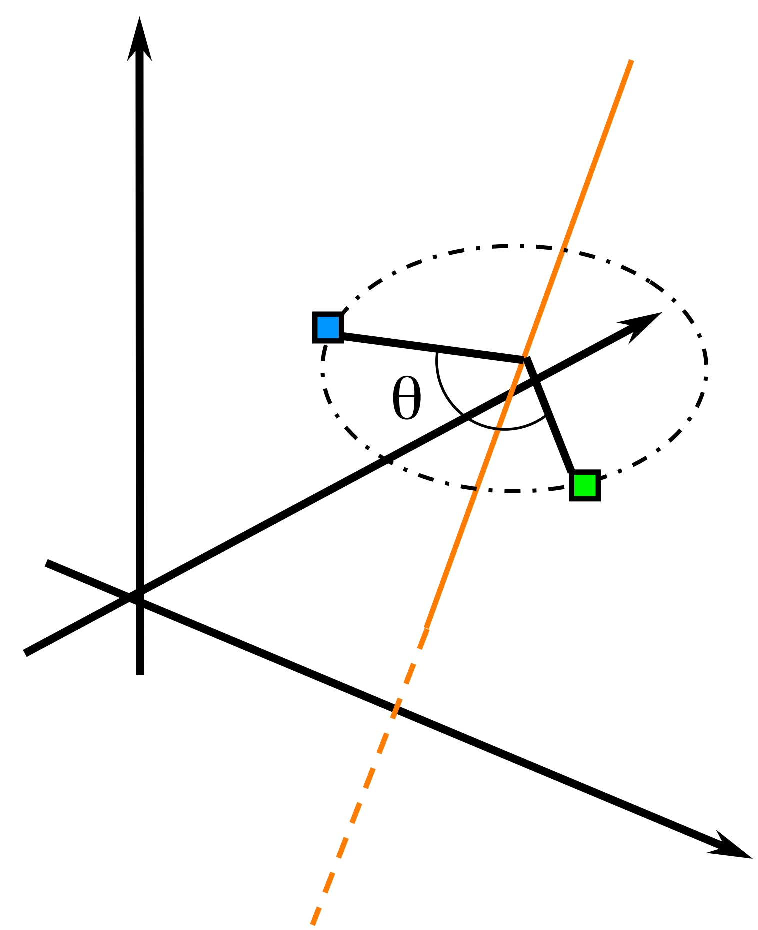
Angle of rotation: θ

- Procedure

1. Transform so that the given axis coincides with the Z axis

2. Rotate by θ

3. Apply inverse of step 1. transforms

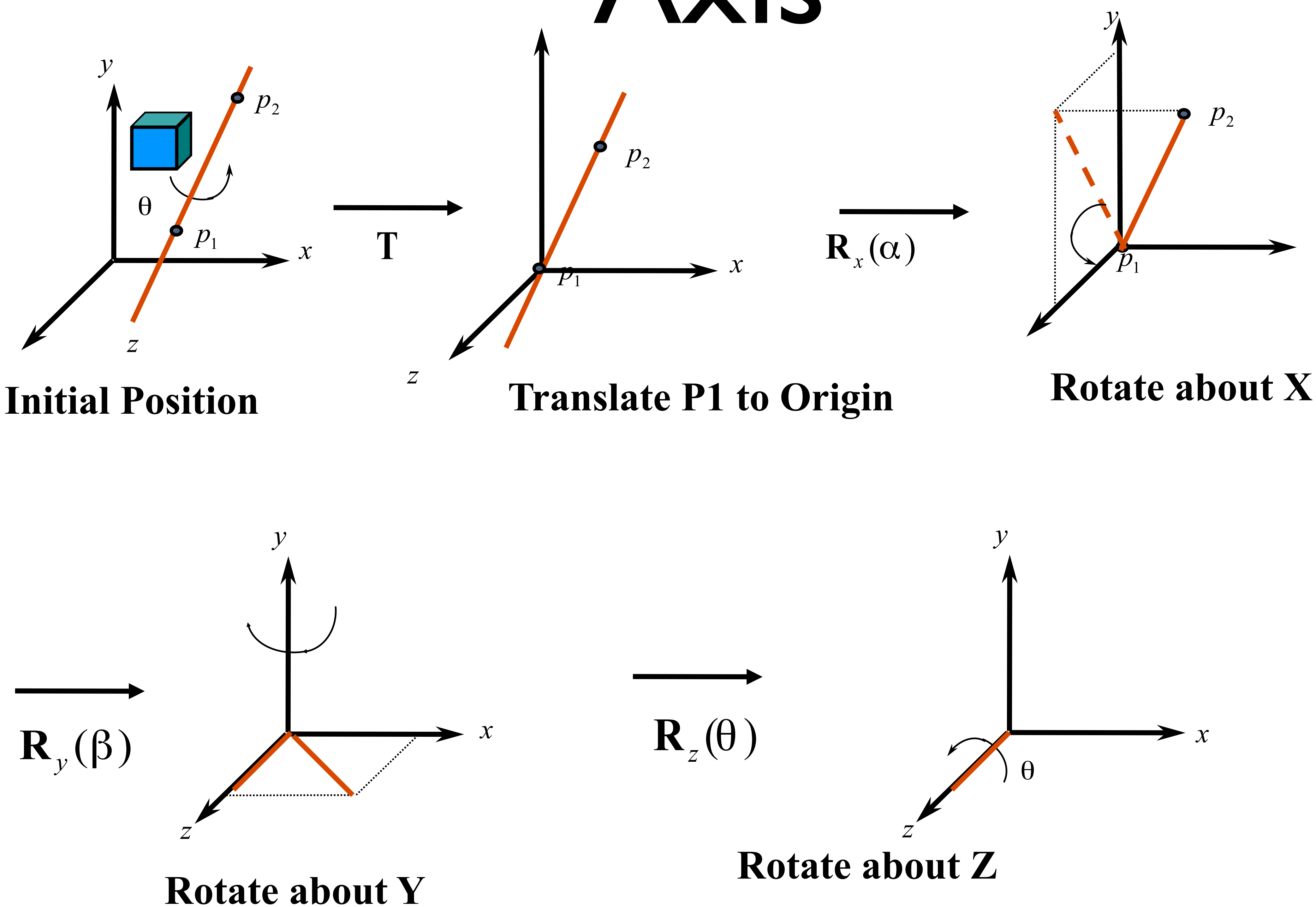


Rotation example (contd.)

- Steps

$T_{-(x_1, y_1, z_1)}$	Makes given axis pass through origin
$R_{(x,\alpha)}$	Makes axis lie in ZX plane
$R_{(y,\beta)}$	Makes axis coincide with the Z axis
$R_{(z,\theta)}$	Applies given rotation
	Apply inverses of aligning transformations

Rotation About Arbitrary Axis



Alternative solution

- Quaternion (10 min reading)
  - what is?
  - basic operations
  - and how to perform rotation
- reference:
  - http://www.cs.ucr.edu/~vbz/resources/quatut.pdf

Reading and discussion

http://www.cad.zju.edu.cn/home/zhx/CG/2017/lib/exe/fetch.php?media=quatut-2-2.pdf

Transformations in OpenGL

- Model-view matrix
- Projection matrix
- Texture matrix

Programming Transformations

- In OpenGL, the transformation matrices are part of the state, they must be defined *prior to* any vertices to which they are to apply.
- In modeling, we often have objects specified in their own coordinate systems and must use transformations to bring the objects into the scene.
- OpenGL provides *matrix stacks* for each type of supported matrix (model-view, projection, texture) to store matrices.

Current Transformation Matrix

- Current Transformation Matrix (CTM)

Is the matrix that is applied to any vertex that is defined subsequent to its setting.

- If we change the CTM, we change the *state* of the system.
- CTM is a 4 x 4 matrix that can be altered by a set of functions.

Changing CTM

- 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)

Rotation About an Arbitrary Point

Task:
Rotate an object by 45.0 degrees about the line from (4.0, 5.0, 6.0) to (5.0, 7.0, 9.0). (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);
```

Order of Transformations

- The transformation matrices appear in *reverse* order to that in which the transformations are applied.
- *In OpenGL, the transformation specified most recently is the one applied first.*

Matrix Stacks

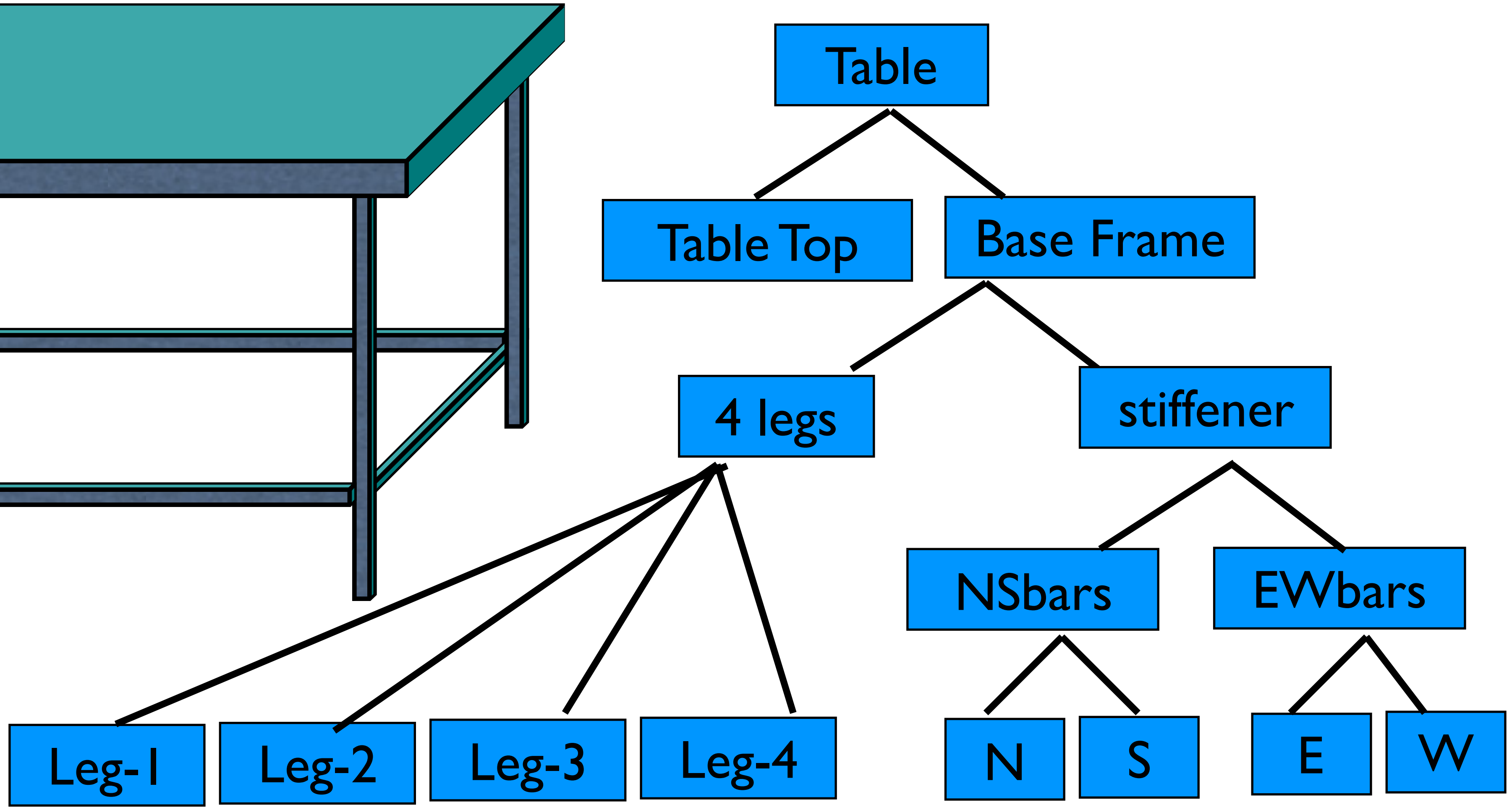
- OpenGL uses matrix stacks mechanism to manage modeling transformation hierarchy.

  glPushMatrix (void);

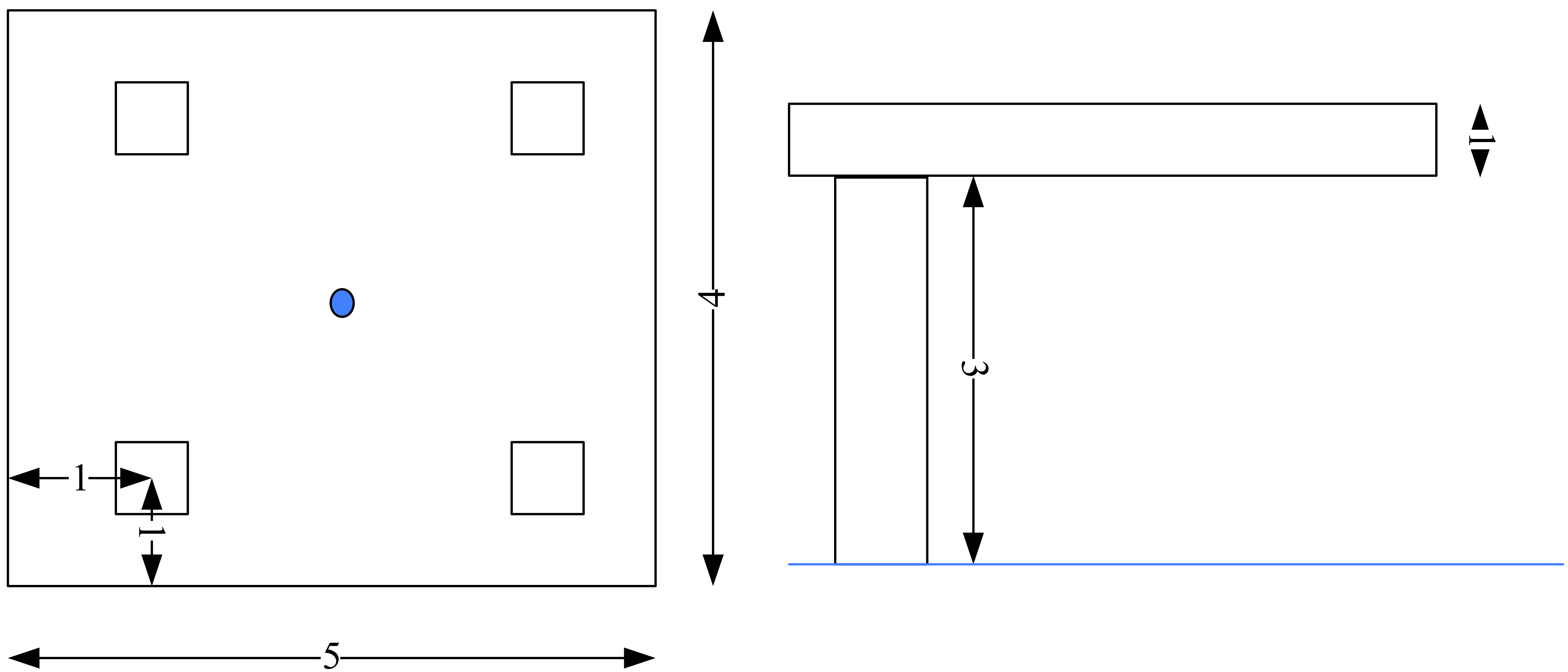
  glPopMatrix (void);

- OpenGL provides matrix stacks for each type of supported matrix to store matrices.
  - Model-view matrix stack
  - Projection matrix stack
  - Texture matrix stack

Example of Modeling Transform hierarchy



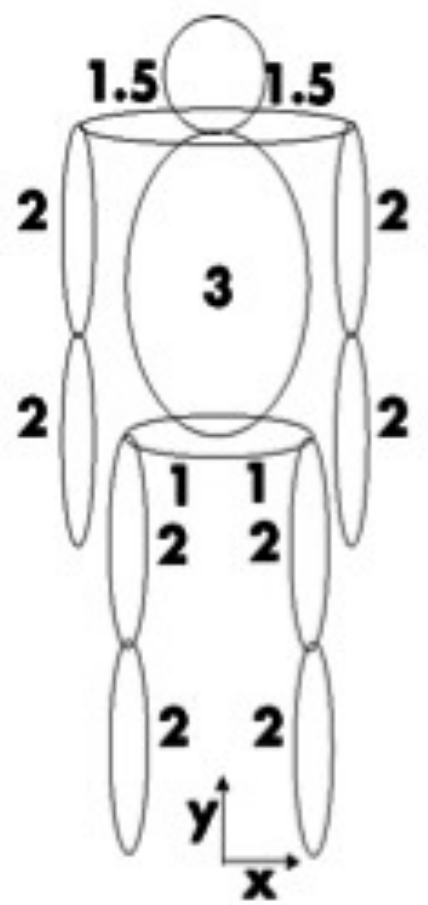
Ex – Desk with 4 legs



By calling glutSolidCube() …

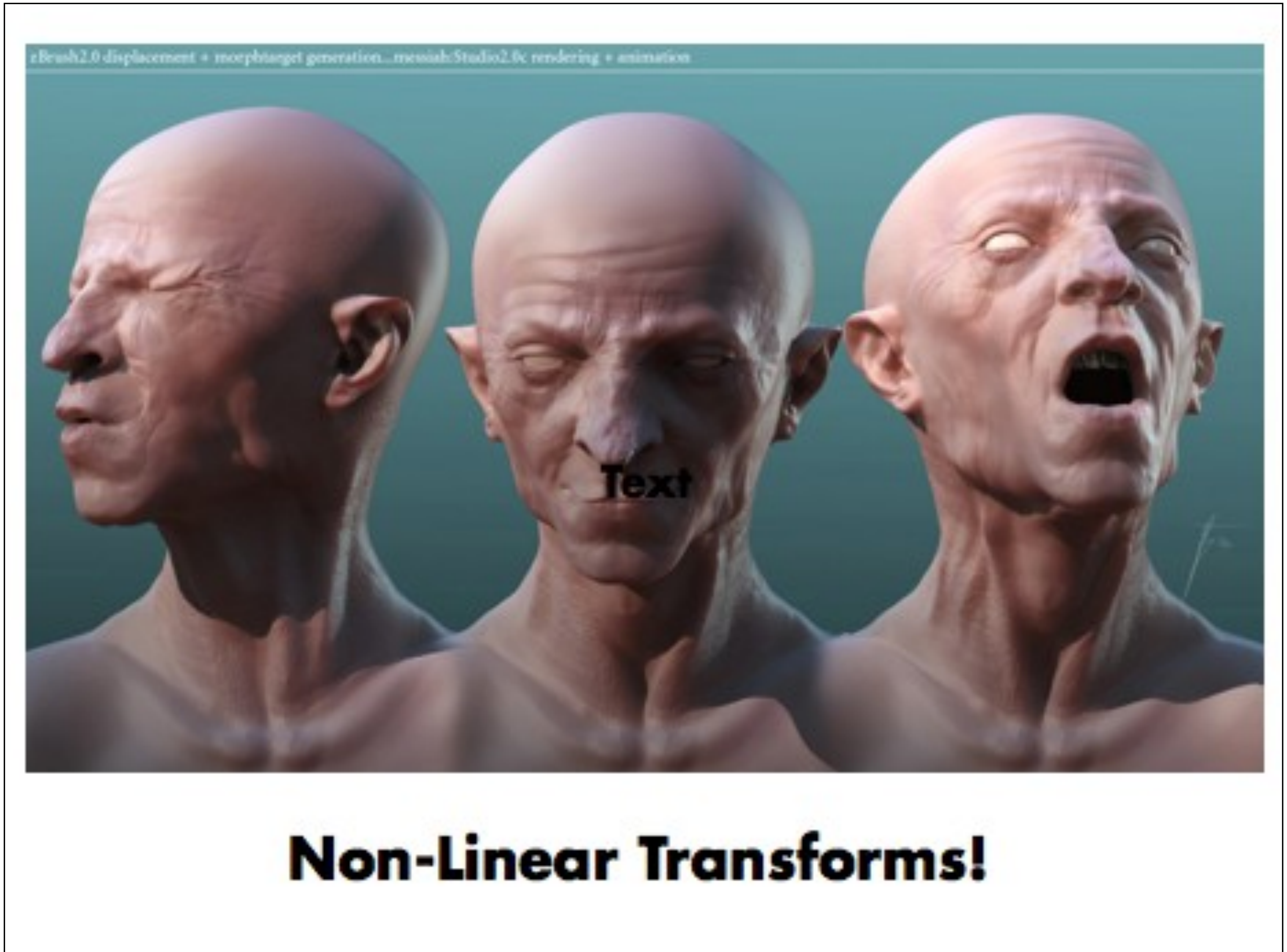
Hierarchal transformations

Skeleton

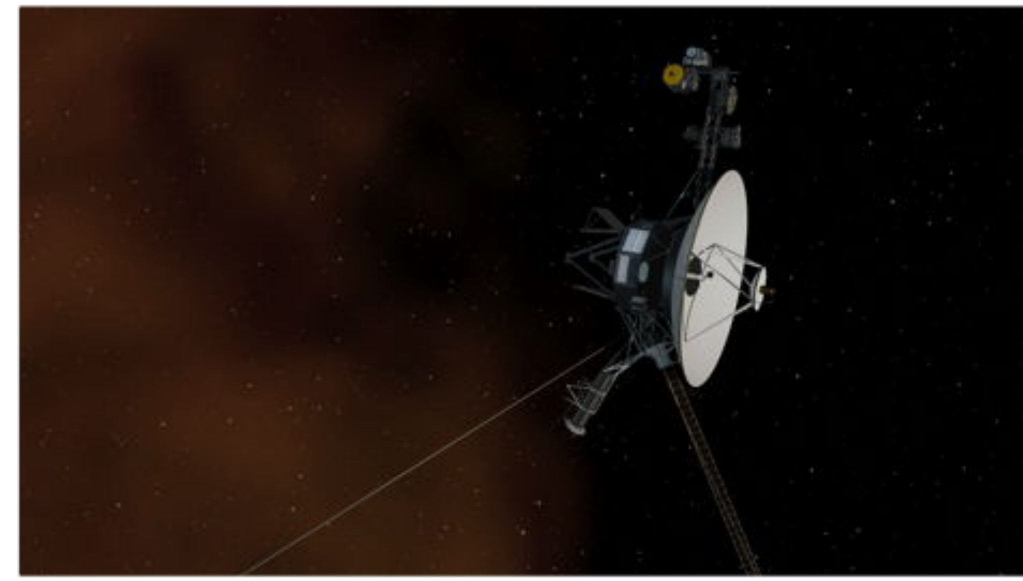



```
body
  torso
    head
    shoulder
      larm
        upperarm
        lowerarm
        hand
      rarm
        upperarm
        lowerarm
        hand
    hips
      lleg
        upperleg
        lowerleg
        foot
      rleg
        upperleg
        lowerleg
        foot
```

CS148 Lecture 4 Pat Hanrahan, Fall 2010


zBrush2.0 displacement + morphtarget generation...messiah:Studio2.0c rendering + animation
Text
Non-Linear Transforms!

Homework 02



旅行者1号已迈进
星际空间

- Build a Solar System
  - requirement:
    - detailed computing steps
    - at least Earth, Moon and Sun
    - implement it in OpenGL/WebGL
    - handout demo and code
  - Deadline: 2019-11-13

Thank You