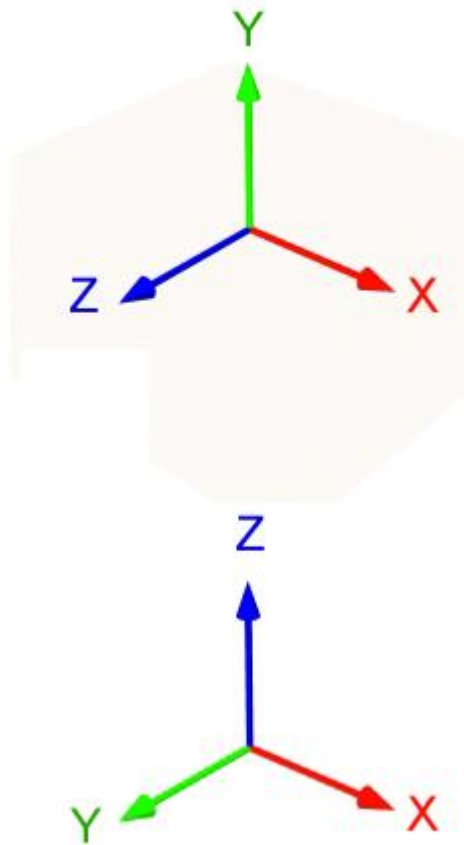


# **Computer Graphics**

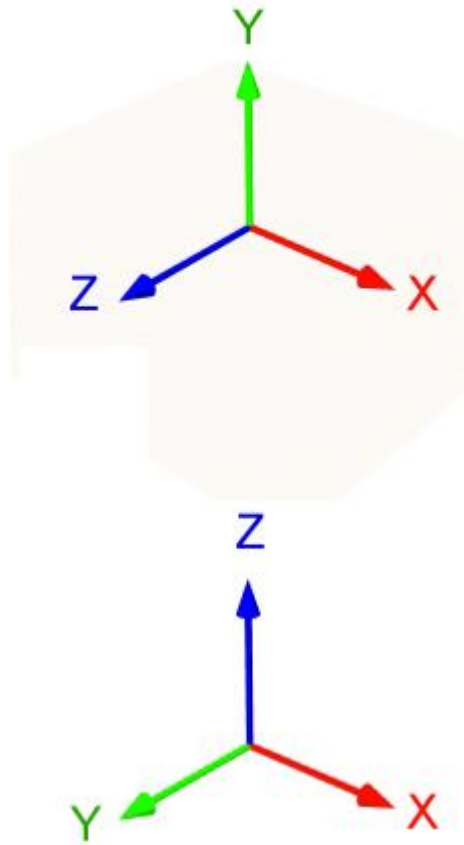
## **Geometric Transformations**

# Change of Coordinate Systems



- an essential role in the pipeline
- a way of assigning numbers to points
- *frame of reference (reference frame)*:  
a coordinate system in which we are currently representing our objects.

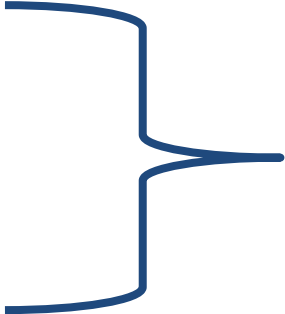
# Change of Coordinate Systems



- a number of systems:
  - **World (Universe) Coordinate System:** the base system, other relate to this, objects are brought together in this system when setting up a scene, its origin is at the center of the scene.
  - **Object (Modelling) Coordinate System:** an object can be specified using this, a frame of reference for a specific object, we must pick some point to be the origin and the orientation.
  - **Camera (Eye, Viewpoint) Coordinate System:** a frame of reference from our camera or eyes, everything is calculated with respect to this, based upon the viewpoint of the observer or camera.

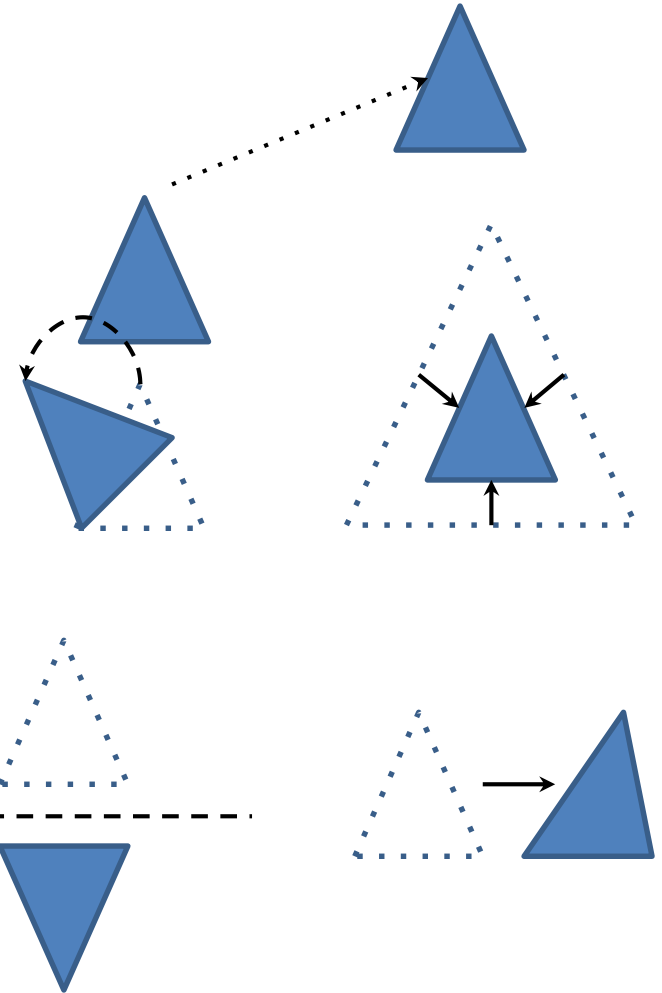
*Geometric transformations are used for change of coordinate systems and for computing object movement and camera movement in animations.*

# Geometric Transformations

- Translation
  - Rotation
  - Scaling
  - Reflection
  - Shearing
- 
- Basic transformations

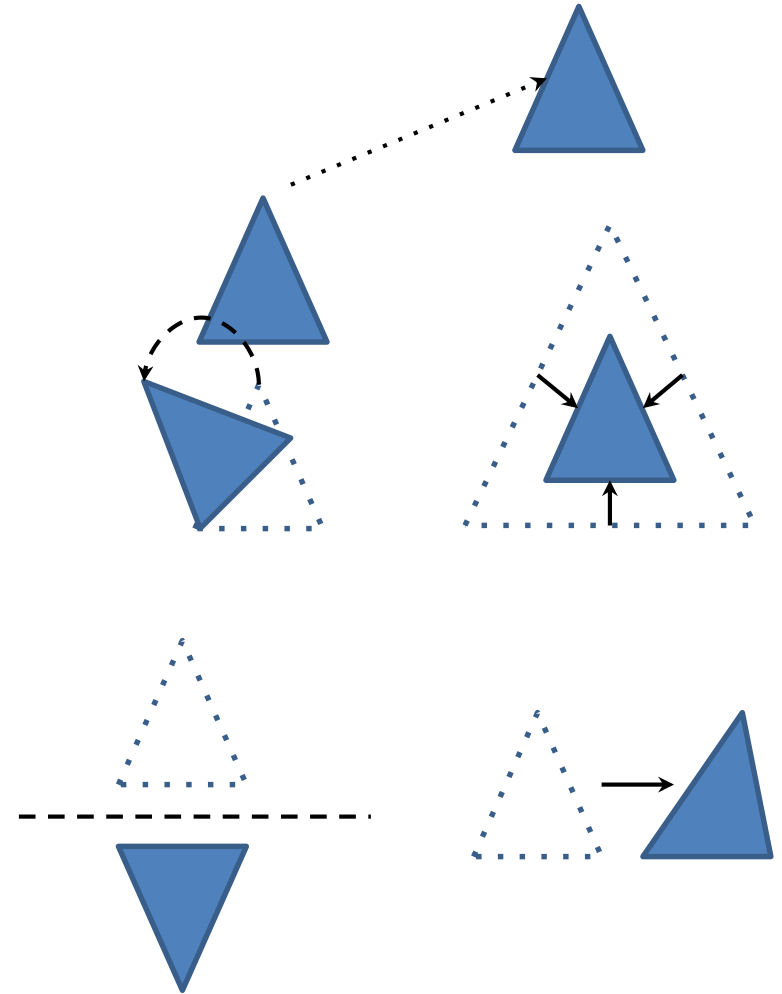
# Geometric Transformations

- Translation:
  - moves the object without deformation (rigid-body transformation)
  - for a line segment, apply the transformation equation to each of the two line endpoints. Redraw the line between new endpoints.
  - for a polygon, apply the transformation equation to coordinates of each vertex and regenerate the polygon using the new set of vertex coordinates



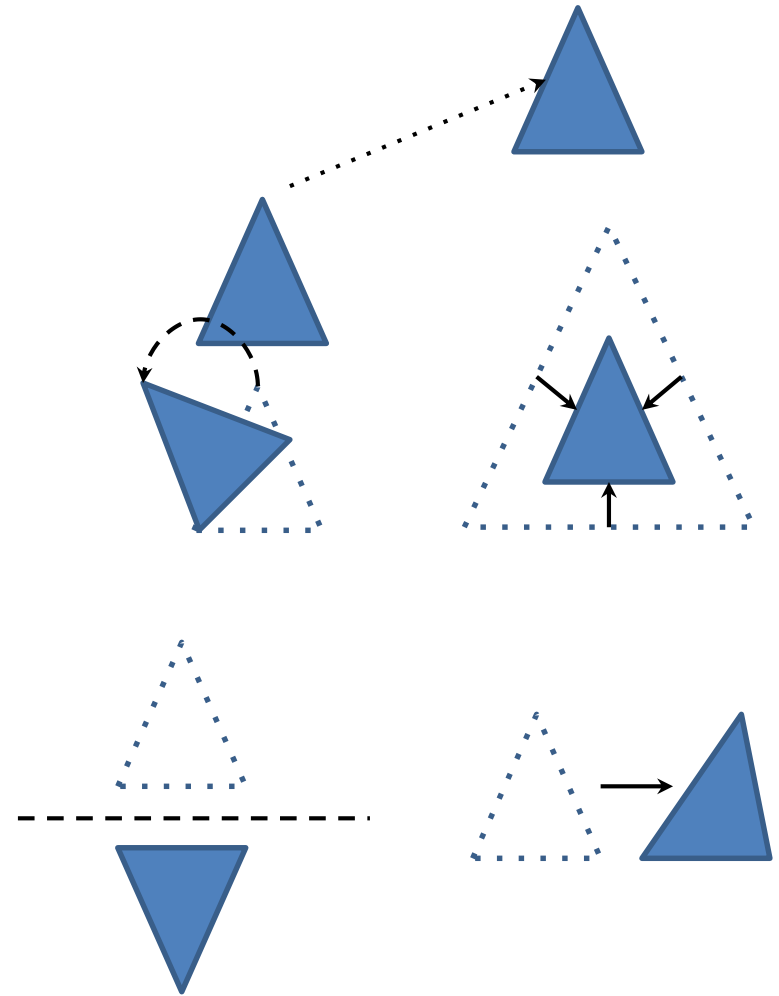
# Geometric Transformations

- Rotation:
  - moves the object without deformation
  - a line is rotated by applying the rotation formula to each of the endpoints and redrawing the line between the new end points
  - a polygon is rotated by applying the rotation formula to each of the vertices and redrawing the polygon using new vertex coordinates



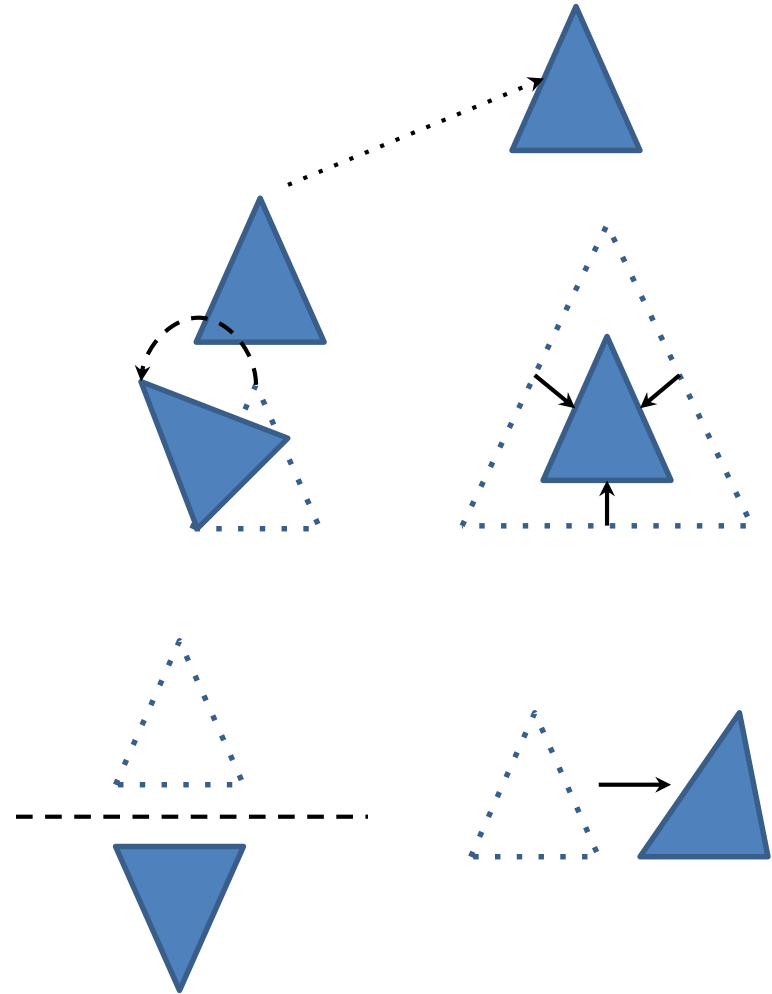
# Geometric Transformations

- **Scaling:**
  - alters the size of an object
  - 2D scaling is performed by multiplying object positions by positive scaling factors (in x and y axes)
  - $sf < 1 \rightarrow$  reduces the size
  - $sf > 1 \rightarrow$  enlarge the object
  - $sf = 1 \rightarrow$  no change



# Geometric Transformations

- Reflection:
  - produces a mirror image of an object
  - rotate the object  $180^\circ$  about the reflection axis
- Shearing:
  - distorts the shape of an object





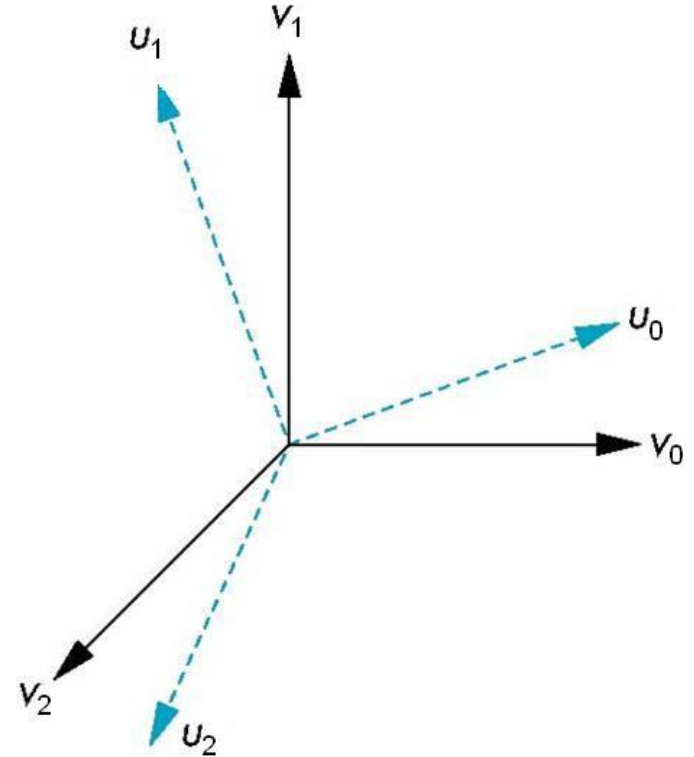
# Different Vector Bases

(1)  $(x, y, z)^T$

(2)  $(1,0,0)^T, (0,1,0)^T, (0,0,1)^T$

(3)  $(v_1, v_2, v_3) \quad (u_1, u_2, u_3)$

(4) 
$$\begin{aligned} u_1 &= \gamma_{11}v_1 + \gamma_{12}v_2 + \gamma_{13}v_3 \\ u_2 &= \gamma_{21}v_1 + \gamma_{22}v_2 + \gamma_{23}v_3 \\ u_3 &= \gamma_{31}v_1 + \gamma_{32}v_2 + \gamma_{33}v_3 \end{aligned}$$



Each of the basis vectors  $(u_0, u_1, u_2)$  are vectors that can be represented in terms of the other basis  $(v)$

# Different Vector Bases

$$(5) \quad \mathbf{M} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} \end{bmatrix}$$

\*  $\mathbf{M}$ :

- 3 x 3 matrix of scalars
- Obtains info to go

from a representation to the other

$$(6) \quad \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \mathbf{M} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix},$$

\* Using (5), we obtain (6) and (7)

or

$$(7) \quad \mathbf{u} = \mathbf{M}\mathbf{v}.$$

# Different Vector Bases

$\mathbf{a} = (\alpha_1, \alpha_2, \alpha_3)$  representation of **vector w** w.r.t.  $(v_1, v_2, v_3)$   
*a is rep. of w w.r.t. v basis*

$\mathbf{b} = (\beta_1, \beta_2, \beta_3)$  representation of **vector w** w.r.t.  $(u_1, u_2, u_3)$   
*b is rep. of w w.r.t. u basis*

$$w = \alpha_0 v_0 + \alpha_1 v_1 + \alpha_2 v_2$$

$$w = \beta_0 u_0 + \beta_1 u_1 + \beta_2 u_2$$



$$w = \mathbf{b} \mathbf{u} = \mathbf{b} \mathbf{M} \mathbf{v} = \mathbf{a} \mathbf{v}$$

$$\mathbf{a} = \mathbf{b} \mathbf{M}$$

Using these two representations of w, we obtain

$$(8) \quad \mathbf{a} = \mathbf{M}^T \mathbf{b}$$

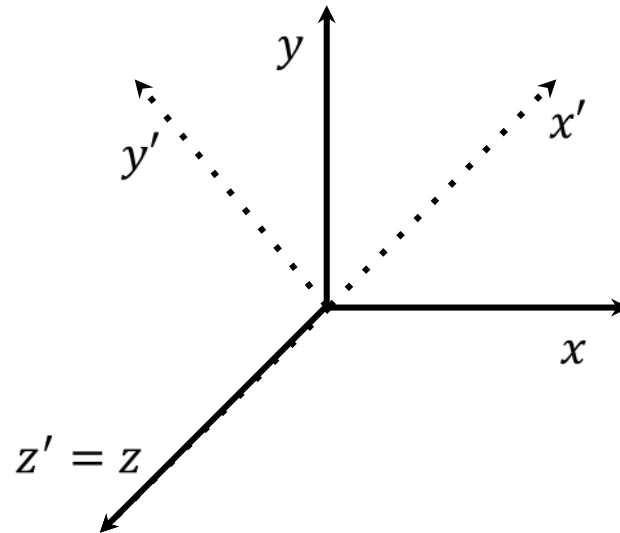
$$(9) \quad \mathbf{b} = (\mathbf{M}^T)^{-1} \mathbf{a}$$

# Example

We've a vector  $\mathbf{a}$ , we want to find coordinates with the new axes  $x'$ ,  $y'$ , and  $z'$

$$\mathbf{a} = (1, 2, 3)^T$$

$$\begin{aligned}x' &= (1, 1, 0) \\y' &= (-1, 1, 0) \\z' &= z\end{aligned}$$



# Example

Form M: new axes are the rows of M

$$\begin{aligned} x' &= (1, 1, 0) \\ y' &= (-1, 1, 0) \\ z' &= z \end{aligned} \quad \Rightarrow \quad \mathbf{M} = \begin{bmatrix} 1 & 1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

inverse of transpose of M

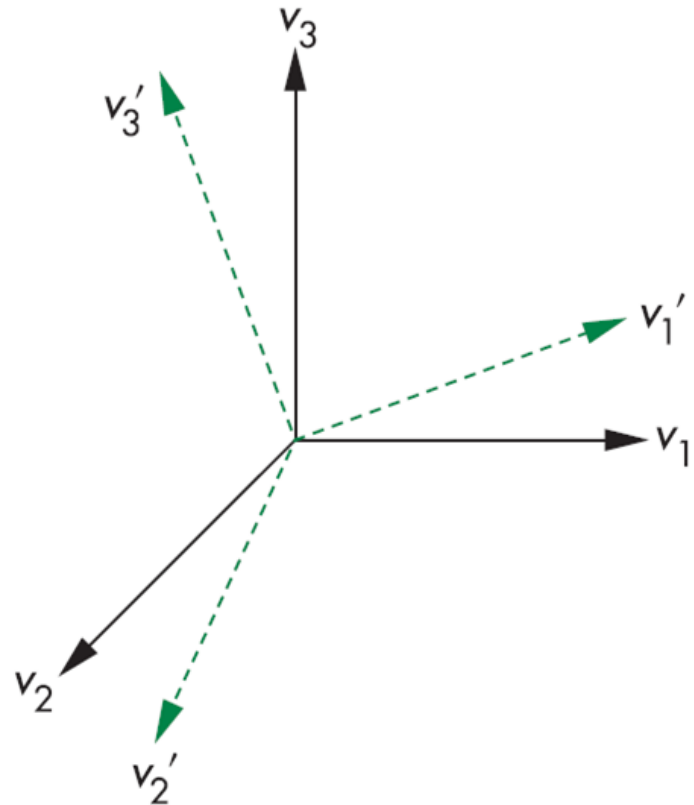
$$(\mathbf{M}^T)^{-1} = \begin{bmatrix} 1 & -1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 0.5 & 0.5 & 0 \\ -0.5 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$(\mathbf{M}^T)^{-1} \mathbf{a} = \begin{bmatrix} 0.5 & 0.5 & 0 \\ -0.5 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 1.5 \\ 0.5 \\ 3 \end{bmatrix}$$

vector a  
in the new system

# Different Vector Bases

- \* We can use basis change to represent rotation and scaling.
- \* It cannot be used to represent translation (Since it leaves the origin unchanged.).

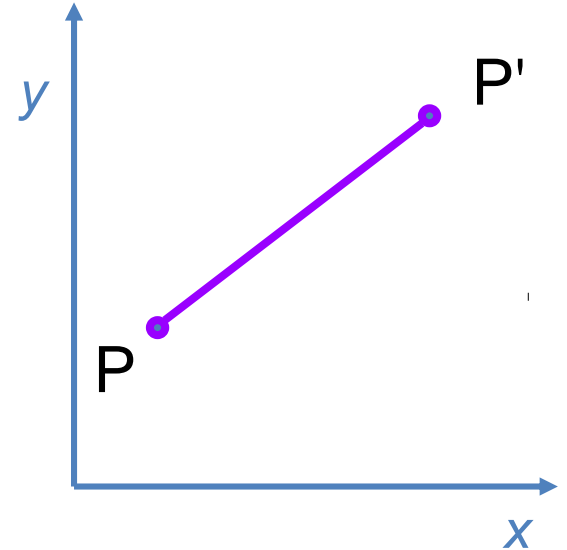


# 2D Translation

We want to translate a point  $P(x, y)$  to  $P'(x', y')$

$$x' = x + t_x, \quad y' = y + t_y$$

distances in x and y



Define column vectors  $P$ ,  $P'$ , and  $T$

$$\mathbf{P} = \begin{bmatrix} x \\ y \end{bmatrix}, \quad \mathbf{P}' = \begin{bmatrix} x' \\ y' \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$
$$\mathbf{P}' = \mathbf{P} + \mathbf{T}$$

**T: translation matrix**

# 2D Scaling

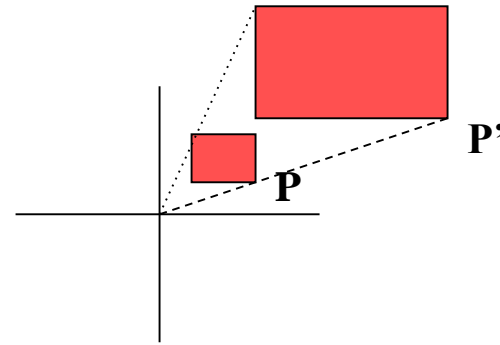
Scaling: altering the size of an object

We want to translate a point P (x,y) to P' (x',y'), by factors  $s_x$ ,  $s_y$  (along the x and y axes)

$$x' = x \cdot s_x, \quad y' = y \cdot s_y$$

**S matrix is formed using the factors**

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$



$$\mathbf{P}' = \mathbf{S} \cdot \mathbf{P}$$

**S: scaling matrix**



# 2D Rotation

Rotate an object w.r.t. an angle in 2D plane, clockwise or counterclockwise

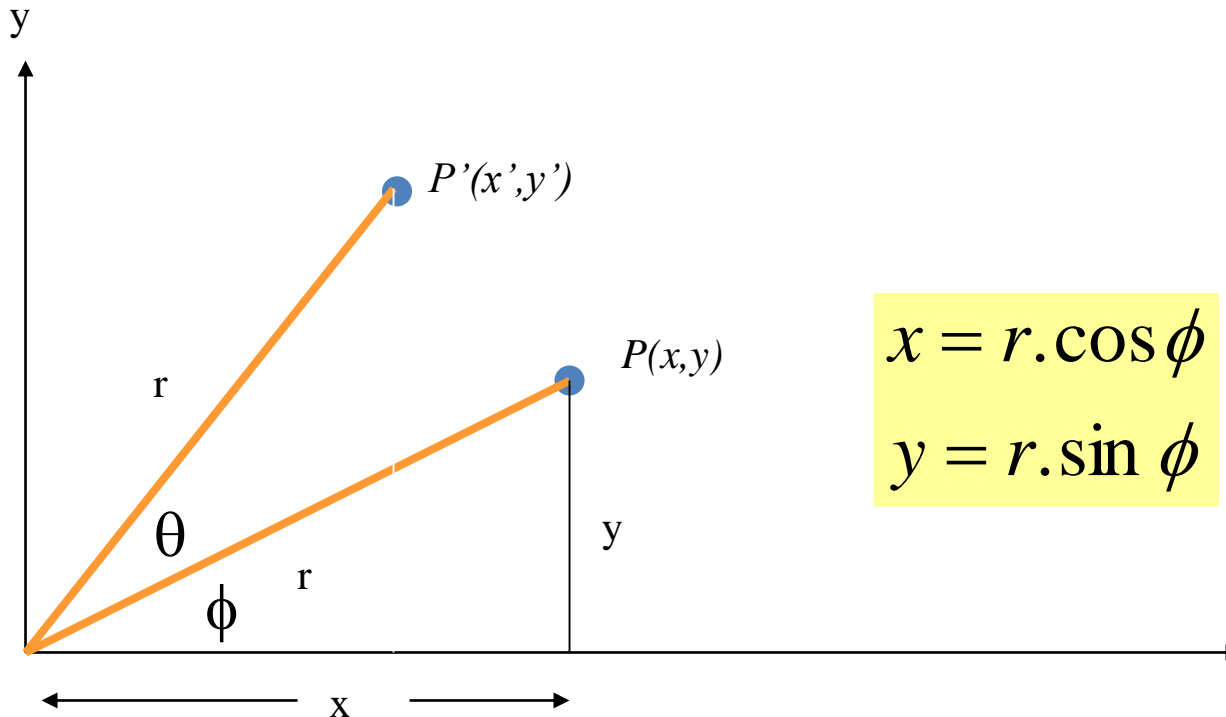
Initial coordinates of P:  $x$  and  $y$

Initial angle of P w.r.t. origin:  $\Phi$

Rotation angle:  $\Theta$

Distance to origin:  $r$

Coordinates (after rotation) of P':  $x'$  and  $y'$



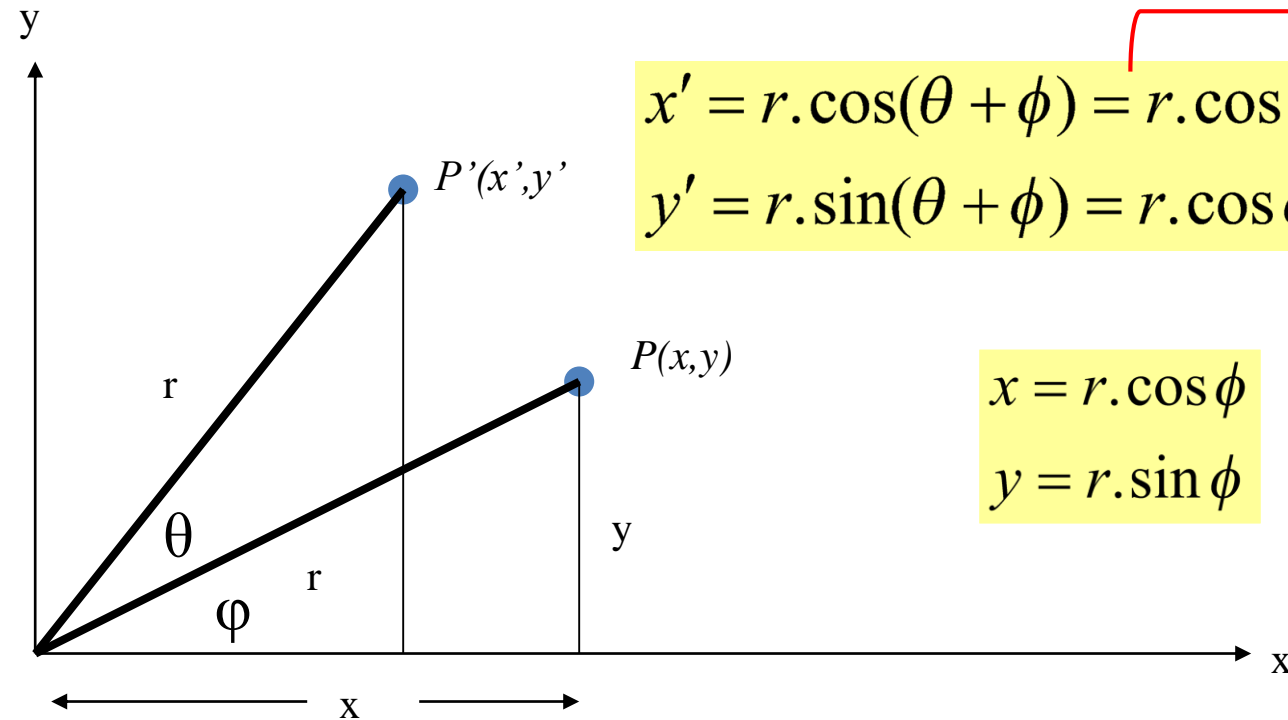
# 2D Rotation

Coordinates (after rotation) of P':  $x'$  and  $y'$

substitution of trigonometric  
equations of cos and sin

$$\begin{aligned}x' &= r.\cos(\theta + \phi) = r.\cos \phi.\cos \theta - r.\sin \phi.\sin \theta \\y' &= r.\sin(\theta + \phi) = r.\cos \phi.\sin \theta + r.\sin \phi.\cos \theta\end{aligned}$$

$$\begin{aligned}x &= r.\cos \phi \\y &= r.\sin \phi\end{aligned}$$



# 2D Rotation

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

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

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

**We substitute x and y**

**Finally we have:**

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

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

# 2D Rotation

$$\begin{aligned}x' &= x \cdot \cos \theta - y \cdot \sin \theta \\y' &= x \cdot \sin \theta + y \cdot \cos \theta\end{aligned}$$

Rewrite the factors of x and y



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

**Gives us this matrix multiplication**

$$\text{Define the matrix } R = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}, \quad P' = R \cdot P$$

**R: rotation matrix**

# Transformations

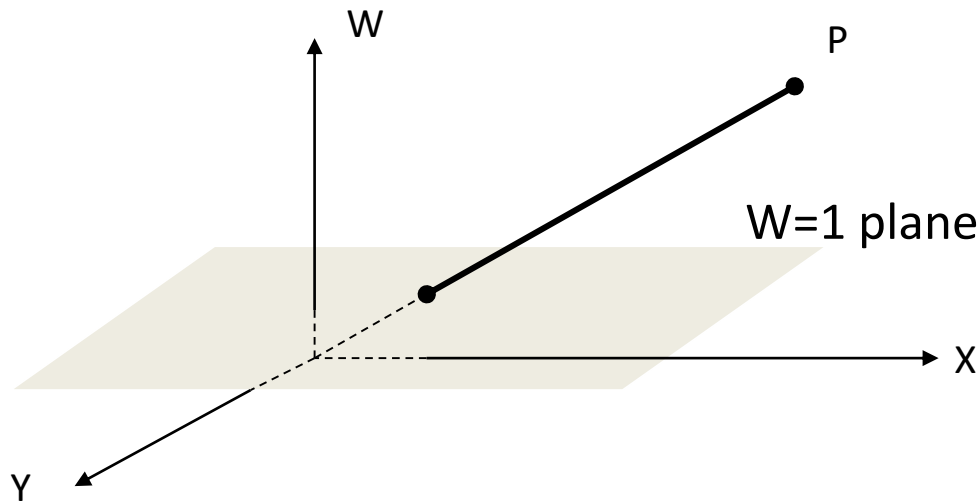
- \* We'd like, all transformations to be multiplications (but translation is addition), so we can concatenate them.
- \* Points are needed to be expressed in homogenous coordinates.
- Translation
  - $P' = T + P$
- Scale
  - $P' = S \cdot P$
- Rotation
  - $P' = R \cdot P$

# Homogeneous Coordinates

- To a point add an extra coordinate,  $W$ .
  - $P(x,y)$  becomes  $P(x,y,W)$
- Two sets of homogeneous coordinates represent the same point if they are a multiple of each other:
  - e.g.  $(2,5,3)$  and  $(4,10,6)$  represent the same point.
- At least one component must be non-zero;  $(0,0,0)$  is not defined.
- If  $W \neq 0$ , Cartesian (homogenised) coordinates:  $(x/W, y/W, 1)$ .
- If  $W=0$ ,  $P$  is at infinity.

# Homogeneous Coordinates

- All triples  $(x,y,W)$  representing the same point describe a line, passing through the origin.
- Homogenised point  $(x,y,1)$ 
  - homogenised points form a plane at  $W=1$ .



# Translation with Homogenised Coordinates

**2D T is now 3D:**

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{aligned} x' &= x + d_x \\ y' &= y + d_y \\ 1 &= 1 \end{aligned}$$



# Concatenation

We perform two translations on the same point P:

$$\begin{aligned}P' &= T(d_{x1}, d_{y1})P \\P'' &= T(d_{x2}, d_{y2})P' \\P'' &= T(d_{x2}, d_{y2})T(d_{x1}, d_{y1})P = T(d_{x2} + d_{x1}, d_{y2} + d_{y1})P\end{aligned}$$

Concatenation is provided summation of distances on the same axis:

$$T(d_{x2}, d_{y2})T(d_{x1}, d_{y1}) = T(d_{x2} + d_{x1}, d_{y2} + d_{y1})$$

# Concatenation

Two translation matrices are multiplied. We obtain compounding, concatenation or composition.

$$\begin{pmatrix} 1 & 0 & d_{x2} \\ 0 & 1 & d_{y2} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & d_{x1} \\ 0 & 1 & d_{y1} \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & d_{x2} + d_{x1} \\ 0 & 1 & d_{y2} + d_{y1} \\ 0 & 0 & 1 \end{pmatrix}$$

Composite Transformation Matrix (CTM)

# Properties of Translation

$$1. T(0,0) = I$$

$$2. T(s_x, s_y) \cdot T(t_x, t_y) = T(s_x + t_x, s_y + t_y)$$

$$3. T(s_x, s_y) \cdot T(t_x, t_y) = T(t_x, t_y) \cdot T(s_x, s_y)$$

$$4. T^{-1}(s_x, s_y) = T(-s_x, -s_y)$$

## Properties of translations:

1. If the distance of translation is zero for both axes, then we get identity, same point or object.
2. Concatenation
3. Translation matrices are commutative.
4. Matrix of the opposite side or negative translation is equal to inverse of the original translation matrix.

# Homogeneous Form of Scaling

Recall the (x,y) form of Scale :

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

In homogeneous coordinates (3 x 3) :

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

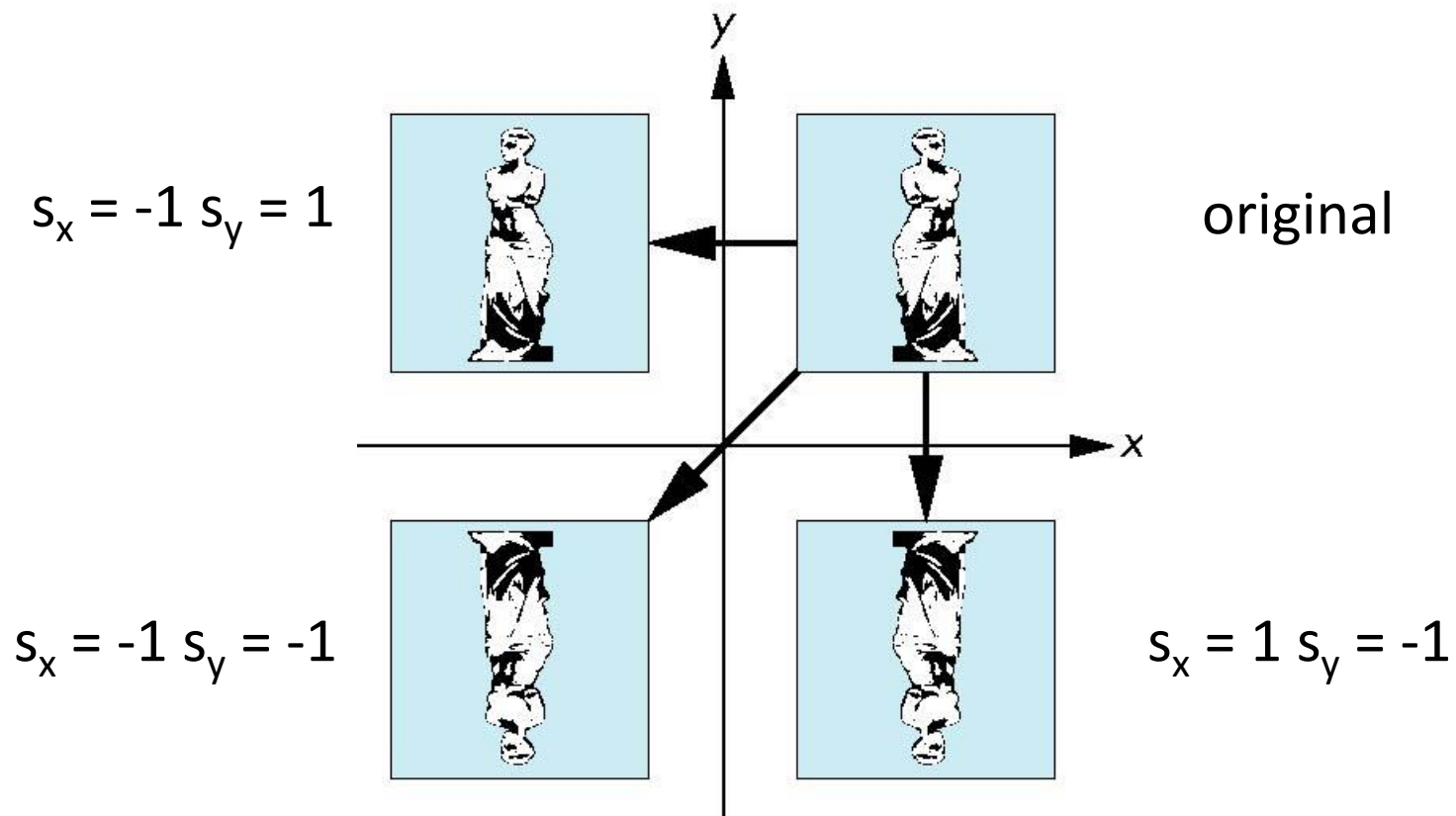
# Concatenation of Scaling

$$(s_{x1}, s_{y1}) \cdot S(s_{x2}, s_{y2})$$
$$\begin{bmatrix} s_{x1} & 0 & 0 \\ 0 & s_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_{x2} & 0 & 0 \\ 0 & s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{x1} \cdot s_{x2} & 0 & 0 \\ 0 & s_{y1} \cdot s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Product of two scaling matrices. Only diagonal elements are nonzero.

# Reflection

Relationship between reflection and scaling  
Reflection corresponds to negative scale factors.



# Homogeneous Form of Rotation

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

$$R^{-1}(\theta) = R(-\theta)$$

Inverse of the rotation matrices are the rotation matrices of negatives of the same angles.

$$R^{-1}(\theta) = R^T(\theta)$$

Rotation matrices are orthogonal, i.e. the inverse is the transpose.

# Orthogonality of Rotation Matrices

$$R^{-1}(\theta) = R(-\theta)$$

$$R^{-1}(\theta) = R^T(\theta)$$

$$\rightarrow R(-\theta) = R^T(\theta)$$

It's really true?

$$R(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad R^T(\theta) = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

SAME



# Other Properties of Rotation

$$R(0) = I$$

$$R(\theta) \cdot R(\phi) = R(\theta + \phi)$$

$$R(\theta) \cdot R(\phi) = R(\phi) \cdot R(\theta)$$

Other properties of rotation:

- Rotation matrix of zero is identity matrix.
- Concatenation
- Rotation matrices are commutative - only if the centres of rotation are the same. Otherwise it matters.

# 2D Composite Transformations

- **Combine transformations:** we have two or more transformations performed on the same object.
  - translate, rotate, translate
  - translate, scale, translate
  - translate, reflect, translate
- e.g. used to rotate or scale an object, with respect to a point that is **not the origin**.
- Implemented by multiplication of the homogeneous matrices.

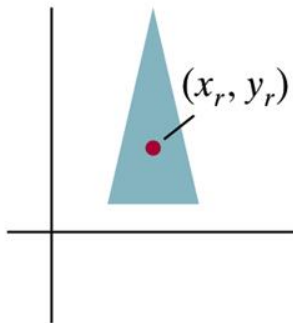
# 2D Composite Transformations

Rotate an object about a specified pivot point  $x, y$ :

First translate the object and pivot point is now at the origin.

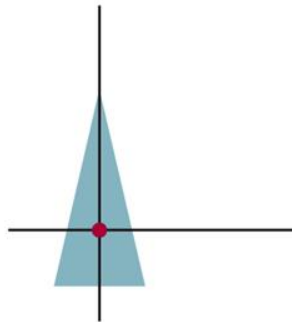
Then rotate the object with reference to the origin.

Again translate it back so the pivot point is returned to its position.



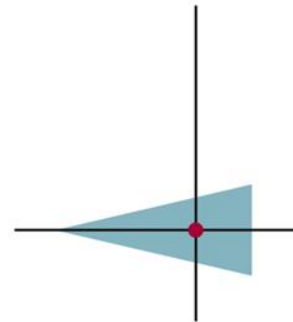
(a)

Original Position  
of Object and  
Pivot Point



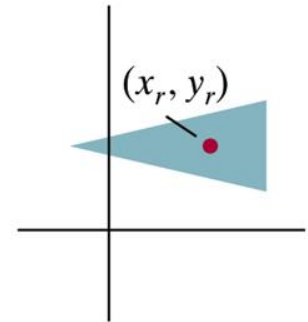
(b)

Translation of  
Object so that  
Pivot Point  
 $(x_r, y_r)$  is at  
Origin



(c)

Rotation  
about  
Origin



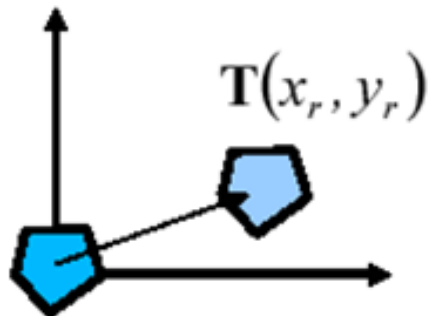
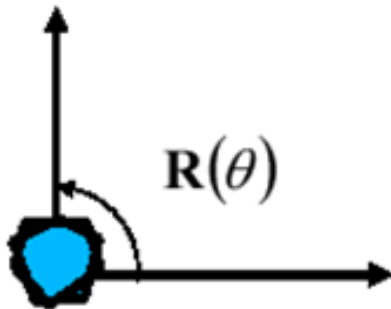
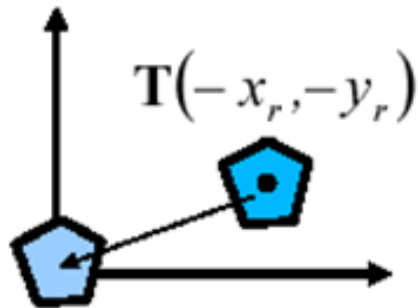
(d)

Translation of  
Object so that  
the Pivot Point  
is Returned  
to Position  
 $(x_r, y_r)$

# 2D Composite Transformations

Translate – rotate – translate back

First translation (to the origin) is on the right!



$$\begin{aligned} T(x_r, y_r) \cdot R(\theta) \cdot T(-x_r, -y_r) = \\ \begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix} = \\ \begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta) + y_r\sin\theta \\ \sin\theta & \cos\theta & y_r(1-\cos\theta) - x_r\sin\theta \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

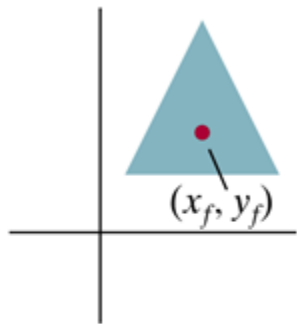
# 2D Composite Transformations

Scale an object with respect to a specified fixed point  $x, y$ :

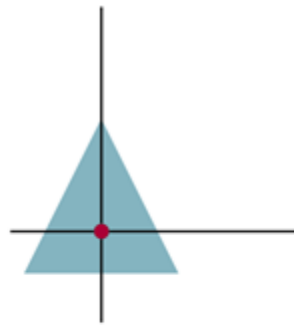
First translate the object and pivot point is now at the origin.

Then scale the object with respect to the origin.

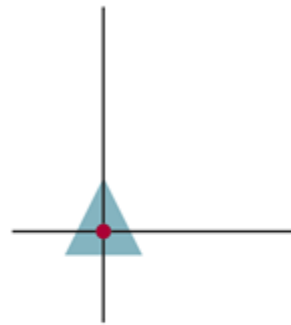
Again translate it back so the pivot point is returned to its position.



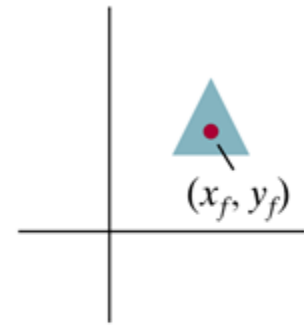
(a)  
Original Position  
of Object and  
Fixed Point



(b)  
Translate Object  
so that Fixed Point  
( $x_f, y_f$ ) is at Origin



(c)  
Scale Object  
with Respect  
to Origin

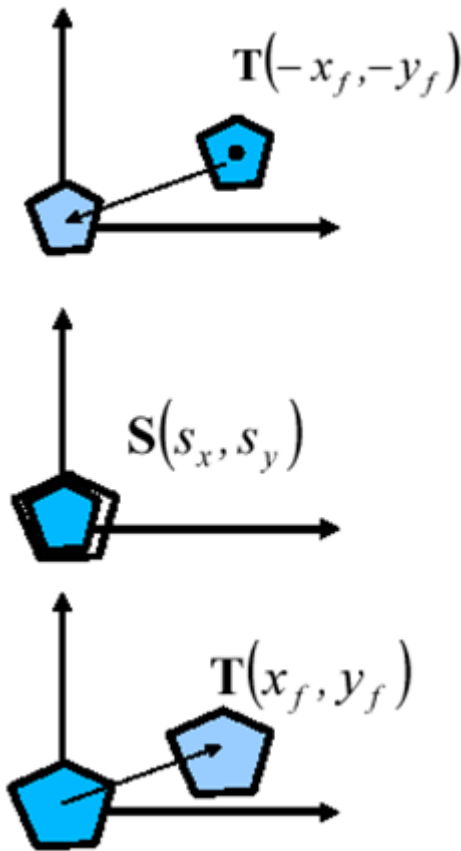


(d)  
Translate Object  
so that the Fixed  
Point is Returned  
to Position ( $x_f, y_f$ )

# 2D Composite Transformations

Translate – scale – translate back

First translation (to the origin) is on the right!



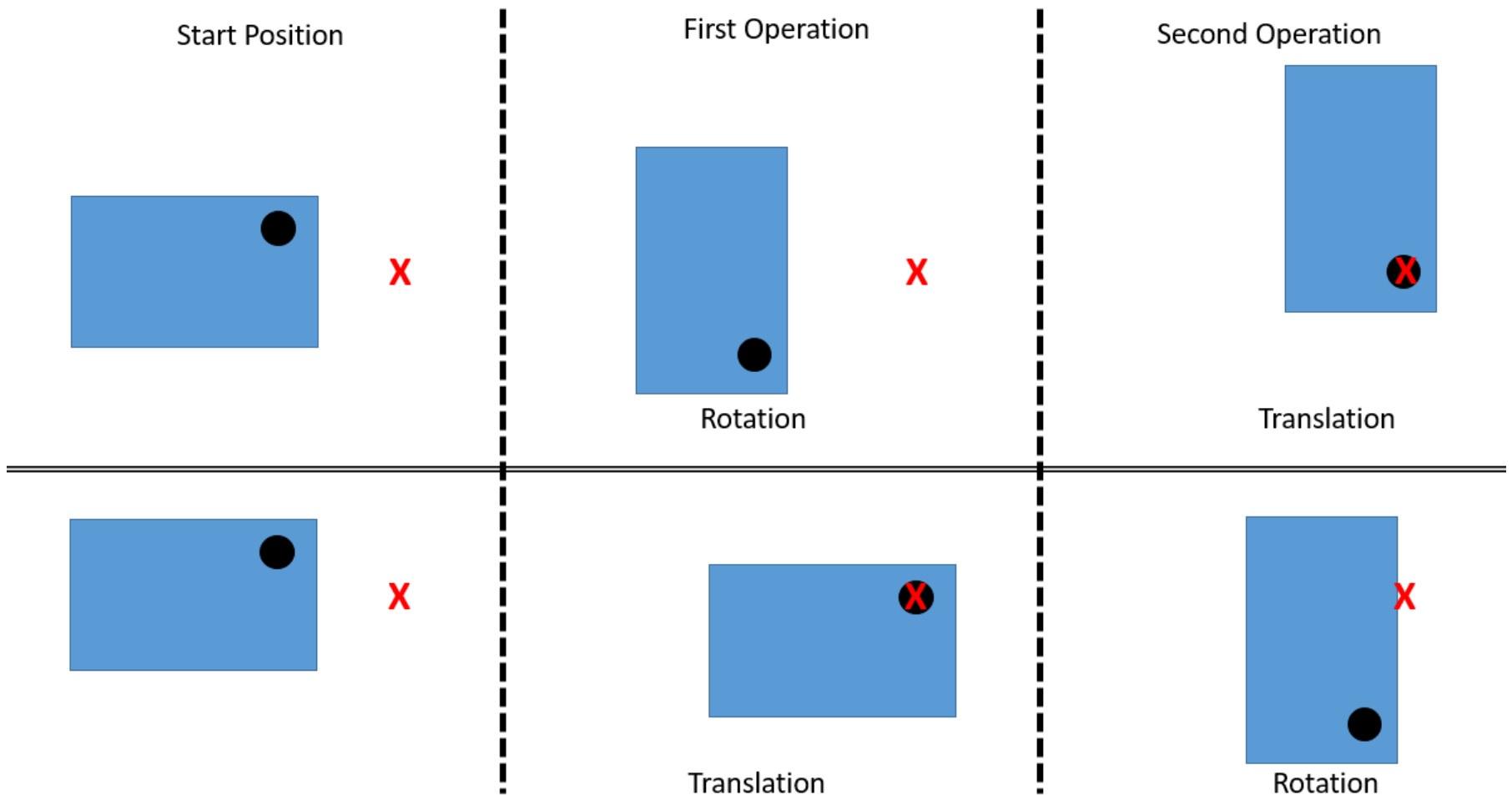
$$\begin{aligned} T(x_f, y_f) \cdot S(s_x, s_y) \cdot T(-x_f, -y_f) = \\ \begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix} = \\ \begin{bmatrix} s_x & 0 & x_f(1-s_x) \\ 0 & s_y & y_f(1-s_y) \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

# 2D Composite Transformations

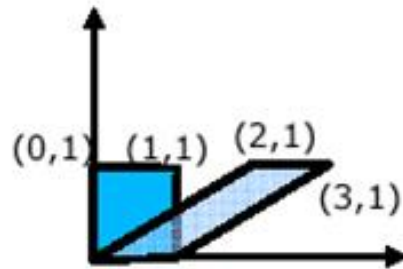
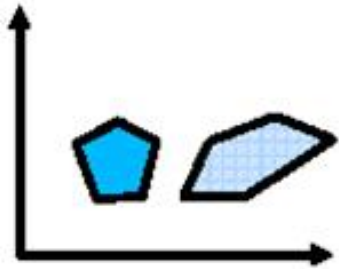
Matrix composition is not commutative!

Be careful when applying a sequence of transformations!

Below: Two different approaches, two results, they differ!



# Shear Transformation



Shearing deforms the shape of an object.

$$x' = x + sh_x \cdot y \quad y' = y$$

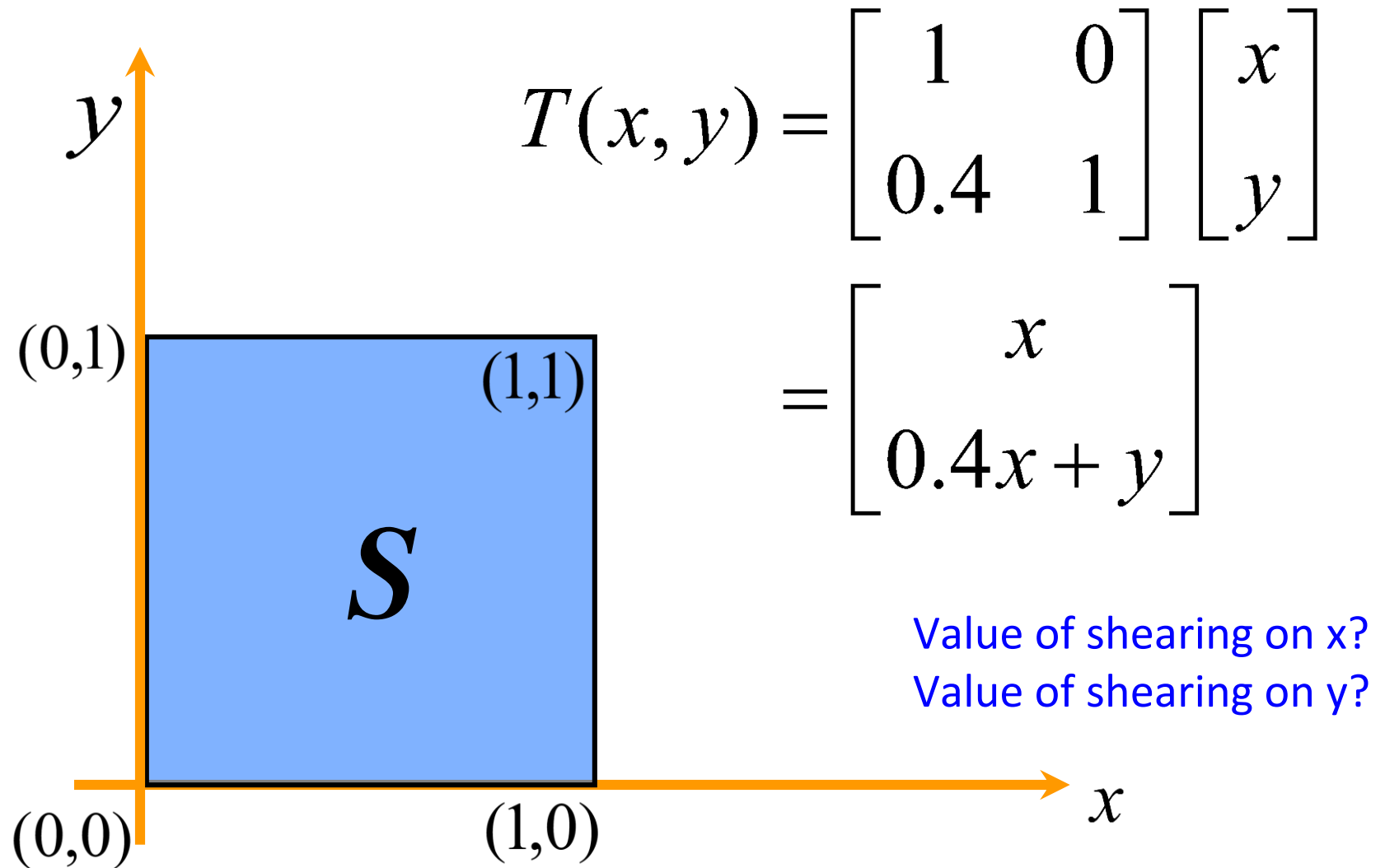
$$\begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Value of shearing on x?

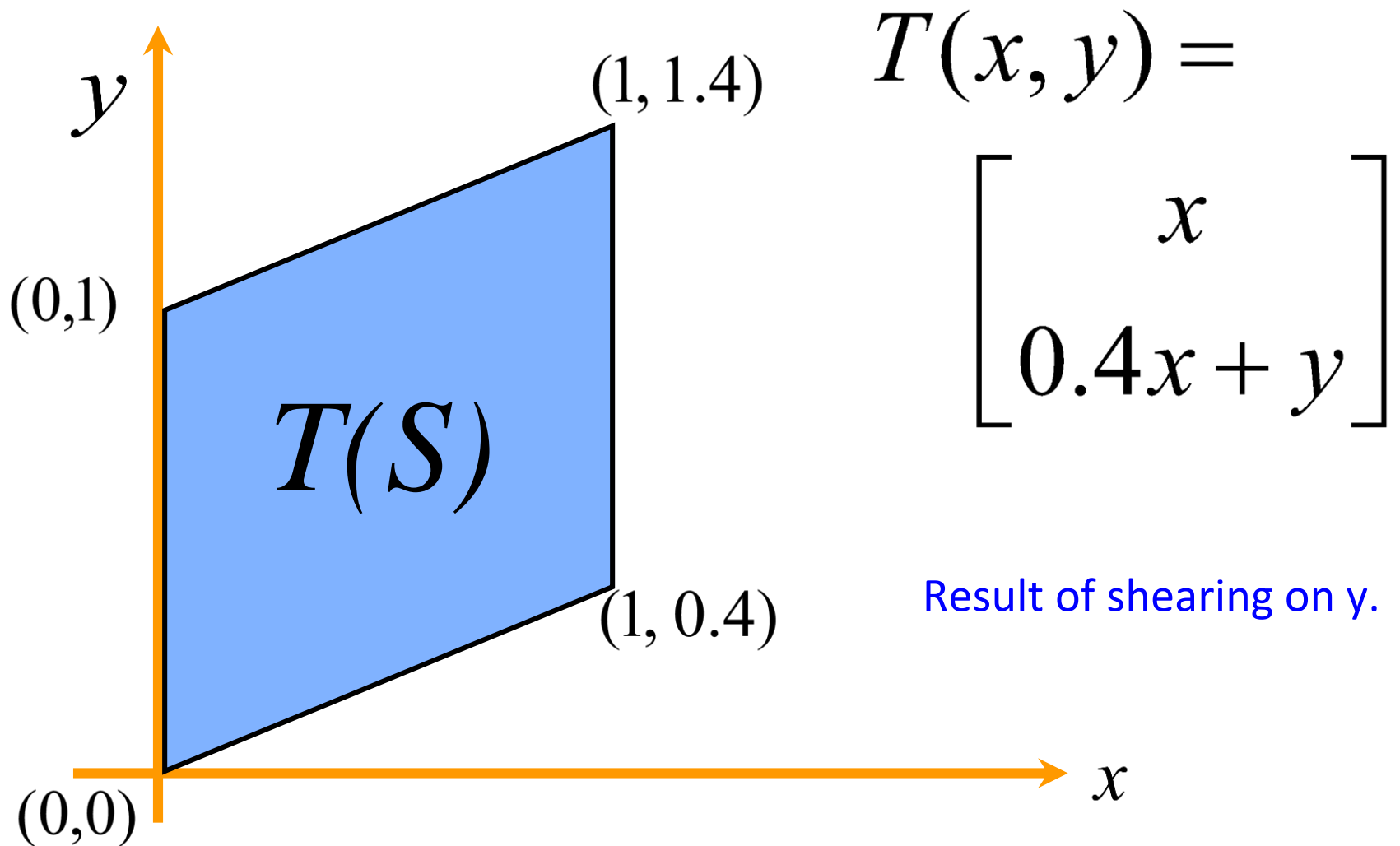
Value of shearing on y?



# Example (not homogeneous)



## Example (not homogeneous)



# Summary (not homogeneous)

The transformation matrices and the results of the products.

Upper side shear on x and at the bottom shear on y.


$$Sh_x = \begin{bmatrix} 1 & sh_x \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x + sh_x y \\ y \end{bmatrix}$$

$$Sh_y = \begin{bmatrix} 1 & 0 \\ sh_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ sh_y \cdot x + y \end{bmatrix}$$

# Double Shear: not commutative!

Double shearing is not commutative: result depends on the order of transformations.

Both changing x and y coordinates at the same time is possible.

$$\begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ b & 1 \end{bmatrix} = \begin{bmatrix} (1+ab) & a \\ b & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ b & 1 \end{bmatrix} \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & a \\ b & (1+ab) \end{bmatrix}$$

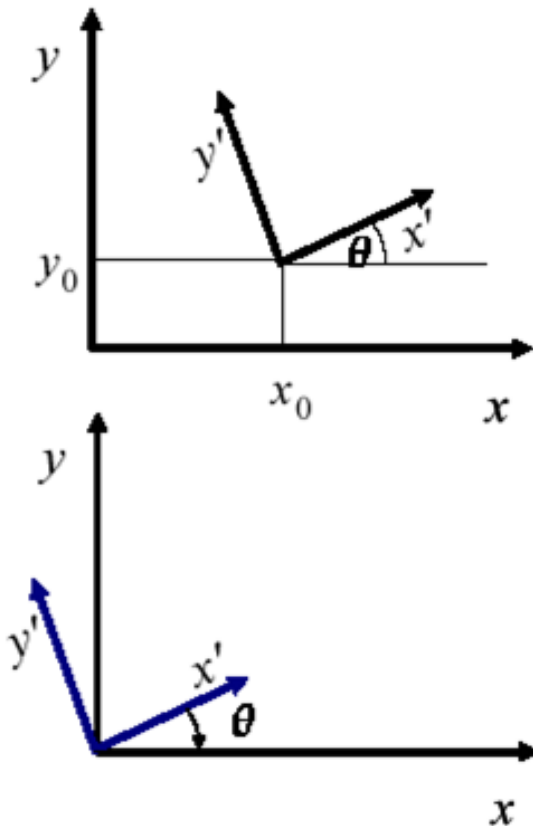
# Transformations Between Coordinate Systems

## \* When used?

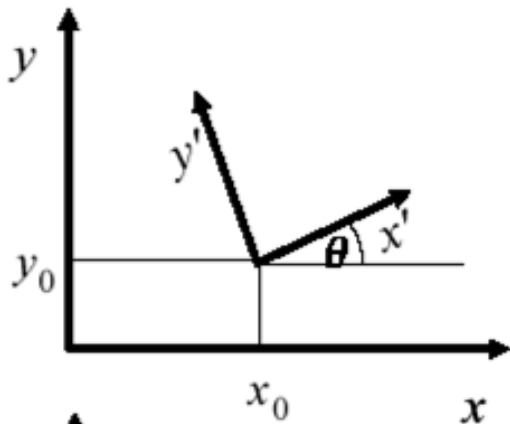
- between different systems, such as from polar to cartesian,
- between two cartesian coordinate systems, such as from window to viewport (from 2D world coordinates to device coordinates).

\* We want to transform from  $(x,y)$  to  $(x',y')$  or superimpose  $(x,y)$  to  $(x',y')$ .

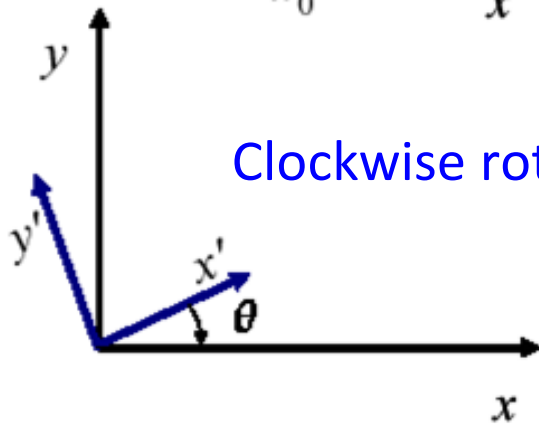
\* Translate to origin and rotate  $x'$  onto  $x$ .



# Transformations Between Coordinate Systems



$$\mathbf{T}(-x_0, -y_0) = \begin{bmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{bmatrix}$$



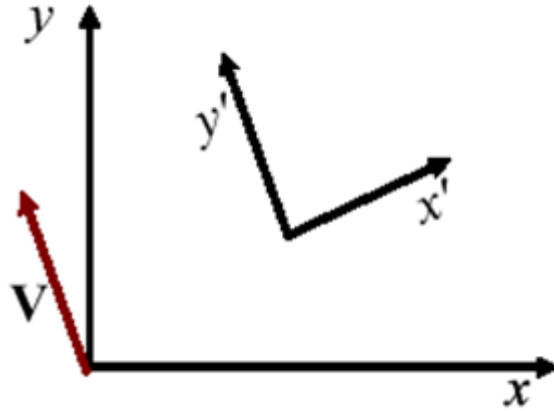
Clockwise rotation, angle is negative

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

$$\mathbf{M}_{xy, x'y'} = \mathbf{R}(-\theta) \cdot \mathbf{T}(-x_0, -y_0)$$

Composite Transformation Matrix

# 2D Rotation: alternative method



- \* First specify a vector  $\mathbf{V}$  for  $y'$
- \* Normalize  $\mathbf{V}$ , it is  $\mathbf{v}$  (unit vector on  $y'$ )
- \* Rotate  $\mathbf{v}$   $90^\circ$  clockwise, it is  $\mathbf{u}$  (unit vector on  $x'$ )

$$\mathbf{v} = \frac{\mathbf{V}}{|\mathbf{V}|} = (v_x, v_y)$$

rotate  $\mathbf{v}$   $90^\circ$

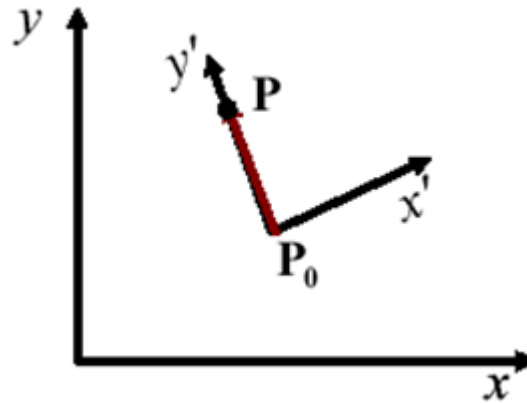
$$\mathbf{u} = (v_y, -v_x) = (u_x, u_y)$$

# 2D Rotation: alternative method

Express the elements of the rotation matrix  $R$ , as elements of a set of orthogonal unit vectors, alternatively.

$$\mathbf{R} = \begin{bmatrix} u_x & u_y & 0 \\ v_x & v_y & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} v_y & -v_x & 0 \\ v_x & v_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

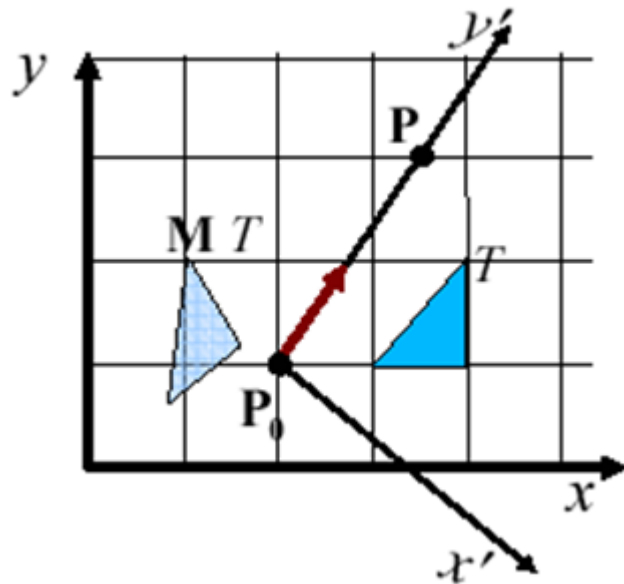
- \* This type of expression can be used for any rotation.
- \* Unit vector  $\mathbf{v}$  calculation is on the right



$$\mathbf{v} = \frac{\mathbf{P} - \mathbf{P}_0}{|\mathbf{P} - \mathbf{P}_0|}$$



# Alternative Method: example



$$\mathbf{P}_0 = (2, 1) \quad \mathbf{P} = (3.5, 3)$$

$$\mathbf{v} = \frac{\mathbf{P} - \mathbf{P}_0}{|\mathbf{P} - \mathbf{P}_0|} = \frac{(1.5, 2)}{\sqrt{1.5^2 + 2^2}} = \left( \frac{1.5}{2.5}, \frac{2}{2.5} \right) = (0.6, 0.8)$$

$$\mathbf{u} = (0.8, -0.6) \quad (u_x = v_y, u_y = -v_x)$$

$$\mathbf{T} = \begin{bmatrix} 3 & 4 & 4 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Assume that there is an object T

$$\mathbf{M} = \mathbf{R}(\mathbf{u}, \mathbf{v}) \cdot \mathbf{T}(-2, -1) =$$

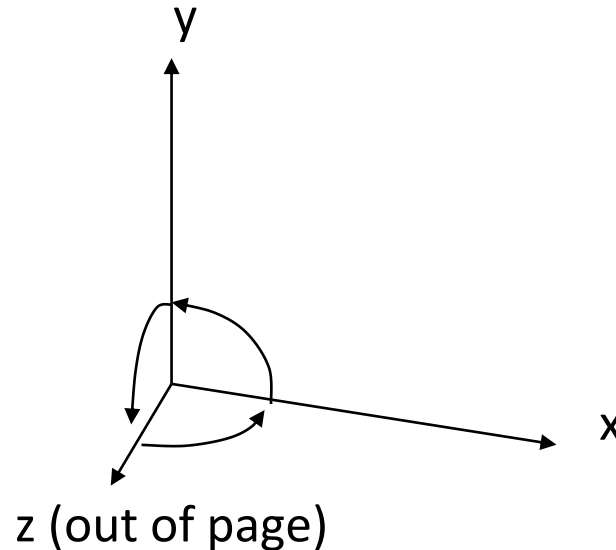
$$\begin{bmatrix} 0.8 & -0.6 & 0 \\ 0.6 & 0.8 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.8 & -0.6 & -1 \\ 0.6 & 0.8 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{M} \cdot \mathbf{T} = \begin{bmatrix} 0.8 & 1 & 1.6 \\ 0.6 & 2 & 1.2 \\ 1 & 1 & 1 \end{bmatrix}$$

Composite Transformation Matrix

Result of applying this transformation to T

# 3D Transformations



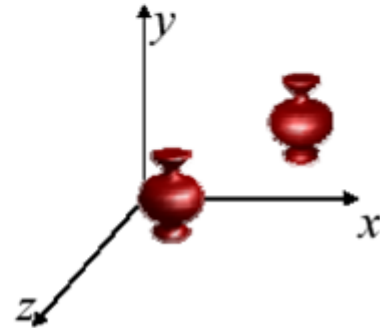
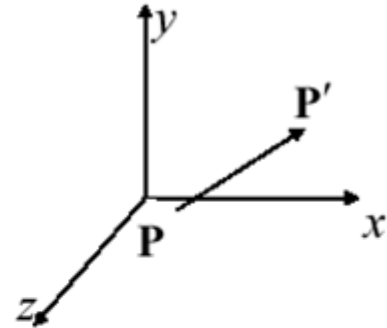
- Homogeneous coordinates are used, just as in 2D case.
- Transformations are now 4 by 4 matrices.
- Use a right-handed coordinate system (positive x to the right, positive y upwards, and positive z out of page, towards us).

# 3D Translation

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\mathbf{P}' = \mathbf{T} \cdot \mathbf{P}$$

Move from P to P'



# 3D Scaling

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

4 by 4 3D scaling matrix

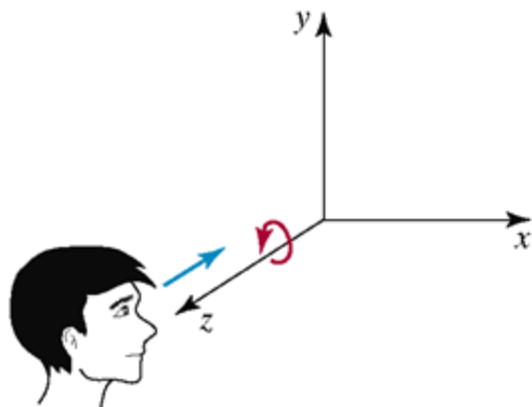
# 3D Rotation

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

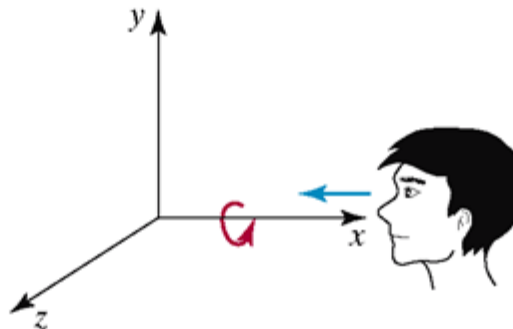
- \* In 2D, we were rotating around the origin.
- \* This corresponds to rotating around z axis, if we think in 3D (The axis we are rotating around is important).

# 3D Rotation

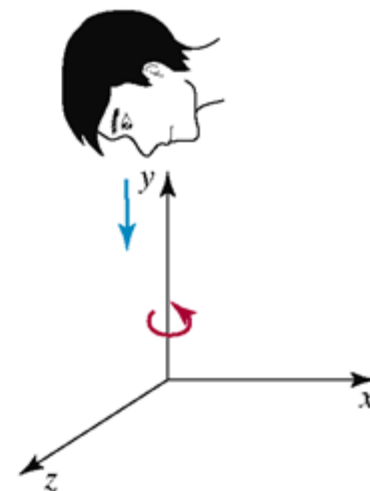
Rotations around z, x, and y axes respectively



(a)

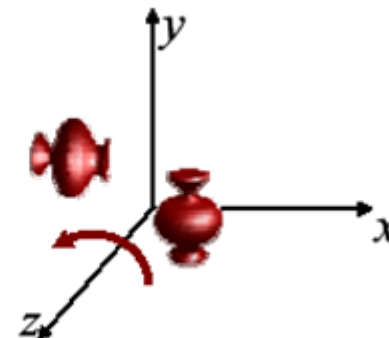
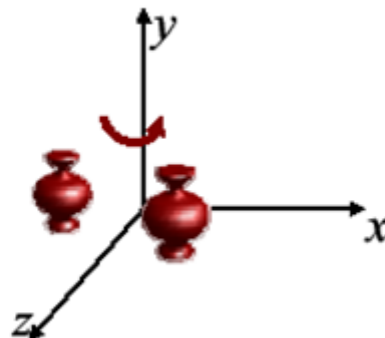
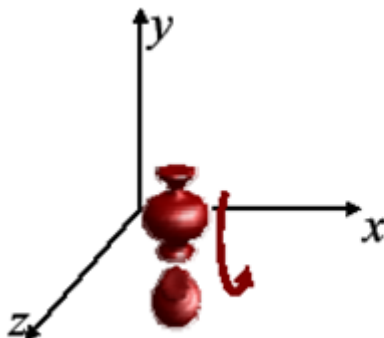


(b)



(c)

Rotations around x, y, and z axes respectively, positive rotations are counterclockwise



# 3D Rotation

• Around  $x$

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

• Around  $y$

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

• Around  $z$

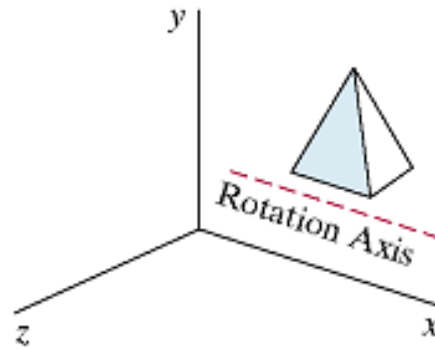
$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{P}' = \mathbf{R}_z(\theta) \cdot \mathbf{P}$$

- Rotation matrices around 3 coordinate axes.
- Please note that, the column and the row of the axis that we rotate the object around, does not include factors with angle values.
- And these columns and rows have value of one, as the first, second and third items for  $x$ ,  $y$  and  $z$  axes respectively.

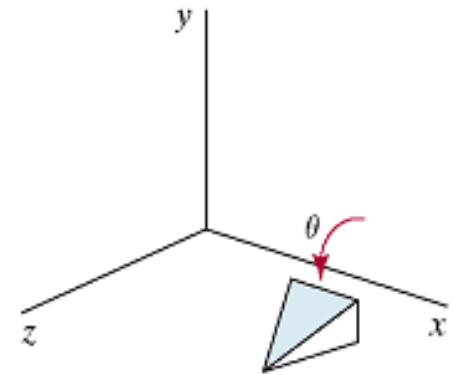
# 3D Rotation

## Rotation around an axis that is parallel to x axis

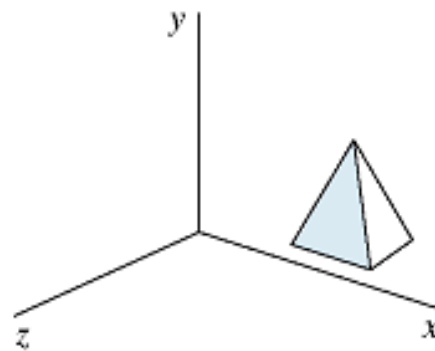
- Same as 2D case
- First translate the object towards the x axis
- Then rotate it around x axis
- Translate it back to the rotation axis



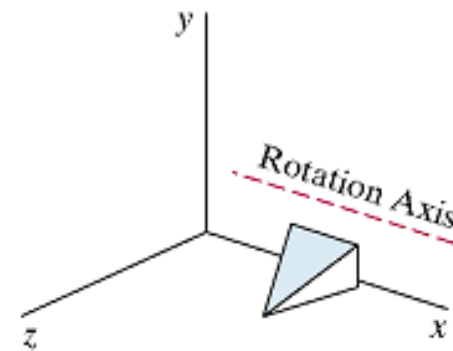
(a)  
Original Position of Object



(c)  
Rotate Object Through Angle  $\theta$



(b)  
Translate Rotation Axis onto x Axis



(d)  
Translate Rotation Axis to Original Position

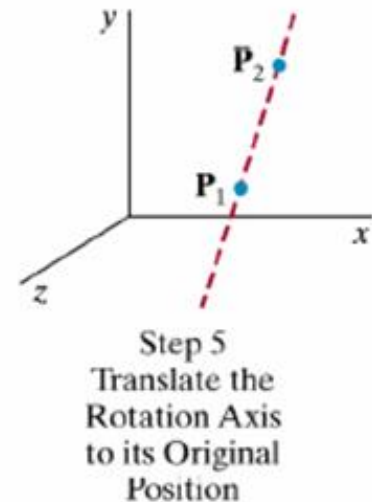
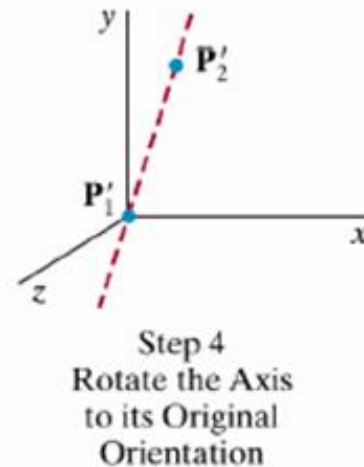
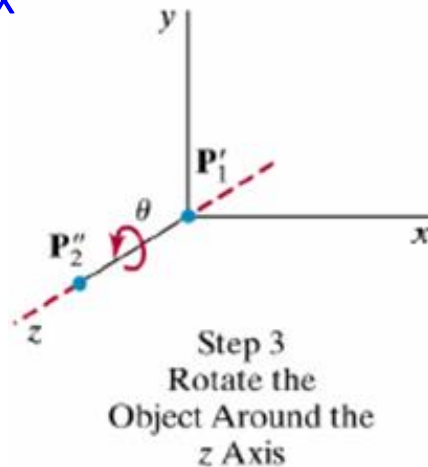
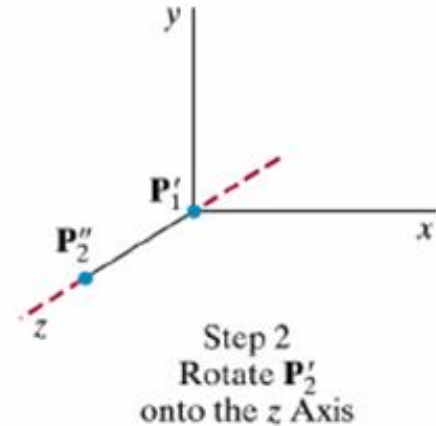
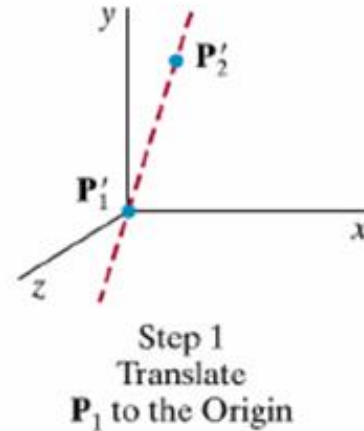
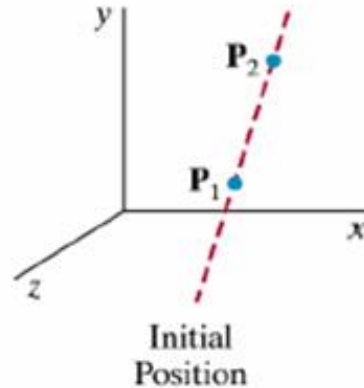
$$\mathbf{P}' = \mathbf{T}^{-1} \cdot \mathbf{R}_x(\theta) \cdot \mathbf{T} \cdot \mathbf{P}$$



# 3D Rotation

## Rotation around an arbitrary axis

- First translate the starting point  $P_1$  to the origin (now  $P_1$  is  $P'_1$ ,  $P_2$  is  $P'_2$ )
- Then rotate  $P'_2$  onto  $z$  axis (rotation around  $x$  and then  $y$  axes)
- Rotate the object around  $z$  axis
- Rotate the axis to its original position
- Translate  $P_1$  to its original position



$$\mathbf{R}(\theta) = \mathbf{T}^{-1} \cdot \mathbf{R}_x^{-1}(\alpha) \cdot \mathbf{R}_y^{-1}(\beta) \cdot \mathbf{R}_z(\theta) \cdot \mathbf{R}_y(\beta) \cdot \mathbf{R}_x(\alpha) \cdot \mathbf{T}$$

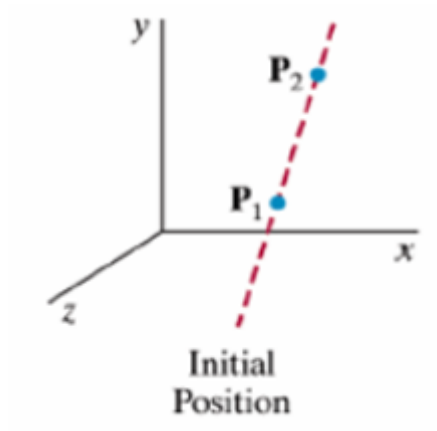
# 3D Rotation

Specify a vector  $V$

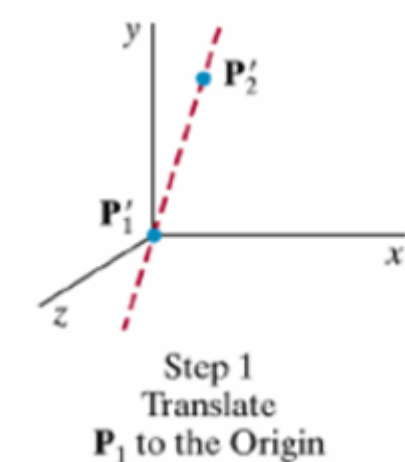
$$\mathbf{V} = \mathbf{P}_2 - \mathbf{P}_1 = (x_2 - x_1, y_2 - y_1, z_2 - z_1)$$

unit vector along  $V$

$$\mathbf{u} = \frac{\mathbf{V}}{|\mathbf{V}|} = (a, b, c)$$



Translate  $P_1$  to the origin using matrix  $T$

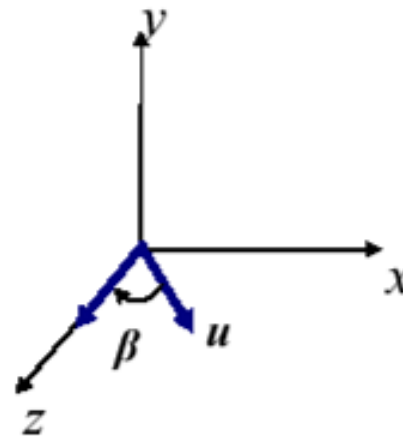
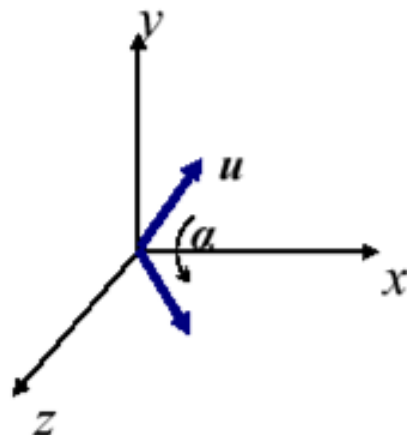


$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

# 3D Rotation

unit vector  $u$  must be aligned with the  $z$  axis:

- Two rotations are needed
- First rotate around  $x$ , get vector  $u$  onto the  $xz$  plane
- Second rotate around  $y$



# 3D Rotation: alternative method of two rotations

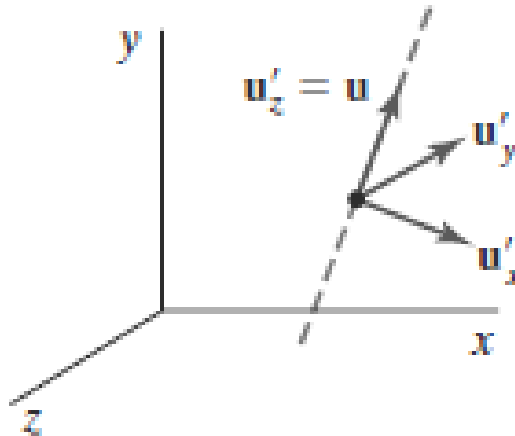
$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Any rotation around origin can be represented by 3 orthogonal unit vectors for 3D case.
- R matrix can be thought of as rotating the  $r_1$ ,  $r_2$ , and  $r_3$  unit vectors onto x, y and z axes.
- We can define an orthogonal coordinate system and form this R matrix.

$$\mathbf{u}'_z = \mathbf{u}$$

$$\mathbf{u}'_y = \frac{\mathbf{u} \times \mathbf{u}_x}{|\mathbf{u} \times \mathbf{u}_x|}$$

$$\mathbf{u}'_x = \mathbf{u}'_y \times \mathbf{u}'_z$$



We defined a new coordinate system.

# 3D Rotation: alternative method of two rotations

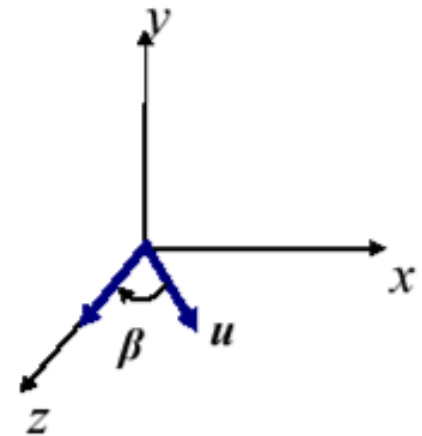
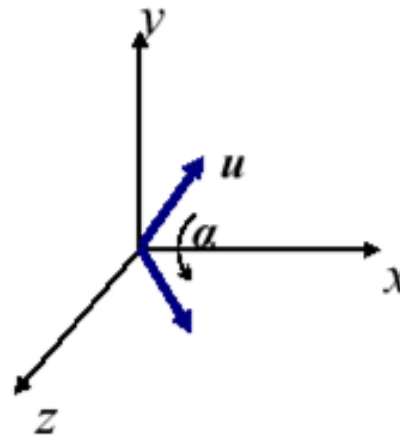
$$\mathbf{u}'_z = \mathbf{u} = (a, b, c)$$

$$d = \sqrt{b^2 + c^2}$$

$$\mathbf{u}'_y = \frac{\mathbf{u} \times \mathbf{u}_x}{|\mathbf{u} \times \mathbf{u}_x|} = \frac{(a, b, c) \times (1, 0, 0)}{|\mathbf{u} \times \mathbf{u}_x|} = \frac{(0, c, -b)}{\sqrt{b^2 + c^2}} = (0, c/d, -b/d)$$

$$\mathbf{u}'_x = \mathbf{u}'_y \times \mathbf{u}'_z = (0, c/d, -b/d) \times (a, b, c) = (d, -a \cdot b/d, -a \cdot c/d)$$

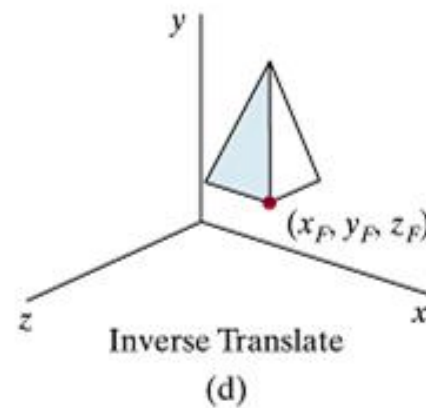
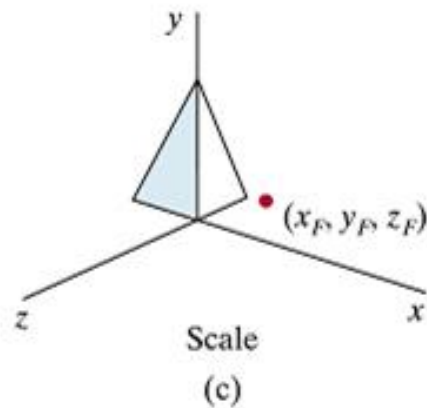
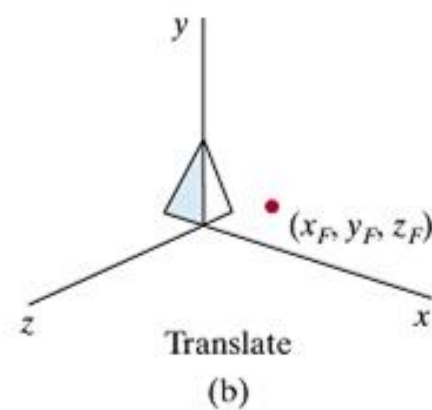
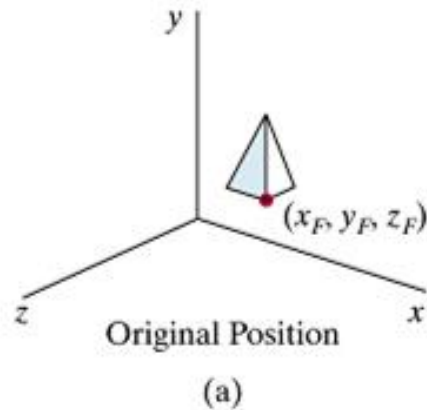
$$\mathbf{R} = \begin{bmatrix} d & \frac{-a \cdot b}{d} & \frac{-a \cdot c}{d} & 0 \\ 0 & \frac{c}{d} & \frac{-b}{d} & 0 \\ a & b & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Please check that if  $\mathbf{R}$  (obtained by using new unit vectors) is equal to these two rotations or not!

$$R_y(\beta) \cdot R_x(\alpha)$$

# Scaling w.r.t. A Fixed Point



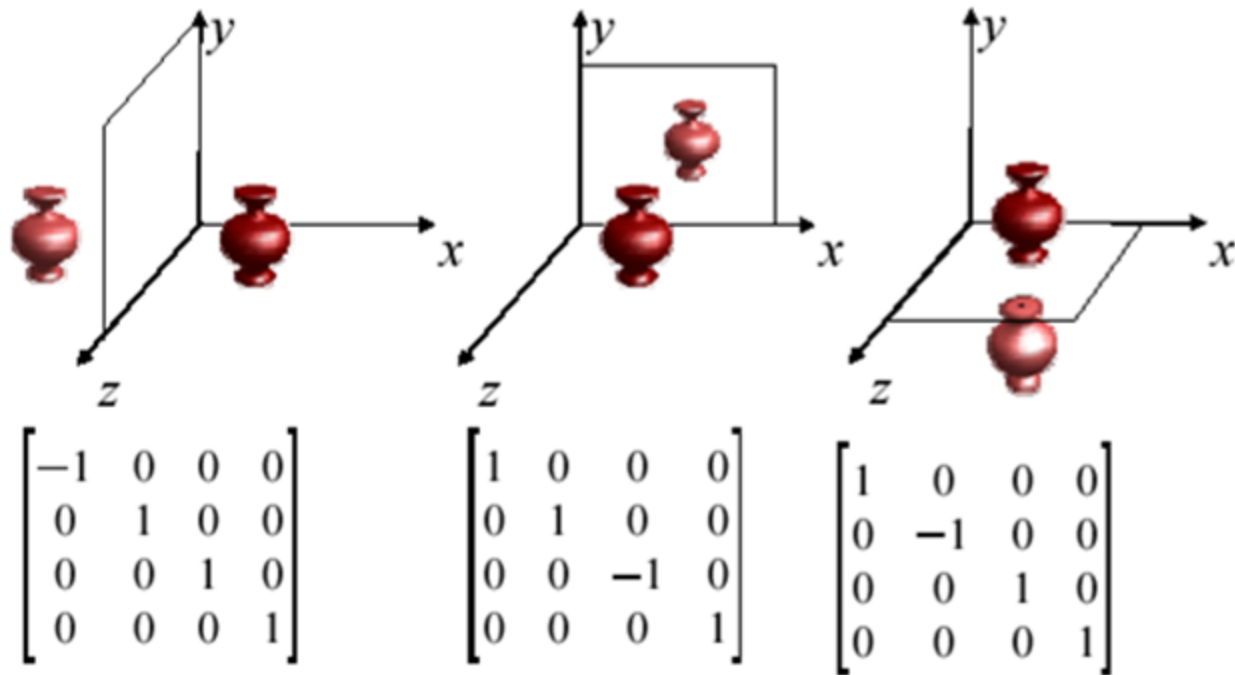
Translation to origin, scaling with reference to origin and then translating (the fixed point) back to the original position.

# Scaling w.r.t. A Fixed Point

$$\begin{aligned}
 \mathbf{T}(x_f, y_f, z_f) \cdot \mathbf{S} \cdot \mathbf{T}(-x_f, -y_f, -z_f) &= \begin{bmatrix} 1 & 0 & 0 & x_f \\ 0 & 1 & 0 & y_f \\ 0 & 0 & 1 & z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_f \\ 0 & 1 & 0 & -y_f \\ 0 & 0 & 1 & -z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} 1 & 0 & 0 & x_f \\ 0 & 1 & 0 & y_f \\ 0 & 0 & 1 & z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 & 0 & -s_x x_f \\ 0 & s_y & 0 & -s_y y_f \\ 0 & 0 & s_z & -s_z z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 &= \mathbf{T}(x_f, y_f, z_f) \cdot \mathbf{S} \cdot \mathbf{T}(-x_f, -y_f, -z_f) = \begin{bmatrix} s_x & 0 & 0 & x_f(1-s_x) \\ 0 & s_y & 0 & y_f(1-s_y) \\ 0 & 0 & s_z & z_f(1-s_z) \\ 0 & 0 & 0 & 1 \end{bmatrix}
 \end{aligned}$$

Composite Transformation Matrix

# 3D Reflection



3D reflection over yz plane, xy plane, xz plane



# 3D Shear

$$SH_z = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$x$  and  $y$  value depends on  $z$  value of the shape

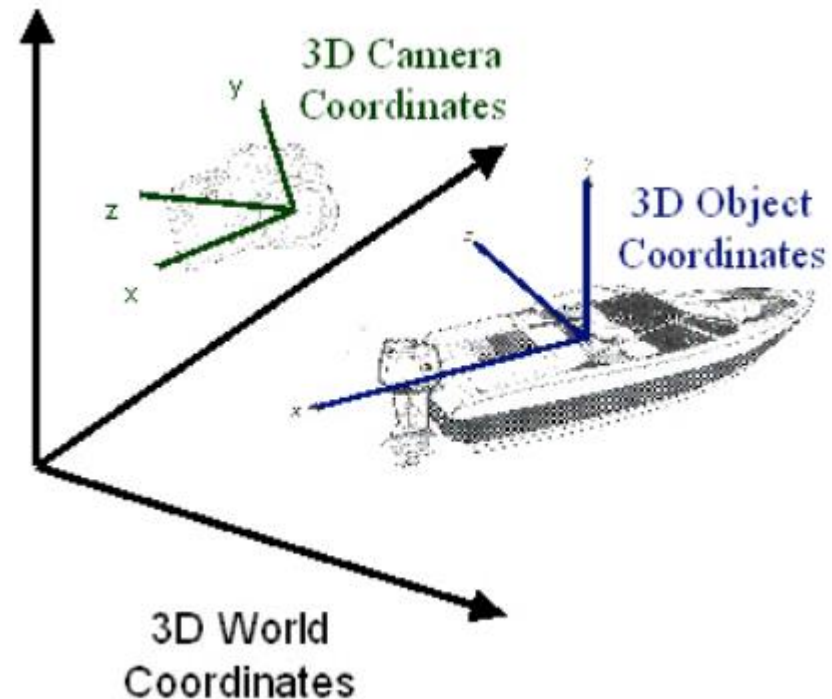
$$SH_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ a & 1 & 0 & 0 \\ b & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$y$  and  $z$  value depends on  $x$  value of the shape

3D shearing, deforming the shape of the object depending on another dimension

# Common Coordinate Systems

- Modelling/object coordinates
- World/scene coordinates
- Camera/eye/viewing coordinates
- Clip/clipping coordinates
- Normalized device coordinates
- Window/screen coordinates

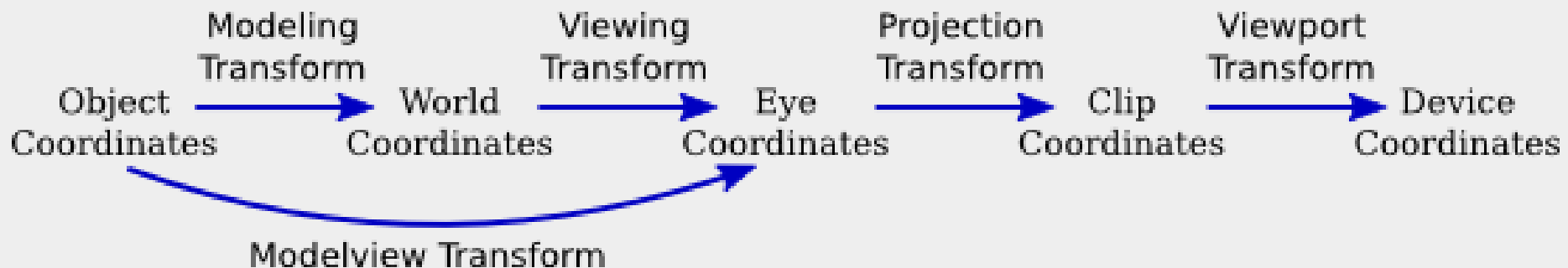


These six were used in fixed-function pipeline.

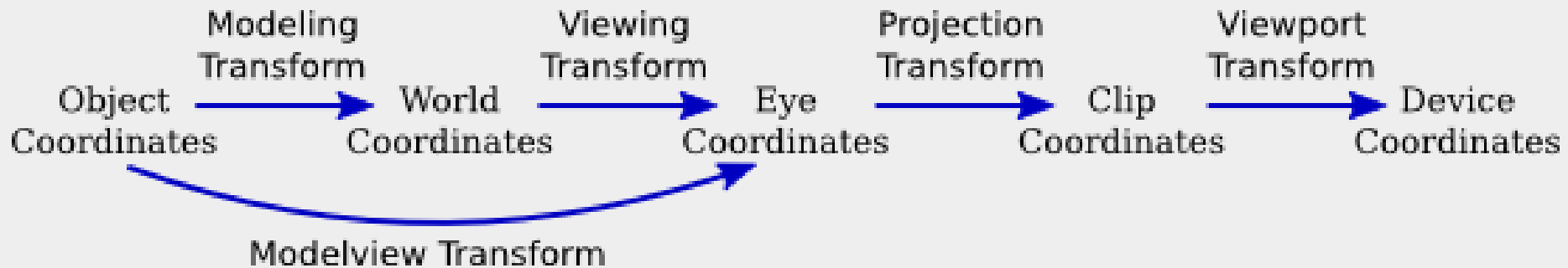
With programmable shaders, we have greater flexibility (We can add other systems or avoid using some traditional ones).

# Model-View Transformation

- Changing from one coordinate system to another (can be done with a 4 by 4 matrix)
- Such matrix can represent change from modelling coordinates to world coordinates
- Another matrix can represent change from world coordinates to camera coordinates
- **Concatenation: Model-view transformation, model-view matrix**



# Projection Transformation



- After modelview transform, objects are in camera (eye) coordinates.
- APIs check whether they lie within the volume: Projection Transformation: Brings all potentially visible objects into a cube centered at the origin in clip coordinates.