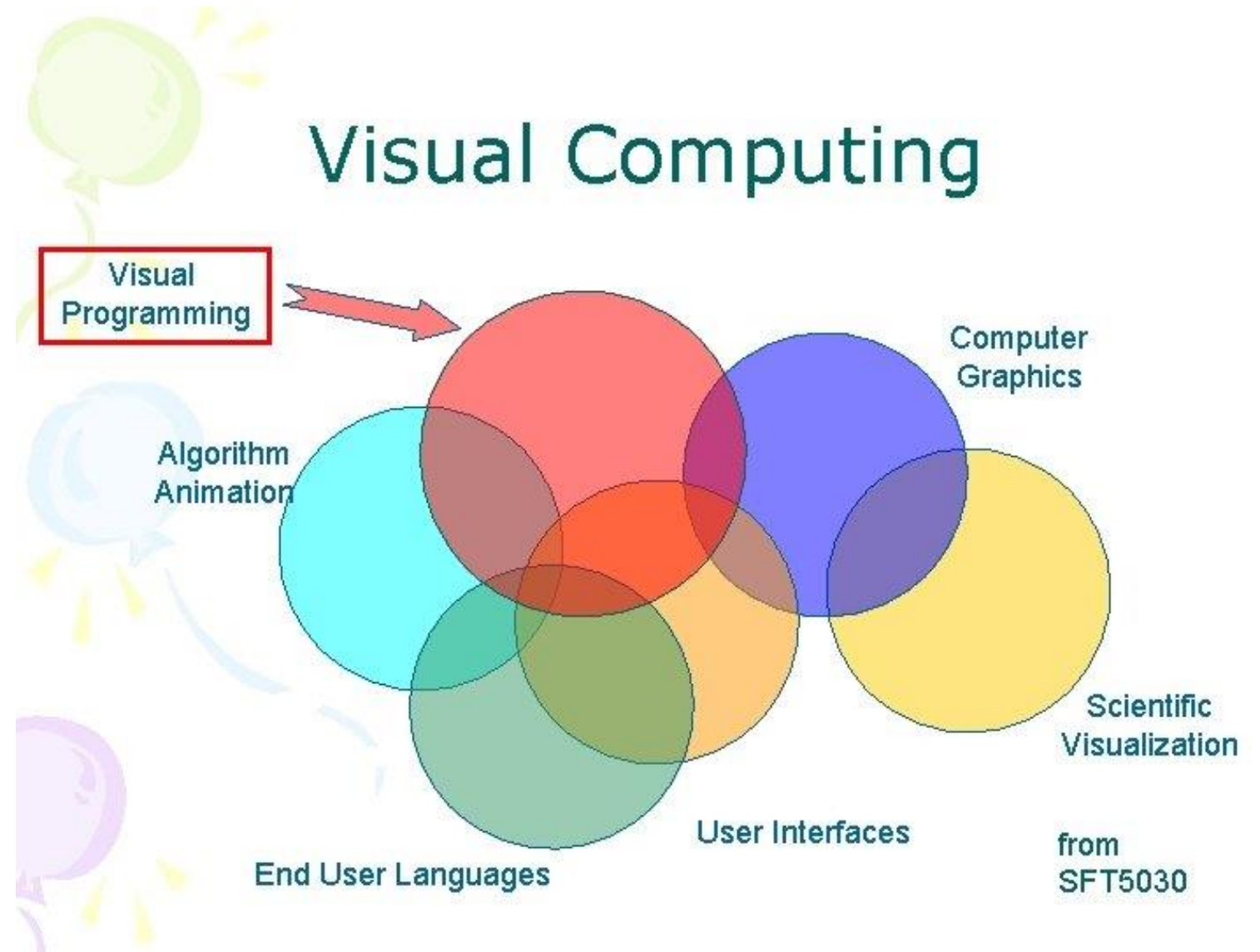


# CSPE 51 – Augmented and Virtual Reality

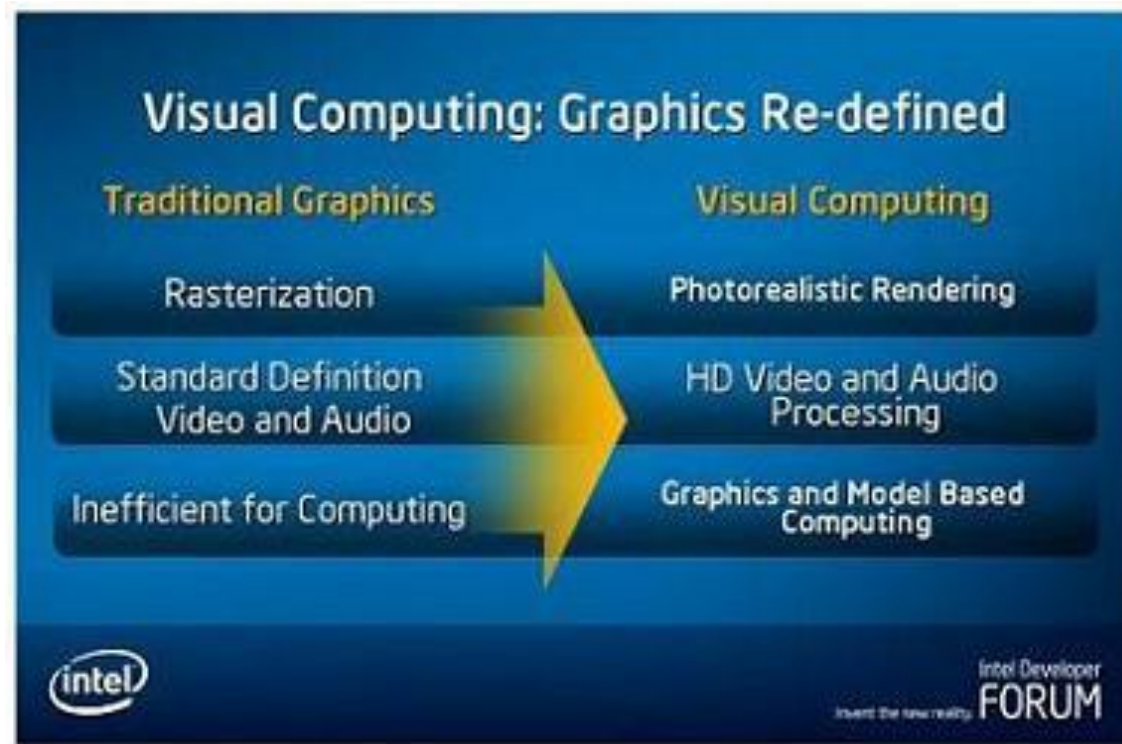
## UNIT - II



## UNIT II :

Visual Computation in Virtual Reality: Fundamentals of Computer Graphics - Software and Hardware Technology on Stereoscopic Display  
- Advanced Techniques in CG: Management of Large Scale Environments & Real Time Rendering.

# Visual Computation in Virtual Reality



# Fundamentals of Computer Graphics

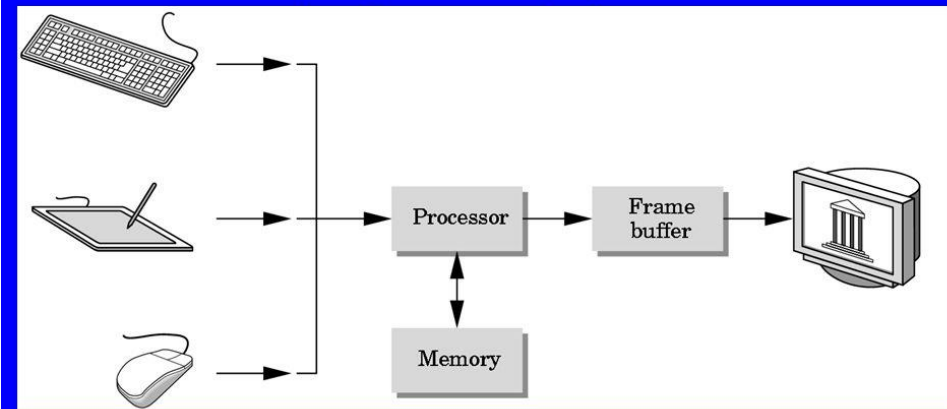
- Components of CG
- Modeling, Storing, Manipulation, Viewing and Rendering
- Graphic system

## Components of an Interactive Graphics System

- Graphics Hardware
  - Graphics Input and Storage devices
  - Graphics Display devices
- Graphics Software
  - General Programming packages
  - Special-purpose applications packages

## Graphics Systems

Five major elements - processor, memory, frame buffer, output devices, input devices





# Input Devices

- ▶ Keyboards, Button Boxes and Dials
- ▶ Mouse Devices
- ▶ Trackballs and Spaceballs
- ▶ Joysticks
- ▶ Data Gloves
- ▶ Digitizers
- ▶ Image Scanners
- ▶ Touch Panels



Activate Windows  
Go to Settings to activate Windows.

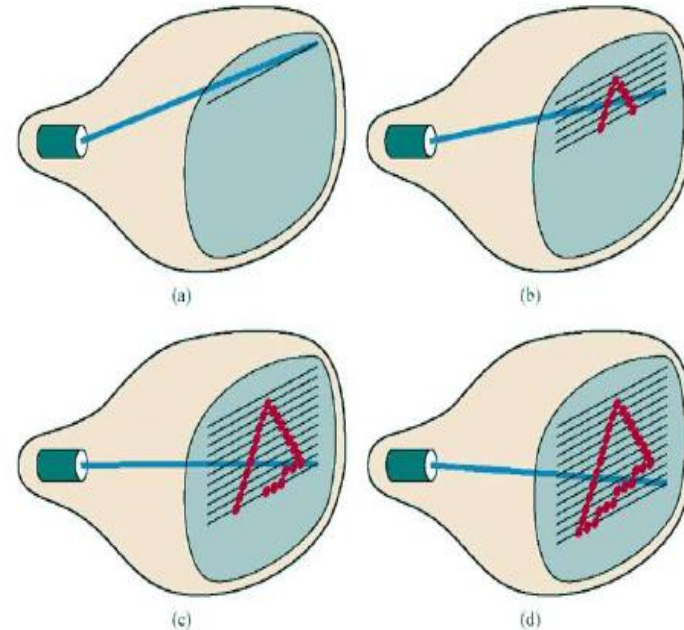
# TYPES OF OUTPUT DEVICES

- Display devices
- Printers
- Speakers ,head phone ,ear buds
- Other output devices

# Raster and Random display

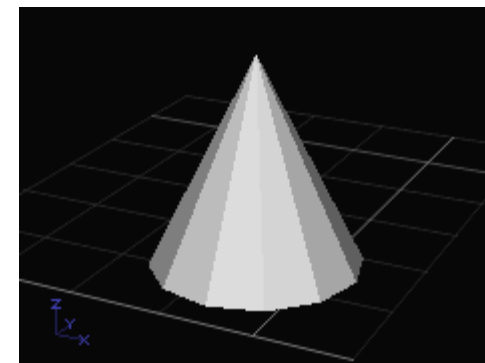
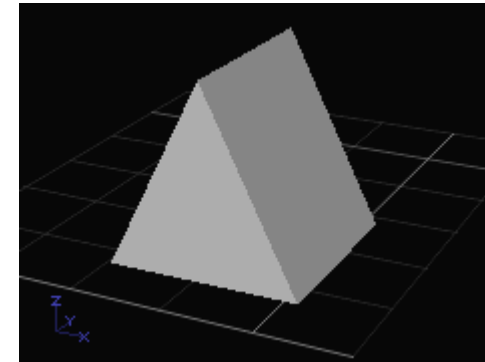
Based on television technology.

- The electron beam is swept across the screen, one row at a time, from top to bottom.
- Each row is called a scan line.
- Picture definition is stored in a memory called refresh buffer or frame buffer.
- Each screen spot is referred as a pixel or pel.
- Refreshed at 24 frames per second (fps) or higher to remove flickering



# Modeling – Output primitives

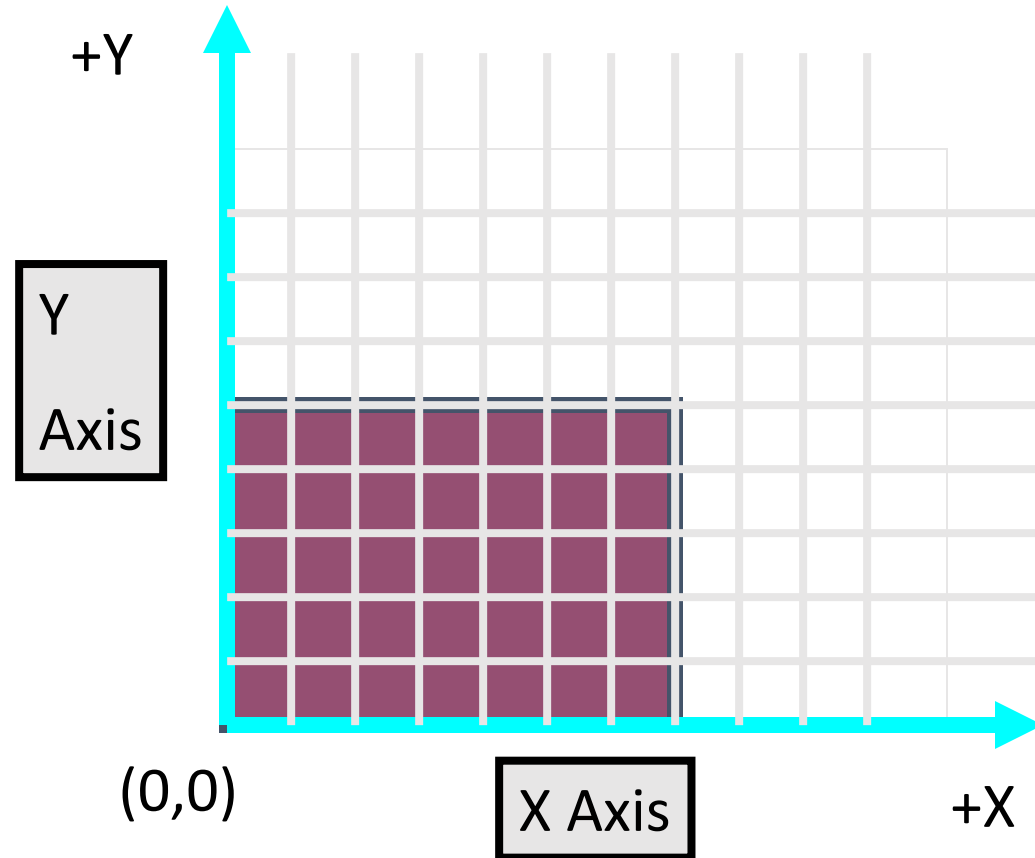
- The basic objects out of which a graphics display is created
- Describes the geometry of objects and – typically referred to as *geometric primitives*.
- **Examples:** point, line, text, filled region, images, quadric surfaces, spline curves
- Each of the output primitives has its own set of *attributes*.





# Two Dimensional Images

- Use **Cartesian coordinates**
  - X (horizontal)
  - Y (vertical)
- Origin is in the lower left
- the image seen on a screen space is called the **world coordinate system**



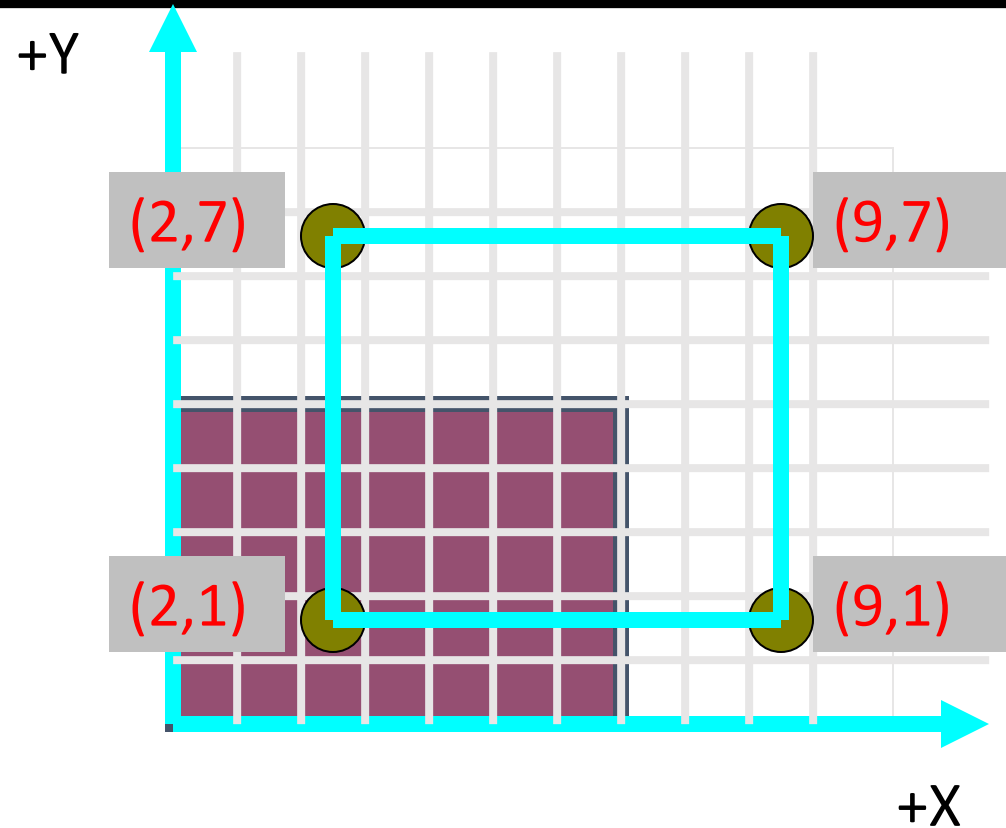
# Partition the space into pixels

**Screen Coordinates** – references to frame buffer locations

1. Define a set of points (vertices) in 2D space.
2. Given a set of vertices, draw lines between consecutive vertices.

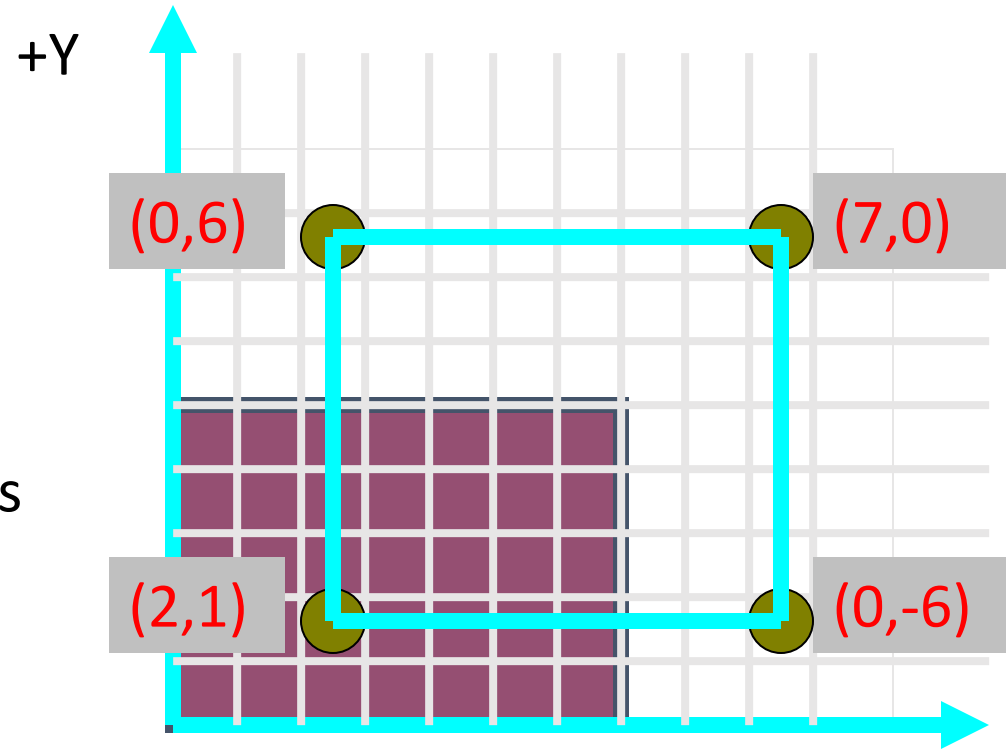
## Pixel

- `?=glSetPixel(?)`
- `?=glGetPixel(?)`
- *Scan line number –  $y$*
- *Column number –  $x$*

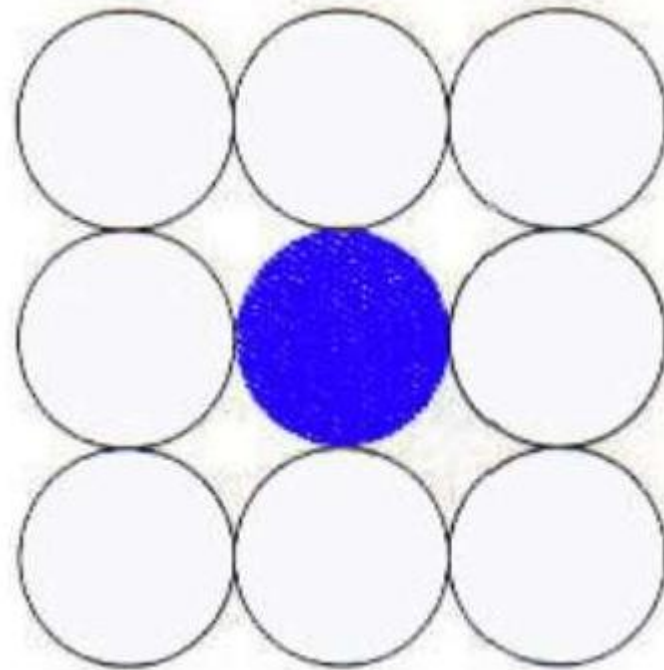
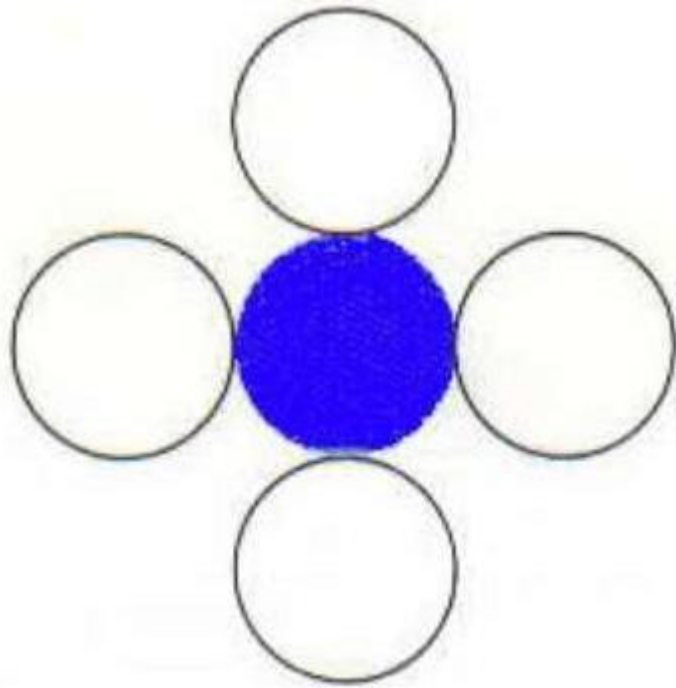


# Absolute and Relative Coordinate Specifications

- **Absolute coordinates** – location specified as a relationship to the origin
- **Relative coordinates** – location specified as a relationship to other points



## 4, 8 – Connectors - Filling



# Manipulation

- Translation
- Rotation
- Scaling
- Reflection
- Shearing



# Outline

- ▶ Basic 2D Transformations (rigid-body transformations):
  - ▶ Translation
  - ▶ Rotation
  - ▶ Scaling
- ▶ Homogenous Representations and Coordinates
- ▶ 2D Composite Transformations
- ▶ Other Transformations:
  - ▶ Reflection
  - ▶ Shearing

# Geometric Transformations

15

- ▶ Sometimes also called modeling transformations
  - ▶ Geometric transformations: Changing an object's position (translation), orientation (rotation) or size (scaling)
  - ▶ Modeling transformations: Constructing a scene or hierarchical description of a complex object
- ▶ Others transformations: reflection and shearing operations

# 1. Translation

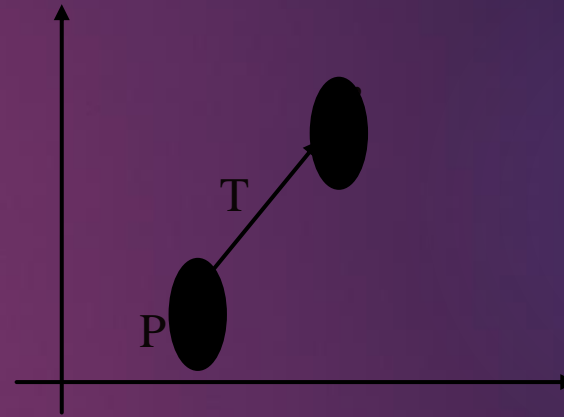
- ▶ 2D Translation

- ▶  $x' = x + t_x, y' = y + t_y$

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

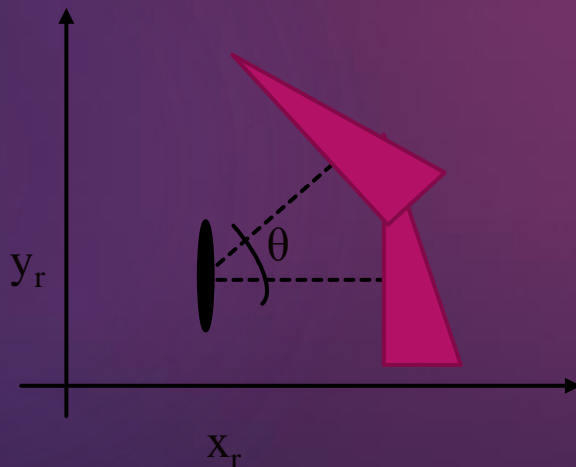
- ▶  $P' = P + T$

- ▶ Translation moves the object without deformation (rigid-body transformation)



## 2. Rotation

- ▶ 2D Rotation
  - ▶ Rotation axis
  - ▶ Rotation angle
  - ▶ rotation point or pivot point  $(x_r, y_r)$



Note:

If  $\theta$  is positive  $\rightarrow$  counterclockwise

If  $\theta$  is negative  $\rightarrow$  clockwise  
rotation

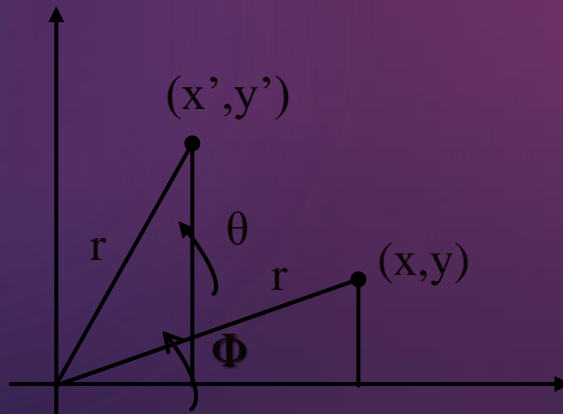
Remember:

$$\cos(a + b) = \cos a \cos b - \sin a \sin b$$

$$\cos(a - b) = \cos a \sin b + \sin a \cos b$$

# Rotation

- ▶ At first, suppose the pivot point is at the origin
- ▶  $x' = r \cos(\theta + \Phi) = r \cos \theta \cos \Phi - r \sin \theta \sin \Phi$   
 $y' = r \sin(\theta + \Phi) = r \cos \theta \sin \Phi + r \sin \theta \cos \Phi$
- ▶  $x = r \cos \Phi, y = r \sin \Phi$
- ▶  $x' = x \cos \theta - y \sin \theta$   
 $y' = x \sin \theta + y \cos \theta$



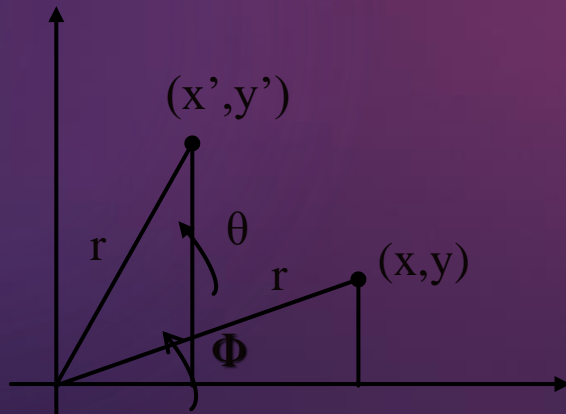


## ► 2D Rotation

►  $P' = R \cdot P$

*Handwritten red notes:*  
A red circle is drawn around the  $P$  in  $P' = R \cdot P$ .  
A red arrow points from the circled  $P$  to the  $P$  in  $P = R \cdot P'$ .

$$R = \begin{bmatrix} \cos \Theta & -\sin \Theta \\ \sin \Theta & \cos \Theta \end{bmatrix}$$



## ► 2D Rotation

- Rotation of a point about any specified position  $(x_r, y_r)$

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

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

- Rotations also move objects without deformation
- A line is rotated by applying the rotation formula to each of the endpoints and redrawing the line between the new endpoints
- A polygon is rotated by applying the rotation formula to each of the vertices and redrawing the polygon using new vertex coordinates

# 3. Scaling

- ▶ Scaling is used to alter the size of an object
- ▶ Simple 2D scaling is performed by multiplying object positions  $(x, y)$  by scaling factors  $s_x$  and  $s_y$

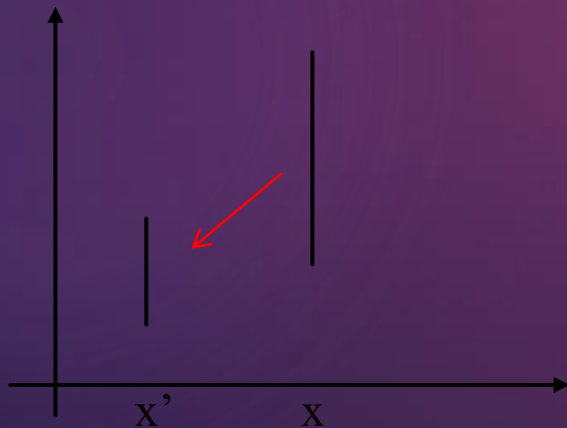
$$x' = x \cdot s_x$$

$$y' = y \cdot s_y$$

$$\text{or } P' = S \cdot P$$

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

- ▶ Any positive value can be used as scaling factor
  - ▶ Values less than 1 reduce the size of the object
  - ▶ Values greater than 1 enlarge the object
  - ▶ If scaling factor is 1 then the object stays unchanged
  - ▶ If  $s_x = s_y$ , we call it uniform scaling
  - ▶ If scaling factor  $< 1$ , then the object moves closer to the origin and  
If scaling factor  $> 1$ , then the object moves farther from the origin



- ▶ We can control the location of the scaled object by choosing a position called the **fixed point**  $(x_f, y_f)$

$$x' - x_f = (x - x_f) s_x \quad y' - y_f = (y - y_f) s_y$$

$$x' = x \cdot s_x + x_f (1 - s_x)$$


$$y' = y \cdot s_y + y_f (1 - s_y)$$

- ▶ Polygons are scaled by applying the above formula to each vertex, then regenerating the polygon using the transformed vertices



# Matrix Representations and Homogeneous Coordinates

- ▶ Many graphics applications involve sequences of geometric transformations
  - ▶ Animations
  - ▶ Design and picture construction applications
- ▶ We will now consider matrix representations of these operations
  - ▶ Sequences of transformations can be efficiently processed using matrices

- 
- ▶  $P' = M_1 \cdot P + M_2$ 
    - ▶ P and P' are column vectors
    - ▶  $M_1$  is a 2 by 2 array containing multiplicative factors
    - ▶  $M_2$  is a 2 element column matrix containing translational terms
    - ▶ For translation  $M_1$  is the identity matrix
    - ▶ For rotation or scaling,  $M_2$  contains the translational terms associated with the pivot point or scaling fixed point

- ▶ To produce a sequence of operations, such as scaling followed by rotation then translation, we could calculate the transformed coordinates one step at a time
- ▶ A more efficient approach is to combine transformations, without calculating intermediate coordinate values

- ▶ Multiplicative and translational terms for a 2D geometric transformation can be combined into a single matrix if we expand the representations to 3 by 3 matrices
  - ▶ We can use the third column for translation terms, and all transformation equations can be expressed as matrix multiplications

- ▶ Expand each 2D coordinate  $(x,y)$  to three element representation  $(x_h, y_h, h)$  called **homogeneous coordinates**
- ▶  $h$  is the **homogeneous parameter** such that  $x = x_h/h, \quad y = y_h/h,$
- ▶  $\rightarrow$  infinite homogeneous representations for a point
- ▶ A convenient choice is to choose  $h = 1$



► 2D Translation Matrix

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

or,  $\mathbf{P}' = \mathbf{T}(t_x, t_y) \cdot \mathbf{P}$

► 2D Rotation Matrix

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

or,  $\mathbf{P}' = \mathbf{R}(\Theta) \cdot \mathbf{P}$

► 2D Scaling Matrix

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

or,  $\mathbf{P}' = \mathbf{S}(s_x, s_y) \cdot \mathbf{P}$

# Inverse Transformations

- ▶ 2D Inverse Translation Matrix

$$T^{-1} = \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix}$$

- ▶ By the way:

$$T^{-1} * T = I$$

► 2D Inverse Rotation Matrix

$$R^{-1} = \begin{bmatrix} \cos \Theta & \sin \Theta & 0 \\ -\sin \Theta & \cos \Theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

► And also:

$$R^{-1} * R = I$$

► 2D Inverse Rotation Matrix:

► If  $\theta$  is negative  $\rightarrow$  clockwise

► In

$$R^{-1} * R = I$$

► Only sine function is affected

► Therefore we can say

$$R^{-1} = R^T$$

► 2D Inverse Scaling Matrix

► Of course:

$$S^{-1} = \begin{bmatrix} \frac{1}{s_x} & 0 & 0 \\ 0 & \frac{1}{s_y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S^{-1} * S = I$$

- ▶ We can setup a sequence of transformations as a **composite transformation matrix** by calculating the product of the individual transformations

- ▶  $P' = M_2 \cdot M_1 \cdot P$   
 $= M \cdot P$





## ► Composite 2D Translations

- If two successive translation are applied to a point P, then the final transformed location P' is calculated as

$$\mathbf{P}' = \mathbf{T}(t_{x_2}, t_{y_2}) \cdot \mathbf{T}(t_{x_1}, t_{y_1}) \cdot \mathbf{P} = \mathbf{T}(t_{x_1} + t_{x_2}, t_{y_1} + t_{y_2}) \cdot \mathbf{P}$$

$$\begin{bmatrix} 1 & 0 & t_{2x} \\ 0 & 1 & t_{2y} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & t_{1x} \\ 0 & 1 & t_{1y} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{1x} + t_{2x} \\ 0 & 1 & t_{1y} + t_{2y} \\ 0 & 0 & 1 \end{bmatrix}$$

► Composite 2D Rotations

$$\mathbf{P}' = \mathbf{R}(\theta_1 + \theta_2) \cdot \mathbf{P}$$

$$\begin{bmatrix} \cos \Theta_2 & -\sin \Theta_2 & 0 \\ \sin \Theta_2 & \cos \Theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos \Theta_1 & -\sin \Theta_1 & 0 \\ \sin \Theta_1 & \cos \Theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\Theta_1 + \Theta_2) & -\sin(\Theta_1 + \Theta_2) & 0 \\ \sin(\Theta_1 + \Theta_2) & \cos(\Theta_1 + \Theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

► Composite 2D Scaling

$$\mathbf{S}(s_{x_2}, s_{y_2}) \cdot \mathbf{S}(s_{x_1}, s_{y_1}) = \mathbf{S}(s_{x_1} \cdot s_{x_2}, s_{y_1} \cdot s_{y_2})$$

$$\begin{bmatrix} s_{2x} & 0 & 0 \\ 0 & s_{2y} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_{1x} & 0 & 0 \\ 0 & s_{1y} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{1x} \cdot s_{2x} & 0 & 0 \\ 0 & s_{1y} \cdot s_{2y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

**Note:**

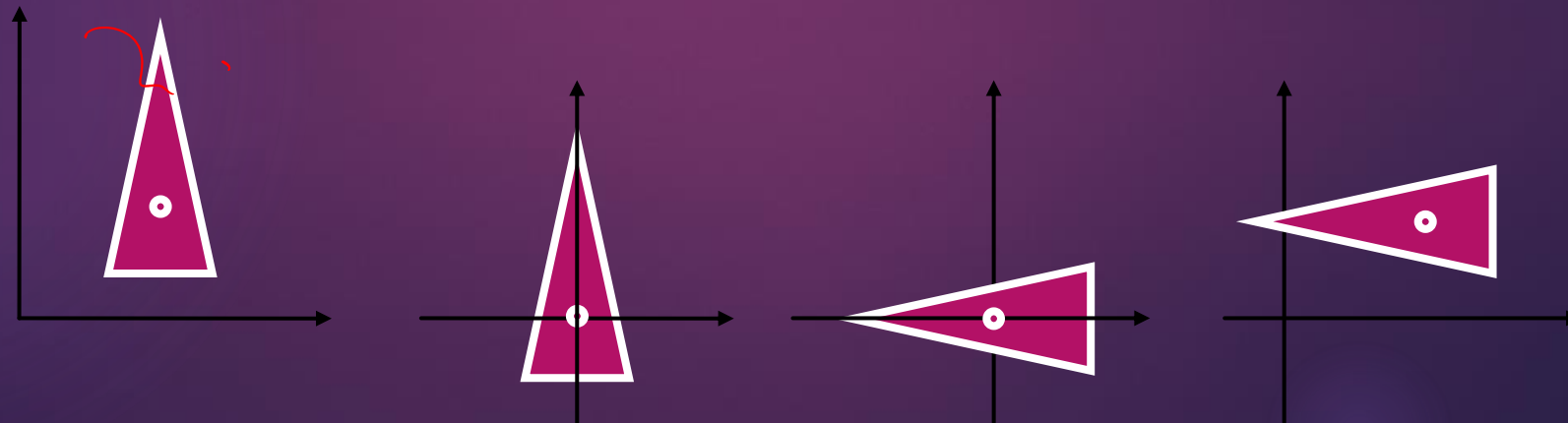
- ▶ Successive translations are additive
- ▶ Successive scalings are multiplicative
  - ▶ For example: If we triple the size of an object twice, the final size is nine (9) times the original

## Rotation w.r.t to pivot point

41

### ► Steps:

1. Translate the object so that the pivot point is moved to the coordinate origin.
2. Rotate the object about the origin.
3. Translate the object so that the pivot point is returned to its original position.



► General 2D Pivot-Point Rotation

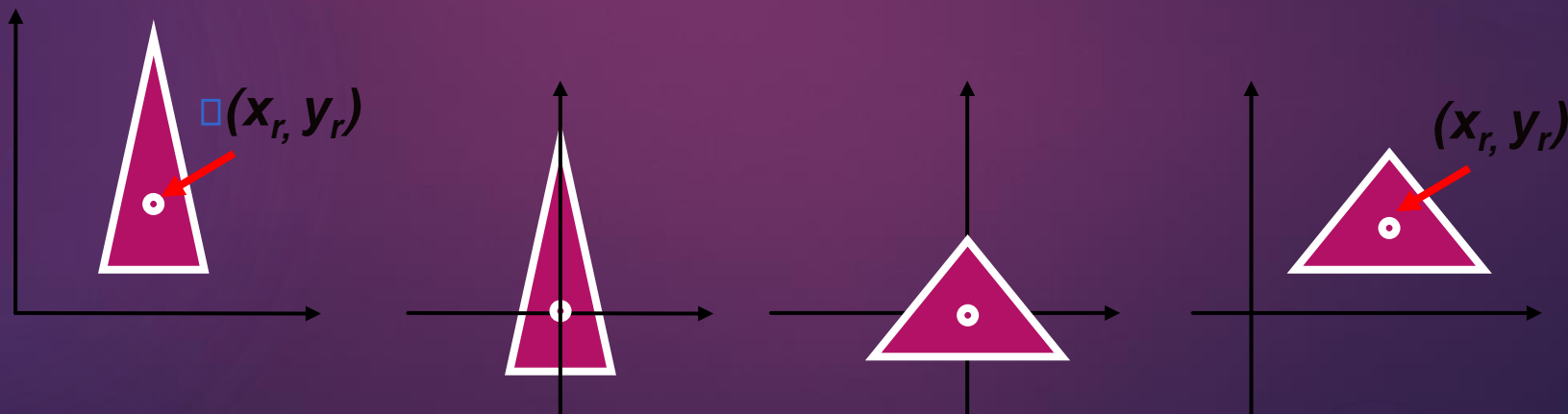
$$\begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos \Theta & -\sin \Theta & 0 \\ \sin \Theta & \cos \Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} \cos \Theta & -\sin \Theta & x_r(1 - \cos \Theta) + y_r \sin \Theta \\ \sin \Theta & \cos \Theta & y_r(1 - \cos \Theta) - x_r \sin \Theta \\ 0 & 0 & 1 \end{bmatrix}$$

# General Fixed Point Scaling

43

► Steps:

1. Translate the object so that the fixed point coincides with the coordinate origin.
2. Scale the object about the origin.
3. Translate the object so that the pivot point is returned to its original position.



# General Fixed Point Scaling (cont.)

44

- General 2D Fixed-Point Scaling:

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

$$\mathbf{T}(x_f, y_f) \cdot \mathbf{S}(s_x, s_y) \cdot \mathbf{T}(-x_f, -y_f) = \mathbf{S}(x_f, y_f, s_x, s_y)$$



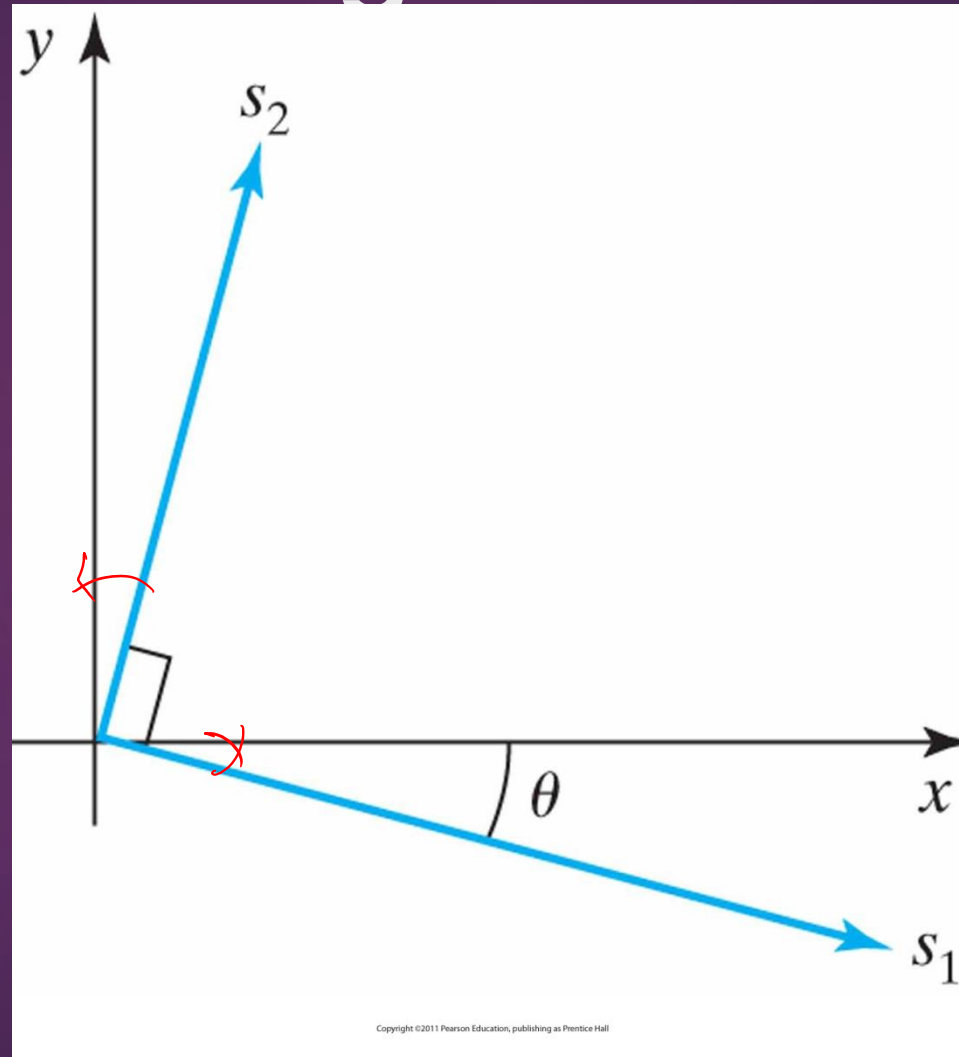
# 2D Composite Transformations

45

- ▶ General 2D scaling directions:
  - ▶ Above: scaling parameters were along x and y directions
  - ▶ What about arbitrary directions?

# General 2D Scaling Directions

46



Scaling parameters  $s_1$  and  $s_2$  along orthogonal directions defined by the angular displacement  $\theta$ .

# General 2D Scaling Directions (cont.)

- General procedure:

1. Rotate so that directions coincides with  $x$  and  $y$  axes
2. Apply scaling transformation  $S(s_1, s_2)$
3. Rotate back

- The composite matrix:

$$R^{-1}(\Theta) * S(s_1, s_2) * R(\Theta) = \begin{bmatrix} s_1 \cos^2 \Theta + s_2 \sin^2 \Theta & (s_2 - s_1) \cos \Theta \sin \Theta & 0 \\ (s_2 - s_1) \cos \Theta \sin \Theta & s_1 \sin^2 \Theta + s_2 \cos^2 \Theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

# 2D Composite Transformations (cont.)

48

- ▶ Matrix Concatenation Properties:
  - ▶ Matrix multiplication is associative !
    - ▶  $M_3 \cdot M_2 \cdot M_1 = (M_3 \cdot M_2) \cdot M_1 = M_3 \cdot (M_2 \cdot M_1)$
    - ▶ A composite matrix can be created by multiplying left-to-right (premultiplication) or right-to-left (postmultiplication)
  - ▶ Matrix multiplication is **not** commutative !
    - ▶  $M_2 \cdot M_1 \neq M_1 \cdot M_2$

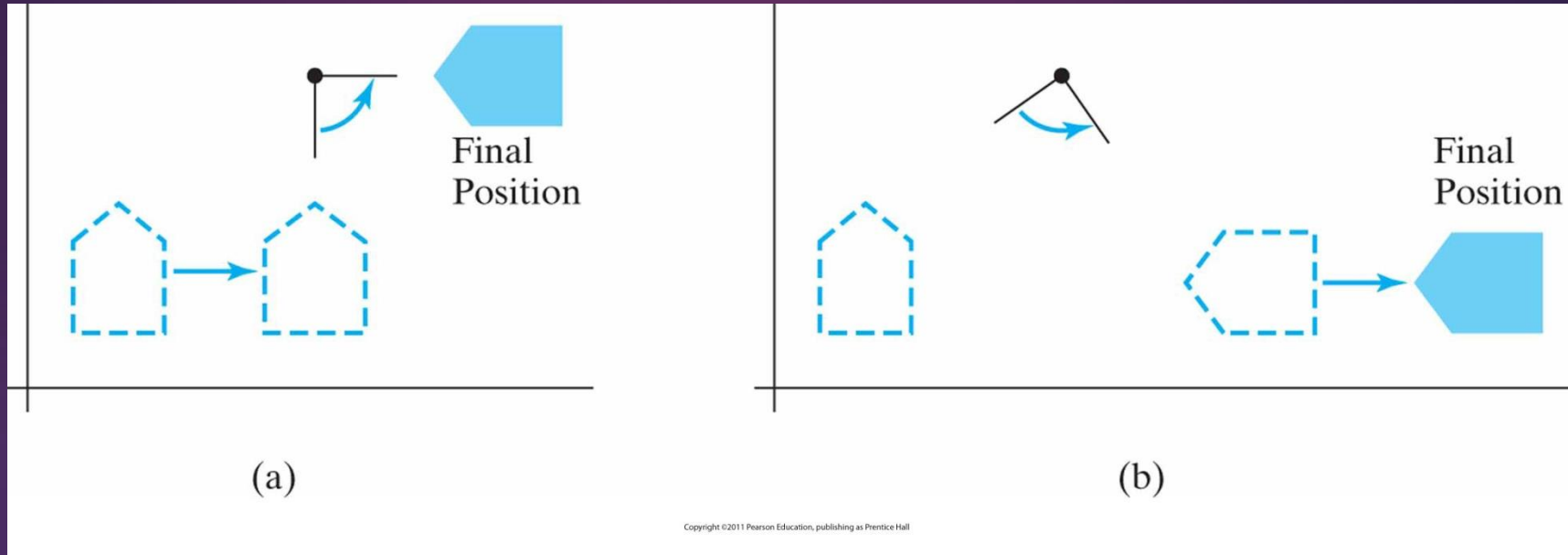
# 2D Composite Transformations (cont.)

49

- ▶ Matrix Concatenation Properties:
  - ▶ But:
    - ▶ Two successive rotations
    - ▶ Two successive translations
    - ▶ Two successive scalings
  - ▶ **are** commutative

# Reversing the order

50



in which a sequence of transformations is performed may affect the transformed position of an object.

In (a), an object is first translated in the  $x$  direction, then rotated counterclockwise through an angle of  $45^\circ$ .

In (b), the object is first rotated  $45^\circ$  counterclockwise, then translated in the  $x$  direction

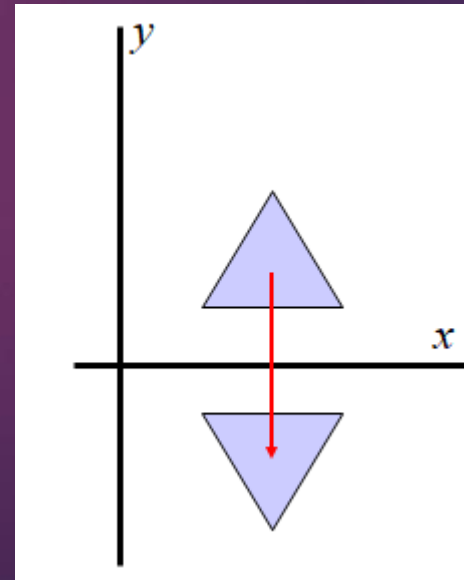
# 4. Reflection

- ▶ Transformation that produces a mirror image of an object
- ▶ **Reflection about**
  1. x axis ( $y=0$ )
  2. y axis ( $x=0$ )
  3. Perpendicular to xy plane
  4.  $y = x$
  5.  $y = -x$
  6. Any line  $y = mx+c$

## i. Reflection about x- axis

- ▶ Image is generated relative to an axis of reflection by rotating the object 180° about the reflection axis
- ▶ Reflection about the line  $y=0$  (the x axis)

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

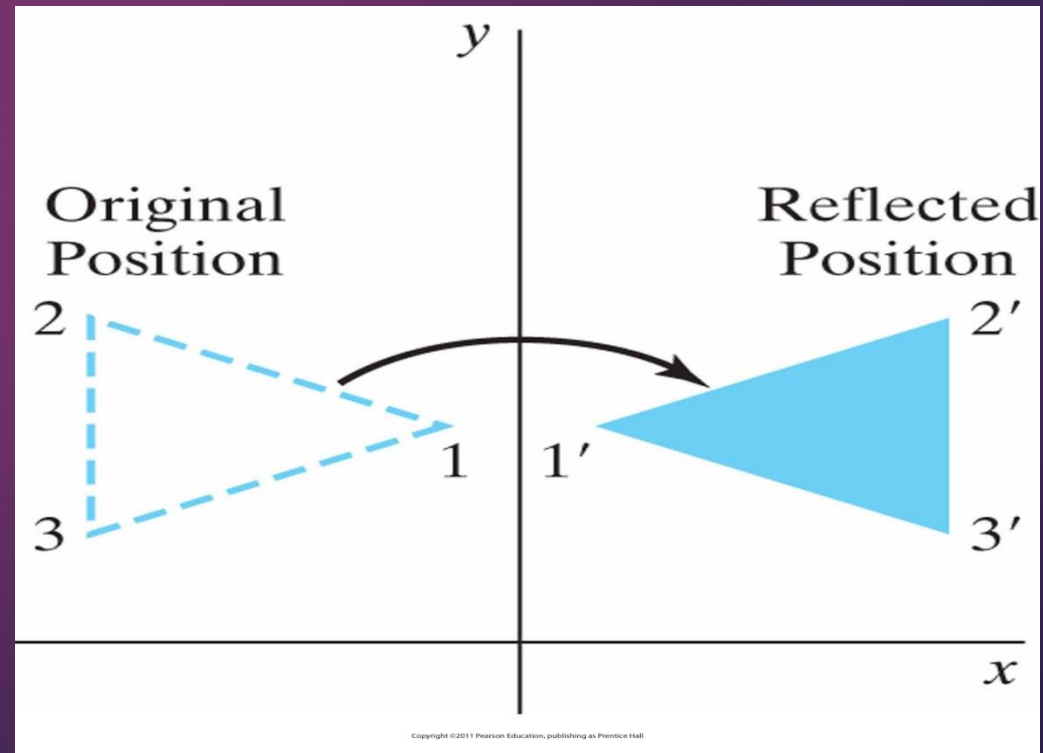




## ii. Reflection about y- axis

- Reflection about the line  $x=0$  (the y axis)

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

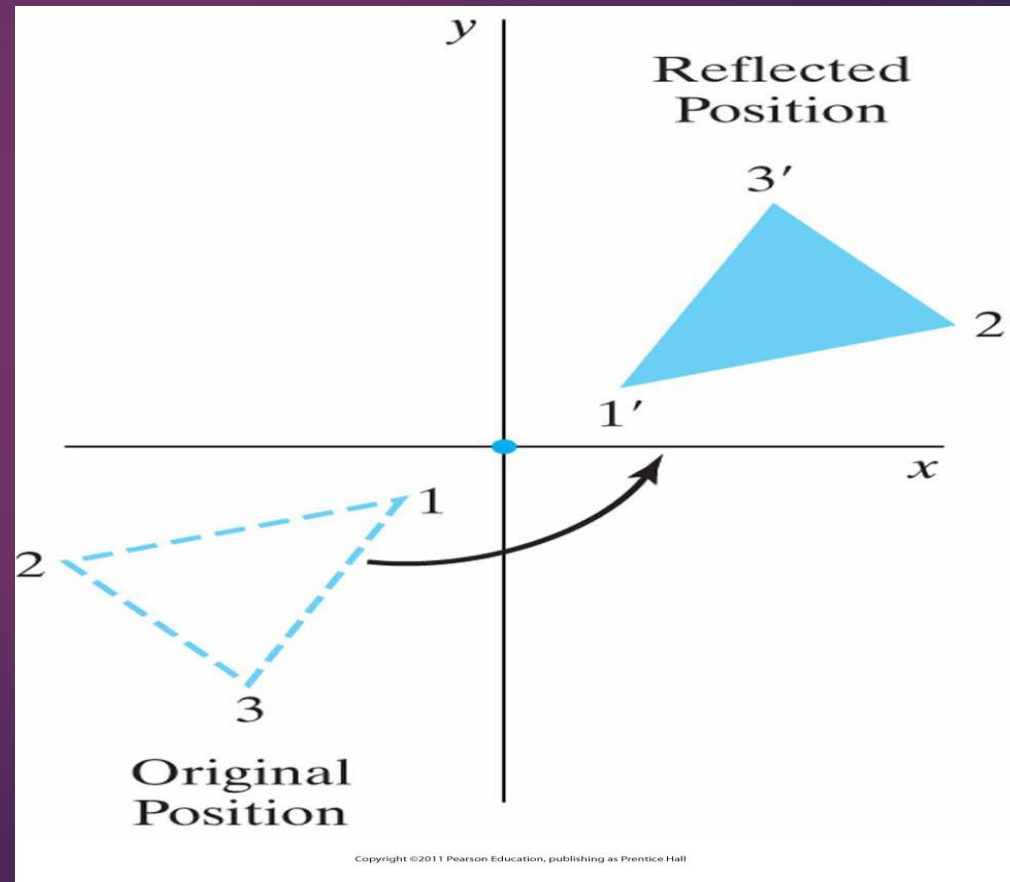


### iii. Reflection about origin

54

- Reflection about the origin

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

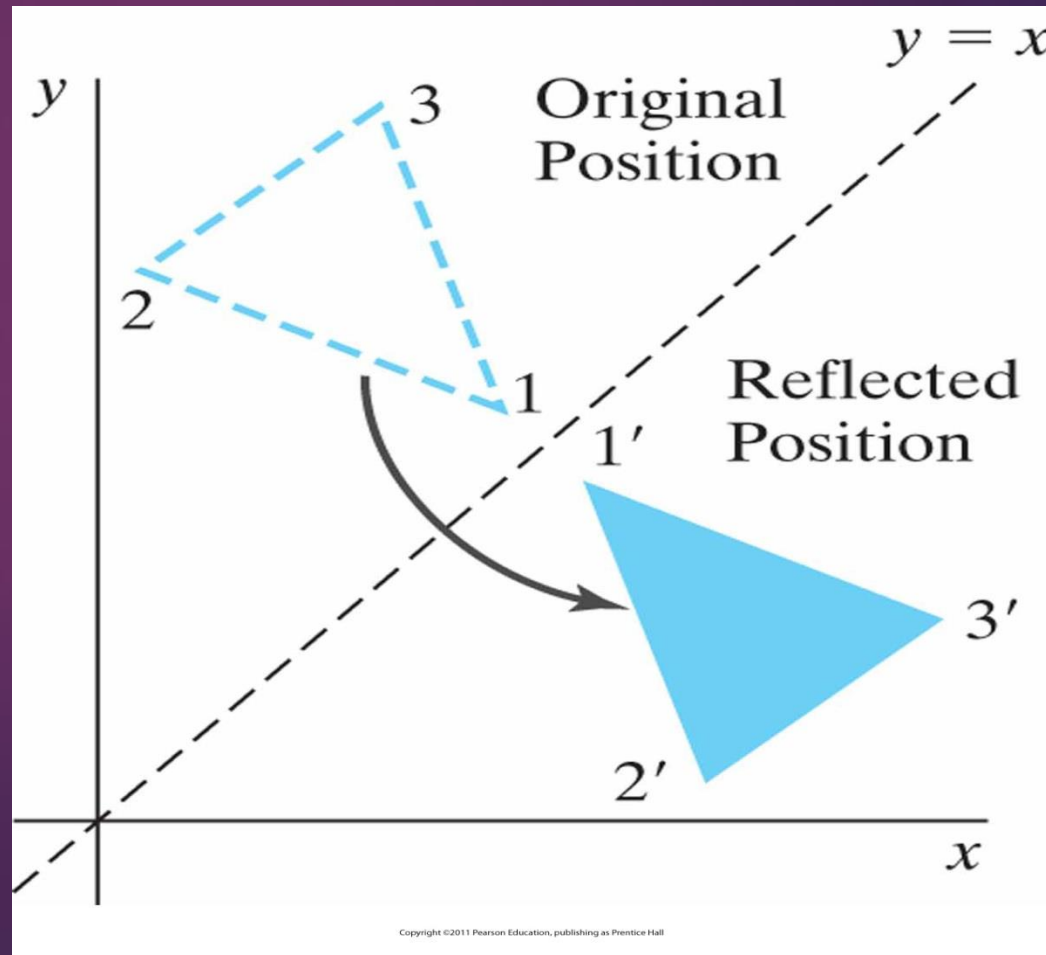


## iv. Reflection about $y = x$

55

► Reflection about the line  $y=x$

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

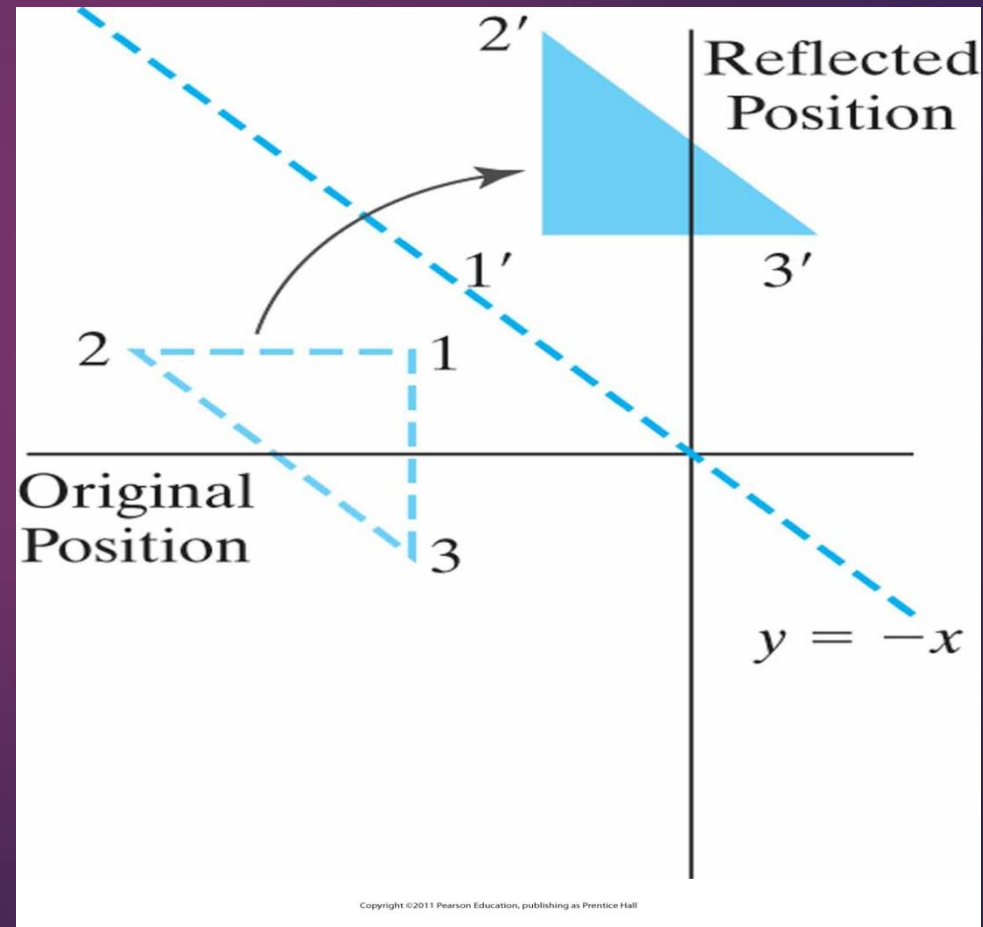


# v. Reflection about $y = x$

56

► Reflection about the line  $y = -x$

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



## vi. Reflection about line $y = mx + c$

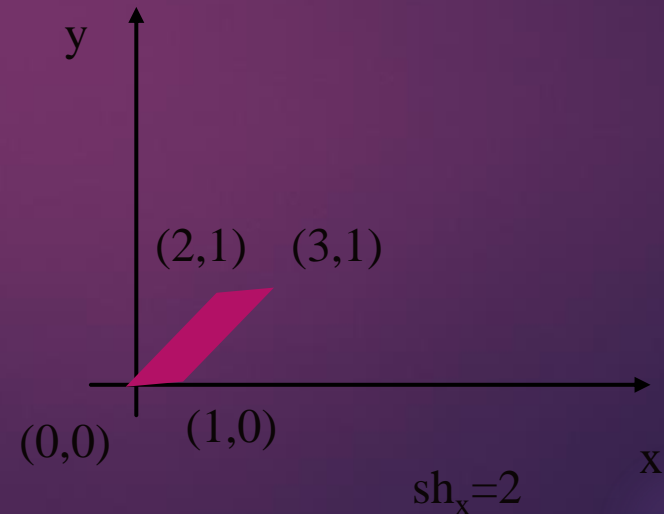
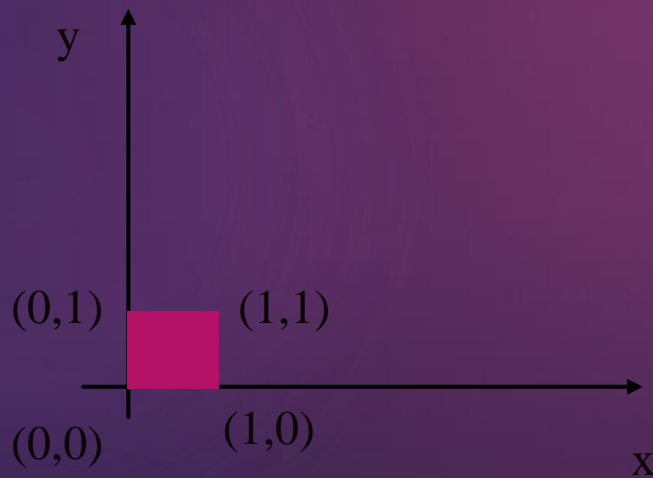
### Steps

1. Translate the line so that it passes through the origin.
2. Rotate the line onto one of the coordinate axes and reflect about that axis
3. Restore the line to its original position with the inverse rotation and translation transformation

# 5. Shear

58

- Transformation that distorts the shape of an object such that the transformed shape appears as the object was composed of internal layers that had been caused to slide over each other



# X and y direction - Shear

59

- An x-direction shear relative to the x axis

$$\begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{aligned} x' &= x + sh_x \cdot y \\ y' &= y \end{aligned}$$

- An y-direction shear relative to the y axis

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

# X axis - Shear

60

- x-direction shear relative to other reference lines

$$\begin{bmatrix} 1 & sh_x & -sh_x * y_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

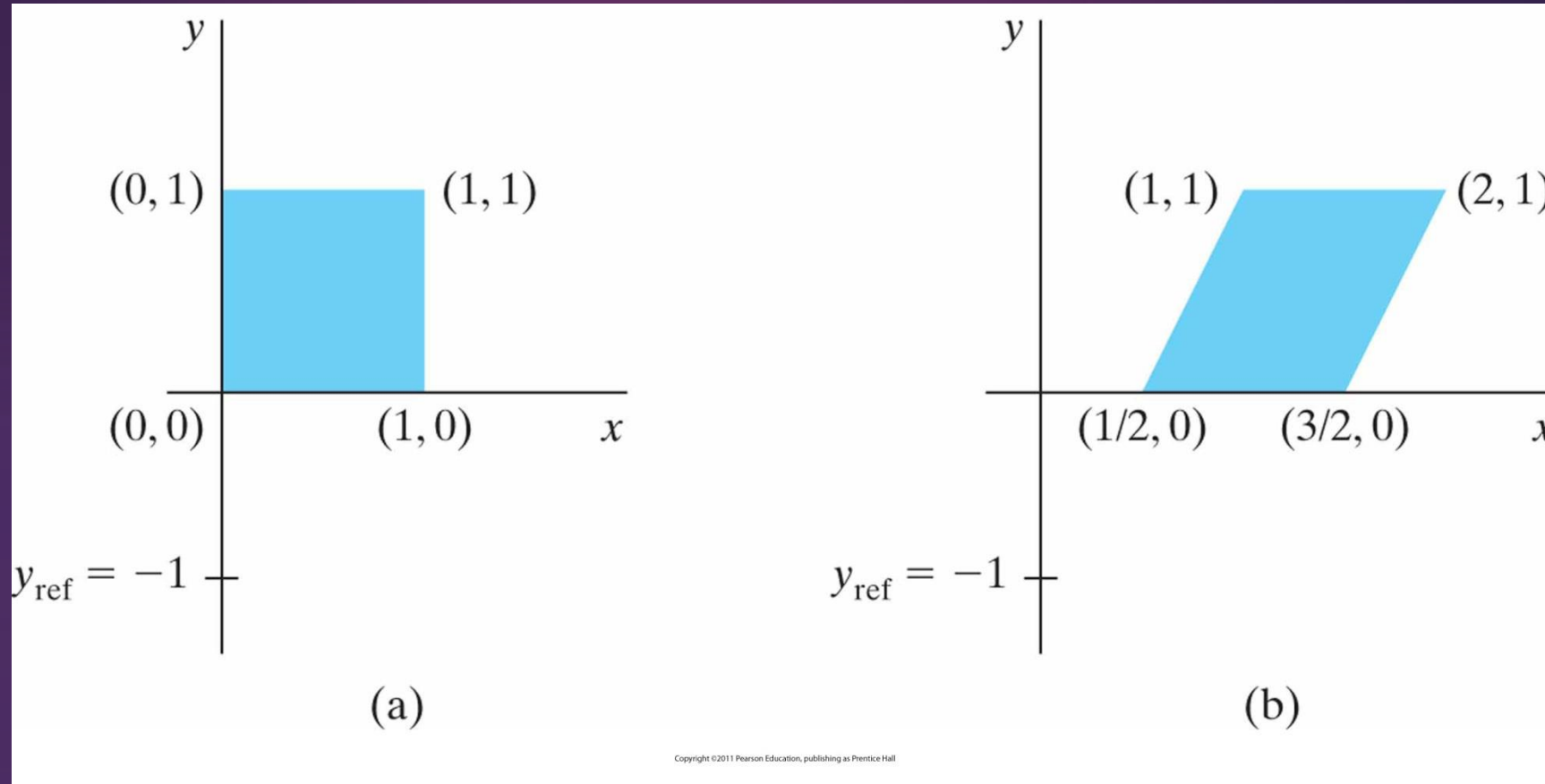
$$x' = x + sh_x * (y - y_{ref})$$

$$y' = y$$



# Example

61



A unit square (a) is transformed to a shifted parallelogram (b) with  $sh_x = 0.5$  and  $y_{\text{ref}} = -1$  in the shear matrix from Slide 56

# Y axis - Shear

62

- y-direction shear relative to the line  $x = x_{ref}$

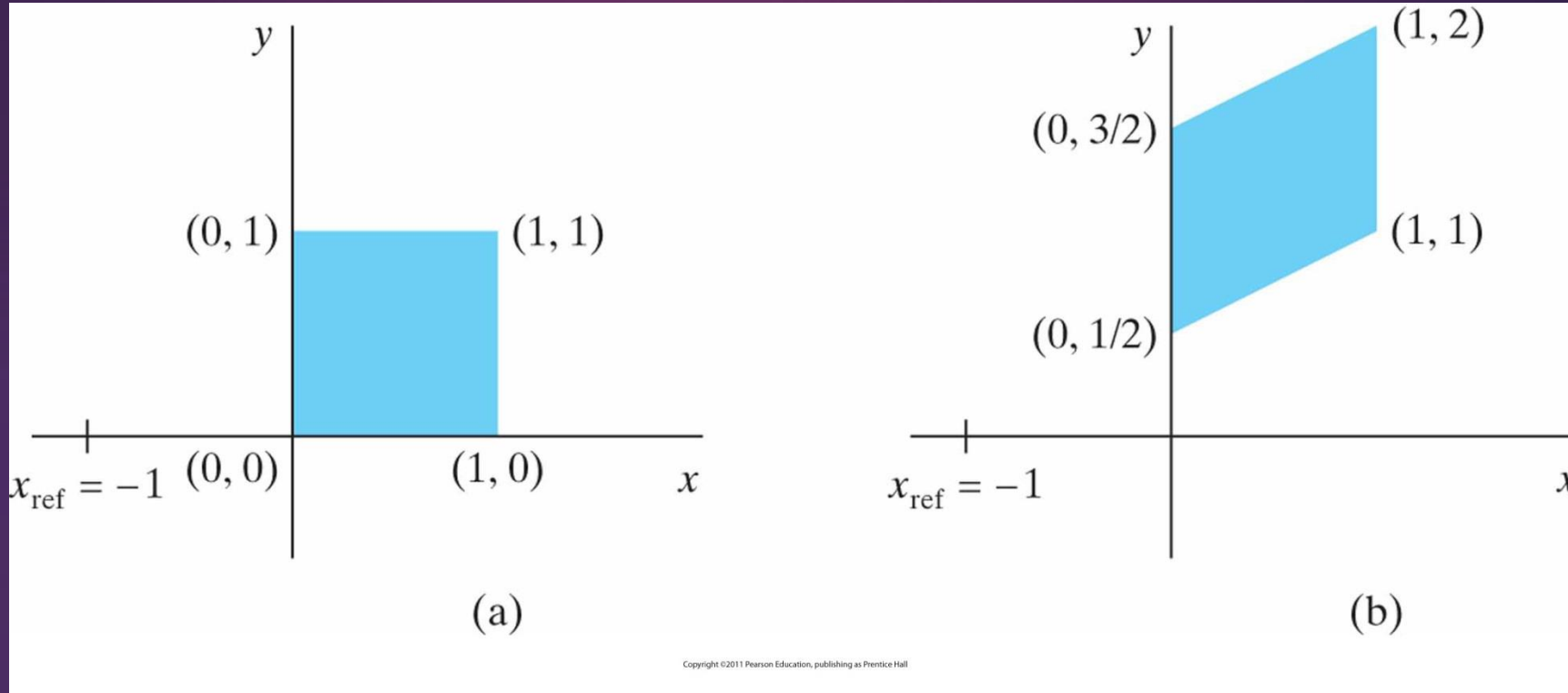
$$\begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & -sh_y * x_{ref} \\ 0 & 0 & 1 \end{bmatrix}$$

$$x' = x$$

$$y' = y + sh_y * (x - x_{ref})$$

# Example

63



A unit square (a) is turned into a shifted parallelogram (b) with parameter values  $sh_y = 0.5$  and  $x_{\text{ref}} = -1$  in the  $y$ -direction shearing transformation from Slide 58

# Transformation Between Coordinate Systems

64

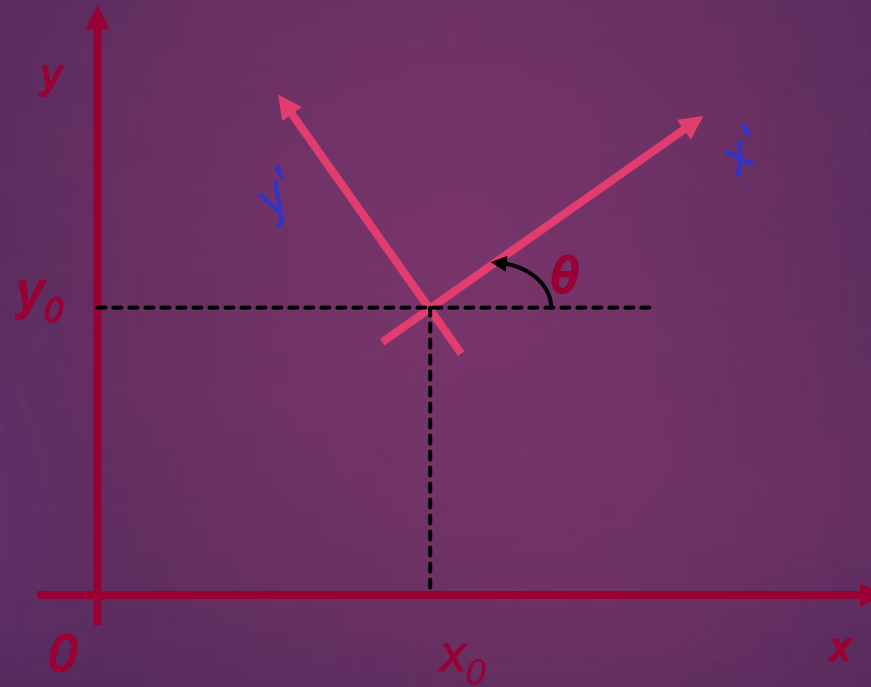
- ▶ Individual objects may be defined in their local cartesian reference system.
- ▶ The local coordinates must be transformed to position the objects within the scene coordinate system.

# Transformation Between Coordinate Systems

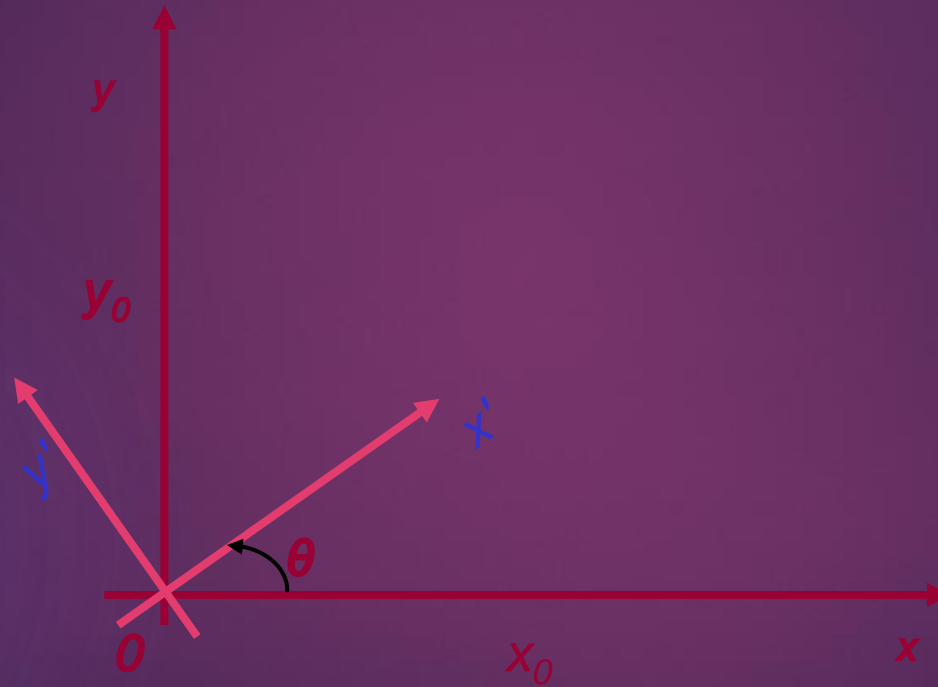
## Steps for coordinate transformation

1. Translate so that the origin  $(x_0, y_0)$  of the  $x'-y'$  system is moved to the origin of the  $x-y$  system.
2. Rotate the  $x'$  axis on to the axis  $x$ .

# Transformation Between Coordinate Systems (cont.)

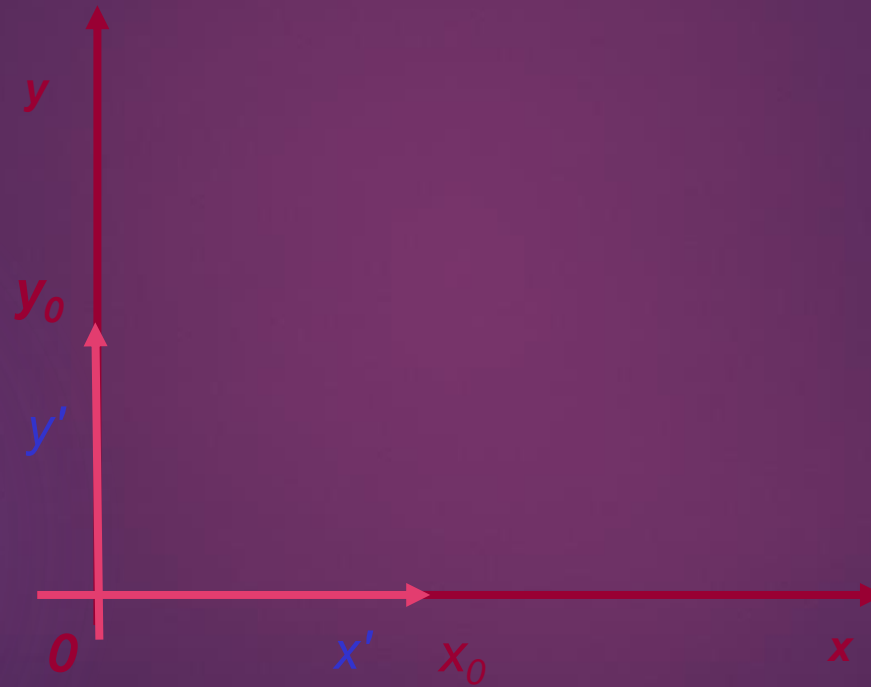


# Transformation Between Coordinate Systems (cont.)



# Transformation Between Coordinate Systems (cont.)

68





# Transformation Between Coordinate Systems (cont.)

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

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

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

# Transformation Between Coordinate Systems (cont.)

An alternative method:

- Specify a vector  $\mathbf{V}$  that indicates the direction for the positive  $y'$  axis. Let

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

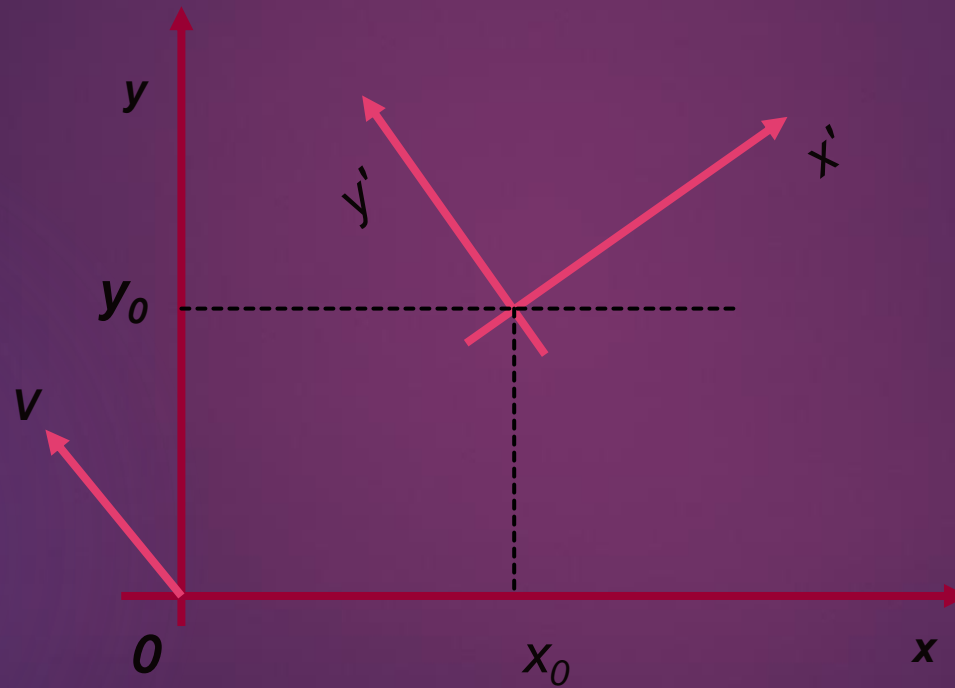
- Obtain the unit vector  $\mathbf{u}=(u_x, u_y)$  along the  $x'$  axis by rotating  $\mathbf{v}$   $90^\circ$  clockwise.

# Transformation Between Coordinate Systems (cont.)

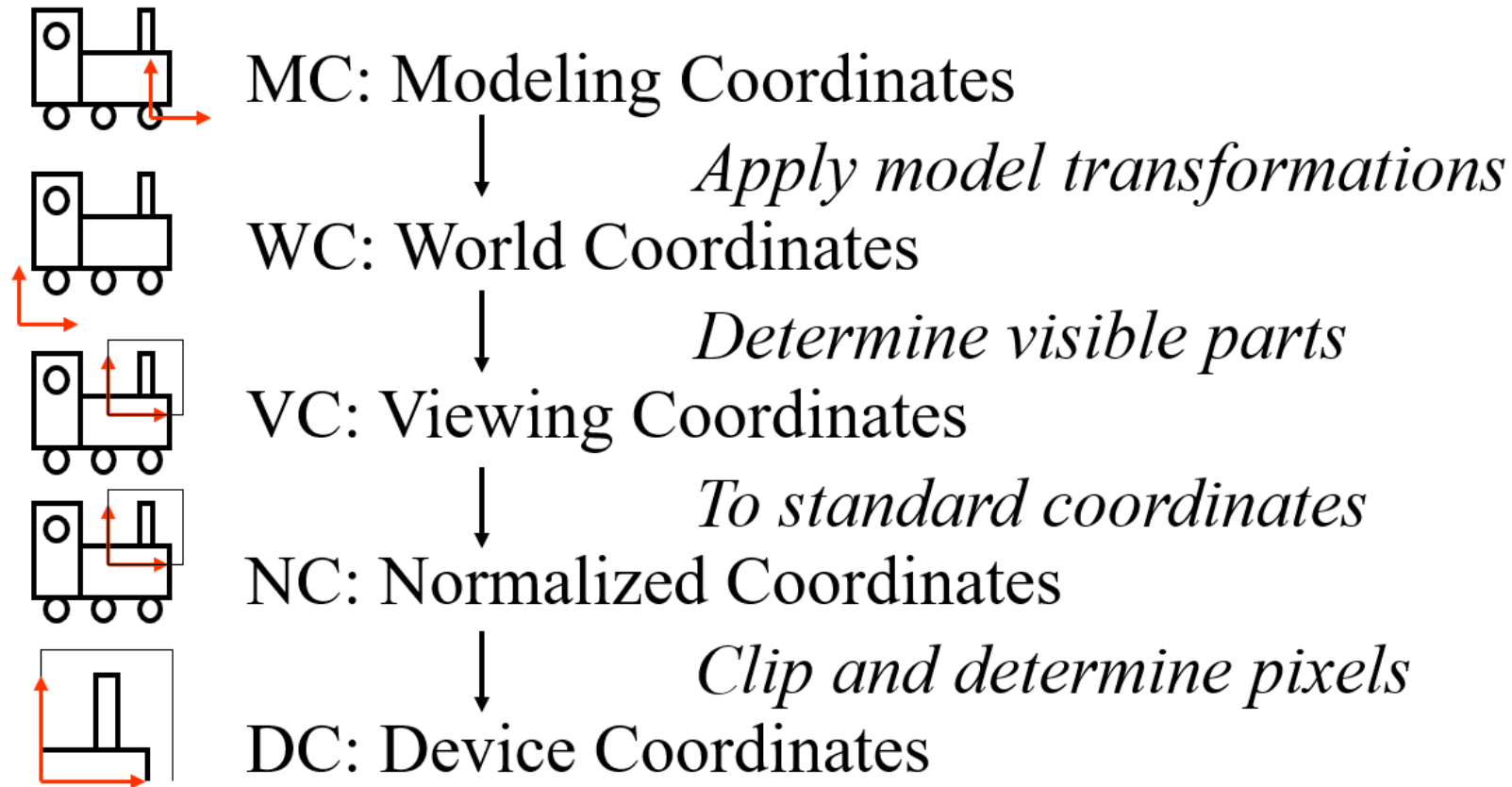
- Elements of any rotation matrix can be expressed as elements of orthogonal unit vectors. That is, the rotation matrix can be written as

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

# Transformation Between Coordinate Systems (cont.)



# 2D viewing pipeline



# References

- Images and videos used in this presentation are referred from the Internet source
- D. Hearn and M. P. Baker, “Computer Graphics - C version”, Pearson Education, 2004.