Unit II
Polygons And Graphical Transformations

Agenda
Part B – Graphical Transformations

- 2D Transformations : Translation, Scaling, Rotation.
- Other Transformations : Reflection, Shearing
- Composite Transformations

Prerequisite

- Linear Algebra
- Matrix Operations
- Homogeneous Coordinate system

Major Application

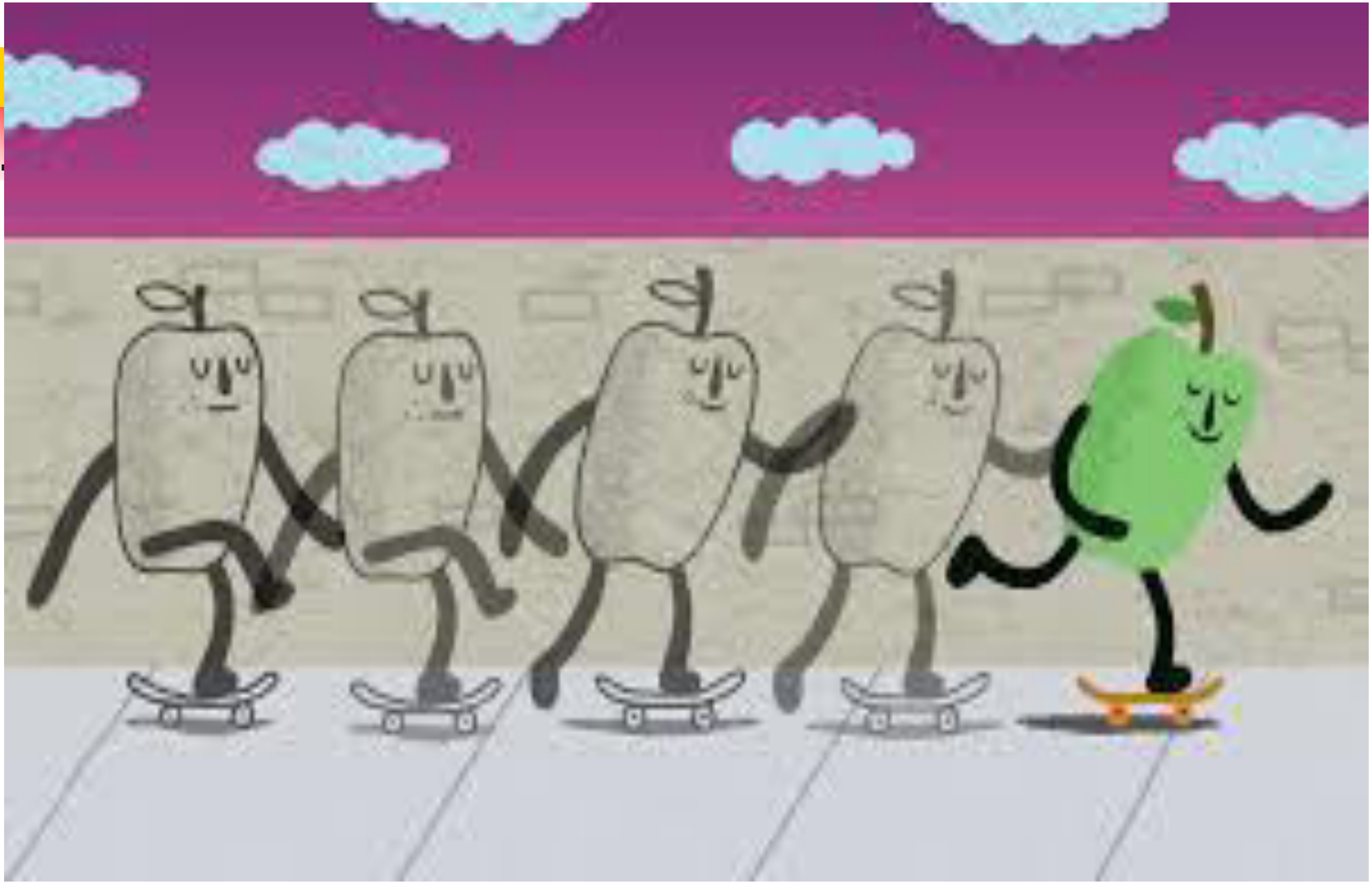
Animation

2D Transformation

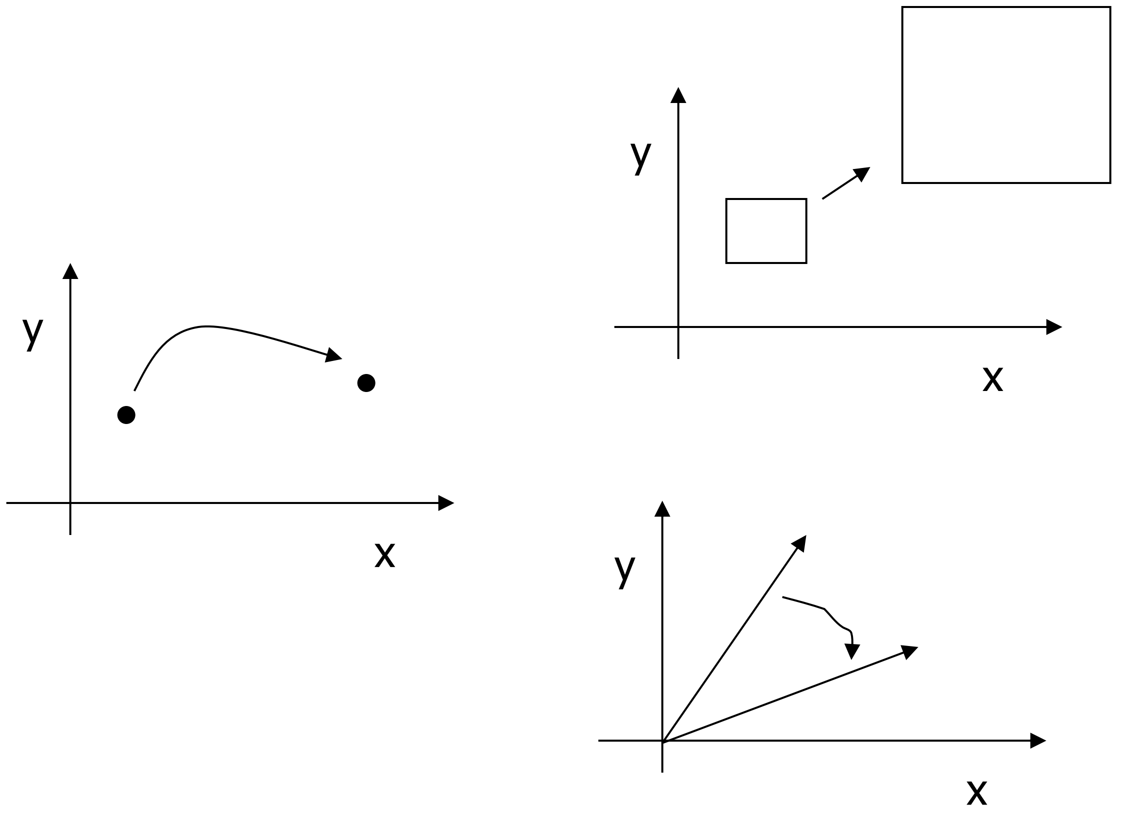
- Given a 2D object, transformation is to change the object's
  - Position (translation)
  - Size (scaling)
  - Orientation (rotation)

  Other Transformations:-

  - Shapes (shear)
  - Reflection



2D Transformations

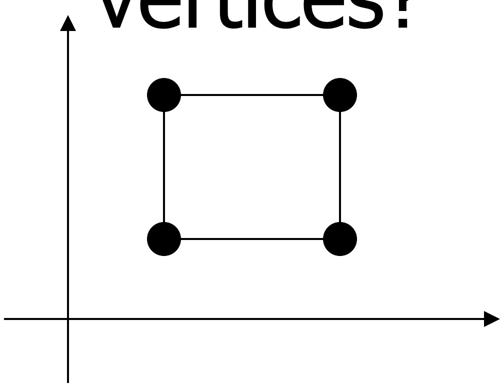


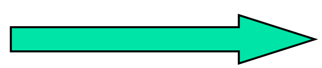
Point representation

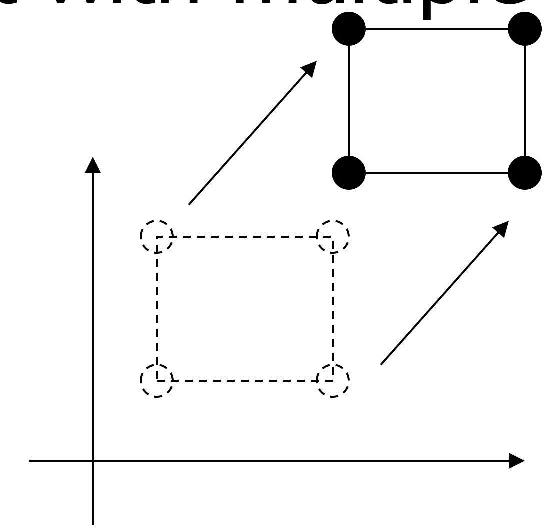
- We can use a column vector (a 2x1 matrix) to represent a 2D point $\begin{vmatrix} x \\ y \end{vmatrix}$
- or row vector [x y]

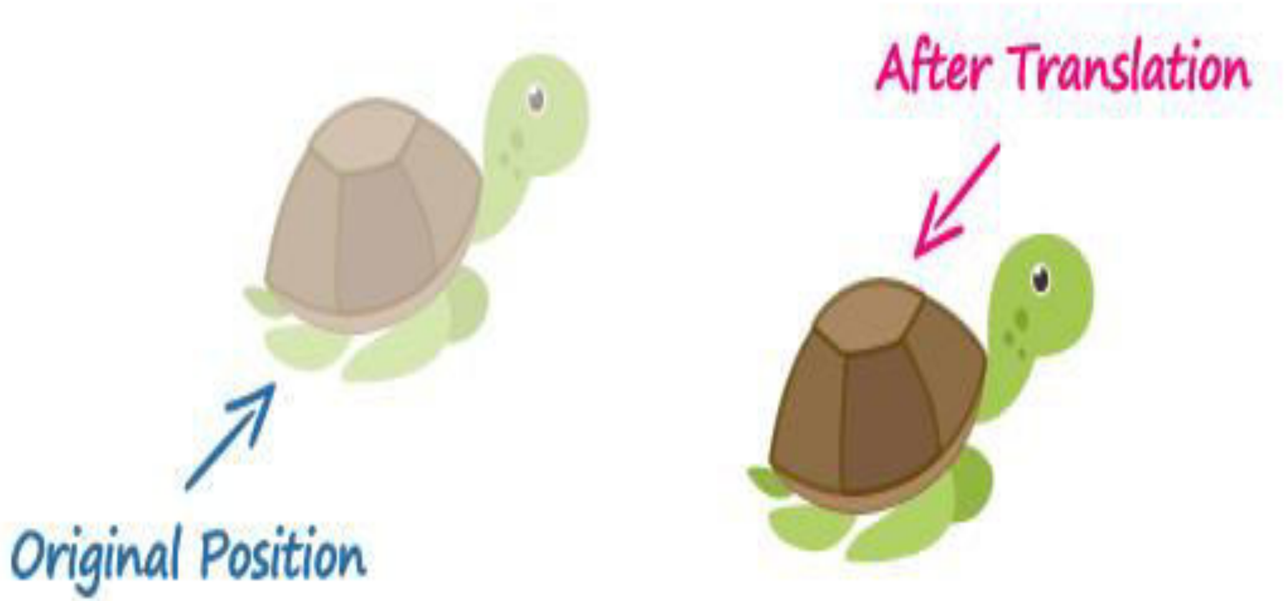
Translation

- Moving one object from one point to other point
- How to translate an object with multiple vertices?







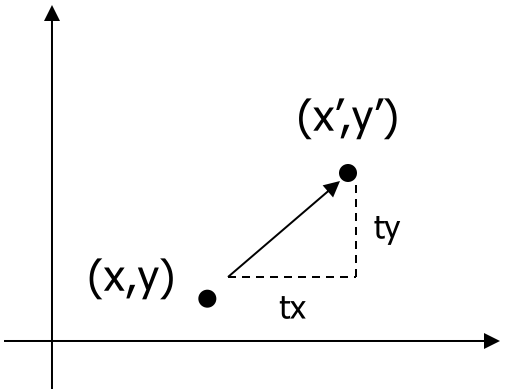

After Translation
Original Position

Translation

- Re-position a point along a straight line from one co-ordinate location to another
- Given a point (x,y), and the translation distance or vector (tx,ty)

The new point: (x', y')

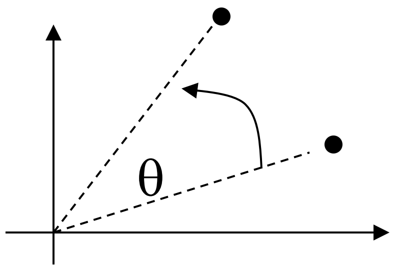
$$x' = x + tx$$
$$y' = y + ty$$



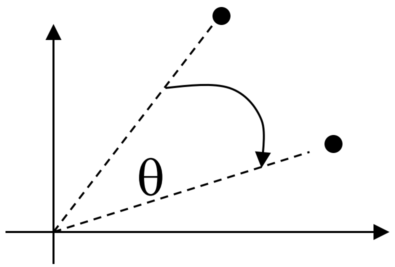
OR $P' = P + T$ where $P' = \begin{vmatrix} x' \\ y' \end{vmatrix}$ $p = \begin{vmatrix} x \\ y \end{vmatrix}$ $T = \begin{vmatrix} tx \\ ty \end{vmatrix}$

2D Rotation

- Default rotation center: Origin (0,0)



$\theta > 0$: Rotate counter clockwise

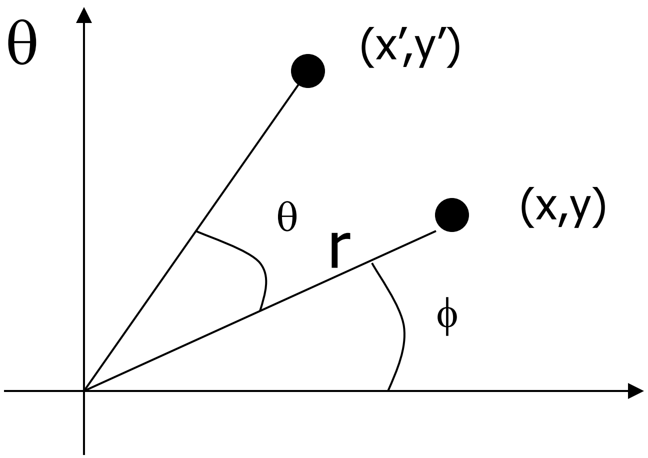


$\theta < 0$: Rotate clockwise

Rotation

(x,y) -> Rotate *about the origin* by θ

$\longrightarrow$ (x′, y′)



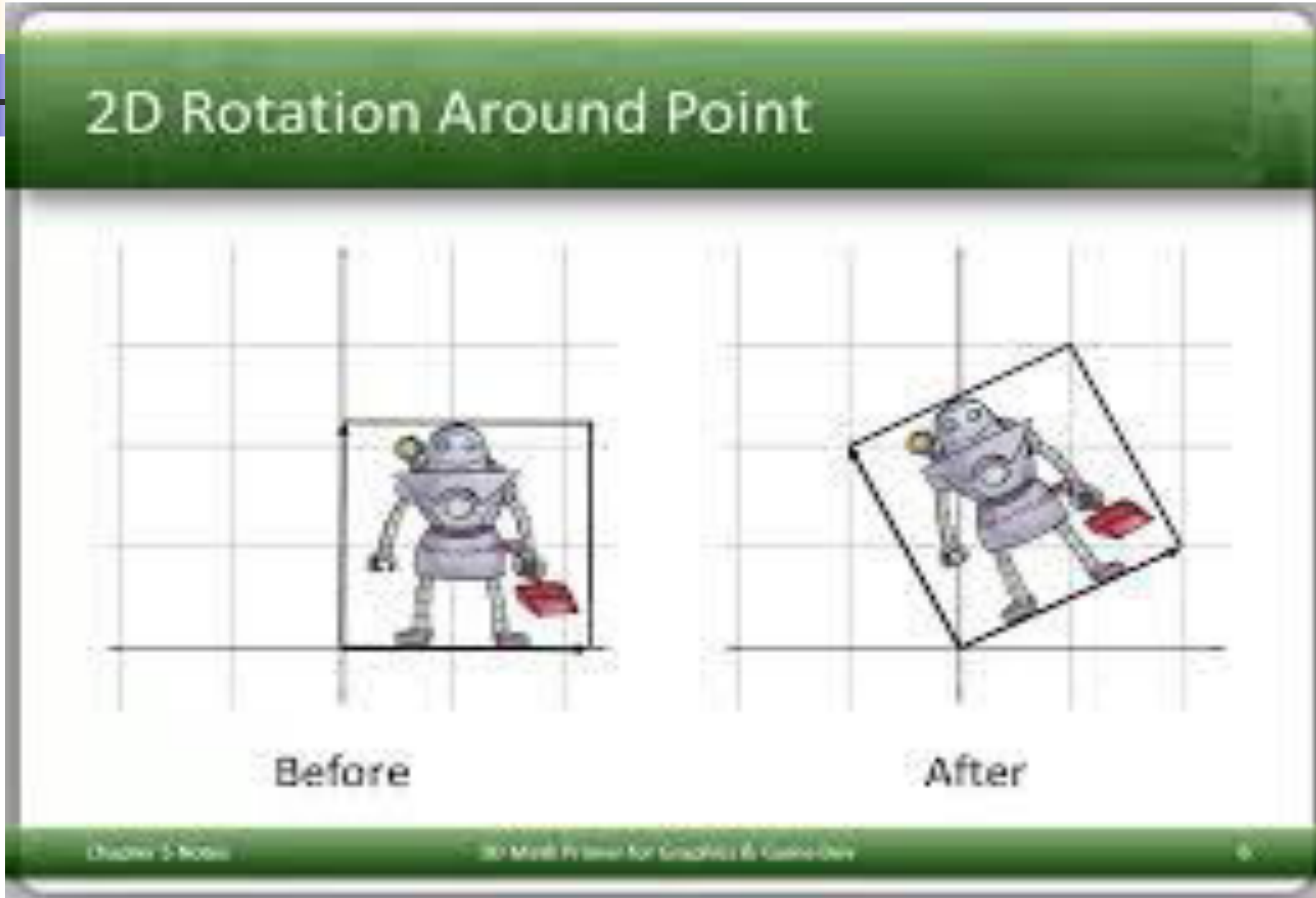
How to compute (x′, y′) ?

The original co-ordinates of the point in polar co-ordinates are

$$x = r\cos(\phi) \quad y = r\sin(\phi)$$

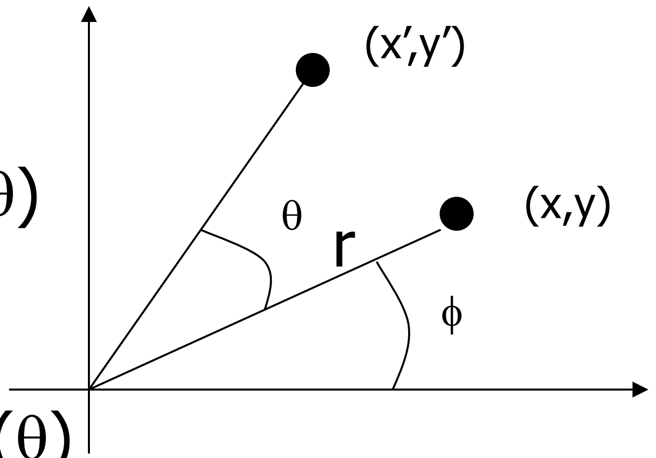
Using standard trigonometric identities, we can express the transformed co-ordinates as

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



2D Rotation Around Point
Before
After

Rotation



$x = r\cos(\phi) \quad y = r\sin(\phi)$

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

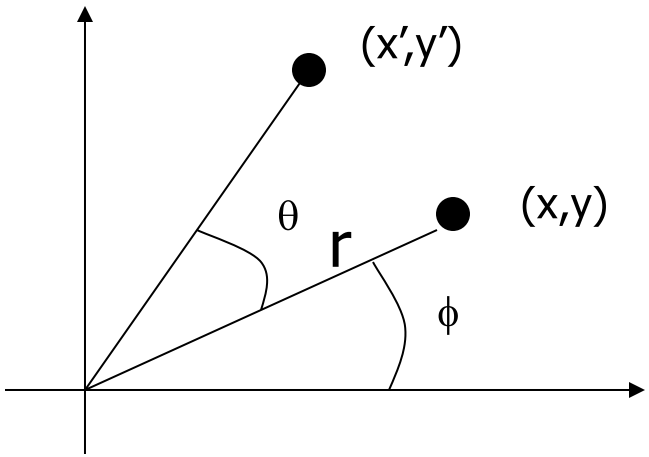
$$
\begin{aligned}
x' &= r\cos(\phi + \theta) \\
&= \boxed{r\cos(\phi)}\cos(\theta) - \boxed{r\sin(\phi)}\sin(\theta) \\
&= x\cos(\theta) - y\sin(\theta)
\end{aligned}
$$

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

Rotation

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

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



Matrix form?

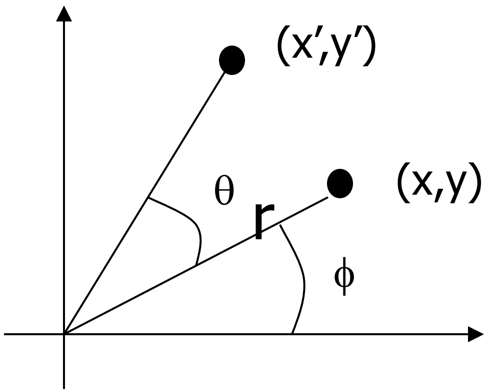
Counter clockwise rotation

$$\begin{vmatrix} x' \\ y' \end{vmatrix} = \begin{vmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{vmatrix} \begin{vmatrix} x \\ y \end{vmatrix}$$

3 x 3?

3x3 2D Rotation Matrix

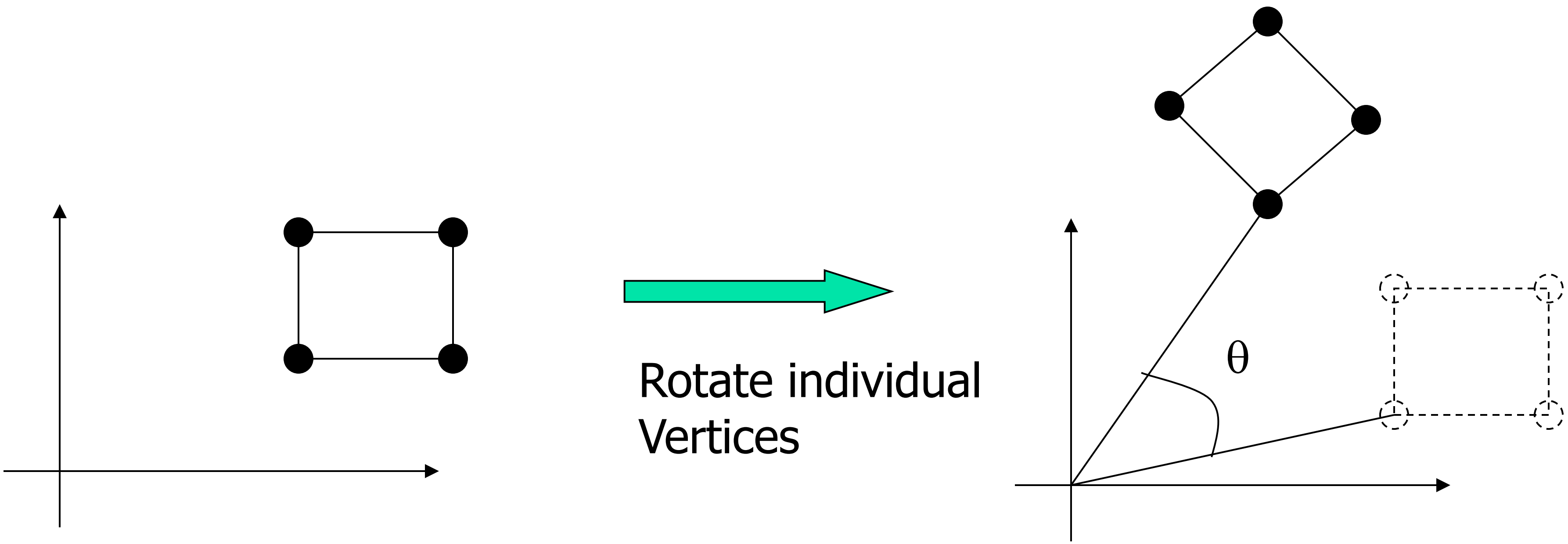
$$\begin{vmatrix} x' \\ y' \end{vmatrix} = \begin{vmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{vmatrix} \begin{vmatrix} x \\ y \end{vmatrix}$$



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

Rotation

- How to rotate an object with multiple vertices?



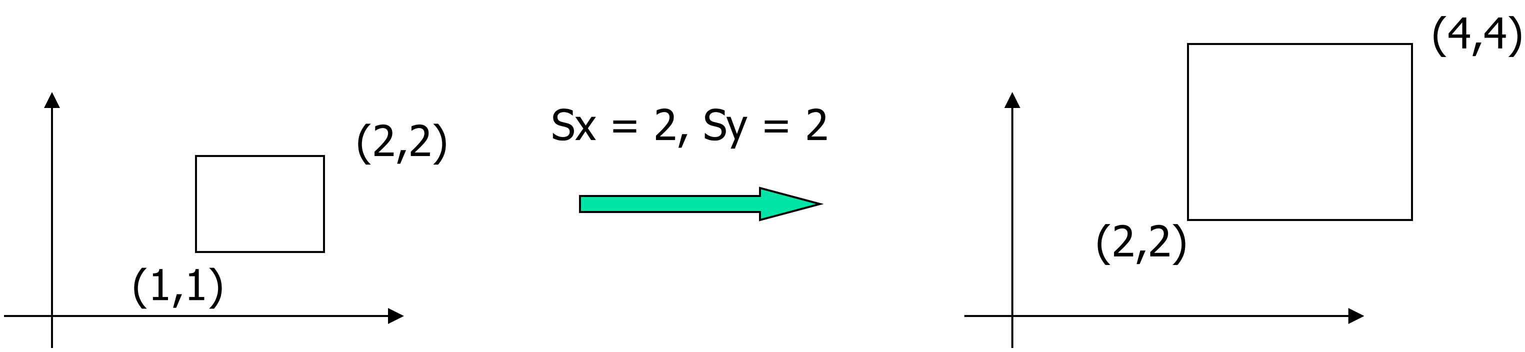
2D Scaling

Enhancing or reducing the size of the object

Scale: Alter the size of an object by a scaling factor (Sx, Sy), i.e.

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

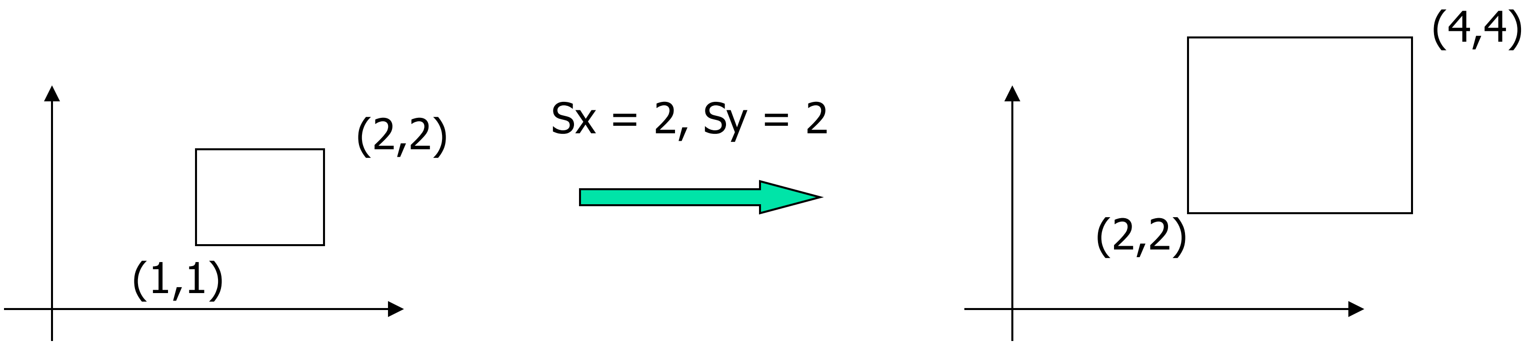
$$\Rightarrow \quad \begin{vmatrix} x' \\ y' \end{vmatrix} = \begin{vmatrix} Sx & 0 \\ 0 & Sy \end{vmatrix} \begin{vmatrix} x \\ y \end{vmatrix}$$



2D Scaling

- Scaling is of two types.
- They are:
- <u>Uniform Scaling</u> :Scaling factors Sx and Sy have the same value
- <u>Differential Scaling</u>: Scaling factors Sx and Sy have unequal values
- E.g. If Sx=2 and Sy=1 and we scale a unit circle with centre(0,0) then it becomes an ellipse with centre (0,0) with major axis 2 and minor axis is 1.

2D Scaling



- Not only the object size is changed, it also moved!!
- Usually this is an undesirable effect

3x3 2D Scaling Matrix

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

$$\Downarrow$$

$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

Put it all together

- Translation: $\begin{vmatrix} x' \\ y' \end{vmatrix} = \begin{vmatrix} x \\ y \end{vmatrix} + \begin{vmatrix} tx \\ ty \end{vmatrix}$

- Rotation: $\begin{vmatrix} x' \\ y' \end{vmatrix} = \begin{vmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{vmatrix} * \begin{vmatrix} x \\ y \end{vmatrix}$

- Scaling: $\begin{vmatrix} x' \\ y' \end{vmatrix} = \begin{vmatrix} Sx & 0 \\ 0 & Sy \end{vmatrix} * \begin{vmatrix} x \\ y \end{vmatrix}$

Why use 3x3 matrices?

- So that we can perform all transformations using matrix/vector multiplications
- This allows us to *pre-multiply* all the matrices together
- The point (x,y) needs to be represented as (x,y,1) -> this is called Homogeneous coordinates!

Linear Transformations

- A general form of a *linear* transformation can be written as:

$x' = ax + by + c$

OR

$y' = dx + ey + f$

$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

3x3 Matrix representations

- Translation:

$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

- Rotation:

$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

- Scaling:

$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

3x3 Matrix representations

Homogenous Co-ordinates for Translation

$$\begin{bmatrix} X' & y' & 1 \end{bmatrix} = \begin{bmatrix} X & y & 1 \end{bmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ tx & ty & 1 \end{pmatrix}$$

Homogenous Co-ordinates for Rotation

$$\begin{bmatrix} X' & y' & 1 \end{bmatrix} = \begin{bmatrix} X & y & 1 \end{bmatrix} \begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\mathrm{Sin}(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

3x3 Matrix representations

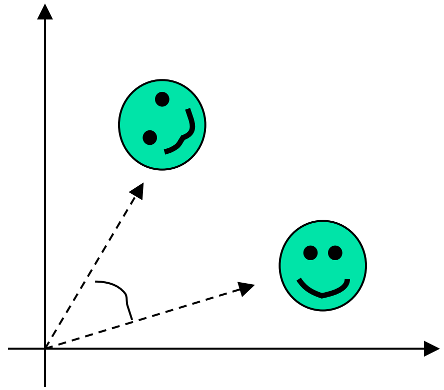
Homogenous Co-ordinates for Scaling

$$\begin{bmatrix} X' & y' & 1 \end{bmatrix} = \begin{bmatrix} X & y & 1 \end{bmatrix} \begin{pmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Rotation Revisit

- The standard rotation matrix is used to rotate about the origin (0,0)

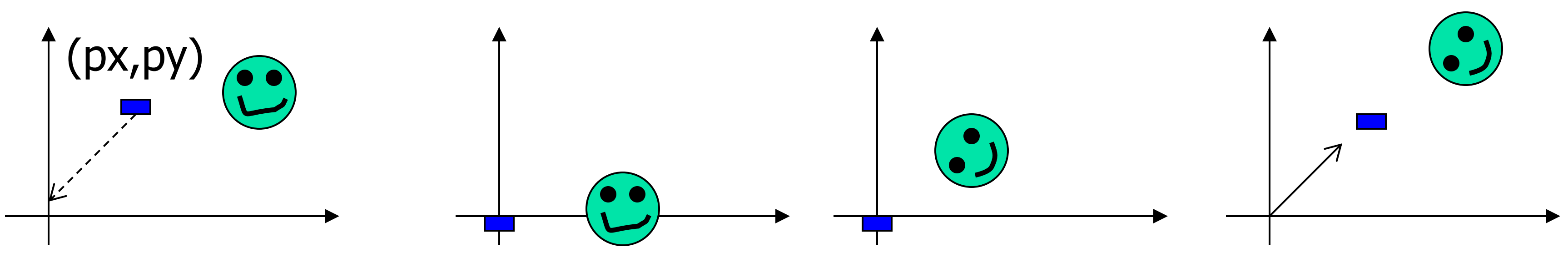
$$\begin{matrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{matrix}$$



- What if I want to rotate about an arbitrary center?

Arbitrary Rotation Center

- To rotate about an arbitrary point P (px,py) by θ:
  - Translate the object so that P will coincide with the origin: T(-px, -py)
  - Rotate the object: R(θ)
  - Translate the object back: T(px,py)



Arbitrary Rotation Center

- Translate the object so that P will coincide with the origin: T(-px, -py)
- Rotate the object: R(θ)
- Translate the object back: T(px,py)

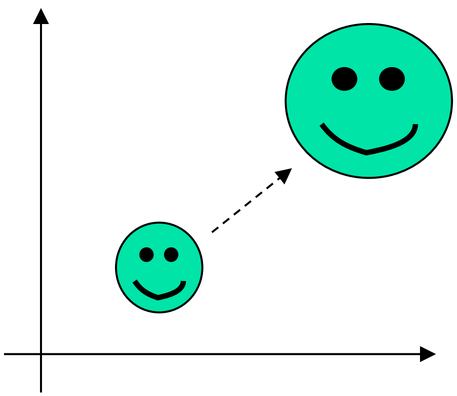
- Put in matrix form: T(px,py) R(θ) T(-px, -py) * P

$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & px \\ 0 & 1 & py \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & -px \\ 0 & 1 & -py \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

Scaling Revisit

- The standard scaling matrix will only anchor at (0,0)

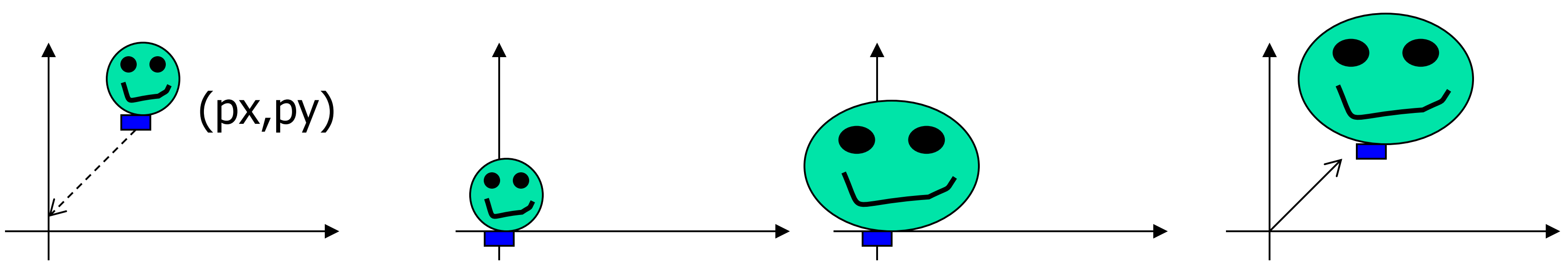
$$\begin{matrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{matrix}$$



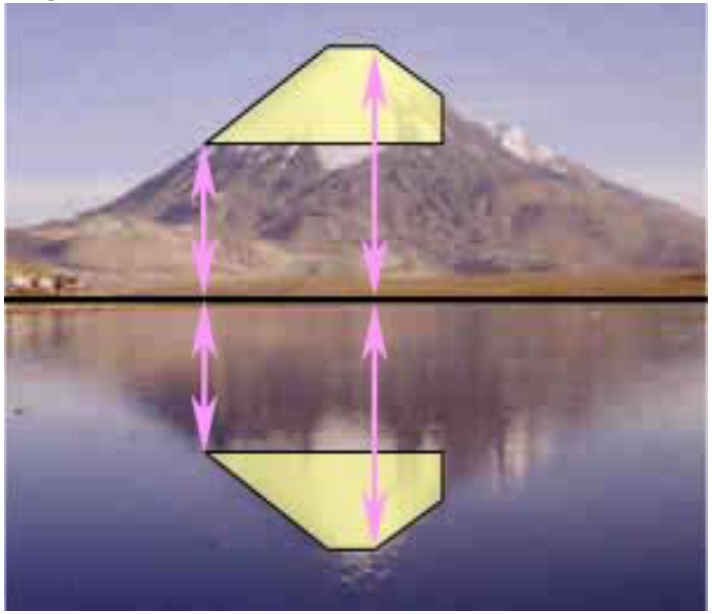
- What if I want to scale about an arbitrary pivot point?

Arbitrary Scaling Pivot

- To scale about an arbitrary pivot point P (px,py):
  - Translate the object so that P will coincide with the origin: T(-px, -py)
  - Scale the object: S(sx, sy)
  - Translate the object back: T(px,py)

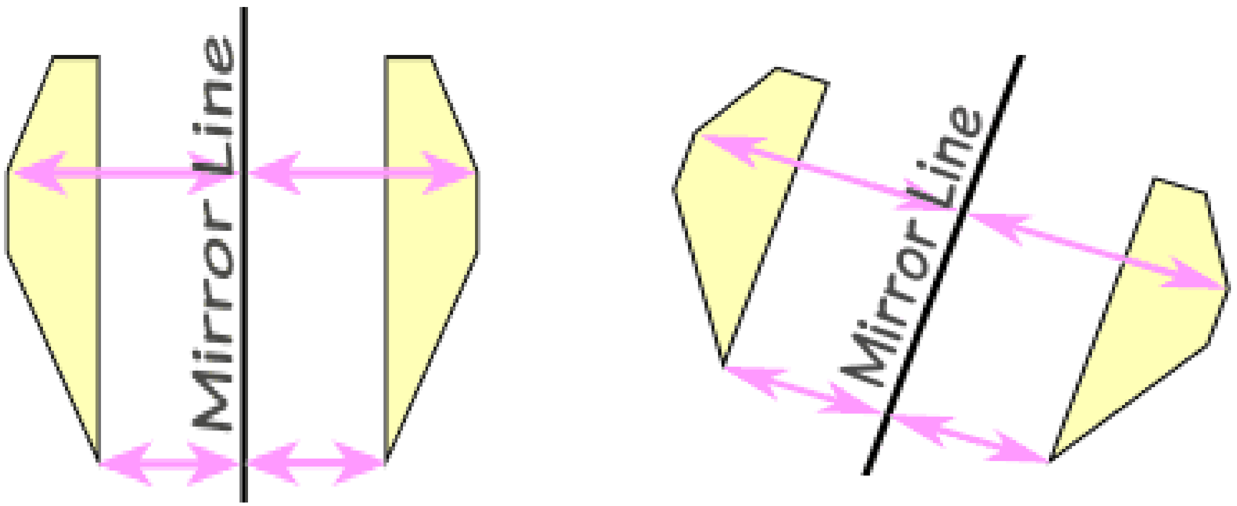


Reflection

- Reflections are everywhere ... in mirrors, glass, and here in a lake.
-  

- Every point is the same distance from the central line !
- The reflection has the same size as the original image
- The central line is called the **Mirror Line**

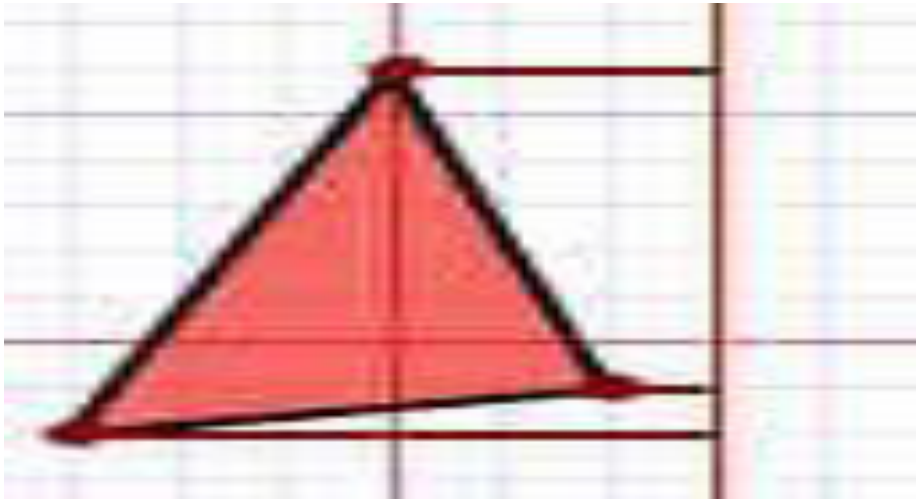
- Mirror Lines can be in **any direction**.



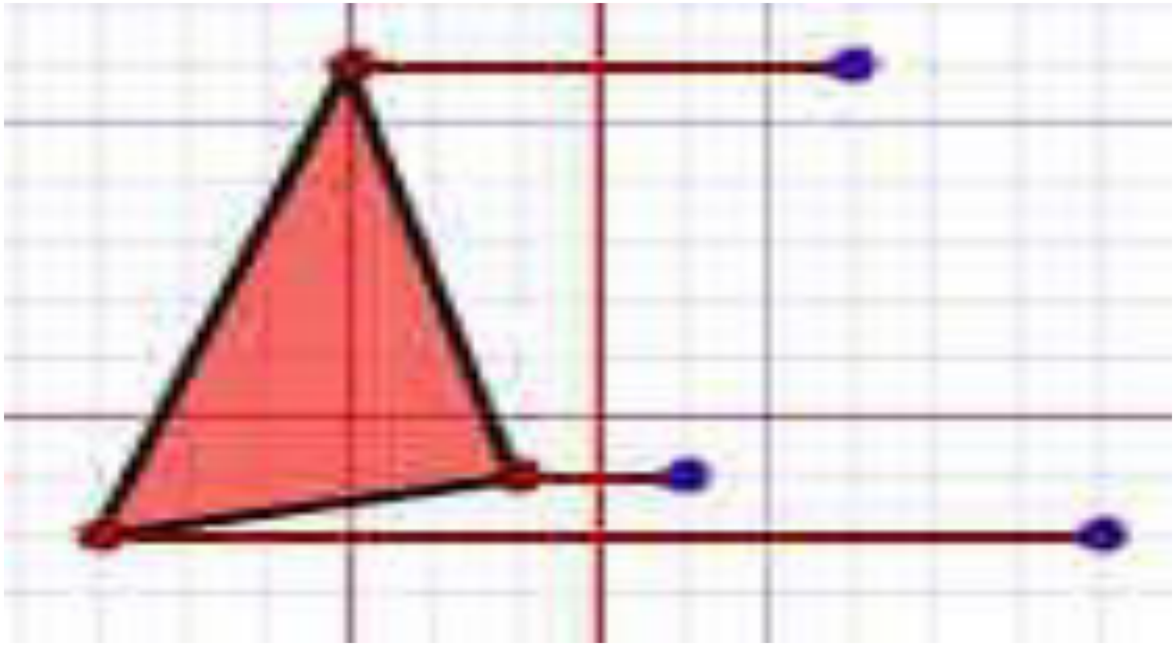
the **reflected image** is always the same size, it just **faces the other way**

How to do it?

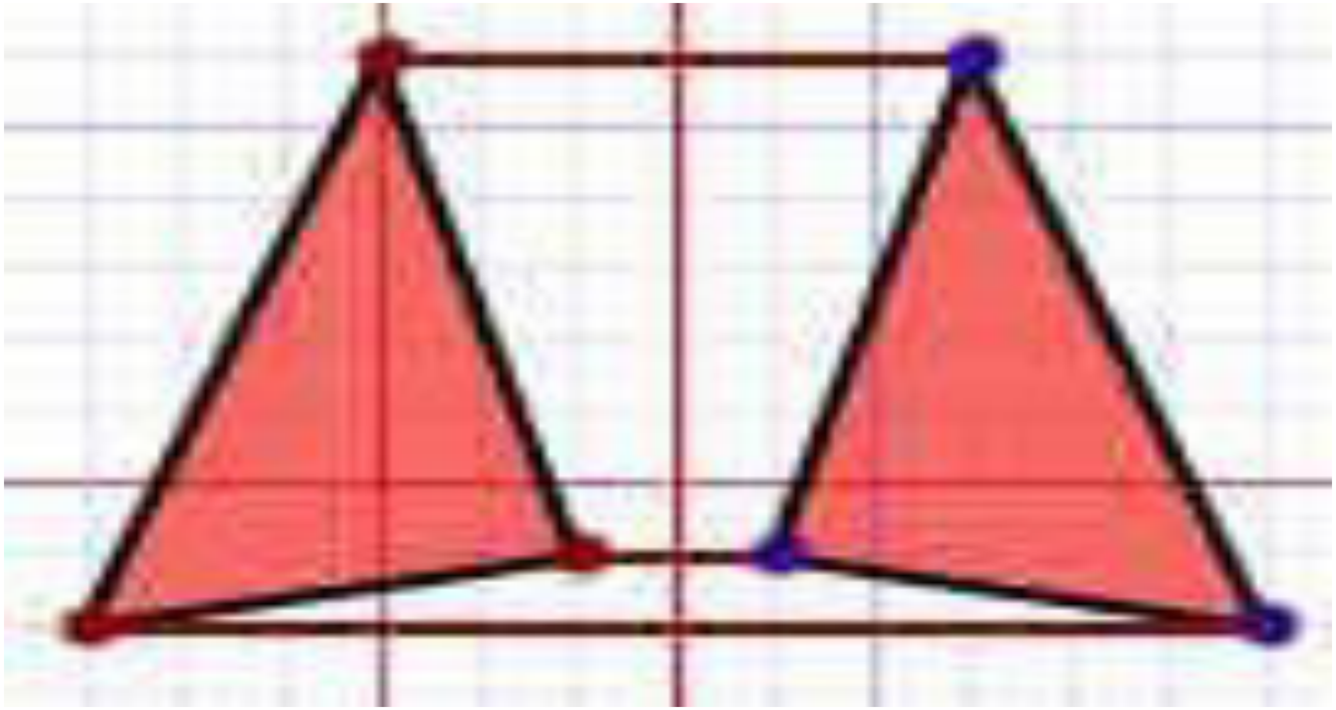
- 1. Measure from the point to the mirror line (must hit the mirror line at a right angle)



- 2. Measure the same distance again on the other side and place a dot.

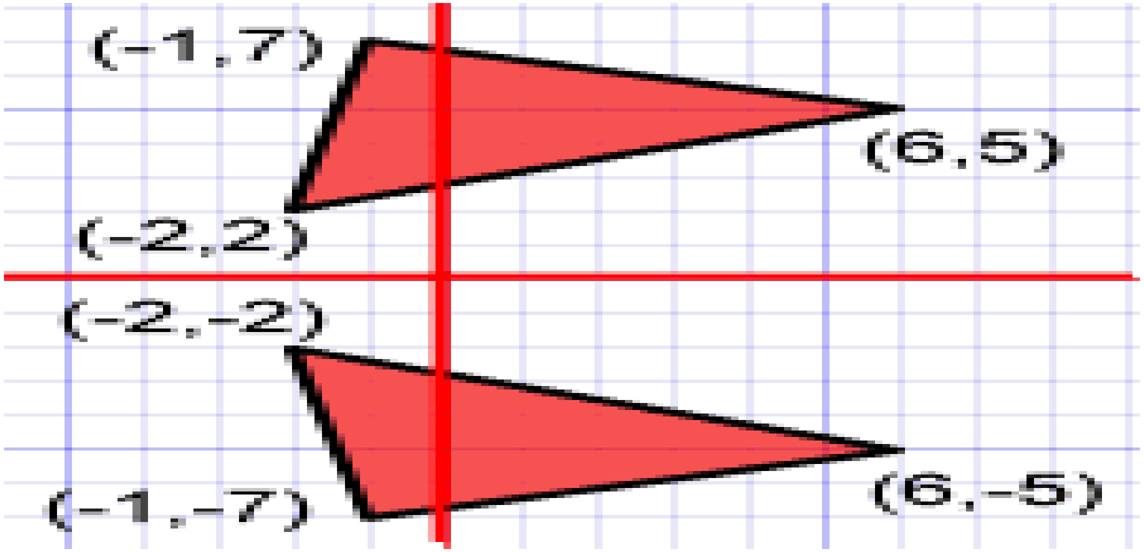


- 3. Then connect the new dots up!



Reflection about X-axis

- If the mirror line is the x-axis, just change each (x,y) into (x,-y)

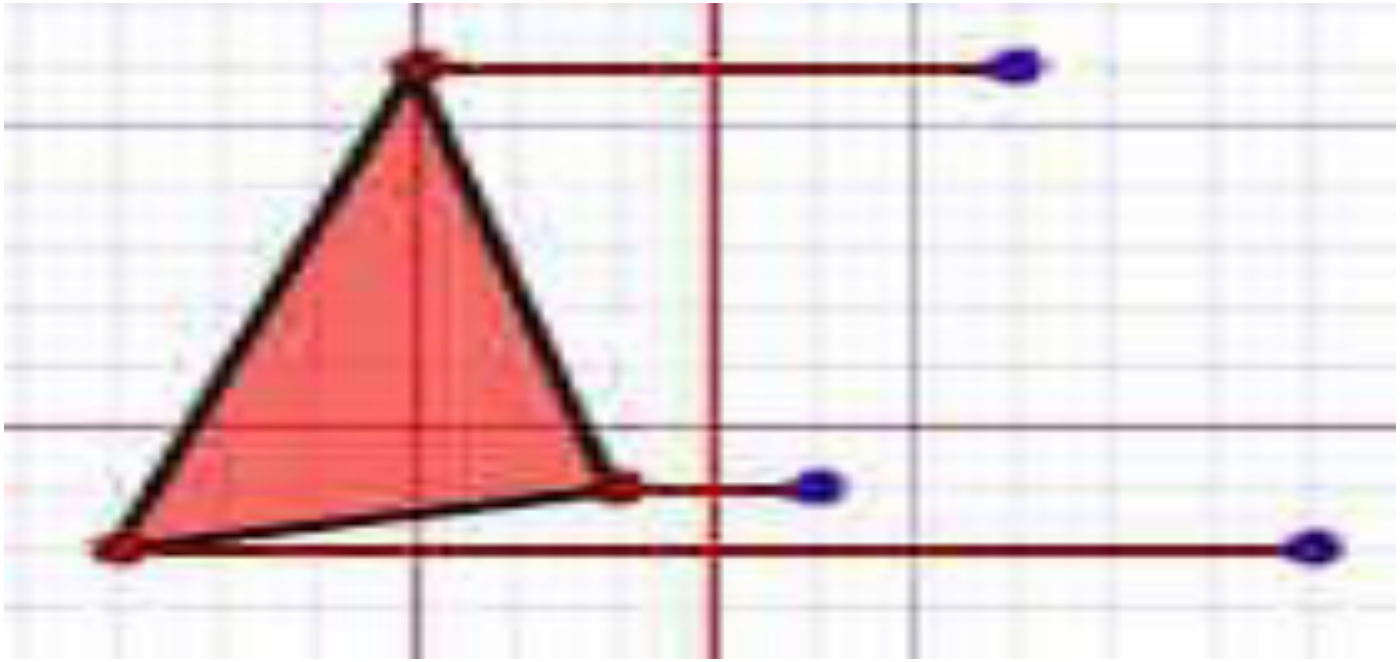


Matrix for reflection about line y=0 / x-axis.

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

Reflection about Y-axis

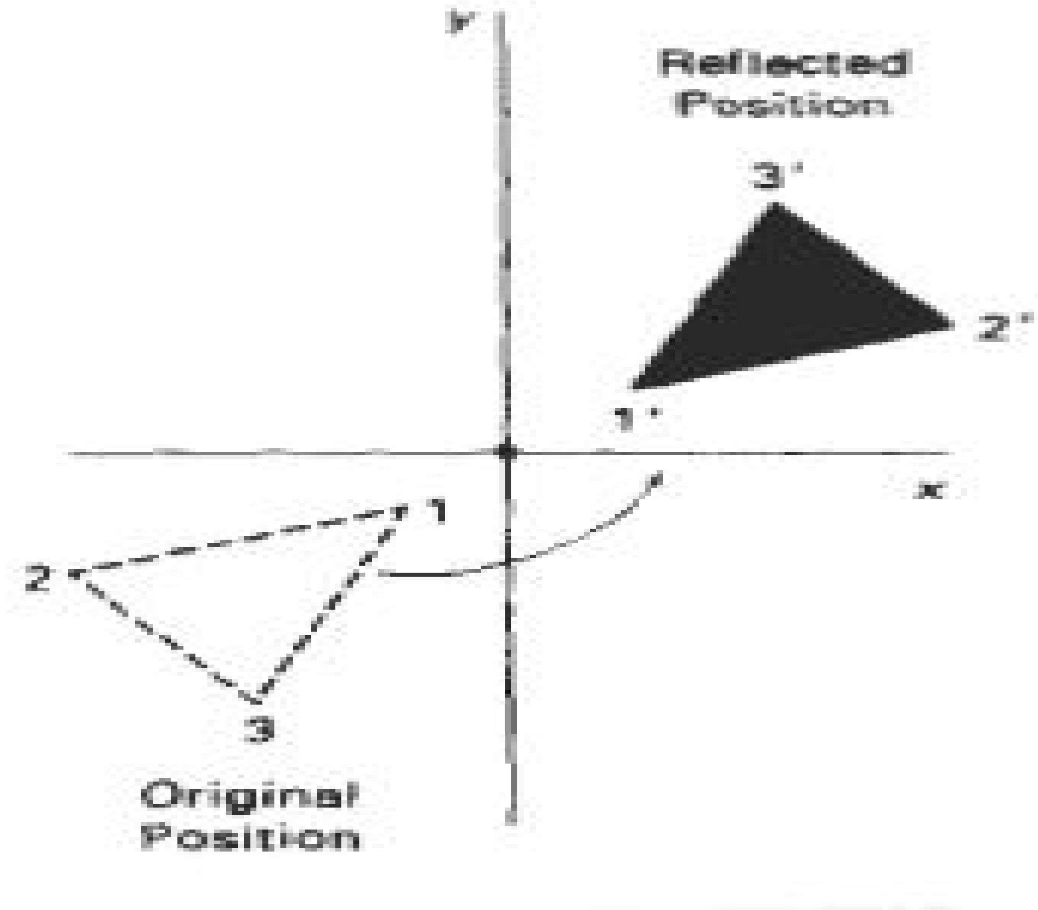
- If the mirror line is the y-axis, just change each (x,y) into (-x,y)



Reflection about line x=0 or y-axis

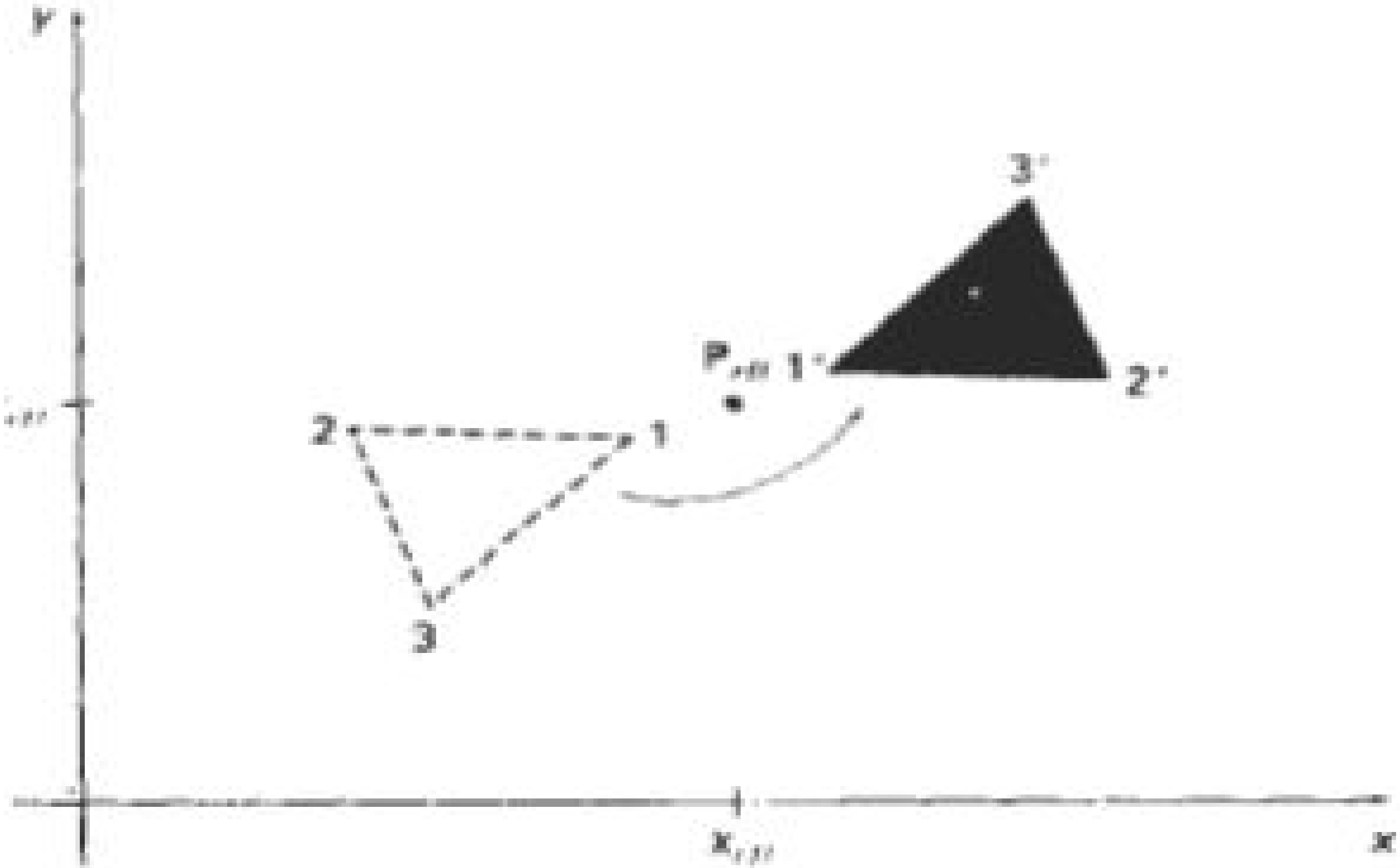
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ref. wrt line perpendicular to xyplane & passing through origin

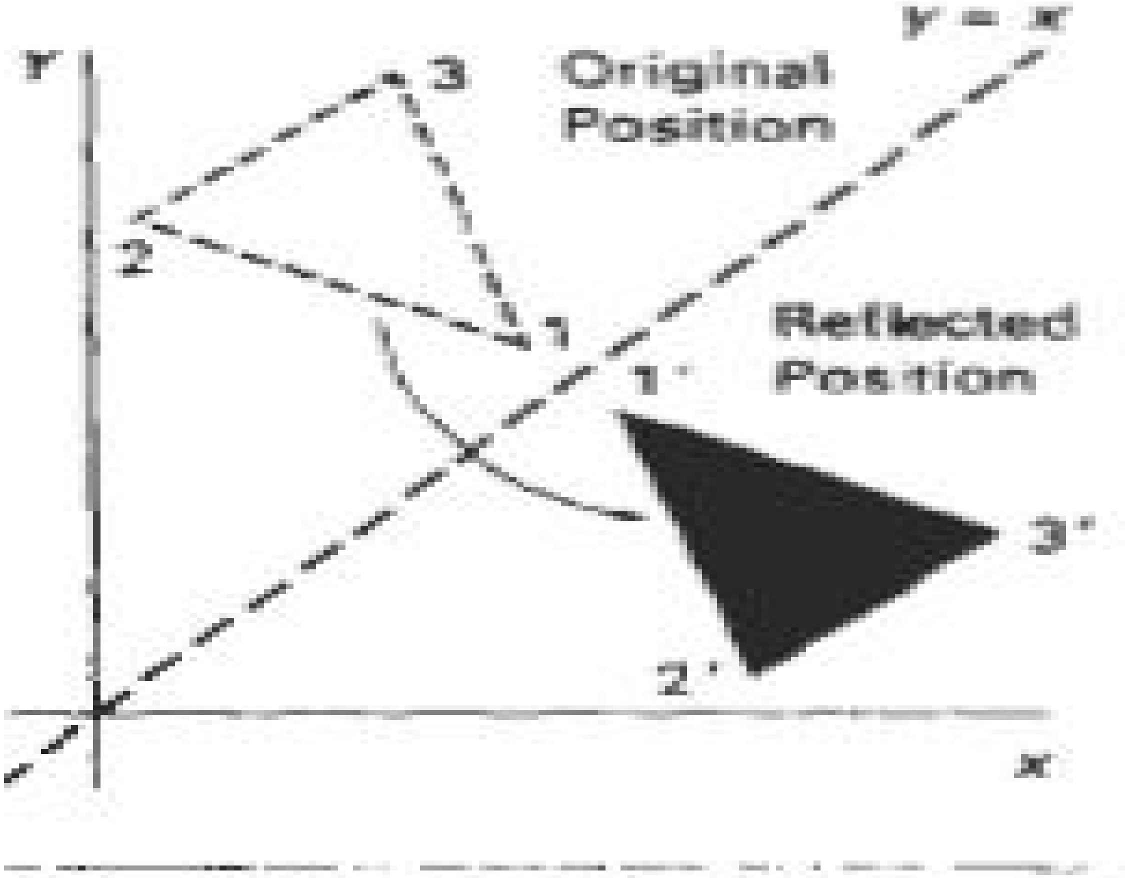


$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ref. wrt axis perp. to xy plane & reference point



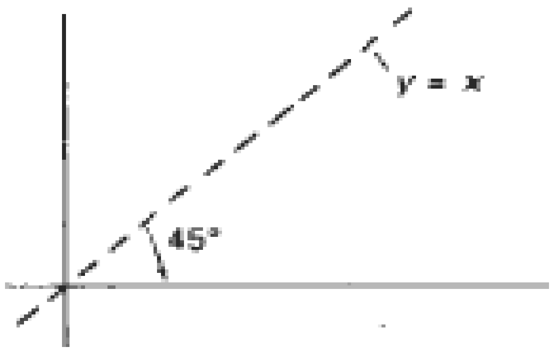
Ref. wrt line y = x

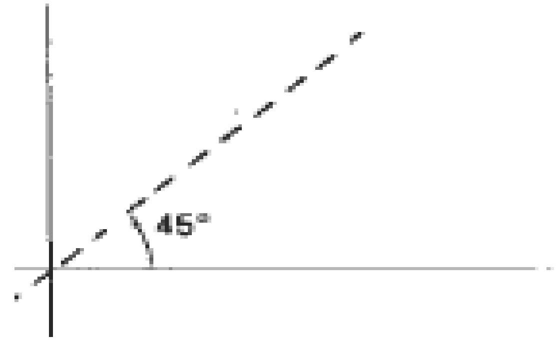


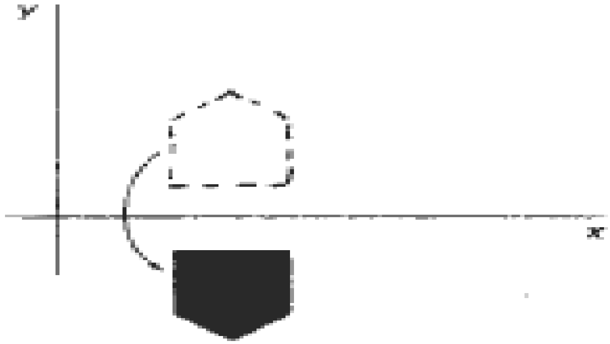
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- The said matrix is derived by following sequence –

- Perform a clockwise rotation by an angle 45.[Onto x-axis]
- Perform reflection wrt x-axis.
- Rotate the line y=x back to its original position through a counterclockwise rotation of 45.







(b)

Ref. wrt line y = -x

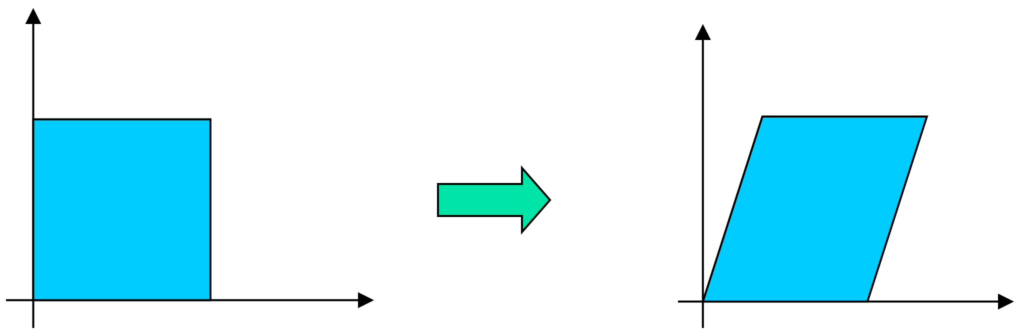
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shearing

- A transformation that distorts the shape of an object such that the transformed shape appears as if the object were composed of internal layers that had been caused to slide over each other.

  1) Shift coordinate x values

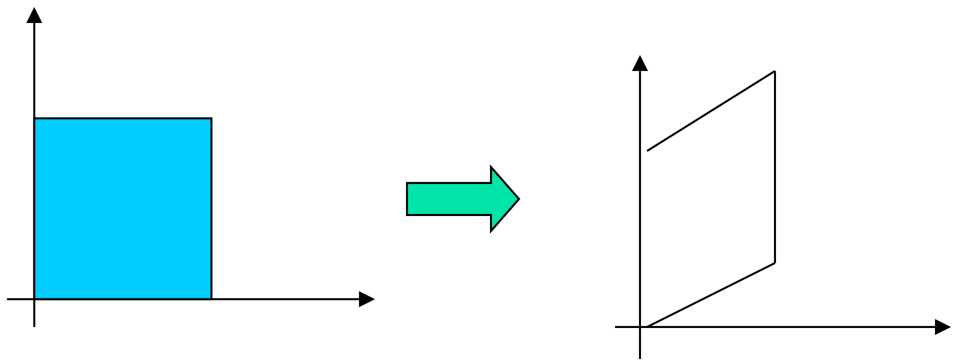
  2) Shift coordinate y values



- Y coordinates are unaffected, but x coordinates are translated linearly with y
- That is:
  - $y' = y$
  - $x' = x + y * h$

$$\begin{vmatrix} x \\ y \\ 1 \end{vmatrix} = \begin{vmatrix} 1 & h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

Shearing in y



$$\left|\begin{matrix} x \\ y \\ 1 \end{matrix}\right| = \left|\begin{matrix} 1 & 0 & 0 \\ g & 1 & 0 \\ 0 & 0 & 1 \end{matrix}\right| * \left|\begin{matrix} x \\ y \\ 1 \end{matrix}\right|$$

Interesting Facts:

- Any 2D rotation can be built using three shear transformations.
- Shearing will not change the area of the object
- Any 2D shearing can be done by a rotation, followed by a scaling, and followed by a rotation

Affine Transformation

- Translation, Scaling, Rotation, Shearing are all affine transformations
- Affine transformation – transformed point P' (x',y') is a linear combination of the original point P (x,y), i.e.

$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} m11 & m12 & m13 \\ m21 & m22 & m23 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

- Any 2D affine transformation can be decomposed into a rotation, followed by a scaling, followed by a shearing, and followed by a translation.

Affine matrix = translation x shearing x scaling x rotation