ΗΥ416
ΓΡΑΦΙΚΑ ΥΠΟΛΟΓΙΣΤΩΝ

Γεωμετρικοί Μετασχηματισμοί

Π. ΤΣΟΜΠΑΝΟΠΟΥΛΟΥ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Κύρια σημεία

- Βασικοί 2D γεωμετρικοί μετασχηματισμοί
- Αναπαραστάσεις με ομογενείς συντεταγμένες
- Αντίστροφοι μετασχηματισμοί
- Σύνθετοι μετασχηματισμοί 2D
- Άλλοι 2D μετασχηματισμοί
- Γεωμετρικοί μετασχηματισμοί σε 3D χώρο
- Προβολές

The Graphics Pipeline

Τοπικές συντεταγμένες

Μετασχηματισμός μοντελοποίησης (GTM)

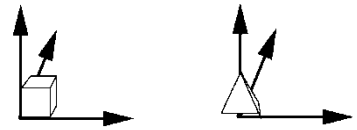
Παγκόσμιες συντεταγμένες

Παγκόσμιες συντεταγμένες

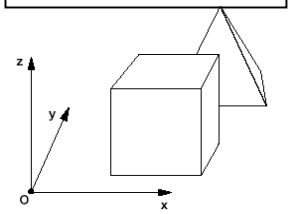
Πίνακας M

Συντεταγμένες παρατηρητή

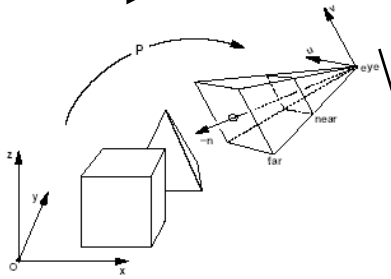
Ορισμός αντικειμένου (Object definition)



Σύνθεση σκηνής, ορισμός κάμερας



Φωτισμός (Lighting)



Πίνακας Q

Κανονική Παράλληλη (normalised device Coordinates)

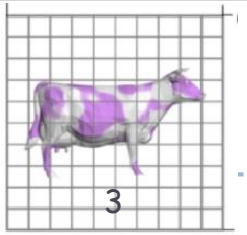
Drop the z and Map to view-port

Προβολή

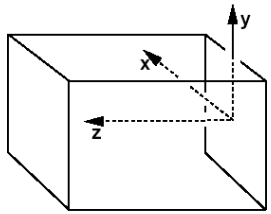
Παράλληλη προβολή

Κανονική Προοπτική (canonical Perspective)

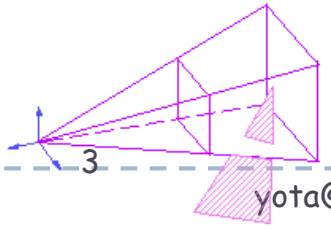
Σύνολο pixels



Rasterisation, Lighting*, Visibility*

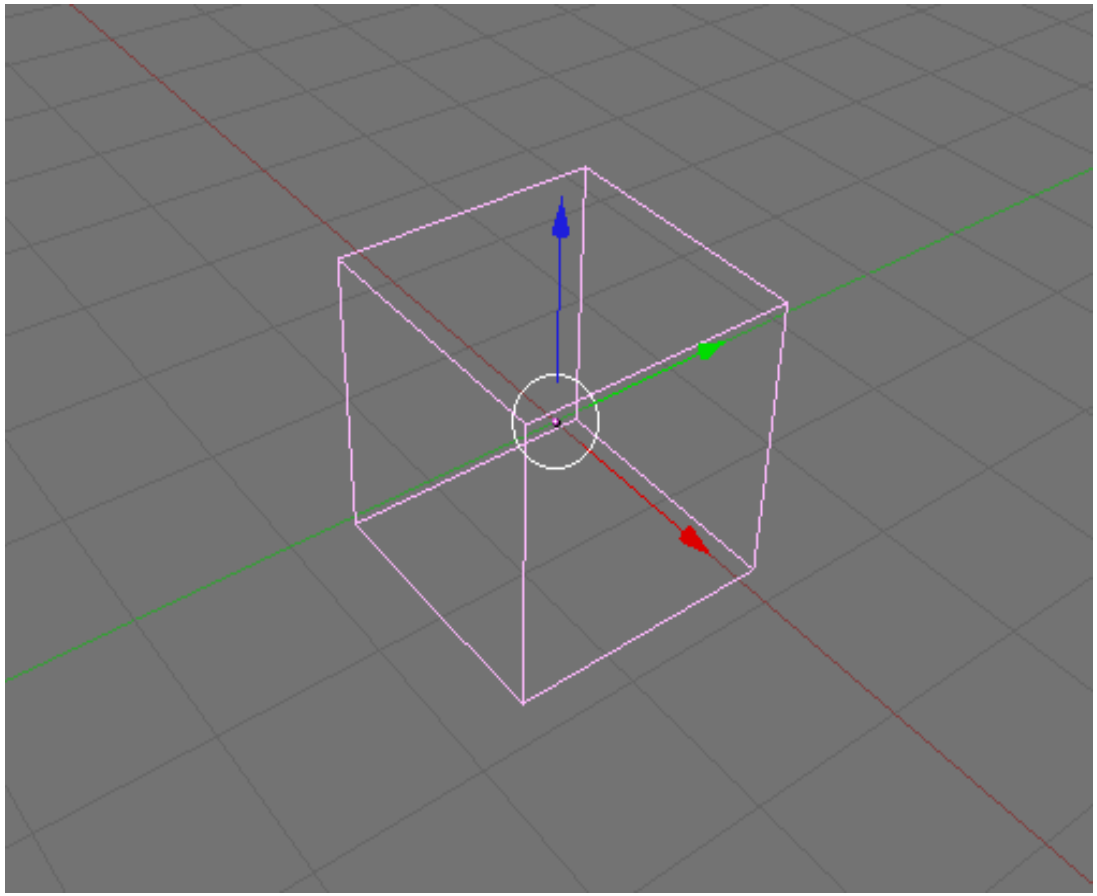


αποκοπή (clipping)



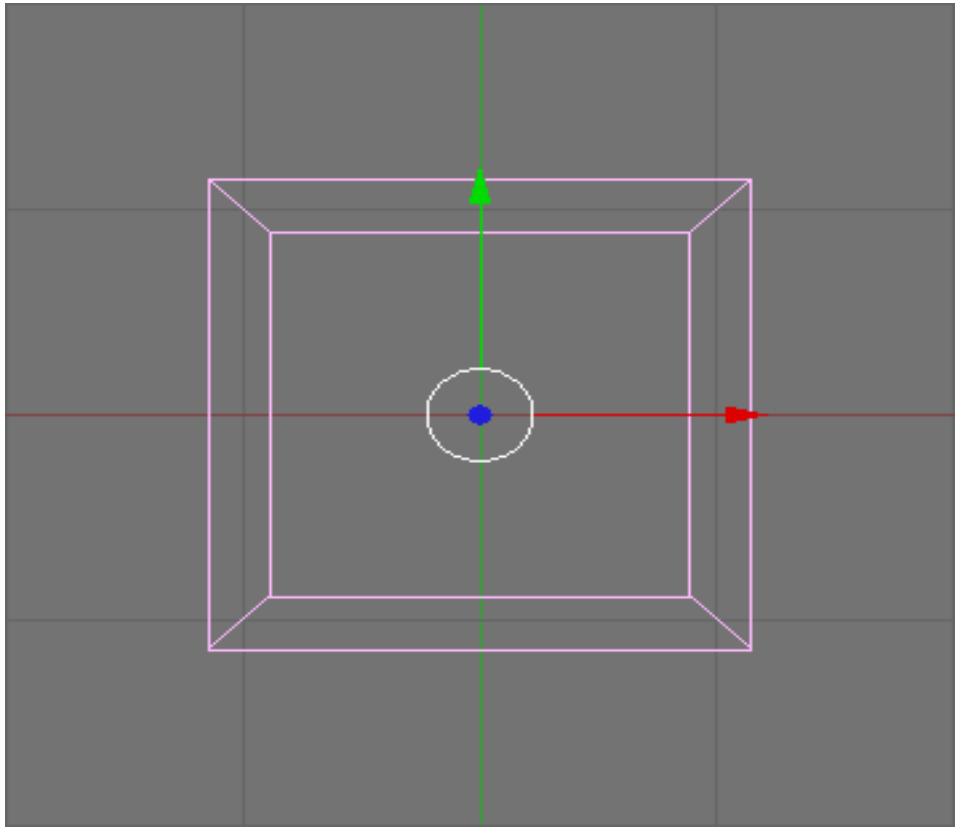
Graphics coordinate systems

- X is red
- Y is green
- Z is blue



Graphics coordinate systems

- If you are on the +z axis, and +y is up, then +x is to the right
- Math fields have +x going to the left



What is a Transformation?

- Maps points (x, y) in one coordinate system to points (x', y') in another coordinate system

$$x' = ax + by + c$$
$$y' = dx + ey + f$$

Basic Three Classes of Transformations

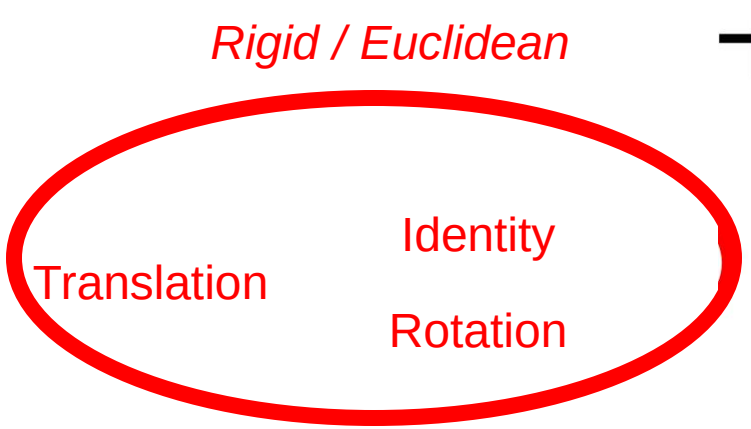
- Rigid Body (ανελαστικοί μετασχηματισμοί ή μετασχηματισμοί στερεών σωμάτων) – Preserves Distance and angles
  - Translation (μετατόπιση) and Rotation (περιστροφή)
- Conformal (σύμμορφος) – Preserves Angles
  - Translation, rotation, and uniform scaling
- Affine (συσχετισμένοι) – Preserves parallelism – Lines remain lines
  - Translation (μετατόπιση), rotation (περιστροφή), scaling (αλλαγή κλίμακας), shear (στρέβλωση), and reflection (κατοπτρισμός)

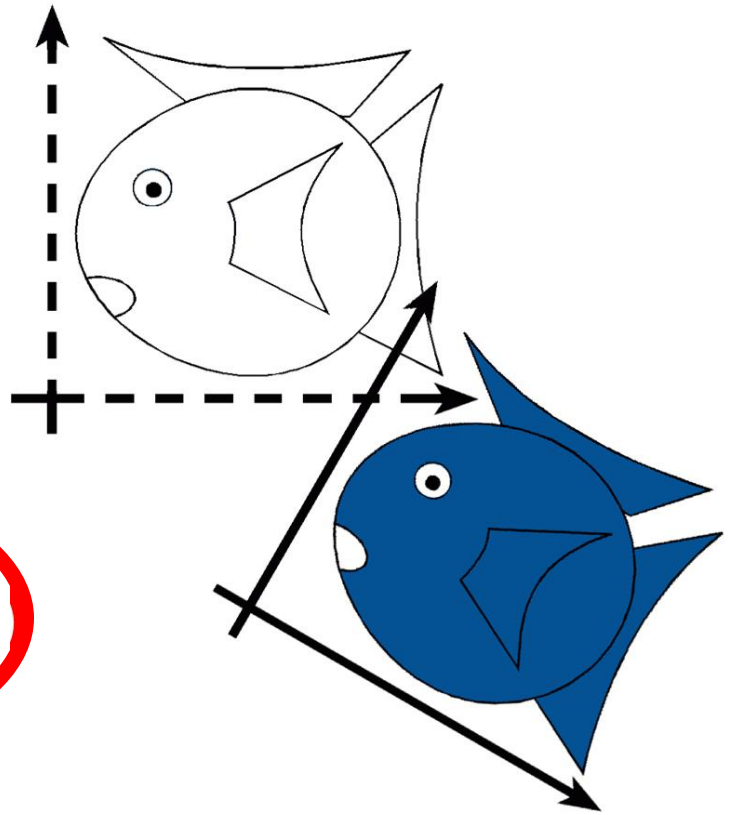
More Classes of Transformations

- Projective Transformations (μετασχηματισμοί προβολών)
  - Parallel (orthographic and oblique) projections
  - Perspective projections

Rigid-Body / Euclidean Transforms

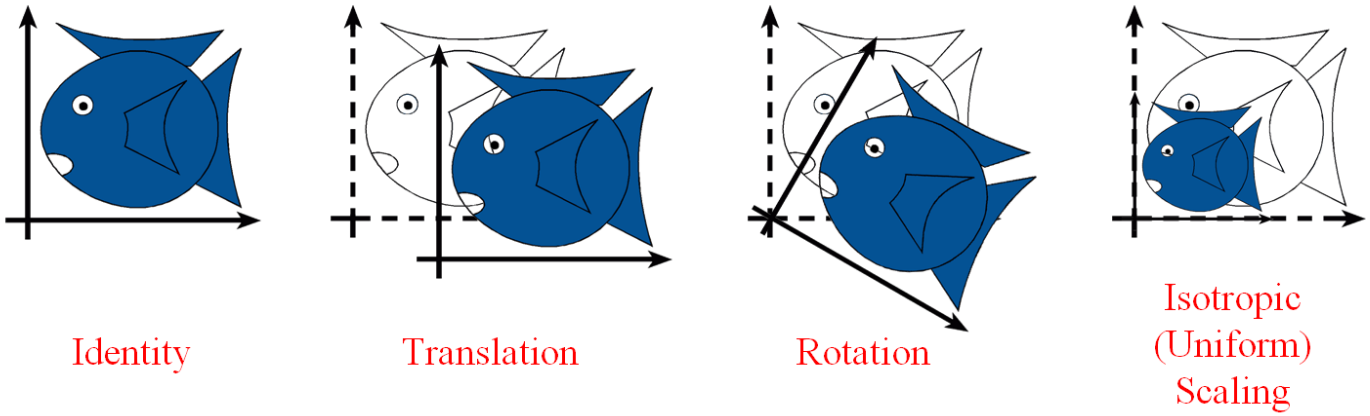
- Preserves distances
- Preserves angles





Transformations

- Simple transformations
  - Euclidean
  - Scaling



Transformations

- Deformable transformations
  - Shearing
  - Tapering
  - Twisting
  - etc..
- Issues
  - Can be combined
  - Are these operations invertible?

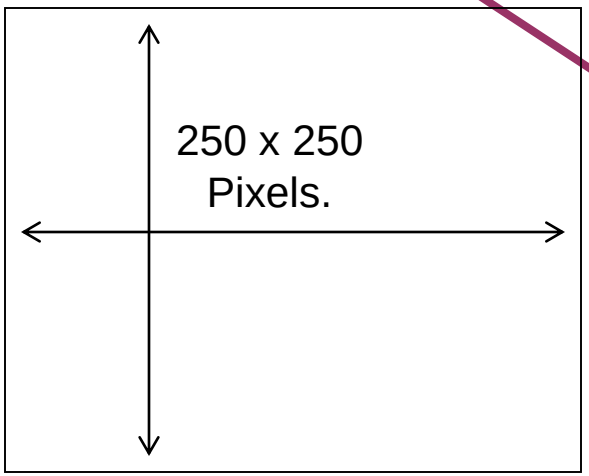
Transformations

- Why use transformations?
  - Position objects in a scene (modeling)
  - Change the shape of objects
  - Create multiple copies of objects
  - Projection for virtual cameras
  - Animations

Viewing in 2D - Viewport

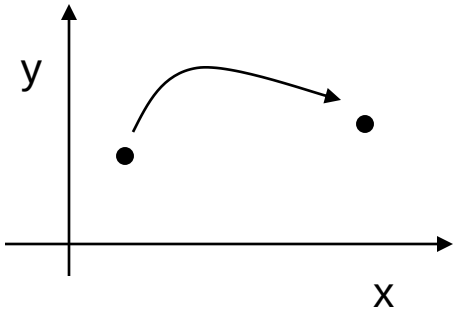
Window in world coordinates.

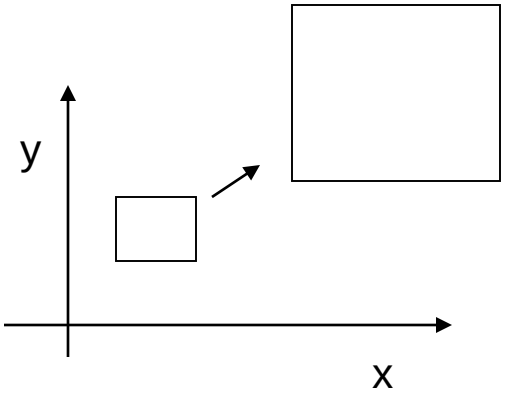
Viewport in Device coords

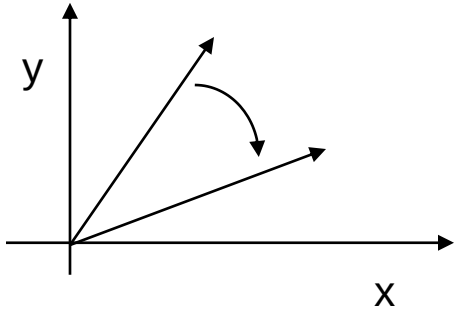


Simple Transformations

2D Transformations







2D Transformation

- Given a 2D object, transformation is to change the object's
  - Position (translation)
  - Size (scaling)
  - Orientation (rotation)
  - Shapes (shear)
- Apply a sequence of matrix multiplication to the object vertices

Point Representation

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

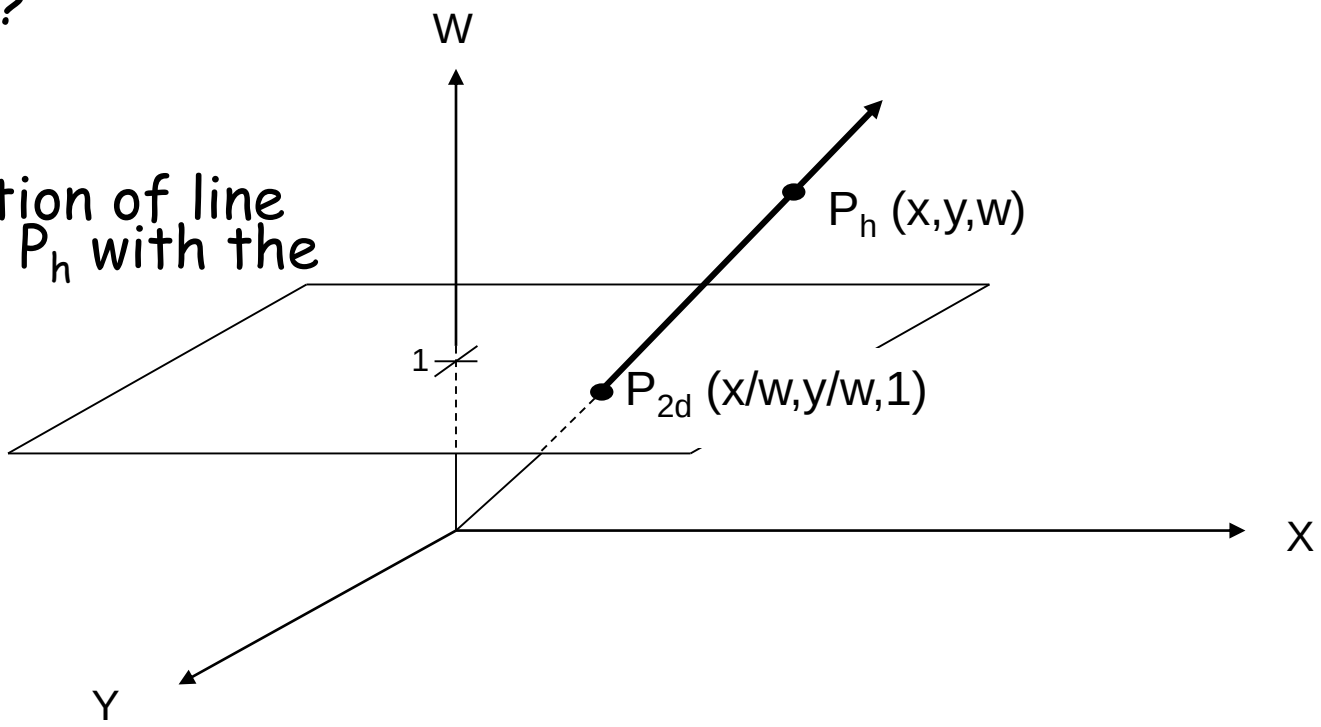
- Or in homogeneous coordinates as

$$\begin{vmatrix} X \\ y \\ 1 \end{vmatrix}$$

2-d Homogeneous coordinates

- What is $\begin{bmatrix} x \\ y \\ w \end{bmatrix}$?

- P_{2d} is intersection of line determined by P_h with the $w = 1$ plane



- So an infinite number of points correspond to $(x, y, 1)$: they constitute the whole line (tx, ty, tw)

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!
- How to represent a vector (v_x,v_y)?

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!
- How to represent a vector (v_x,v_y) → $(v_x,v_y,0)$

How are Transforms Represented?

$$x' = ax + by + c$$

$$y' = dx + ey + f$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ d & e \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} c \\ f \end{pmatrix}$$

$$p' = M\,p + t$$

How are Transforms Represented?
(Homogeneous coordinates)

$$x' = ax + by + c$$

$$y' = dx + ey + f$$

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

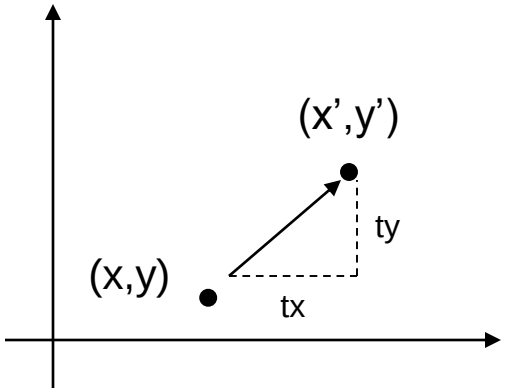
$$p' = M\,p$$

2D Translation

- Re-position a point along a straight line
- Given a point (x,y), and the translation distance (t_x,t_y)

The new point: (x', y')

$$x' = x + t_x$$
$$y' = y + t_y$$



or $P' = P + T$ where $P' = \begin{vmatrix} x' \\ y' \end{vmatrix}$ $p = \begin{vmatrix} x \\ y \end{vmatrix}$ $T = \begin{vmatrix} t_x \\ t_y \end{vmatrix}$

3x3 2D Translation Matrix

$$\begin{vmatrix} x' \\ y' \end{vmatrix} = \begin{vmatrix} x \\ y \end{vmatrix} + \begin{vmatrix} t_x \\ t_y \end{vmatrix}$$

Use 3 x 1 vector
(homogeneous coordinates)

$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

Note that now it becomes a matrix-vector multiplication

3D Translation

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

$$T(t_x, t_y, t_z) \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

Properties of Translation

$$T(0,0,0)\,\mathbf{v} = \mathbf{v}$$

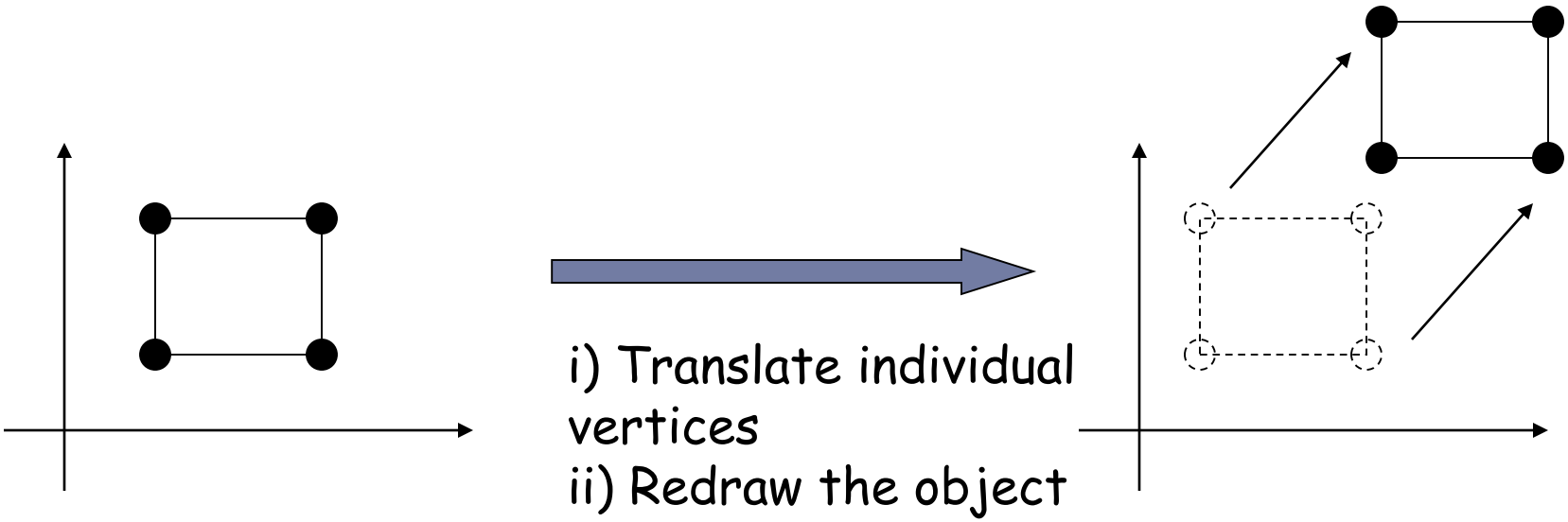
$$T(s_x, s_y, s_z)\,T(t_x, t_y, t_z)\mathbf{v} = T(s_x + t_x, s_y + t_y, s_z + t_z)\,\mathbf{v}$$

$$T(s_x, s_y, s_z)\,T(t_x, t_y, t_z)\mathbf{v} = T(t_x, t_y, t_z)\,T(s_x, s_y, s_z)\mathbf{v}$$

$$T^{-1}(t_x, t_y, t_z)\,\mathbf{v} = T(-t_x, -t_y, -t_z)\,\mathbf{v}$$

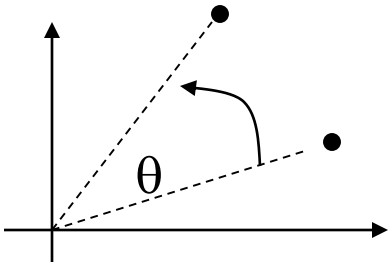
Translation of objects

- How to translate an object with multiple vertices?

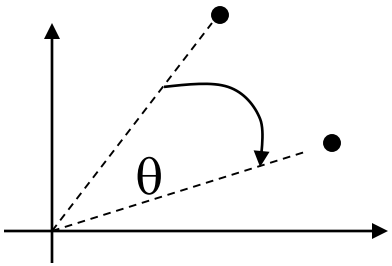


2D Rotation

- Default rotation center: Origin (0,0)



θ> 0 : Rotate counter clockwise



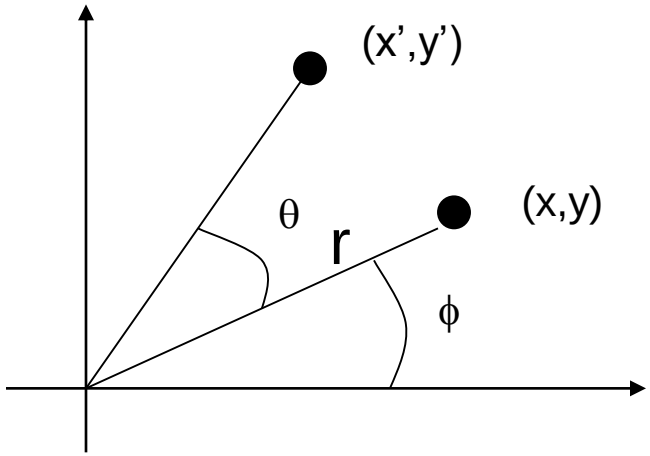
θ< 0 : Rotate clockwise

2D Rotation

(x,y) → Rotate *about the origin* by θ → (x', y')

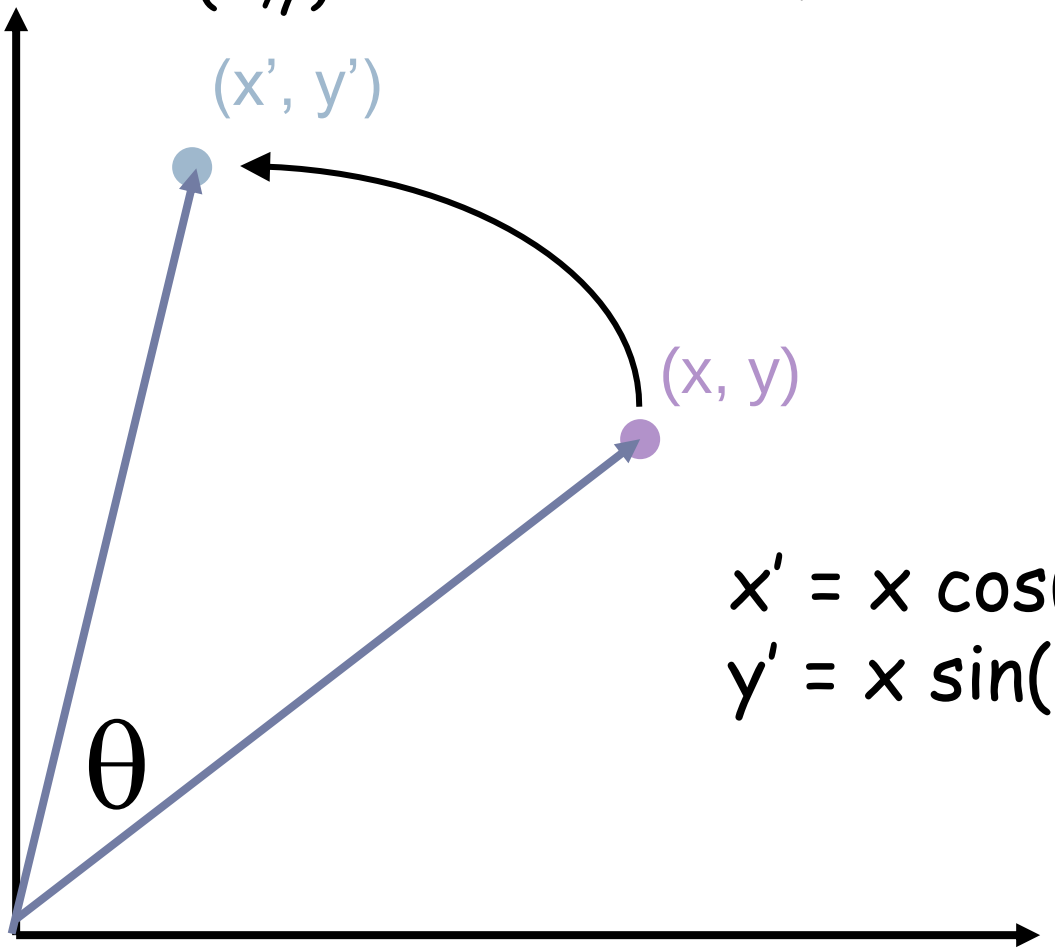
How to compute (x', y') ?

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



2-D Rotation

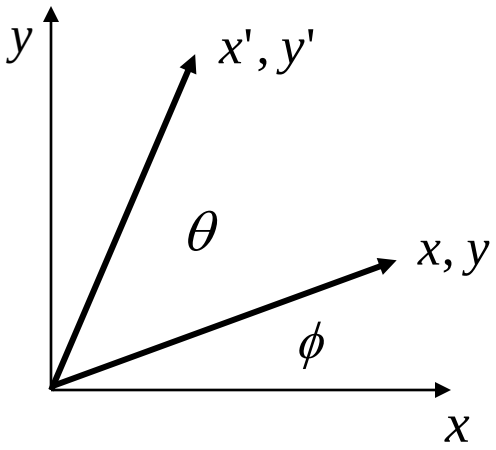
(x,y) → Rotate *about the origin* by θ → (x', y')



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

Rotations (2D)

How to compute (x', y') ?



$$x = r\cos\phi$$

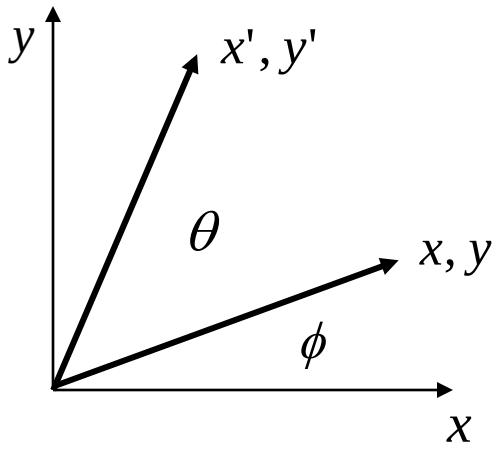
$$y = r\sin\phi$$

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

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

Rotations (2D)

How to compute (x', y') ?



$$x = r\cos\phi$$

$$y = r\sin\phi$$

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

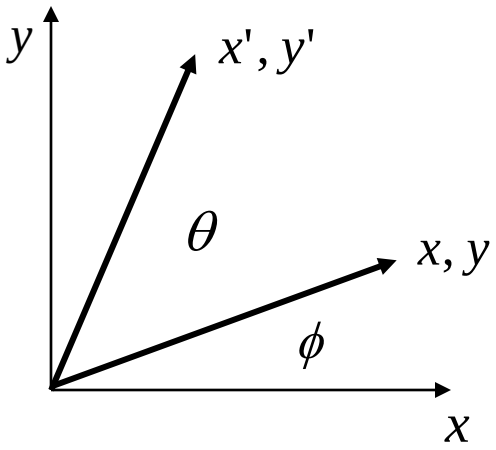
$$y' = r\sin(\phi+\theta)$$

$$\cos(\phi+\theta) = \cos\phi\cos\theta - \sin\phi\sin\theta$$

$$\sin(\phi+\theta) = \cos\phi\sin\theta + \sin\phi\cos\theta$$

Rotations (2D)

How to compute (x', y') ?



$$x = r\cos\phi$$

$$y = r\sin\phi$$

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

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

$$\cos(\phi+\theta) = \cos\phi\cos\theta - \sin\phi\sin\theta$$

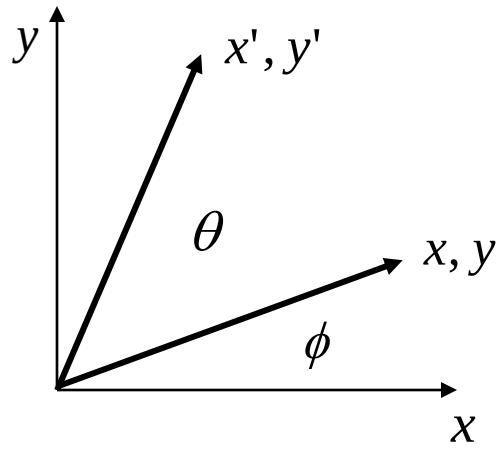
$$\sin(\phi+\theta) = \cos\phi\sin\theta + \sin\phi\cos\theta$$

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

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

Rotations (2D)

How to compute (x', y') ?



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

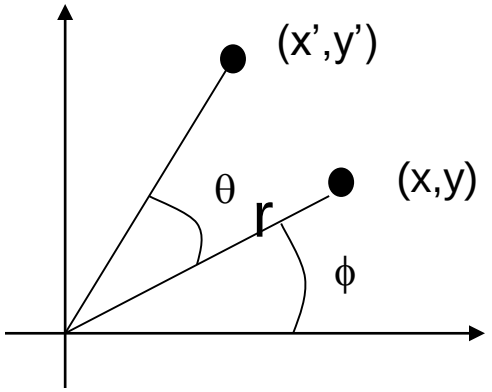
$$\cos(\phi+\theta) = \cos\phi\cos\theta - \sin\phi\sin\theta$$
$$\sin(\phi+\theta) = \cos\phi\sin\theta + \sin\phi\cos\theta$$

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

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

3x3 2D Rotation Matrix

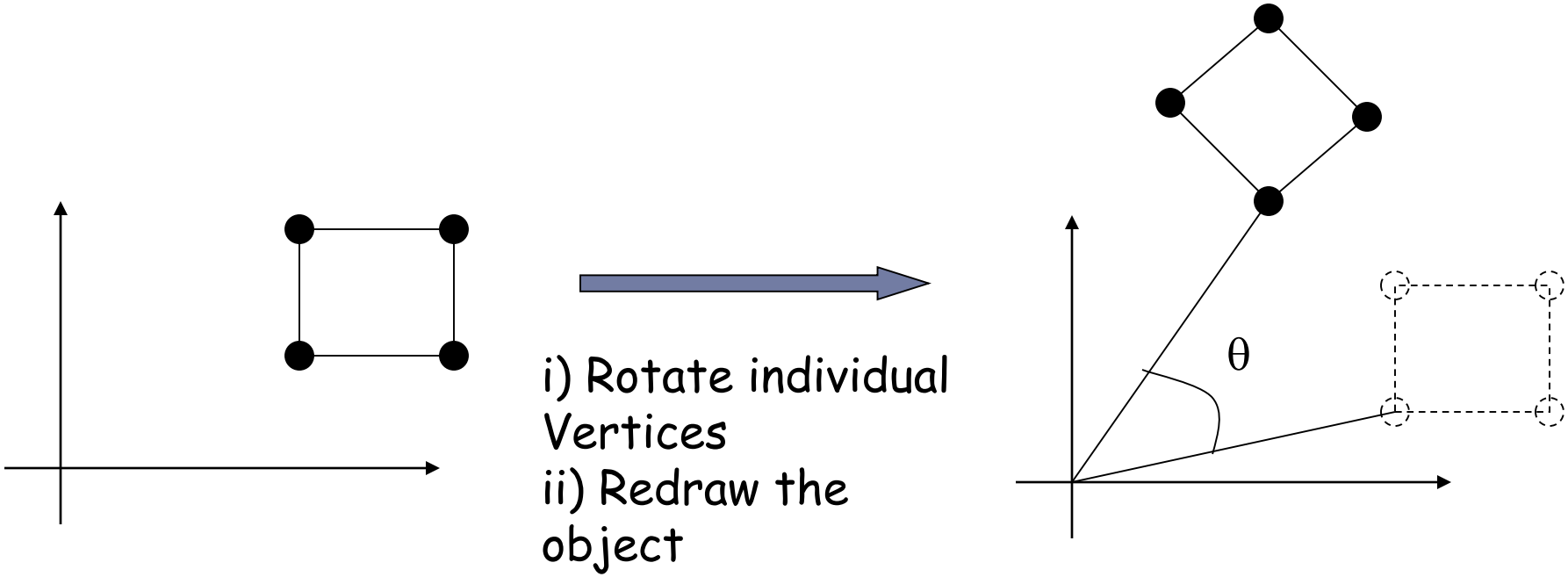
$$\begin{vmatrix} x' \\ y' \end{vmatrix} = \begin{vmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{vmatrix} \begin{vmatrix} x \\ y \end{vmatrix}$$



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

2D Rotation of objects

- How to rotate an object with multiple vertices?

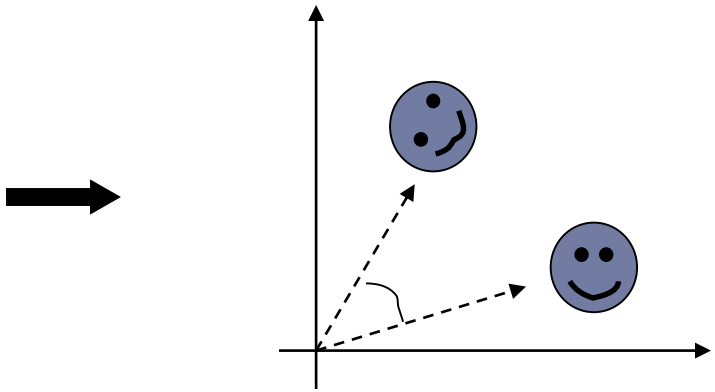


i) Rotate individual Vertices
ii) Redraw the object

More on Rotation

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

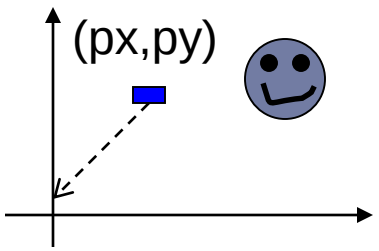
$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



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

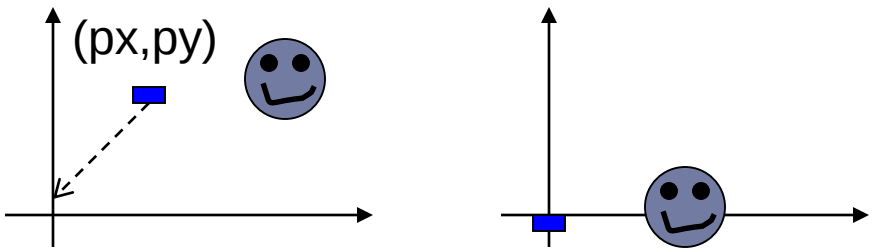
Arbitrary Rotation Center

- To rotate about an arbitrary point P (px,py) by θ:



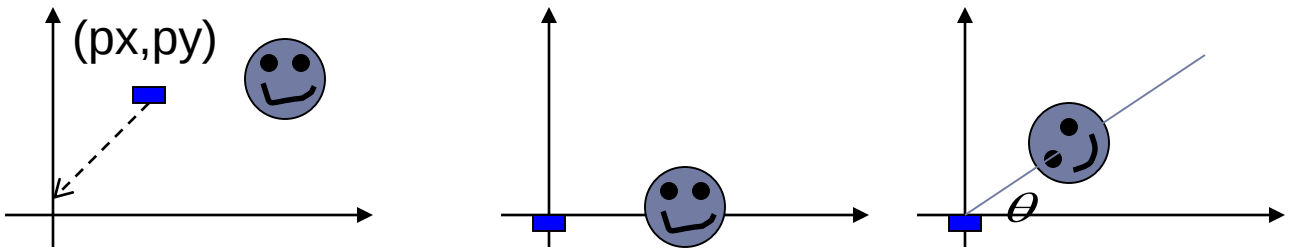
Arbitrary Rotation Center

- 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)



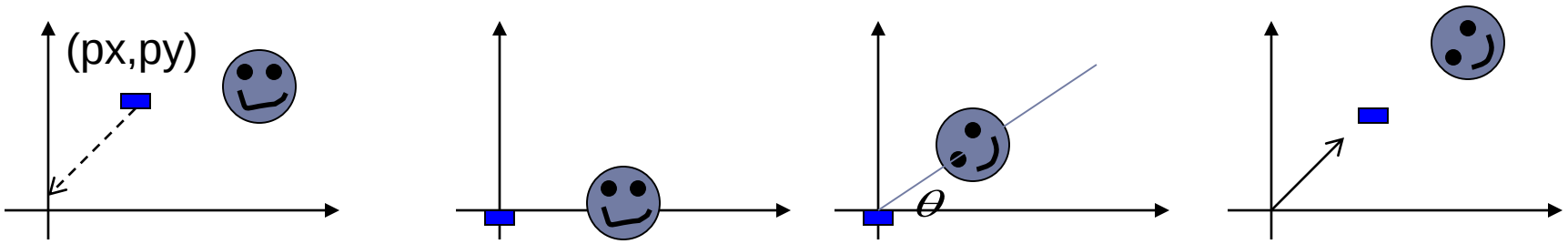
Arbitrary Rotation Center

- 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)



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(q)
- Translate the object back: T(px,py)

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

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