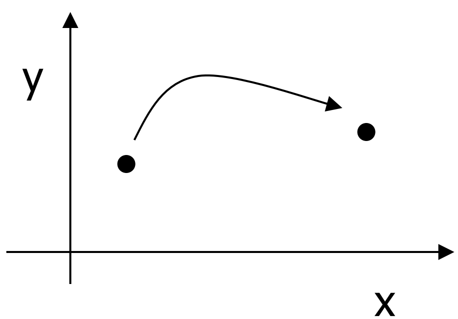
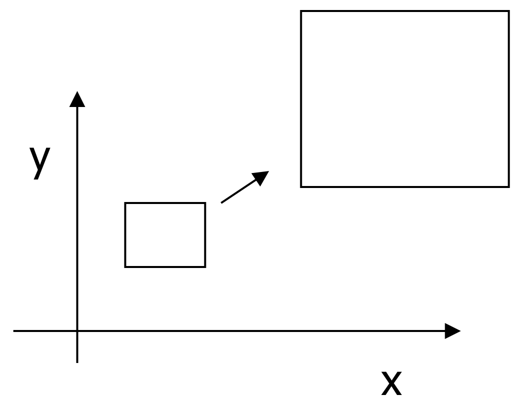
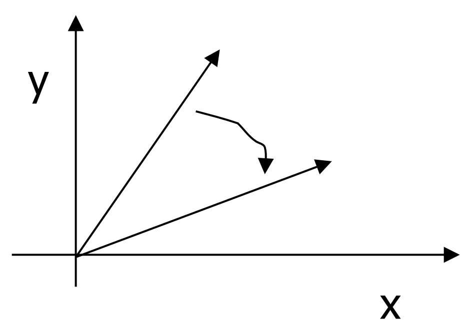
2D Transformations

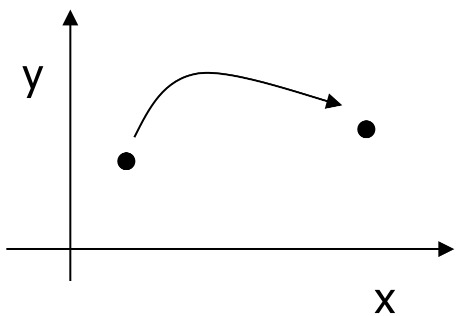
2D Transformations

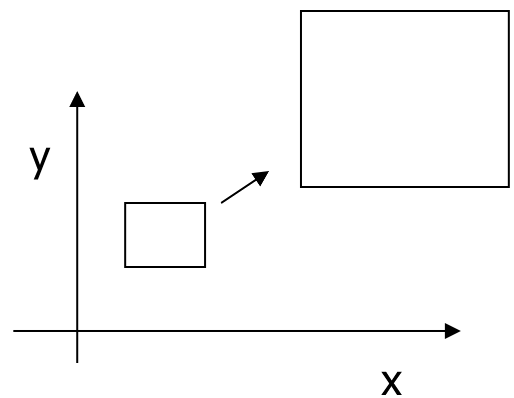


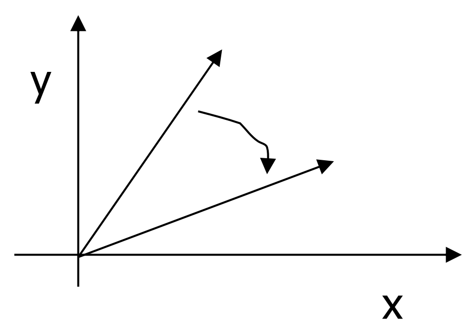




2D Transformations







OpenGl transformations:

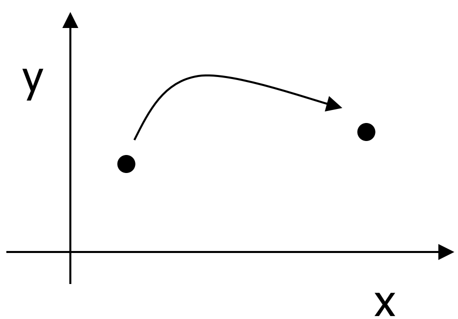
glTranslatef (tx, ty, tz);

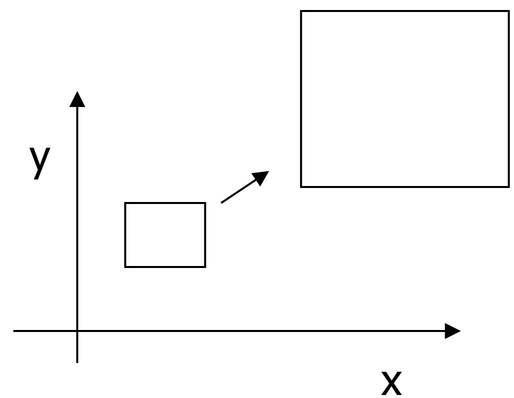
glRotatef (theta, vx, vy, vz)

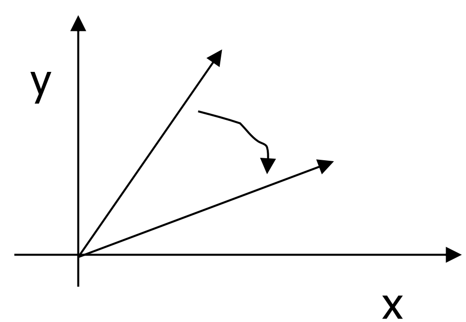
glScalef (sx,sy,sz)

……

2D Transformations







Applications:

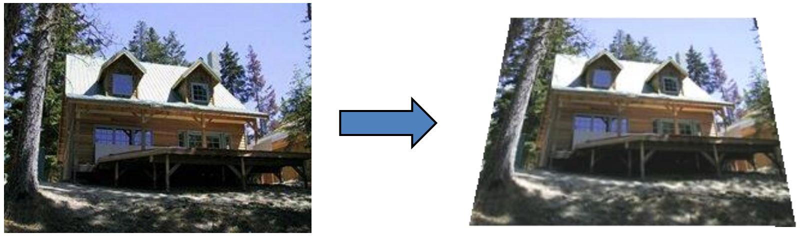
- Animation
- Image/object manipulation
- Viewing transformation
- etc.

Applications of 2D Transformations

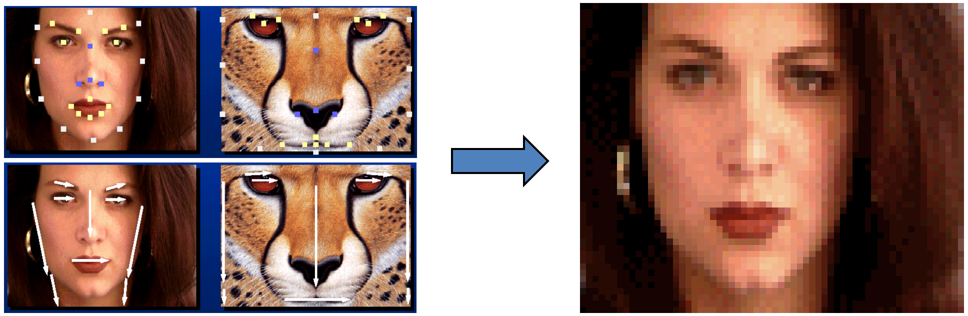
- 2D geometric transformations

- Animation ([demo](#) ,[demo](#))

- Image warping



- Image morphing



2D Transformation

- Required readings: HB 7-1 to 7-5, 7-8
- Given a 2D object, transformation is to change the object's
  - Position (translation)
  - Size (scaling)
  - Orientation (rotation)
  - Shapes (shear)
- Apply a sequence of matrix multiplications to the object vertices

Point Representation

- We can use a column vector (a 2x1 matrix) to represent a 2D point $\begin{vmatrix} x \\ y \end{vmatrix}$
- A general form of *linear* transformation can be written as:

$x' = ax + by + c$

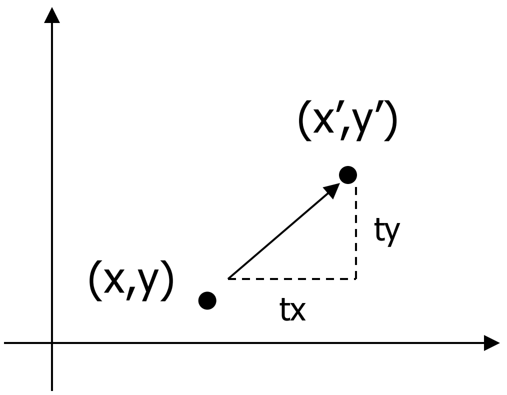
$y' = dx + ey + f$

OR

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

Translation

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



The new point: (x', y')

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

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

3x3 2D Translation Matrix

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

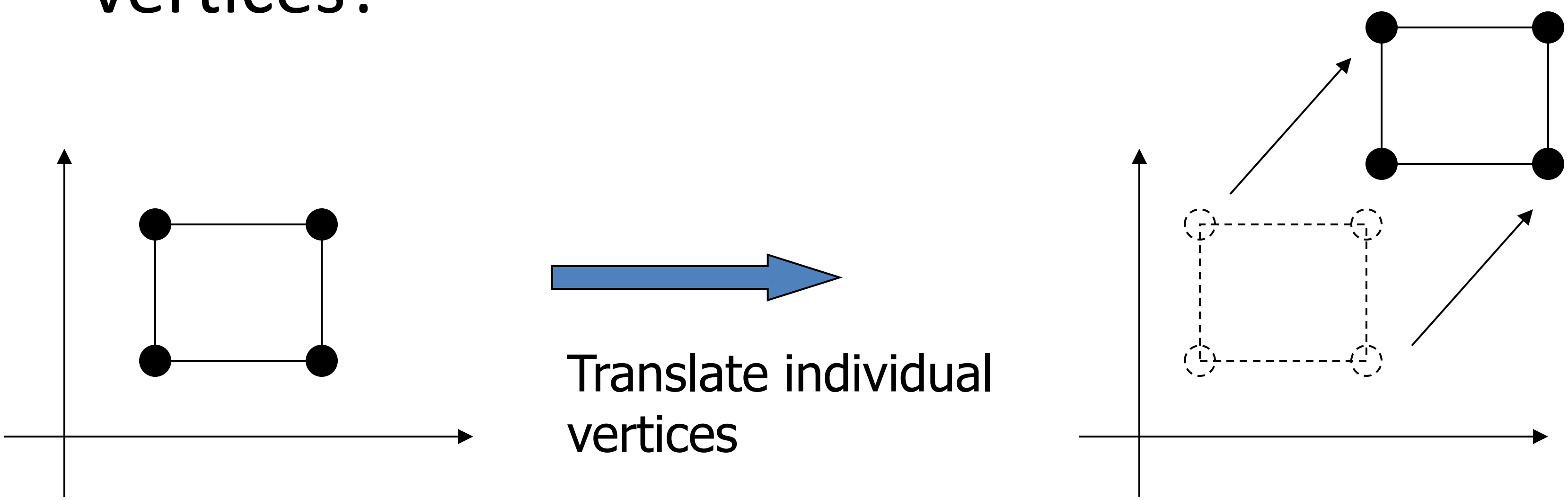
Use 3 x 1 vector

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

- Note that now it becomes a matrix-vector multiplication

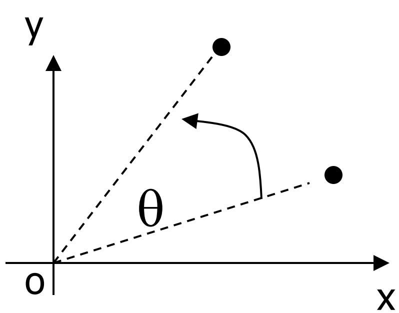
Translation

- How to translate an object with multiple vertices?

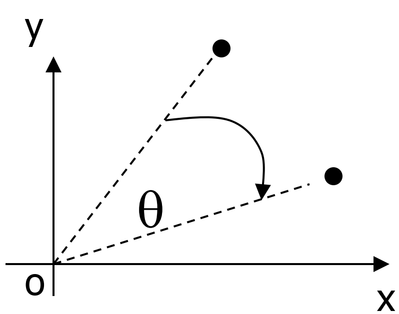


2D Rotation

- Default rotation center: Origin (0,0)



$\theta > 0$: Rotate counter clockwise



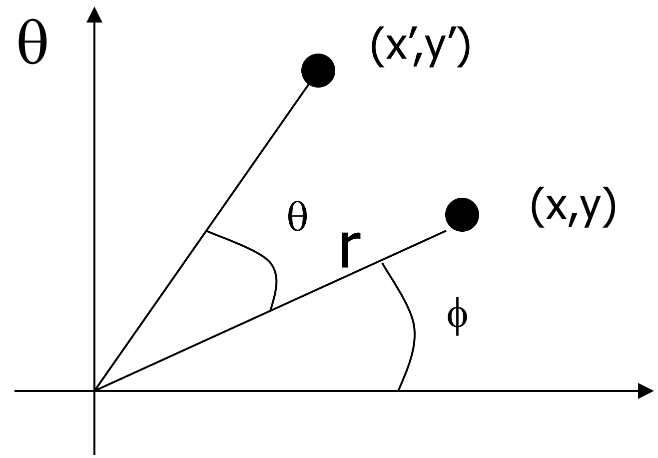
$\theta < 0$: Rotate clockwise

2D Rotation

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

$\longrightarrow$ (x', y')

How to compute (x', y') ?

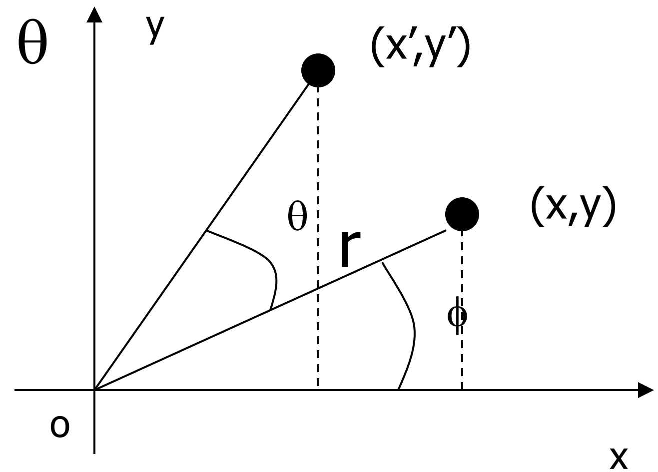


2D Rotation

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

$\longrightarrow$ (x′, y′)

How to compute (x′, y′) ?



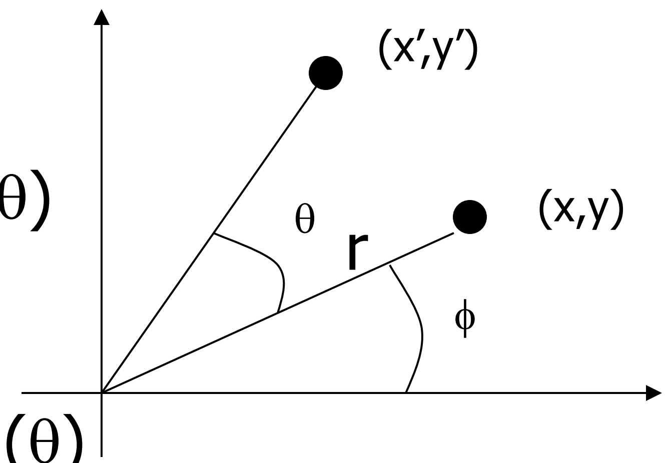
$x = r\cos(\phi) \quad y = r\sin(\phi)$

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

2D Rotation

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

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

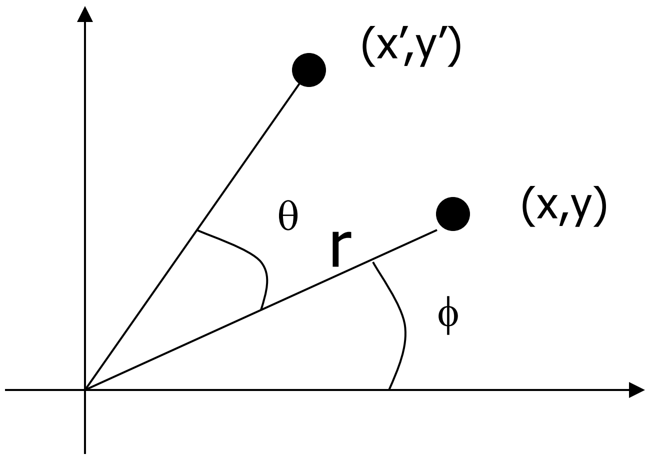


$$
\begin{aligned}
x' &= r\cos(\phi + \theta) \\
&= r\cos(\phi)\cos(\theta) - r\sin(\phi)\sin(\theta) \\
&= x\cos(\theta) - y\sin(\theta)
\end{aligned}
$$

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

2D Rotation

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



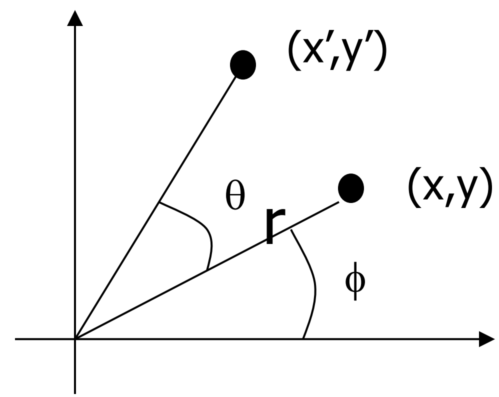
Matrix form?

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

3 x 3?

3x3 2D Rotation Matrix

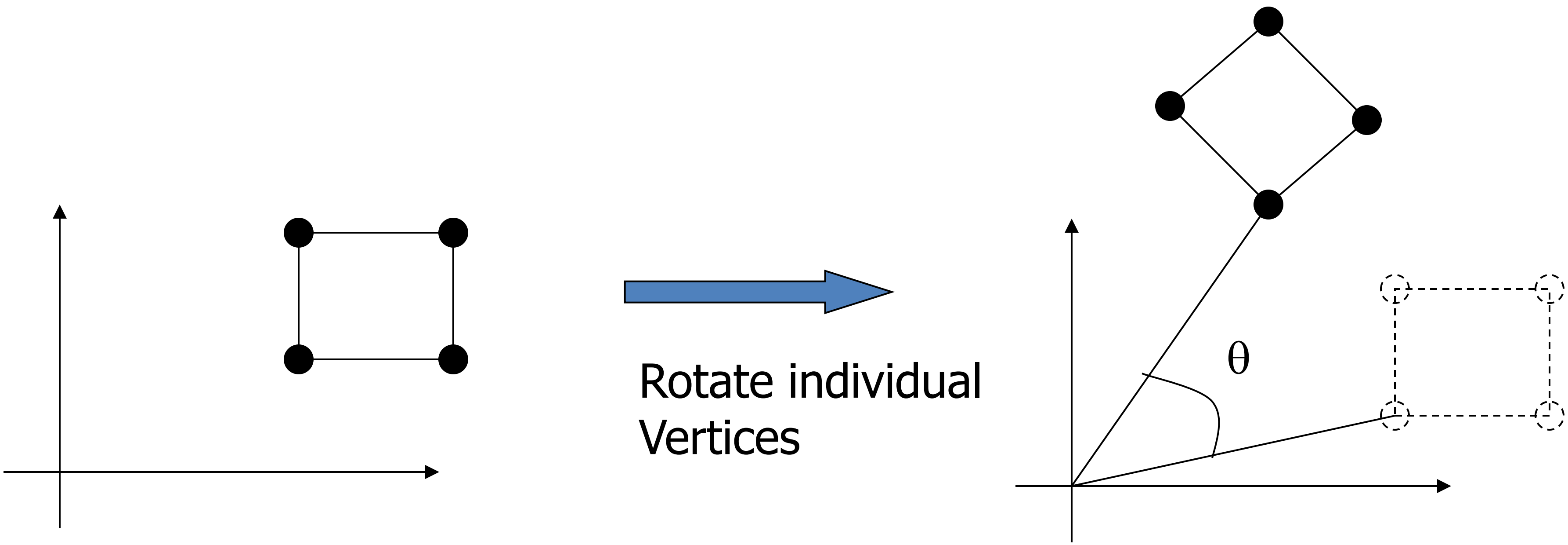
$$\begin{vmatrix} x' \\ y' \end{vmatrix} = \begin{vmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{vmatrix} \begin{vmatrix} x \\ y \end{vmatrix}$$



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

2D Rotation

- How to rotate an object with multiple vertices?

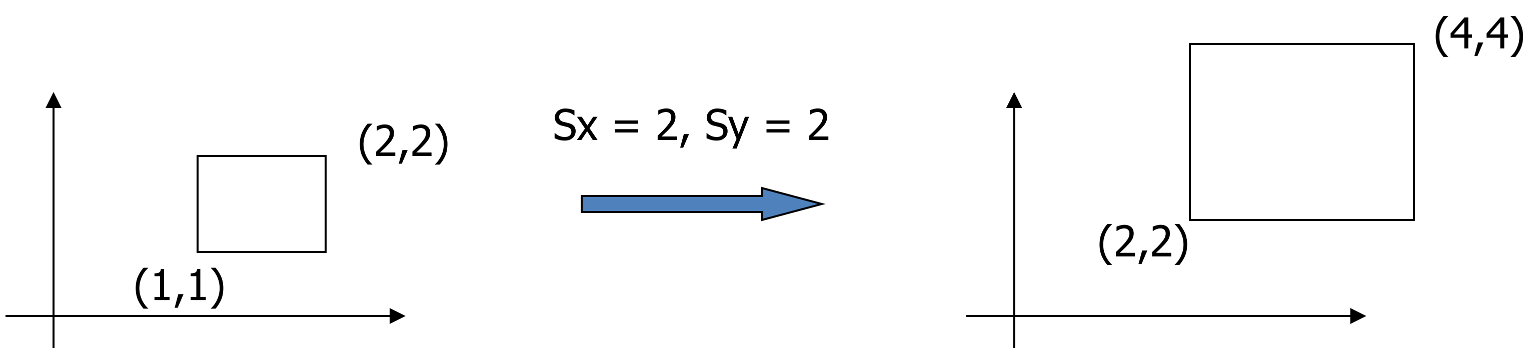


2D Scaling

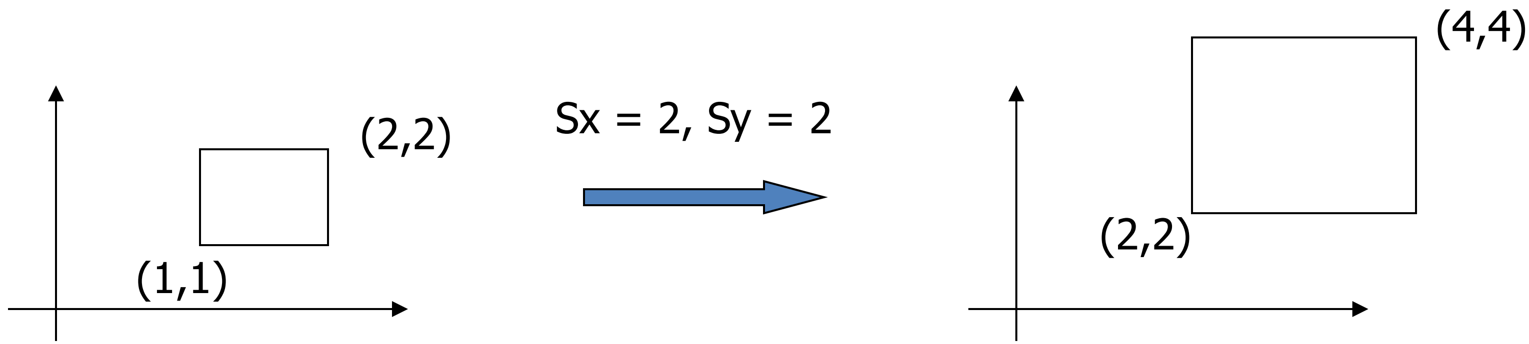
Scale: Alter the size of an object by a scaling factor (Sx, Sy), i.e.

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

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



2D Scaling



- Not only the object size is changed, it also moved<u>!!</u>
- Usually this is an undesirable effect
- We will discuss later (soon) how to fix it

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:

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

- Rotation:

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

- Scaling:

$$\begin{vmatrix} x' \\ y' \end{vmatrix} = \begin{vmatrix} s_x & 0 \\ 0 & s_y \end{vmatrix} \cdot \begin{vmatrix} t_x \\ t_y \end{vmatrix}$$

Or, 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:

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

Why use 3x3 matrices?

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

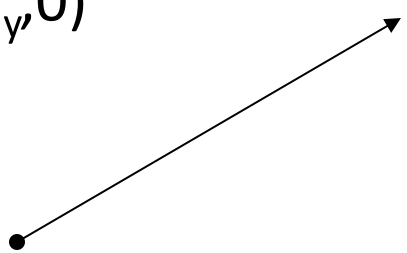
- The point (x,y) needs to be represented as

(x,y,1) -> this is called Homogeneous coordinates!

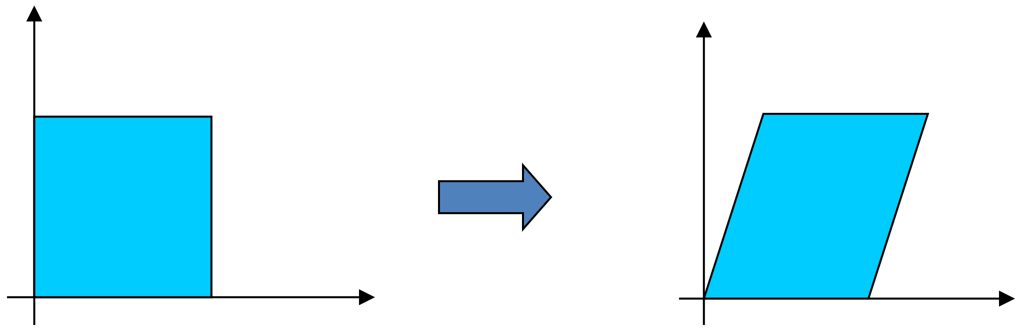
- How to represent a vector (v_x,v_y)?

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)? $(v_x,v_y,0)$



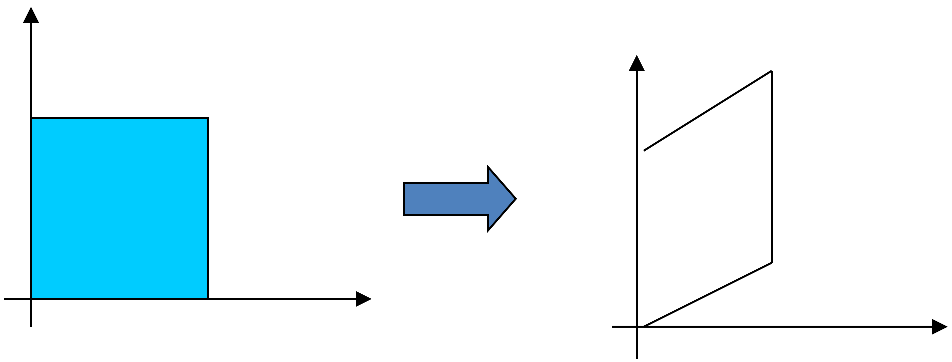
Shearing



- 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



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

Interesting Facts:

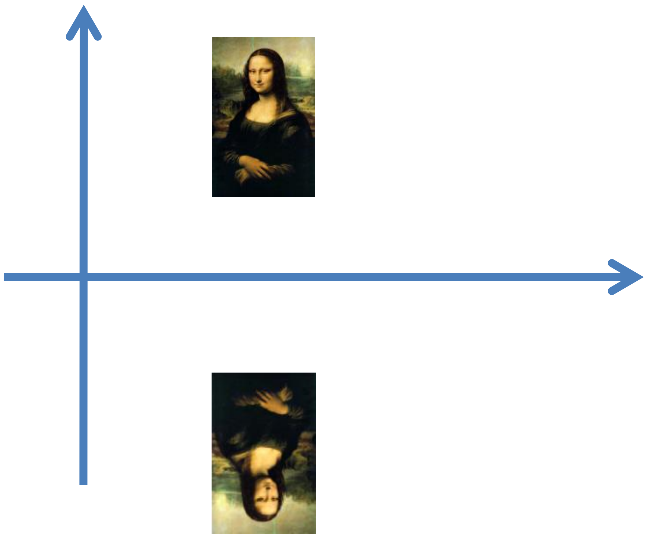
- A 2D rotation is three shears
- Shearing will not change the area of the object
- Any 2D shearing can be done by a rotation, followed by a scaling, and followed by a rotation

Reflection

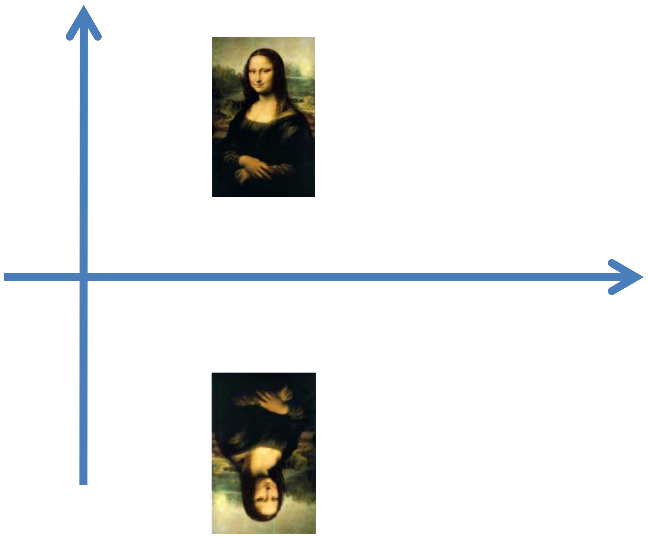
Reflection

Reflection

Reflection about X-axis

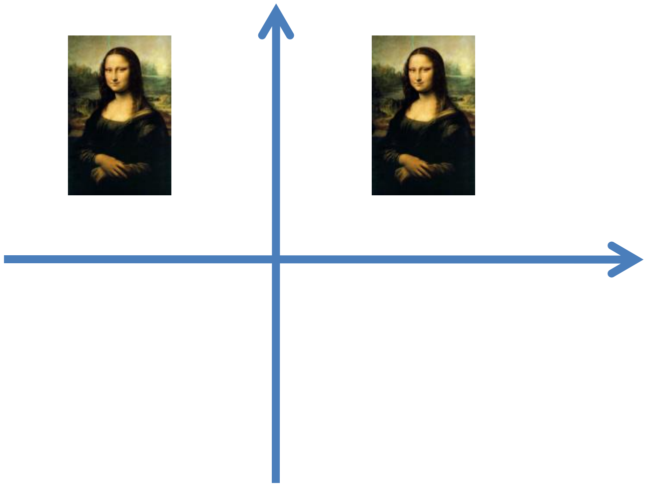


Reflection about X-axis

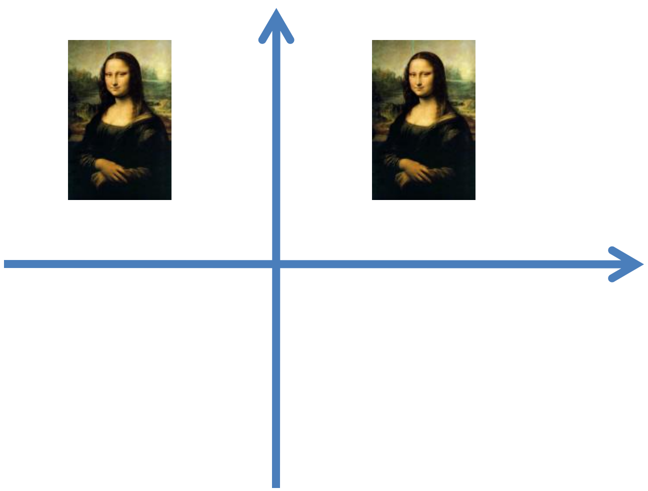


$$\begin{vmatrix} x \\ y \\ 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

Reflection about Y-axis

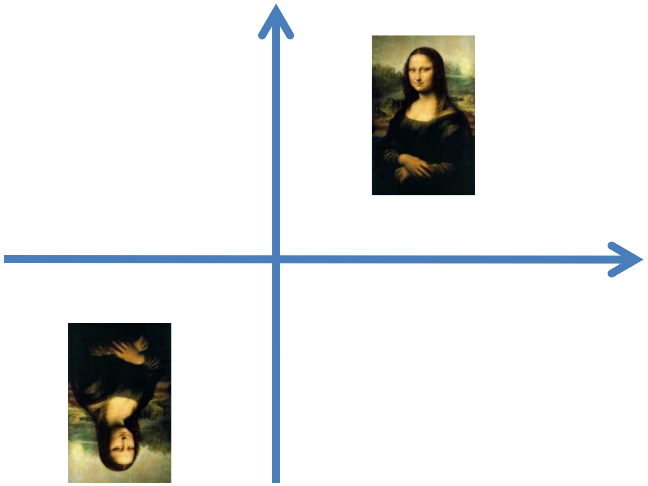


Reflection about Y-axis

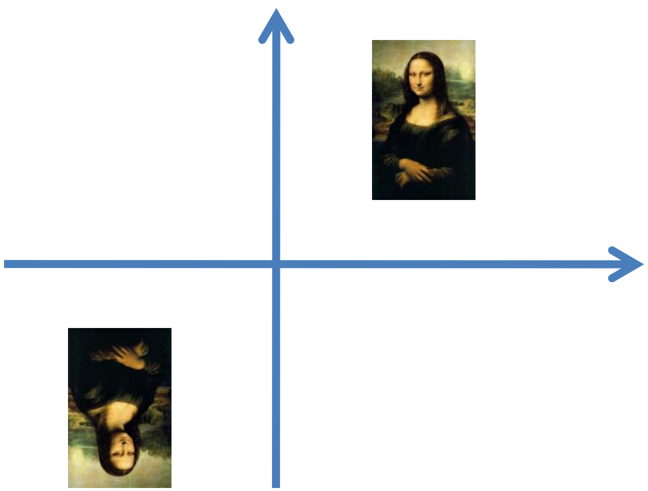


$$\begin{vmatrix} x \\ y \\ 1 \end{vmatrix} = \begin{vmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

What’s the Transformation Matrix?



What's the Transformation Matrix?

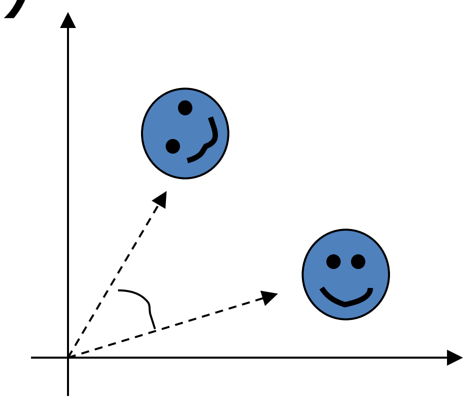


$$\begin{vmatrix} x \\ y \\ 1 \end{vmatrix} = \begin{vmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

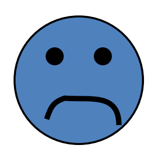
Rotation Revisit

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

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

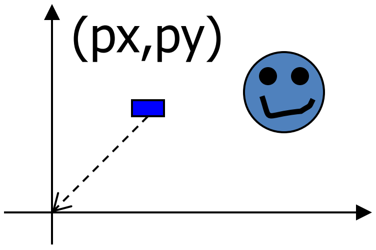


- What if I want to rotate about an arbitrary center?



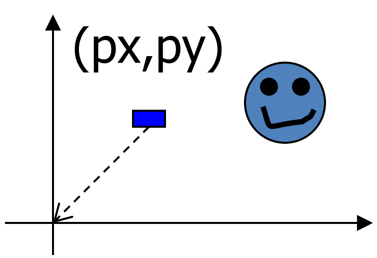
Arbitrary Rotation Center

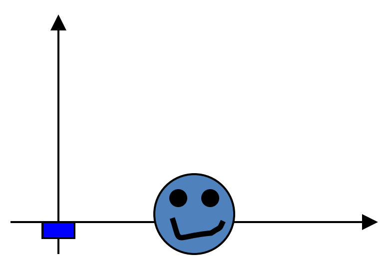
- To rotate about an arbitrary point P (px,py) by θ:



Arbitrary Rotation Center

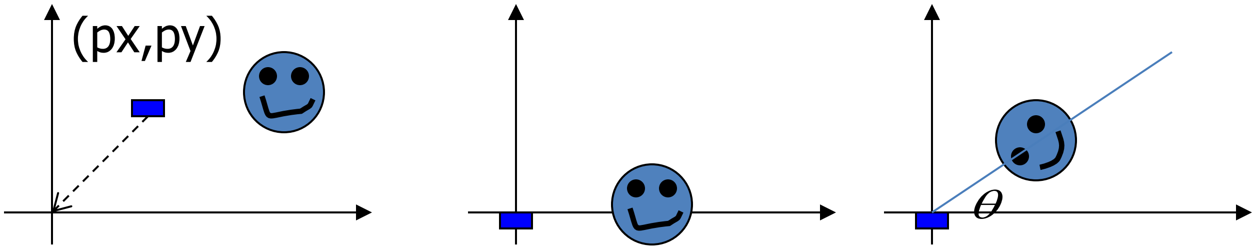
- To rotate about an arbitrary point P (px,py) by θ:
  - Translate the object so that P will coincide with the origin: T(-px, -py)





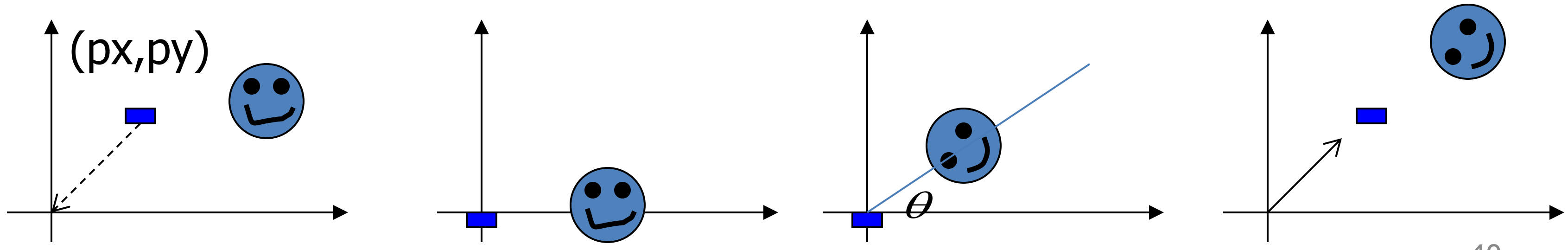
Arbitrary Rotation Center

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

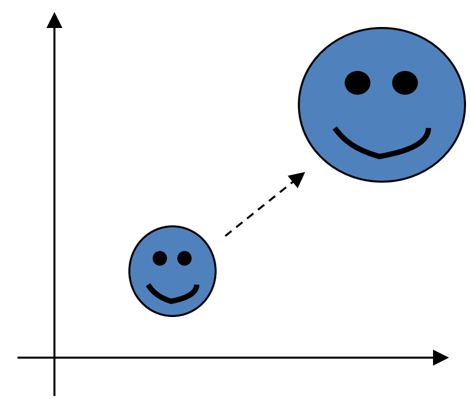
- Put in matrix form: T(px,py) R(θ) T(-px, -py) * P

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

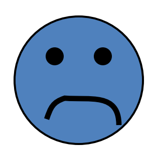
Scaling Revisit

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

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

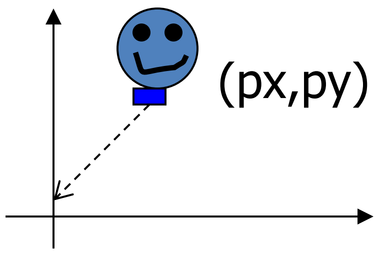


- What if I want to scale about an arbitrary pivot point?



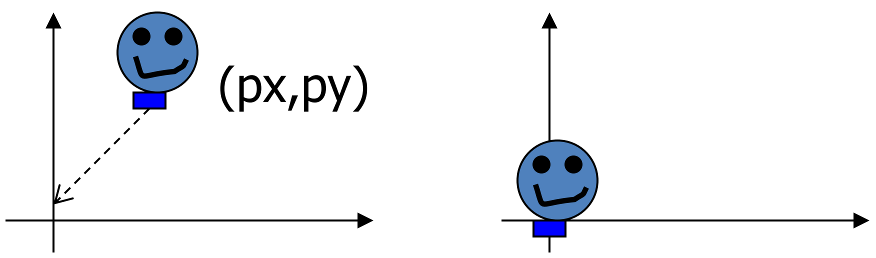
Arbitrary Scaling Pivot

- To scale about an arbitrary fixed point P (px,py):



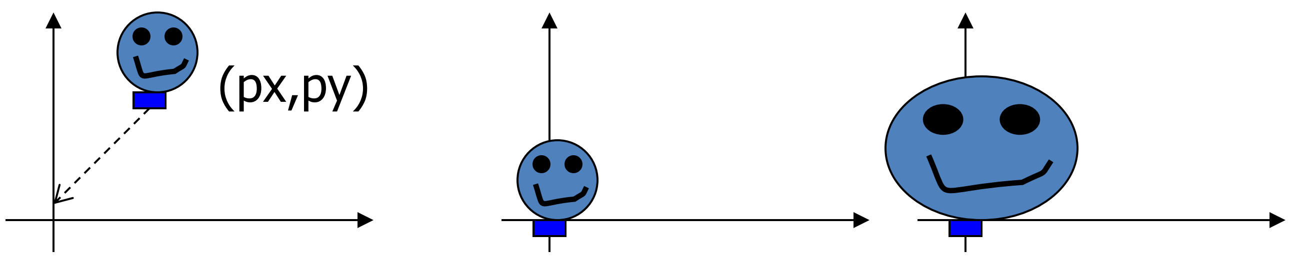
Arbitrary Scaling Pivot

- To scale about an arbitrary fixed point P (px,py):
  - Translate the object so that P will coincide with the origin: T(-px, -py)



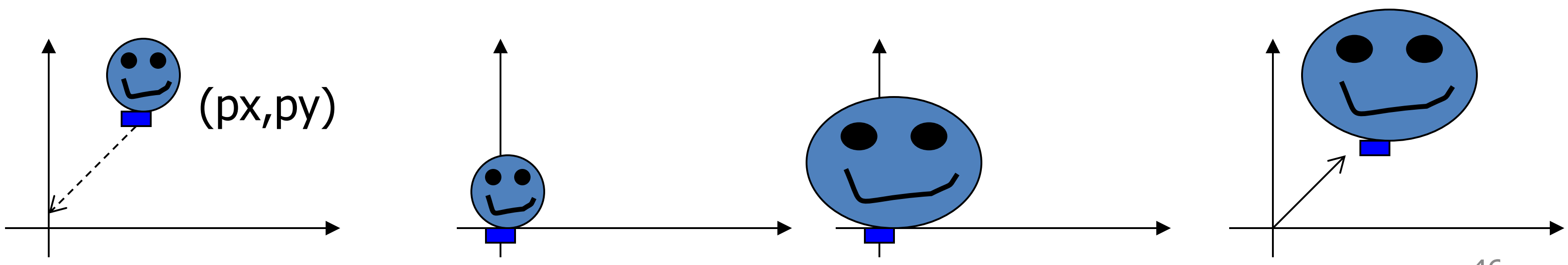
Arbitrary Scaling Pivot

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

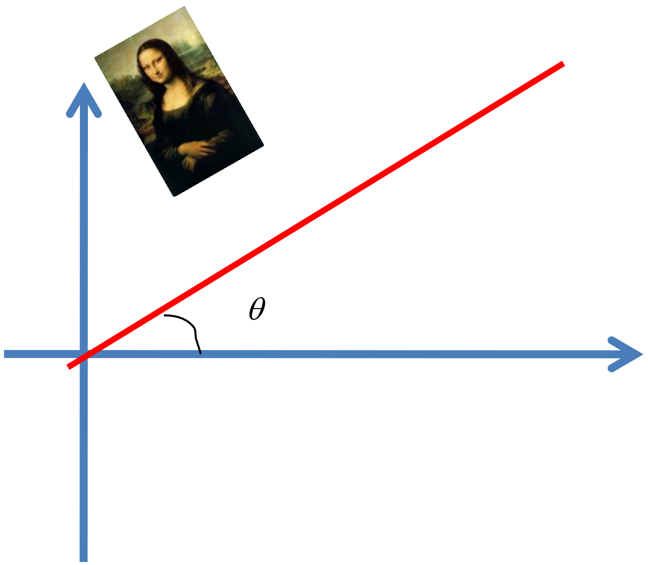


Arbitrary Scaling Pivot

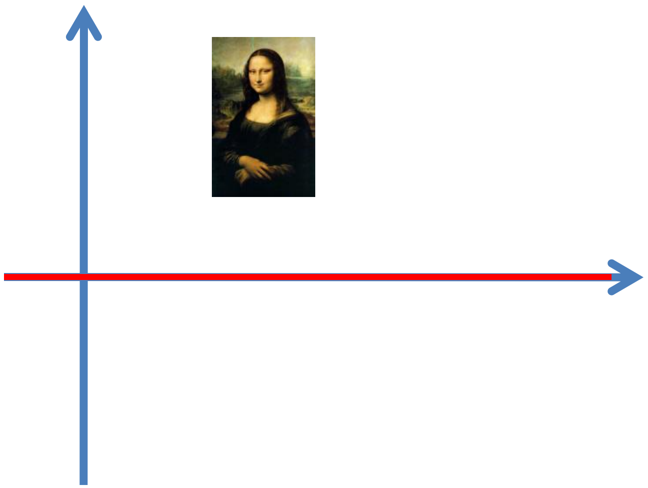
- To scale about an arbitrary fixed point P (px,py):
  - Translate the object so that P will coincide with the origin: T(-px, -py)
  - Scale the object: S(sx, sy)
  - Translate the object back: T(px,py)



Reflection about An Arbitrary Line

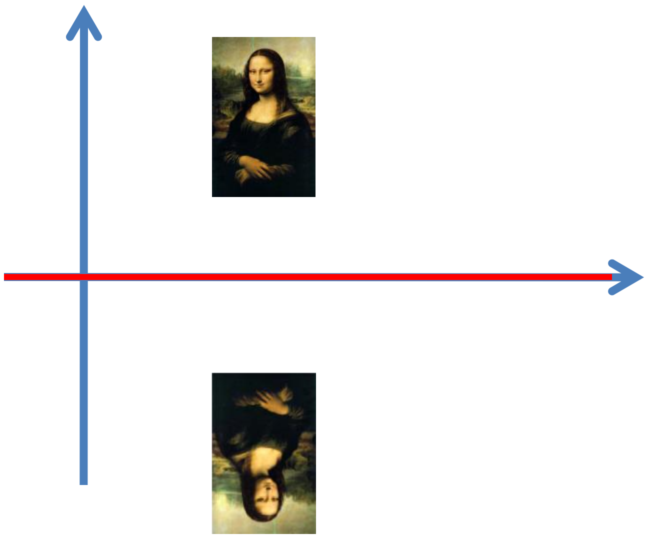


Reflection about An Arbitrary Line



– Rotate the object to align the reflection vector with x axis: $R(-\theta)$

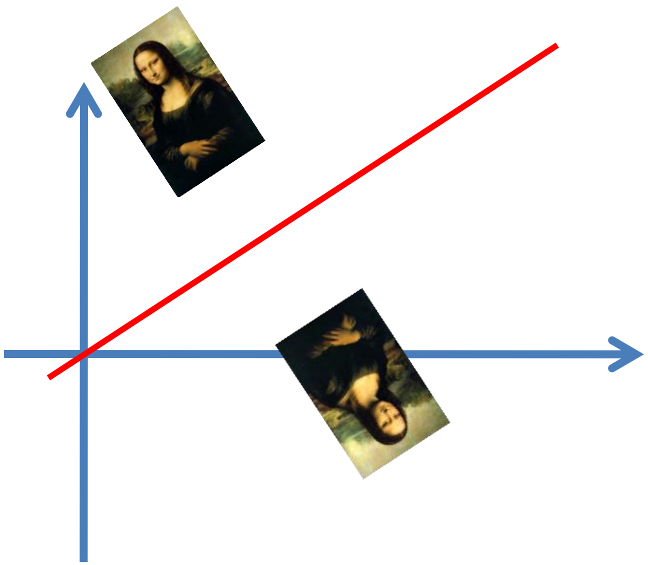
Reflection about An Arbitrary Line



– Rotate the object to align the reflection vector with x axis: $R(-\theta)$

– Reflect the object

Reflection about An Arbitrary Line



– Rotate the object to align the reflection vector with x axis: $R(-\theta)$

– Reflect the object

– Rotate the object back: $R(\theta)$

Affine Transformation

- Translation, Scaling, Rotation, Shearing are all affine transformation

Affine Transformation

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

$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ 0 & 0 & 1 \end{vmatrix} \cdot \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

Affine Transformation

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

$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ 0 & 0 & 1 \end{vmatrix} \cdot \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

- 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

Composing Transformation

- Composing Transformation – the process of applying several transformation in succession to form one overall transformation
- If we apply transforming a point P using M1 matrix first, and then transforming using M2, and then M3, then we have:

(M3 x (M2 x (M1 x P)))

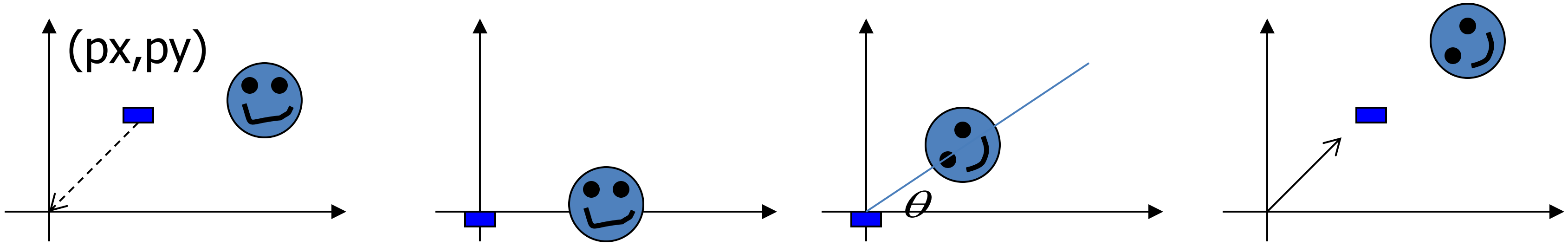
Composing Transformation

- Composing Transformation – the process of applying several transformation in succession to form one overall transformation
- If we apply transforming a point P using M1 matrix first, and then transforming using M2, and then M3, then we have:

$$(M3 \times (M2 \times (M1 \times P))) = \underbrace{M3 \times M2 \times M1}_{M} \times P$$

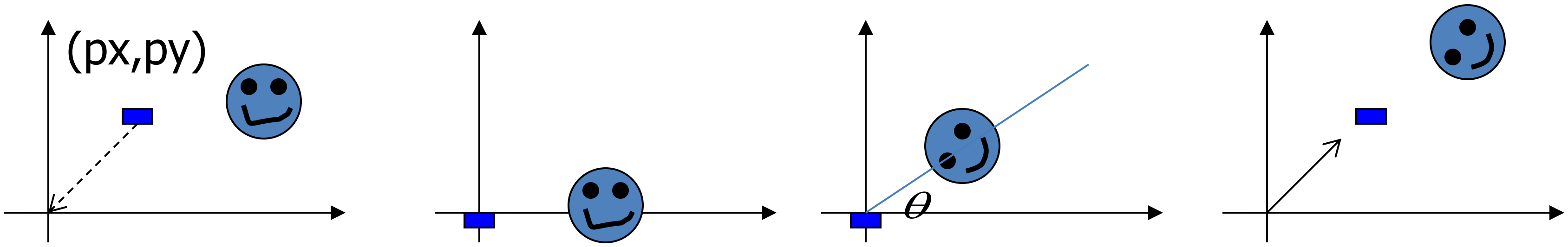
(pre-multiply)

Arbitrary Rotation Center



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

Arbitrary Rotation Center



$$
\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \underbrace{\begin{vmatrix} 1 & 0 & px \\ 0 & 1 & py \\ 0 & 0 & 1 \end{vmatrix}}_{M_3} \underbrace{\begin{vmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{vmatrix}}_{M_2} \underbrace{\begin{vmatrix} 1 & 0 & -px \\ 0 & 1 & -py \\ 0 & 0 & 1 \end{vmatrix}}_{M_1} \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}
$$

$$M = M_3 * M_2 * M_1$$

Composing Transformation

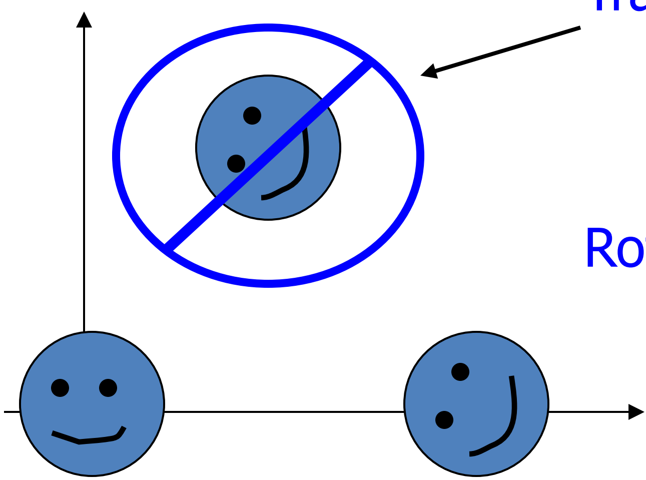
- Matrix multiplication is associative

  M3 x M2 x M1 = (M3 x M2) x M1 = M3 x (M2 x M1)

- Transformation products may not be commutative A x B != B x A

Transformation Order Matters!

- Example: rotation and translation are not commutative



Translate (5,0) and then Rotate 60 degree

OR

Rotate 60 degree and then translate (5,0)??

Rotate and then translate !!

Composing Transformation

- Matrix multiplication is associative

  M3 x M2 x M1 = (M3 x M2) x M1 = M3 x (M2 x M1)
- Transformation products may not be commutative A x B != B x A
- Some cases where A x B = B x A

A	B
translation	translation
scaling	scaling
rotation	rotation

Finding Affine Transformations

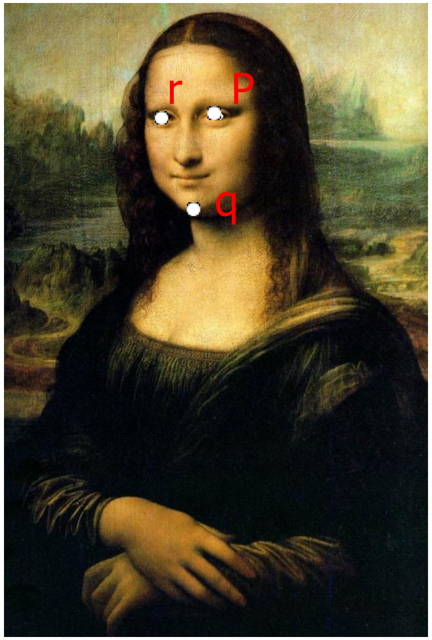
- How many points determines affine transformation

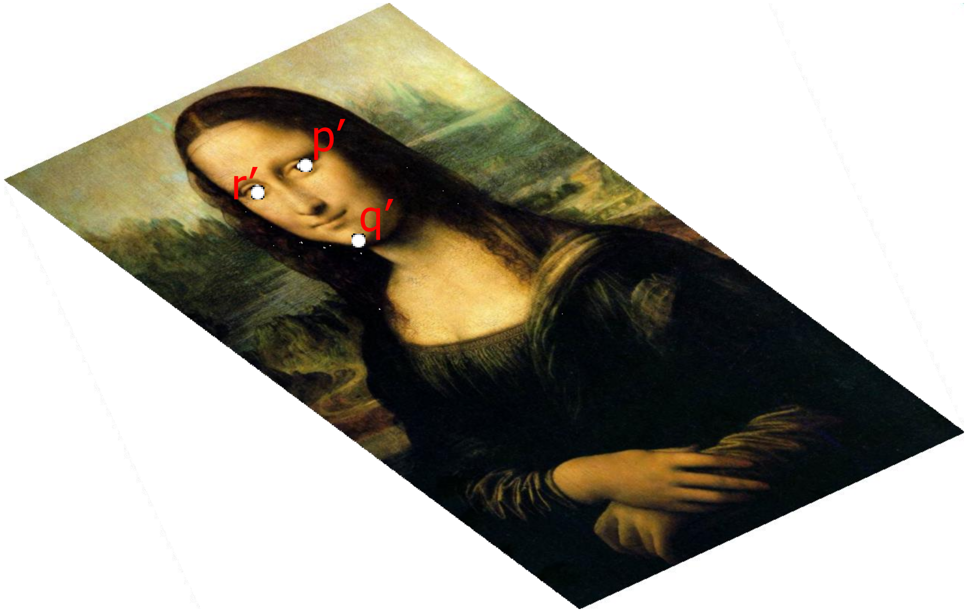
Finding Affine Transformations

- How many points determines affine transformation

Finding Affine Transformations

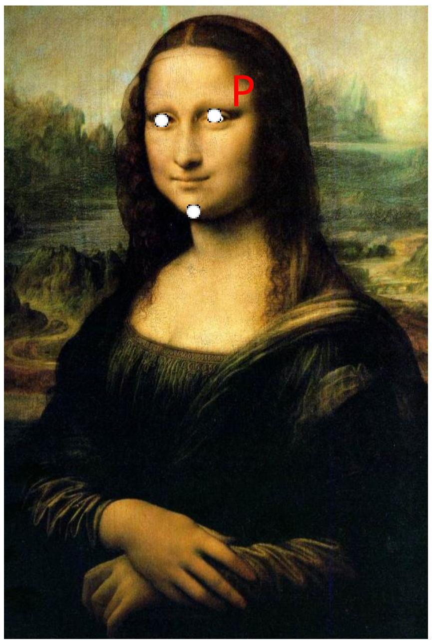
- Image of 3 points determines affine transformation

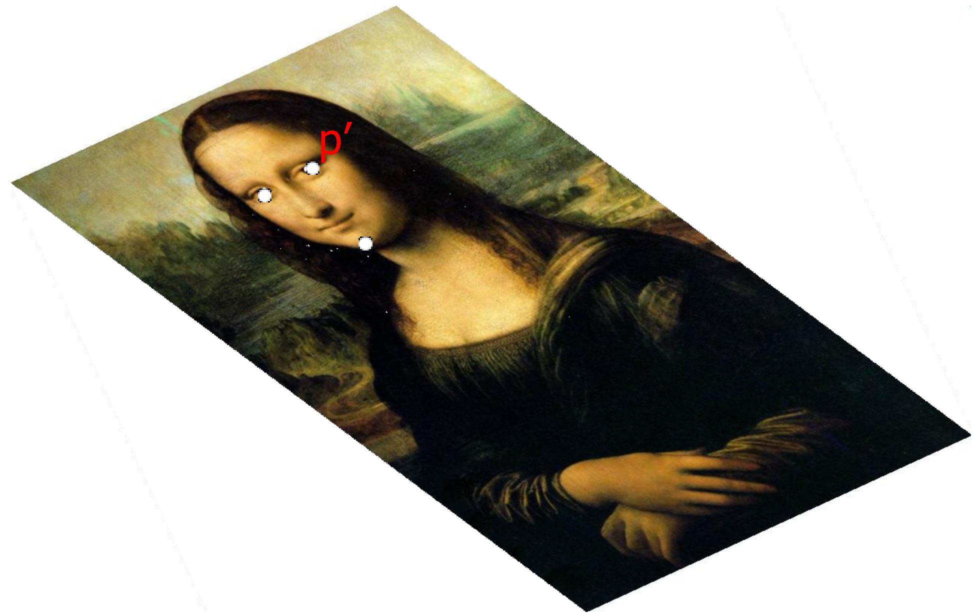




Finding Affine Transformations

- Image of 3 points determines affine transformation

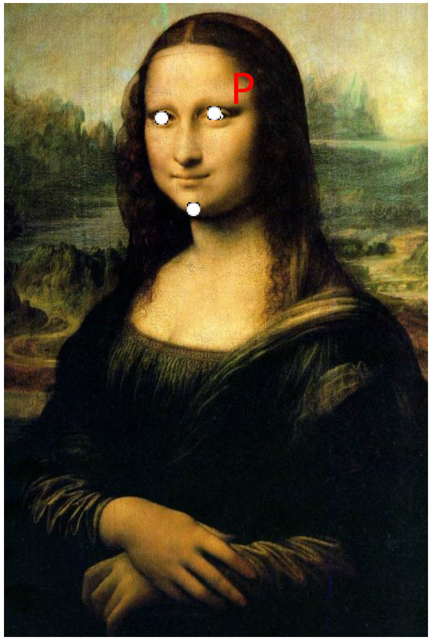


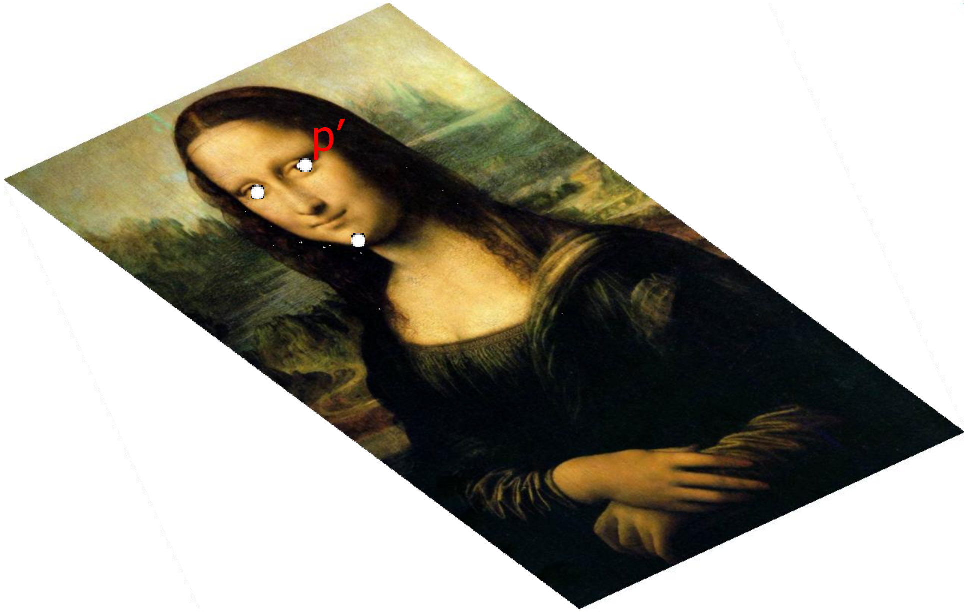


$$\begin{pmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ 1 \end{pmatrix} = \begin{pmatrix} p'_x \\ p'_y \\ 1 \end{pmatrix}$$

Finding Affine Transformations

- Image of 3 points determines affine transformation





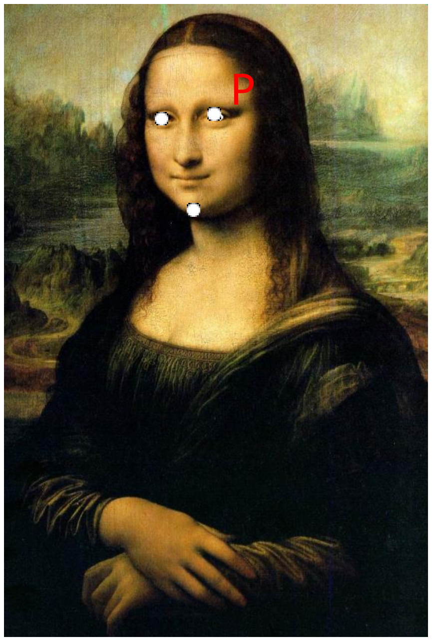
- Each pair gives us 2 linear equations on 6 unknowns!

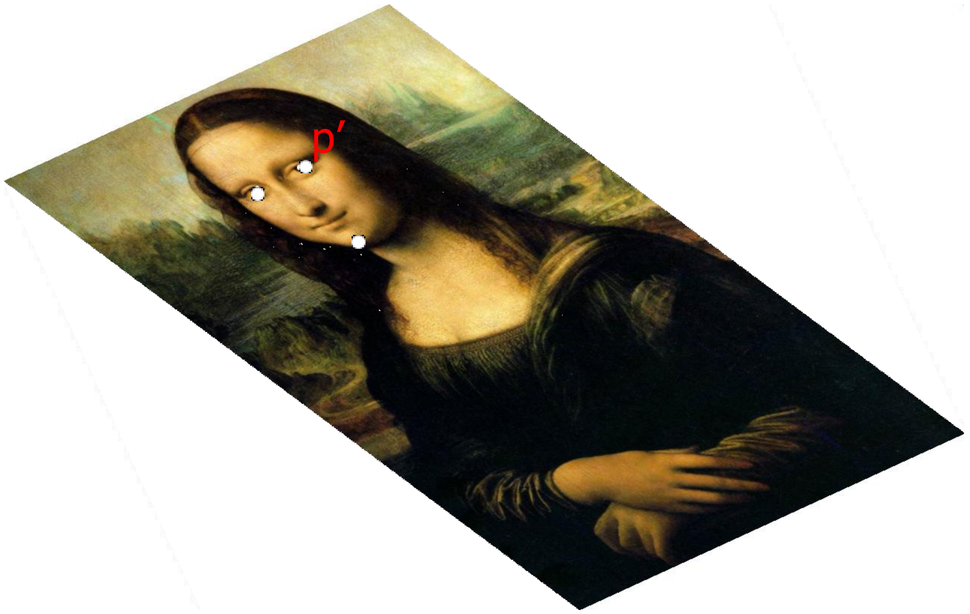
- In total, 6 unknowns 6 linear equations.

$$\begin{pmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ 1 \end{pmatrix} = \begin{pmatrix} p'_x \\ p'_y \\ 1 \end{pmatrix}$$

Finding Affine Transformations

- Image of 3 points determines affine transformation

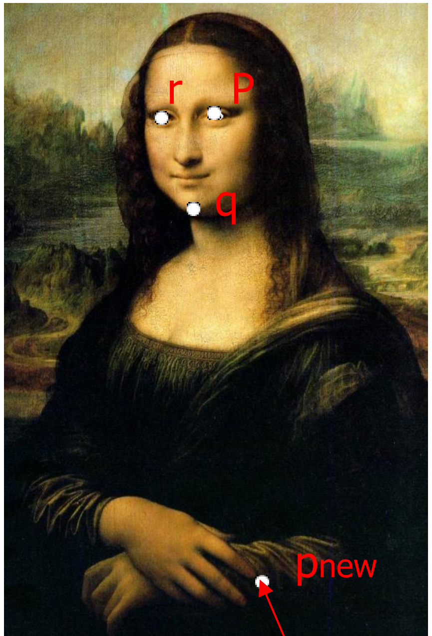


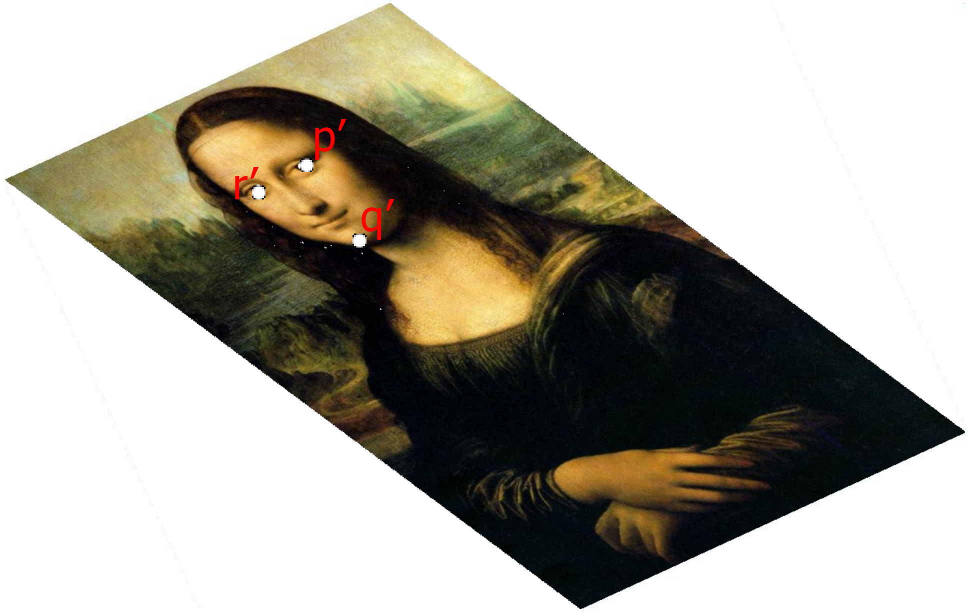


$$\begin{pmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ 1 \end{pmatrix} = \begin{pmatrix} p'_x \\ p'_y \\ 1 \end{pmatrix}$$

Finding Affine Transformations

- Image of 3 points determines affine transformation

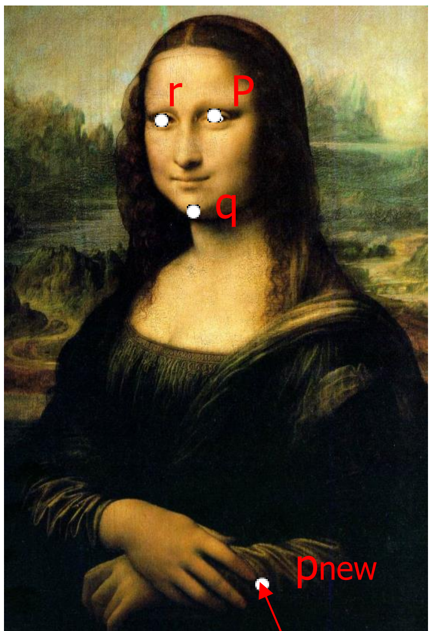


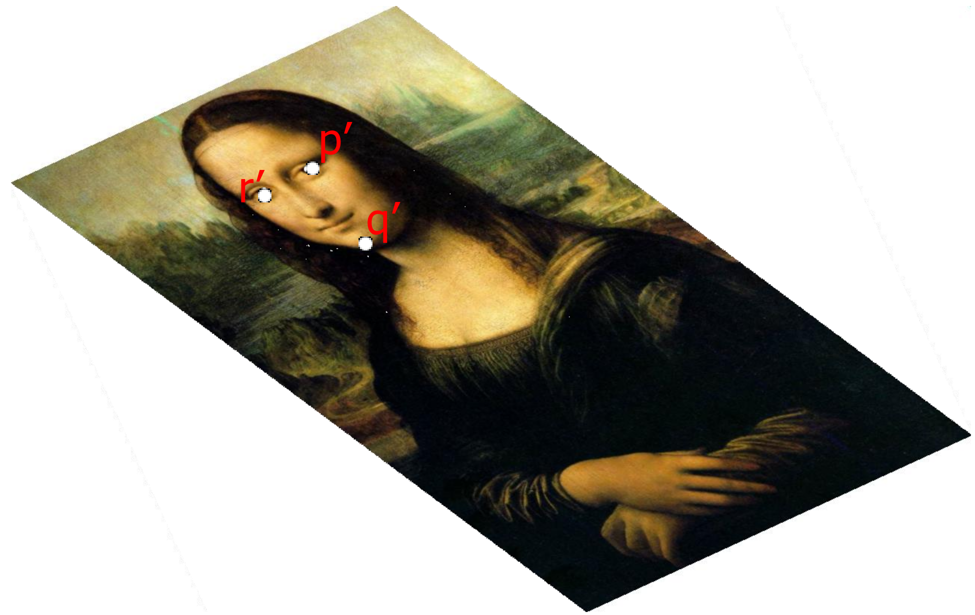


What's the corresponding point in the right image?

Finding Affine Transformations

- Image of 3 points determines affine transformation





What's the corresponding point in the right image?

$$\begin{pmatrix} p'_{new} \\ p'_{new} \\ 1 \end{pmatrix} = \begin{pmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_{new} \\ p_{new} \\ 1 \end{pmatrix}$$

Next Lecture

- 2D coordinate transformations
- 3D transformations
- Lots of vector and matrix operations!