Affine Transformations

Affine Transformations

Homogeneous Coordinates

And related issues

Affine Transformation

- Maps parallel lines to parallel lines
- Common affine transforms
  - Translation
  - Rotation
  - Reflection
  - Scale
  - Shear

Linear Combinations & Dot Products

- A *linear combination* of the vectors
  $v_1, v_2, \ldots v_n$
  is any vector of the form
  $\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$
  where α_i is a real number (i.e. a scalar)

- *Dot Product*:

$$u \cdot v = \sum_{k=1}^{n} u_k v_k$$

a real value $\mathbf{u_1 v_1 + u_2 v_2 + \ldots + u_n v_n}$ written as

$u \bullet v$

Matrices and Matrix Operators

- A *n*-dimensional vector:

$$\begin{bmatrix} x_1 \\ . \\ . \\ . \\ x_n \end{bmatrix}$$

- Matrix Operations:
  - Addition/Subtraction
  - Identity
  - Multiplication
    - Scalar
    - Matrix Multiplication

$$A + B = B + A$$
$$A + (B + C) = (A + B) + C$$
$$(cd)A = c(dA)$$
$$1A = A$$
$$c(A + B) = cA + cB$$
$$(c + d)A = cA + dA$$

Matrix Multiplication

- Sum over rows & columns
- Recall: multiplication is not commutative
- *Identity Matrix*:
  1s on diagonal
  0s everywhere else

$$c_{ij} = \sum_{s=1}^{m} a_{is} b_{sj}$$

$$\begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix}$$

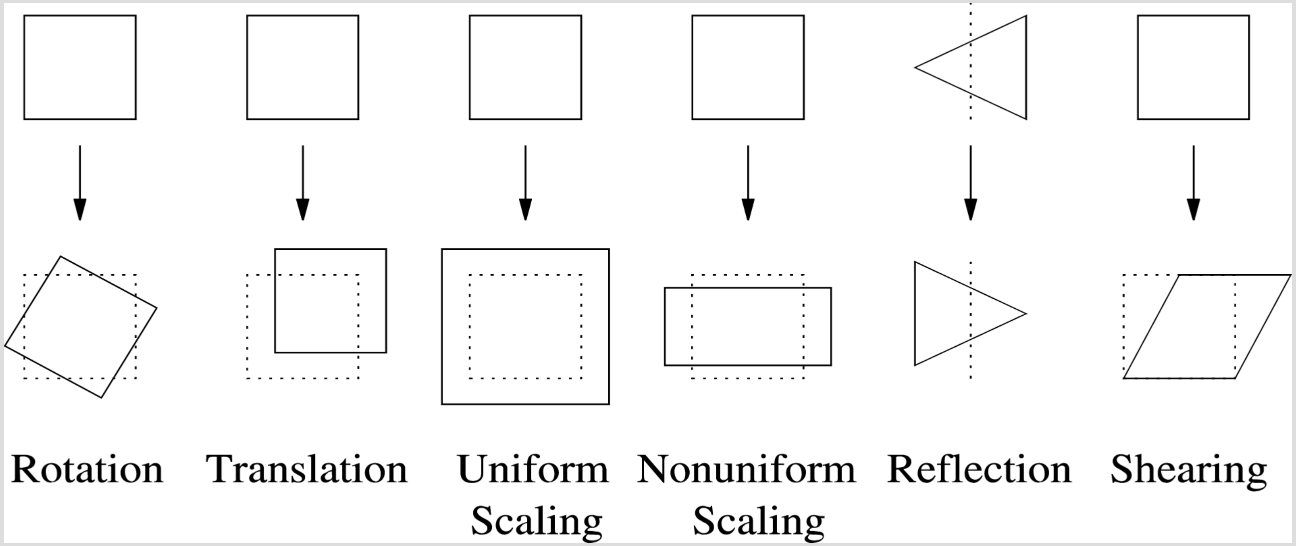
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \quad \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix}$$

2D Affine Transformations

All represented as matrix operations on vectors!

Parallel lines preserved, angles/lengths not

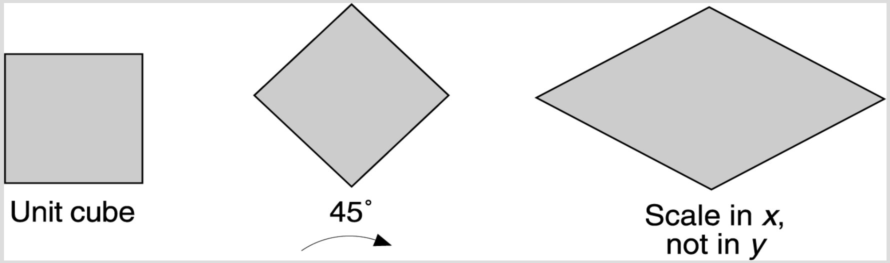
- Scale
- Rotate
- Translate
- Reflect
- Shear

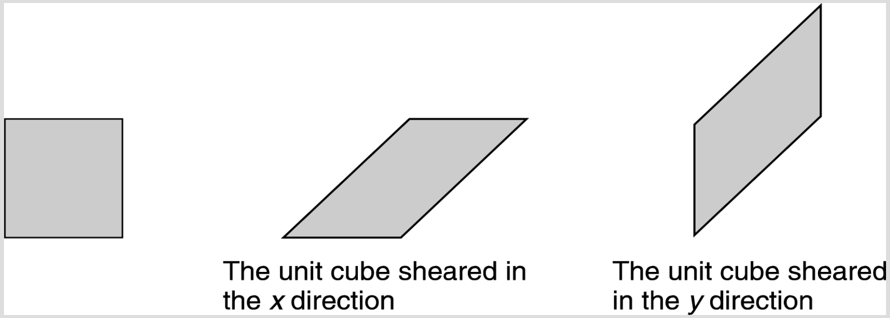


2D Affine Transformations

- **Example 1:** rotation and non uniform scale on unit cube
- **Example 2:** shear first in x, then in y

  Note:
  - Preserves parallels
  - Does not preserve lengths and angles

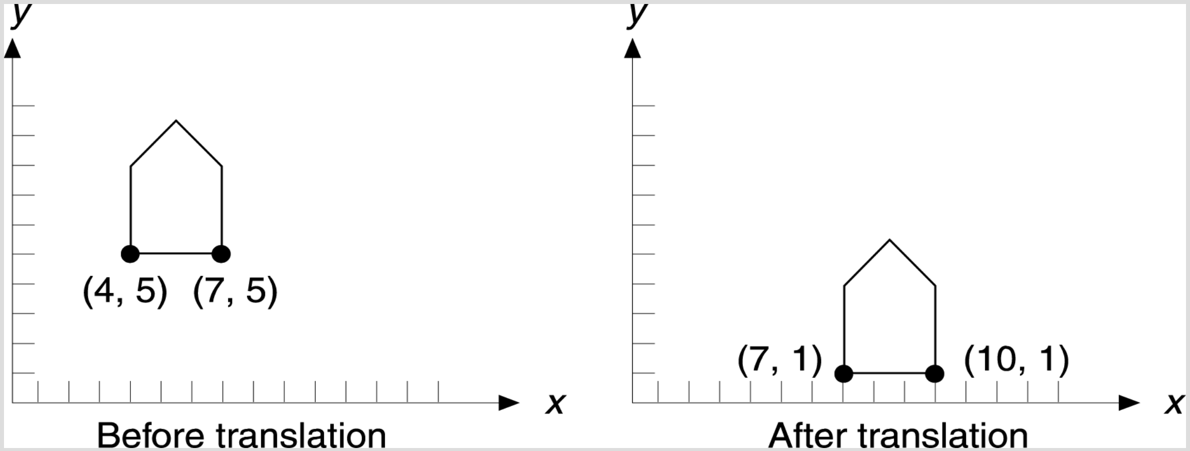




2D Transforms: Translation

- Rigid motion of points to new locations

$$x' = x + d_x$$
$$y' = y + d_y$$



- Defined with column vectors:

as
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

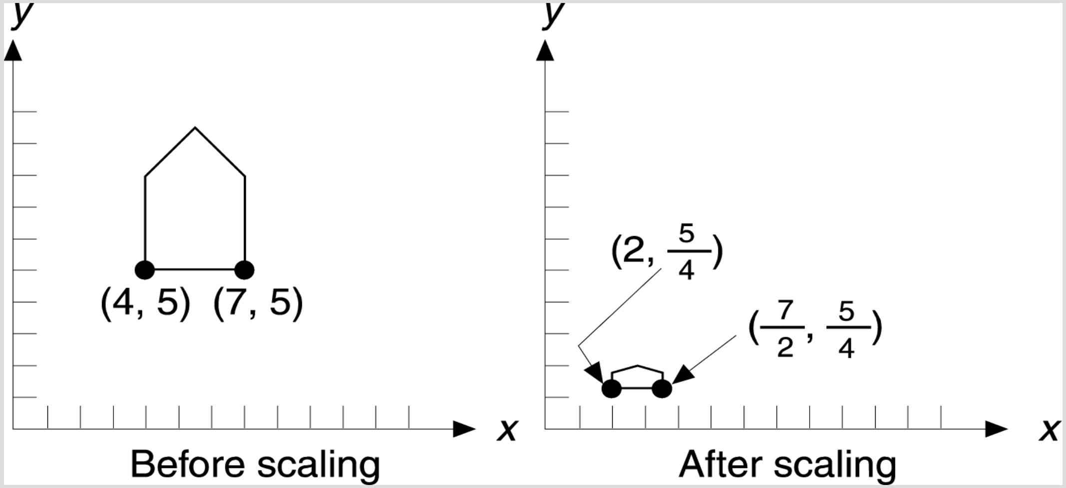
$$P' = P + T$$

2D Transforms: Scale

- Stretching of points along axes:

$$x' = s_x \cdot x$$

$$y' = s_y \cdot y$$



In matrix form:

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

or just:

$$P' = S \cdot P$$

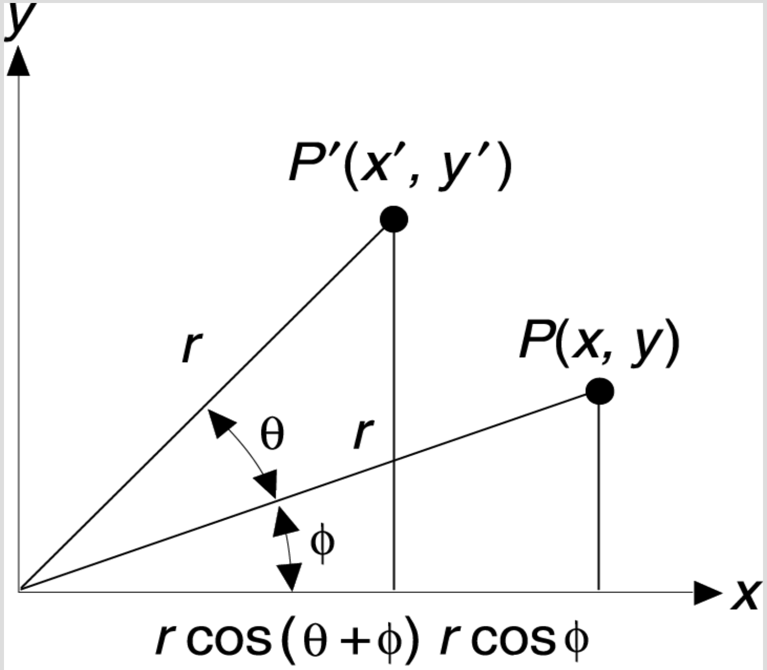
2D Transforms: Rotation

- Rotation of points about the origin

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

Positive Angle: CCW

Negative Angle: CW



Matrix form:

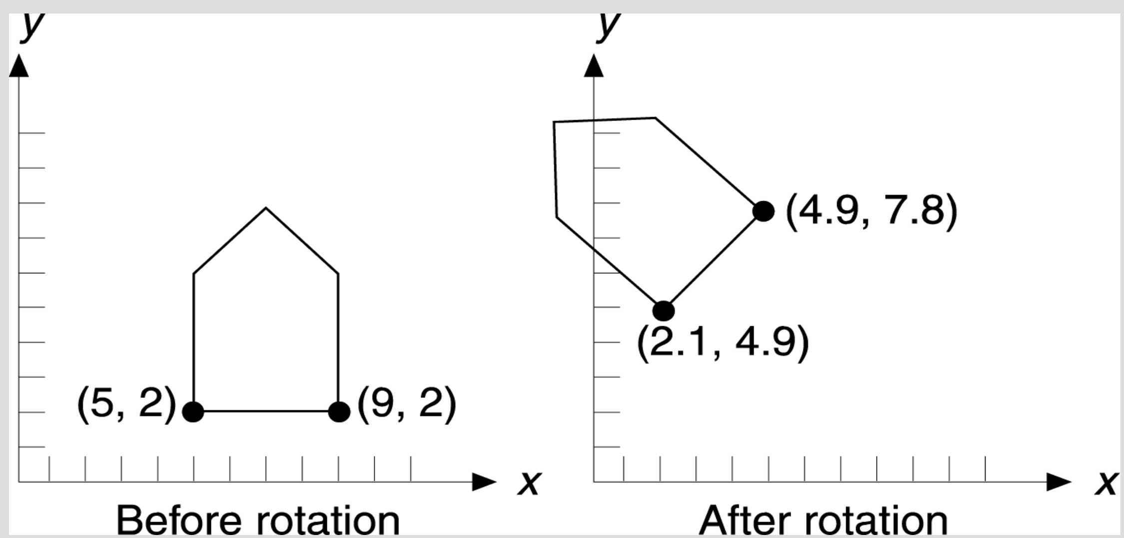
or just:

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

$$P' = R \cdot P$$

2D Transforms: Rotation

- Substitute the 1st two equations into the 2nd two to get the general equation



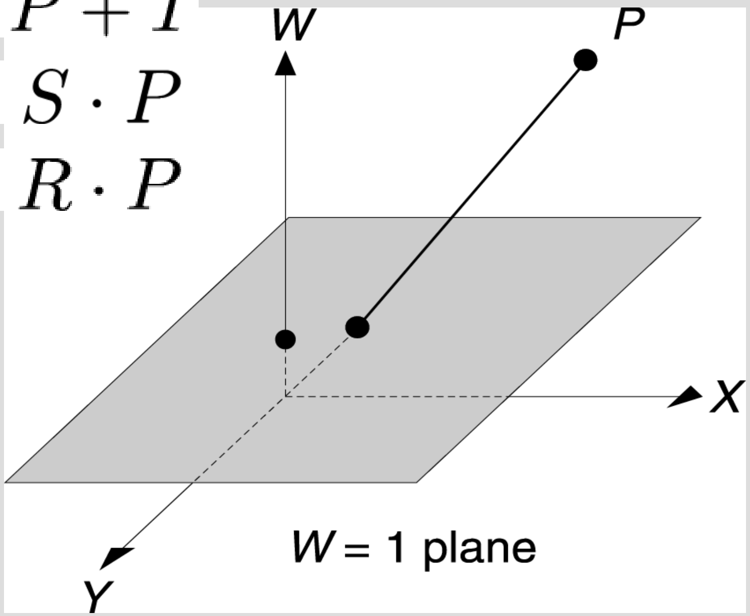
$$x = r \cdot cos\ \phi$$
$$y = r \cdot sin\ \phi$$

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

Homogeneous Coordinates

- Observe: *translation* is treated differently from *scaling* and *rotation*
- ***Homogeneous coordinates***: allows all transformations to be treated as matrix multiplications

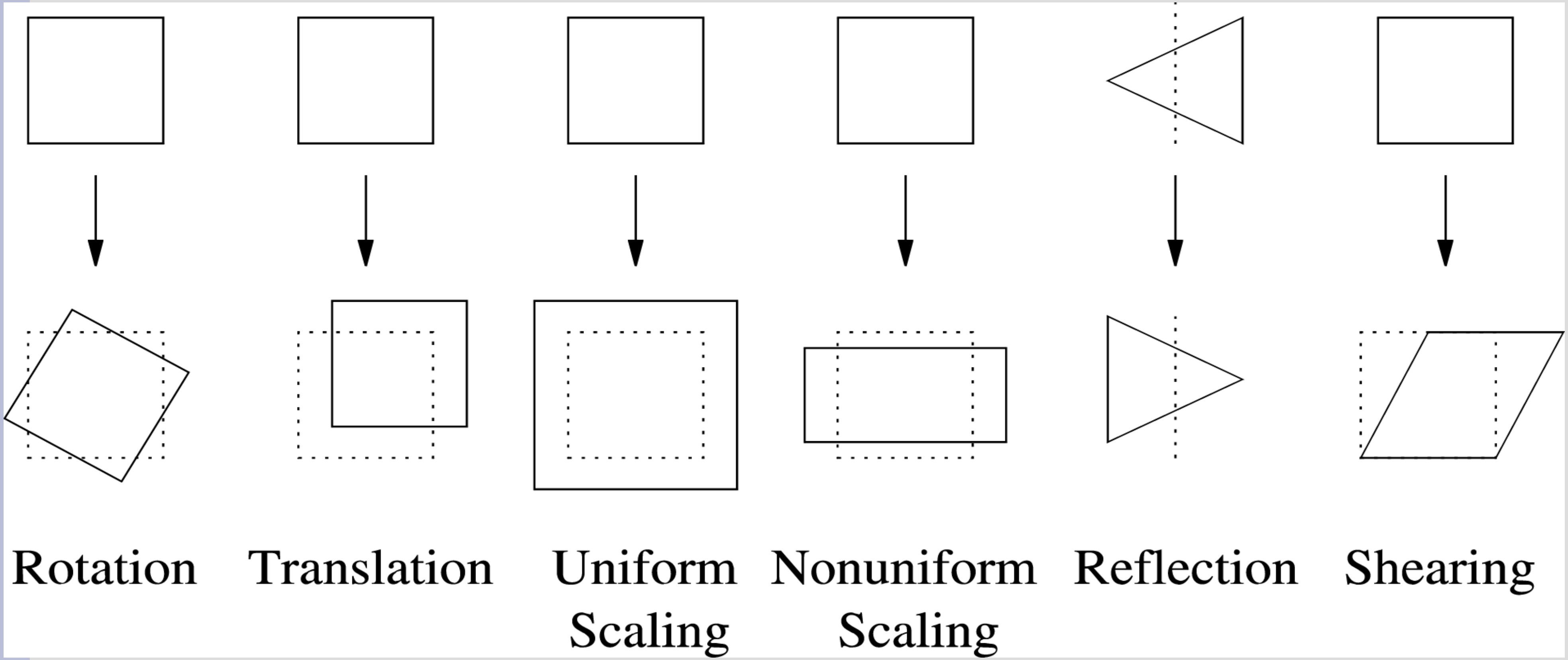
$$P' = P + T$$
$$P' = S \cdot P$$
$$P' = R \cdot P$$



Example: A 2D point *(x,y)* is the line *(wx,wy,w),* where *w* is any real #, in 3D homogenous coordinates.

To get the point, *homogenize* by dividing by *w (i.e. w=1)*

Recall our Affine Transformations

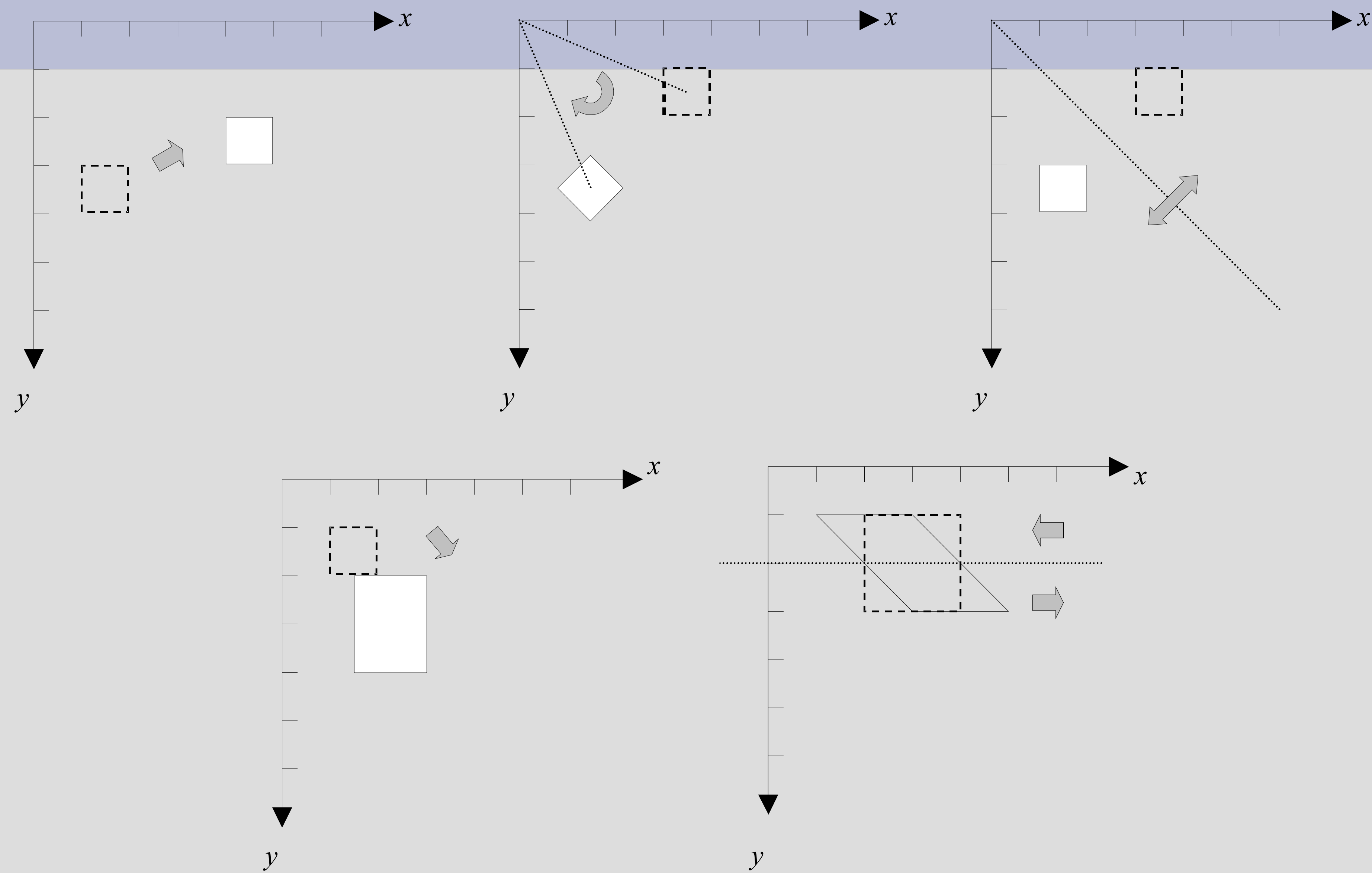


Matrix Representation of 2D Affine Transformations

- Translation: $$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$
- Scale: $$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$
- Rotation: $$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} cos\ \theta & -sin\ \theta & 0 \\ sin\ \theta & cos\ \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$
- Shear: $$SH_x = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  Reflection: $$F_y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Affine Transformation

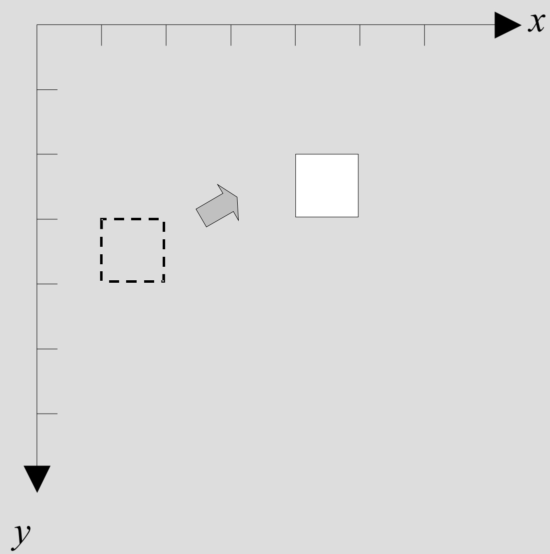


Transformation Matrix

$$\begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \frac{2}{1+k^2} - 1 & \frac{2k}{1+k^2} & 0 \\ \frac{2k}{1+k^2} & \frac{2k^2}{1+k^2} - 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & s & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Translations

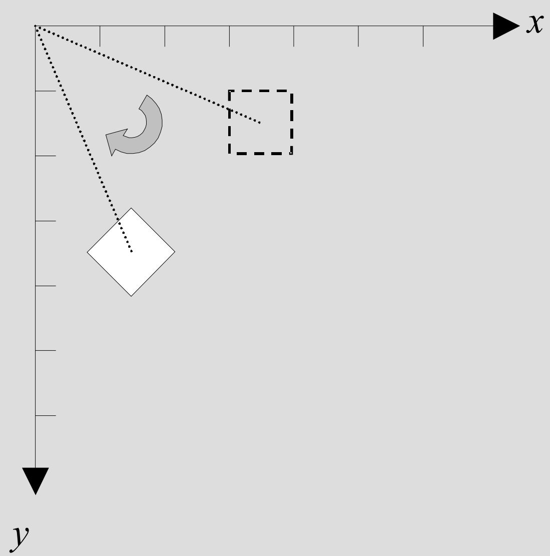


$$\begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix}$$

AffineTransform class of Java 2D

void setToTranslation(double tx, double ty)

Rotations



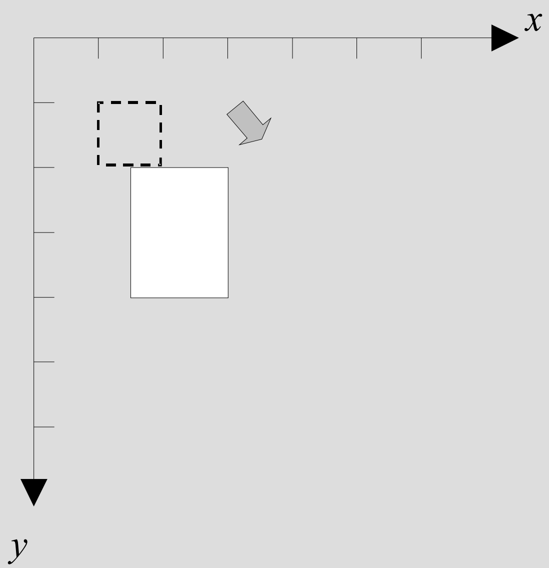
Rotation about the origin

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

AffineTransform class of Java 2D

void setToRotation(double theta)
void setToRotation(double theta, double x, double y)

Scaling

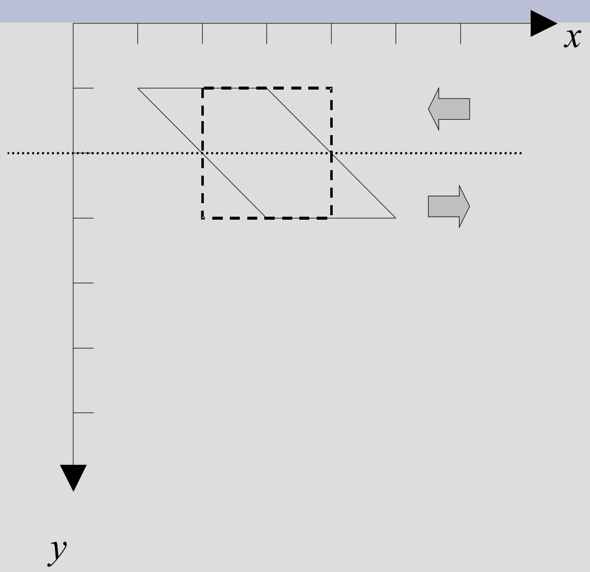


$$\begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

AffineTransform class of Java 2D

void setToScale(double sx, double sy)

Shearing

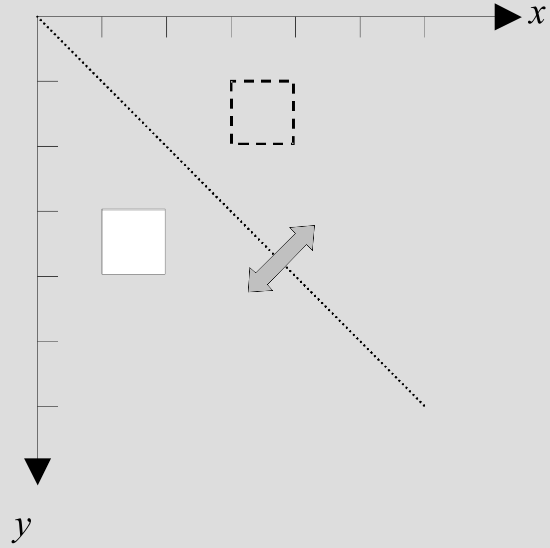


$$\begin{bmatrix} 1 & s & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

AffineTransform class of Java 2D

void setToShear(double shx, double shy)

Reflection



Reflection about the line y = kx

$$\begin{bmatrix} \frac{2}{1+k^2}-1 & \frac{2k}{1+k^2} & 0 \\ \frac{2k}{1+k^2} & \frac{2k^2}{1+k^2}-1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In AffineTransform class of Java 2D, must set first two rows of transformation matrix directly with one of:

```
AffineTransform(double m00, double m10, double m01,
                double m11, double m02, double m12)
AffineTransform(double[] flatmatrix)
void setTransform(double m00, double m10, double m01,
                  double m11, double m02, double m12)
```

AffineTransform class

- For object transformations:
  - To transform shapes
    - Shape createTransformedShape(Shape shape)
  - To transform points or sets of points
    - There are several methods named transform for this
  - To transform vectors
    - There are methods named deltaTransform
- For viewing transformations:
  - Methods of the Graphics2D class
    - void setTransform(AffineTransform tx)
    - void transform(AffineTransform tx)

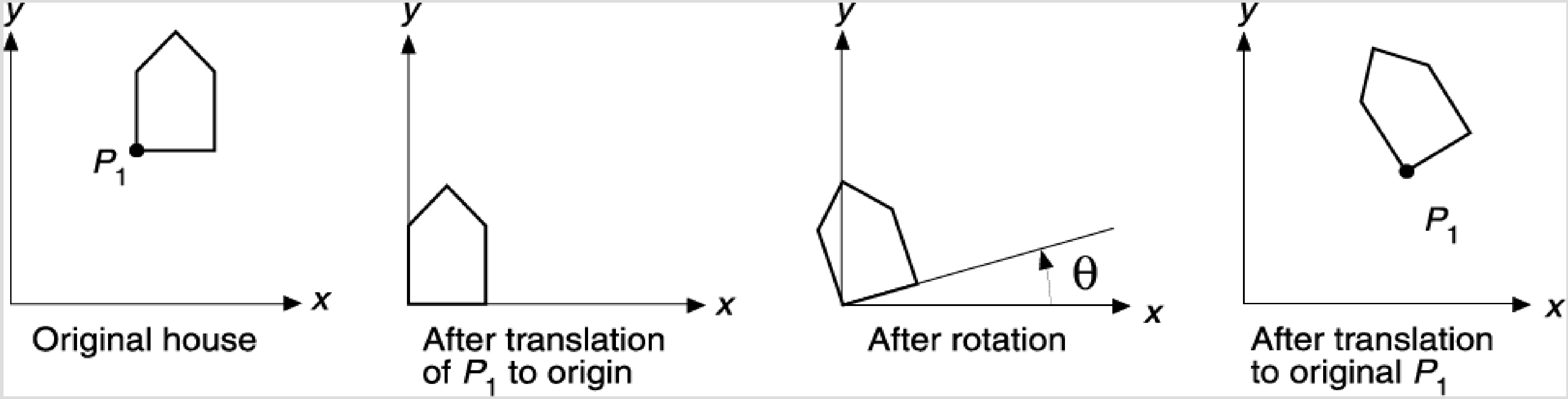
Composition of 2D Transforms

- Rotate about a point *P1*
  - Translate *P1* to origin
  - Rotate
  - Translate back to *P1*

$$T(x_1, y_1) \cdot R(\theta) \cdot T(-x_1, -y_1)$$

$$= \begin{bmatrix} 1 & 0 & x_1 \\ 0 & 1 & y_1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} cos\ \theta & -sin\ \theta & 0 \\ sin\ \theta & cos\ \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} cos\ \theta & -sin\ \theta & x_1(1 - cos\ \theta) + y_1 sin\ \theta \\ sin\ \theta & cos\ \theta & y_1(1 - cos\ \theta) - x_1 sin\ \theta \\ 0 & 0 & 1 \end{bmatrix}$$



Composition of 2D Transforms

- Scale object around point *P1*
  - *P1* to origin
  - Scale
  - Translate back to *P1*

$$T(x_1, y_1) \cdot S(S_x, S_y) \cdot T(-x_1, -y_1)$$

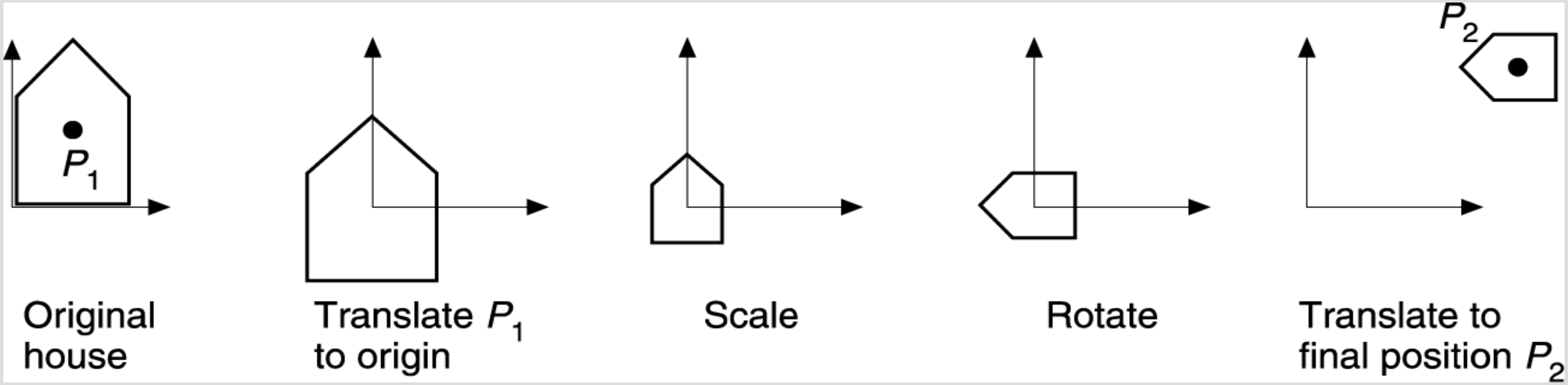
$$= \begin{bmatrix} 1 & 0 & x_1 \\ 0 & 1 & y_1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} S_x & 0 & x_1(1 - S_x) \\ 0 & S_y & y_1(1 - S_y) \\ 0 & 0 & 1 \end{bmatrix}$$

Composition of 2D Transforms

- Scale + rotate object around point *P1* and move to *P2*
  - *P1* to origin
  - Scale
  - Rotate
  - Translate to *P2*

$$T(x_2, y_2) \cdot R(\theta) \cdot S(s_x, s_y) \cdot T(-x_1, -y_1)$$



Composition of Transformations

- In Java 2D (AffineTransform class):
  - void rotate(double theta)
  - void rotate(double theta, double x, double y)
  - void scale(double sx, double sy)
  - void shear(double shx, double shy)
  - void translate(double tx, double ty)
  - void concatenate(AffineTransform tx)
  - void preConcatenate(AffineTransform tx)
- The setTo* methods clears the existing transform
- The above methods append an additional transform matrix on the right
  - preConcatenate appends on the left