

CSD1130

Game Implementation Techniques

Lecture 8

Overview

- 2D Transformations
- Homogeneous Coordinates and Matrix Representation
- Composition of 2D Transformations

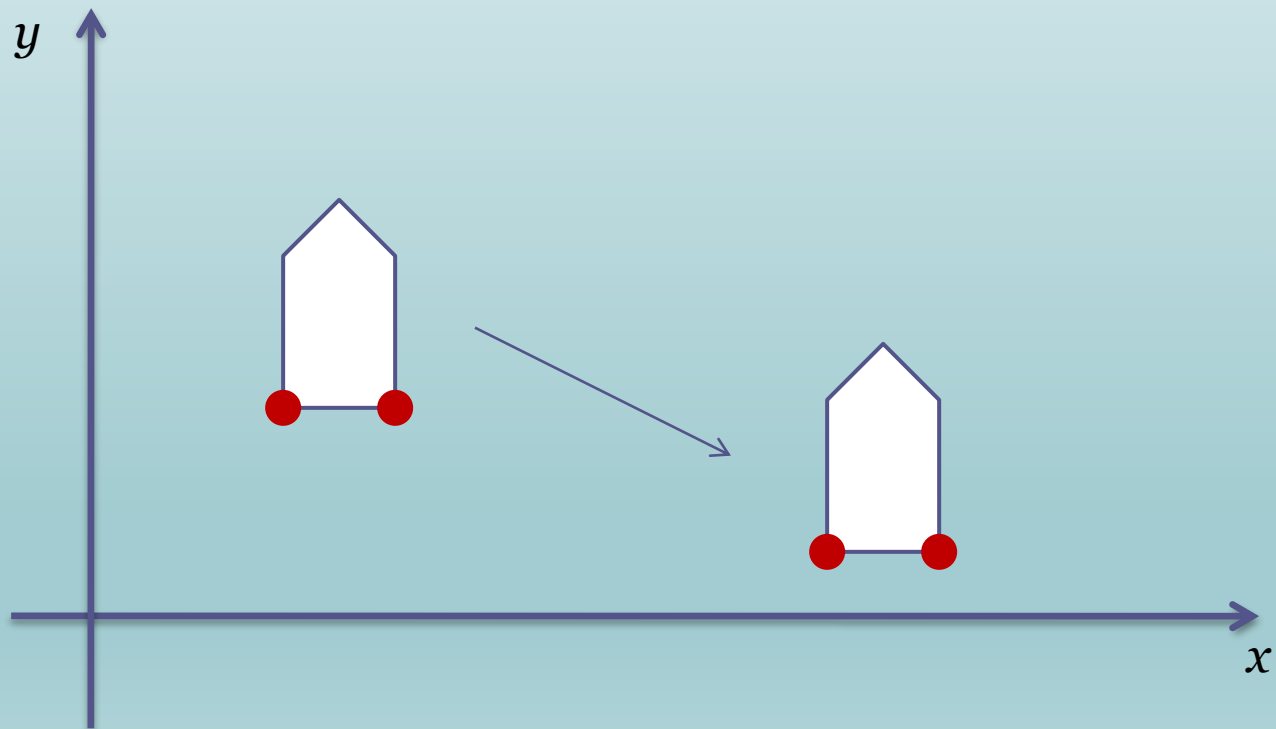
Translation (1/2)

- For each $P(x, y)$ to be moved by d_x, d_y units parallel to the x and y axis we get a new point $P'(x', y')$

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

$$P = \begin{bmatrix} x \\ y \end{bmatrix}, \quad P' = \begin{bmatrix} x' \\ y' \end{bmatrix}, \quad T = \begin{bmatrix} d_x \\ d_y \end{bmatrix} \quad \Rightarrow \quad P' = P + T$$

Translation (2/2)



Scaling (1 / 2)

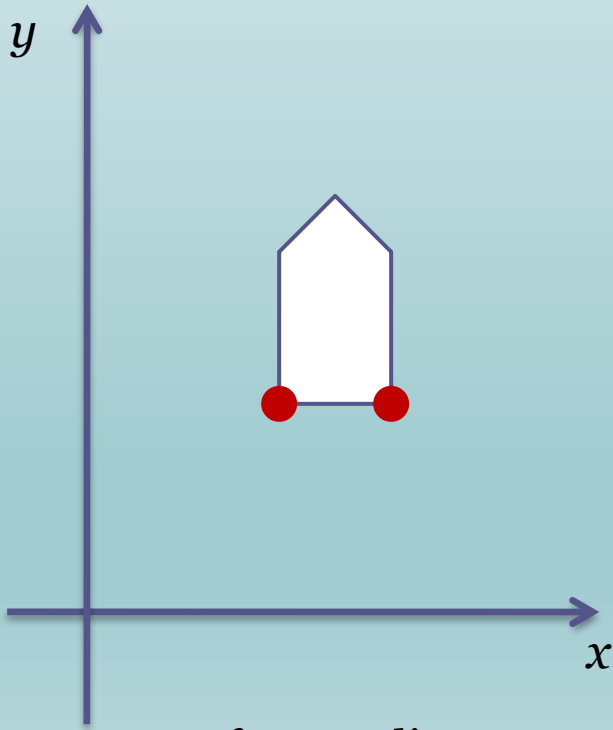
- Points can be stretched by S_x and S_y along the x -axis and y -axis respectively into new points by the multiplications

$$x' = S_x \cdot x \quad y' = S_y \cdot y$$

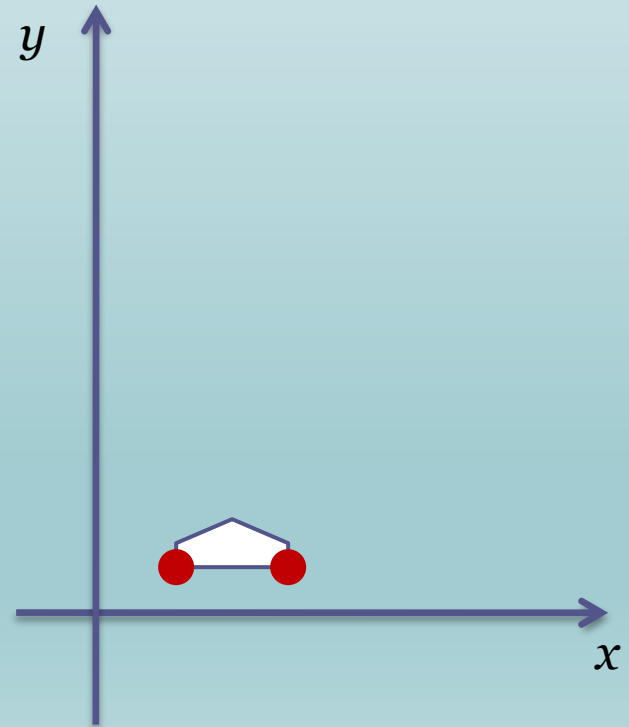
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \quad \text{or} \quad P' = S \cdot P$$

Scaling (2/2)

- Scaling about the origin



Before Scaling



After Scaling

Rotation (1 / 5)

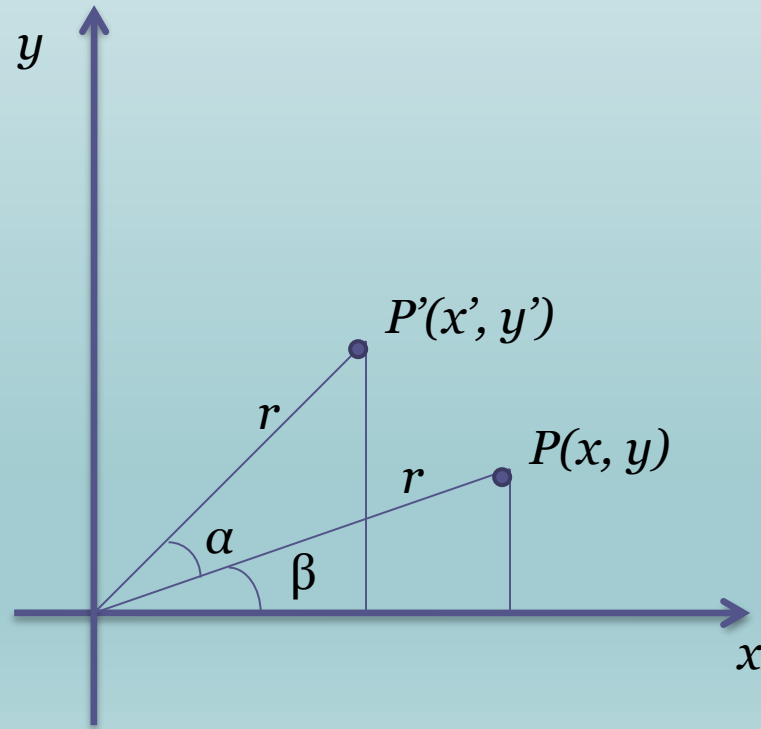
- Points can be rotated through an angle α about the origin, defined as

$$x' = x \cdot \cos \alpha - y \cdot \sin \alpha$$

$$y' = x \cdot \sin \alpha + y \cdot \cos \alpha$$

- These two equations are derived from the following

Rotation (2/5)



Rotation (3/5)

- Rotation by α transforms $P(x, y)$ into $P'(x', y')$
- Rotation along the origin, meaning the distances from the origin to P and P' is r

$$x = r \cdot \cos \beta, \quad y = r \cdot \sin \beta$$

$$x' = r \cdot \cos(\alpha + \beta) = r \cdot \cos \beta \cdot \cos \alpha - r \cdot \sin \beta \cdot \sin \alpha$$

$$y' = r \cdot \sin(\alpha + \beta) = r \cdot \cos \beta \cdot \sin \alpha + r \cdot \sin \beta \cdot \cos \alpha$$

Rotation (4/5)

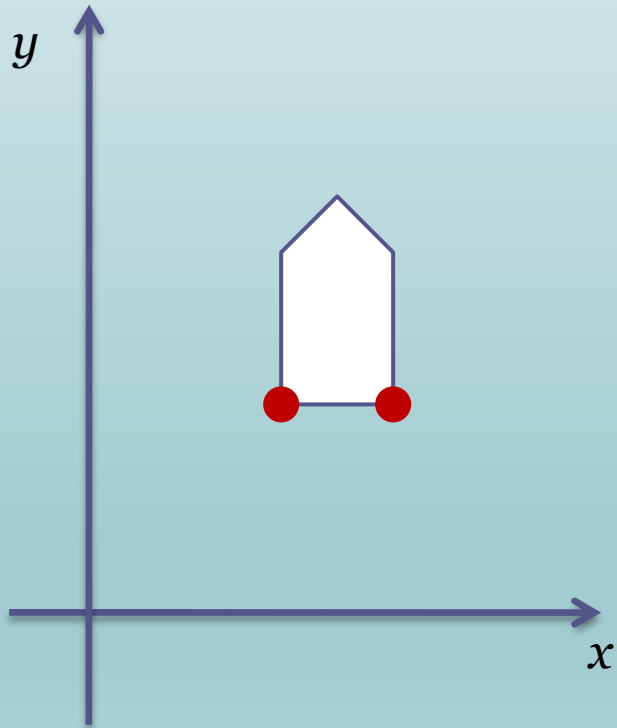
- The rotation matrix would be

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \quad \text{or} \quad P' = R \cdot P$$

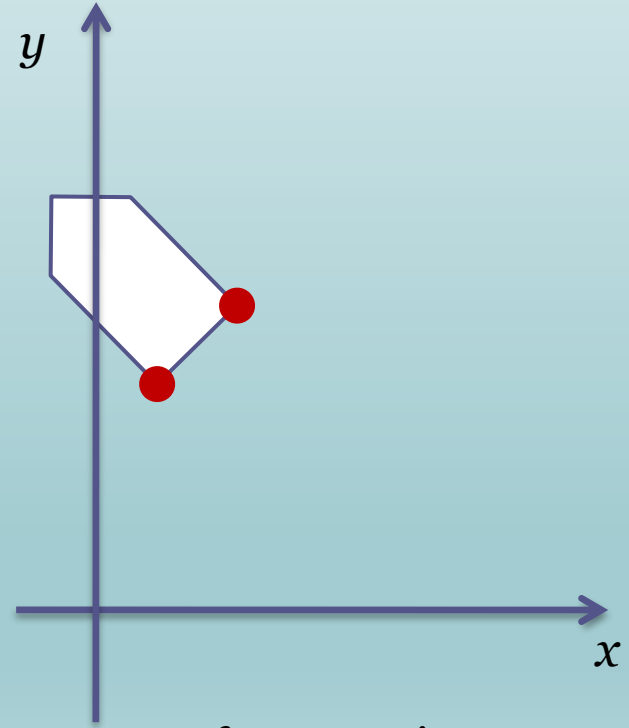
- Where positive angles are measured counterclockwise from x towards y

$$\cos(-\alpha) = \cos \alpha \qquad \sin(-\alpha) = -\sin \alpha$$

Rotation (5/5)



Before Rotation



After Rotation

Overview

- 2D Transformations
- Homogeneous Coordinates and Matrix Representation
- Composition of 2D Transformations

Homogeneous (1 / 3)

$$P' = P + T$$

$$P' = S \cdot P$$

$$P' = R \cdot P$$

- Problem: Translation can not be represented as 2x2 matrix
- Solution: Use **homogeneous coordinates** by adding a third coordinate w

Homogeneous (2/3)

- Instead of points represented by pairs (x, y) we use triples (x, y, w)
- Points (x, y, w) and (x', y', w') are considered equal (represent the same 2D point) if one is scalar multiple of the other:

$$\text{e.g. } (4, 2, 1) \equiv (8, 4, 2) \equiv (16, 8, 4)$$

Homogeneous (3/3)

- Divide by w to obtain a 2D point:

$$(24, 16, 4) \equiv \left(\frac{24}{4}, \frac{16}{4}, \frac{4}{4}\right) = (6, 4, 1)$$

- For **points** we use $w = 1$

$$(6, 4) \mapsto (6, 4, 1)$$

- For **direction vectors** we use $w = 0$ (points at ∞)

Translation

- In the 3x3 matrix form for homogeneous coordinates, the translation equation is:

$$P' = T(d_x, d_y) \cdot P$$

$$T(d_x, d_y) = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix}$$

Exercise

- What happens if a point P is translated by $T(d_{x1}, d_{y1})$ to P' and then translated by $T(d_{x2}, d_{y2})$ to P'' ?

Scaling

$$P' = S(S_x, S_y) \cdot P$$

$$S(S_x, S_y) = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- What about successive scaling?

Rotation

$$P' = R(\alpha) \cdot P$$

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

- What about successive rotations?

Identity

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

Rigid-Body Transformations

- Has the form

$$\begin{bmatrix} \cos \alpha & -\sin \alpha & d_x \\ \sin \alpha & \cos \alpha & d_y \\ 0 & 0 & 1 \end{bmatrix}$$

- The upper 3x3 matrix is orthogonal
 - Preserves angles and lengths
- A unit square after transformation will remain a unit square

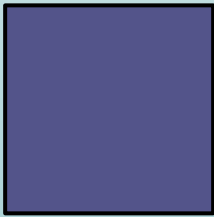
Affine Transformations (1/2)

- What about the product of an arbitrary sequence of rotation, translation and scale matrices?

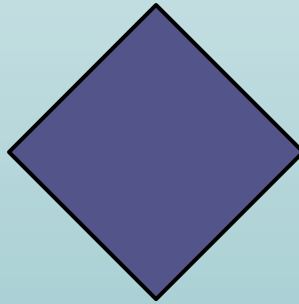
Answer: Affine transformations

- Affine transformations, have the property of preserving parallelism of lines but **not** lengths and angles
 - That also goes for a sequence of rotation, scale and translation operations

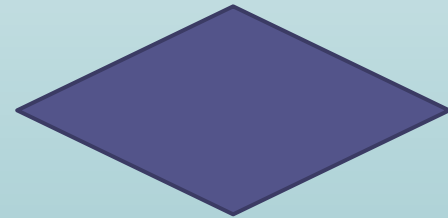
Affine Transformations (2/2)



Unit Cube



Rotation
about 45°



Scale in x ,
not in y

Overview

- 2D Transformations
- Homogeneous Coordinates and Matrix Representation
- **Composition of 2D Transformations**

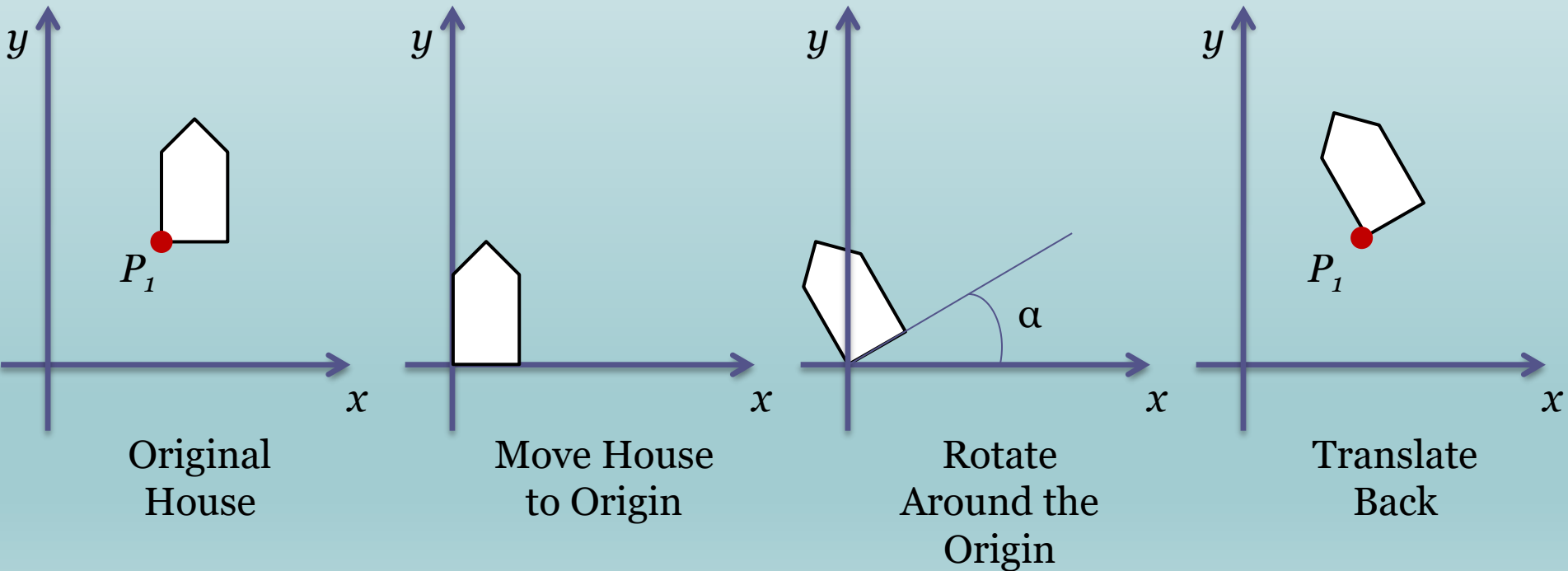
Composition of 2D Transformations

- Purpose:
 - Gain efficiency by applying a single composed transformation to a point, rather than a series of them

Rotation About Some Arbitrary Point (1 / 3)

- To rotate about an arbitrary point $P_1(x_1, y_1)$ we need a sequence of transformations:
 1. Translate P_1 by $(-x_1, -y_1)$ (i.e. move to origin)
 2. Rotate by α
 3. Translate P_1 by (x_1, y_1) (i.e. translate back)

Rotation About Some Arbitrary Point (2/3)



Rotation About Some Arbitrary Point (3/3)

- The transformation is

$$\begin{aligned} T(x_1, y_1) \cdot R(\alpha) \cdot T(-x_1, -y_1) = \\ \begin{bmatrix} 1 & 0 & x_1 \\ 0 & 1 & y_1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{bmatrix} = \\ \begin{bmatrix} \cos \alpha & -\sin \alpha & x_1(1 - \cos \alpha) + y_1 \sin \alpha \\ \sin \alpha & \cos \alpha & y_1(1 - \cos \alpha) - x_1 \sin \alpha \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

Scale About Some Arbitrary Point

- The transformation is

$$\begin{aligned} &T(x_1, y_1) \cdot S(s_x, s_y) \cdot T(-x_1, -y_1) = \\ &\begin{bmatrix} 1 & 0 & x_1 \\ 0 & 1 & y_1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{bmatrix} = \\ &\begin{bmatrix} S_x & 0 & x_1(1-S_x) \\ 0 & S_y & y_1(1-S_y) \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

Matrix Multiplications (review)

- Matrix multiplication is associative:

$$\begin{aligned} \mathbf{C}(\mathbf{B}(\mathbf{A}\mathbf{x})) &= \mathbf{C}((\mathbf{B}\mathbf{A})\mathbf{x}) = (\mathbf{C}(\mathbf{B}\mathbf{A}))\mathbf{x} \\ &= ((\mathbf{C}\mathbf{B})\mathbf{A})\mathbf{x} \\ &= (\mathbf{C}\mathbf{B})(\mathbf{A}\mathbf{x}) \end{aligned}$$

- Matrix multiplication is **not** commutative

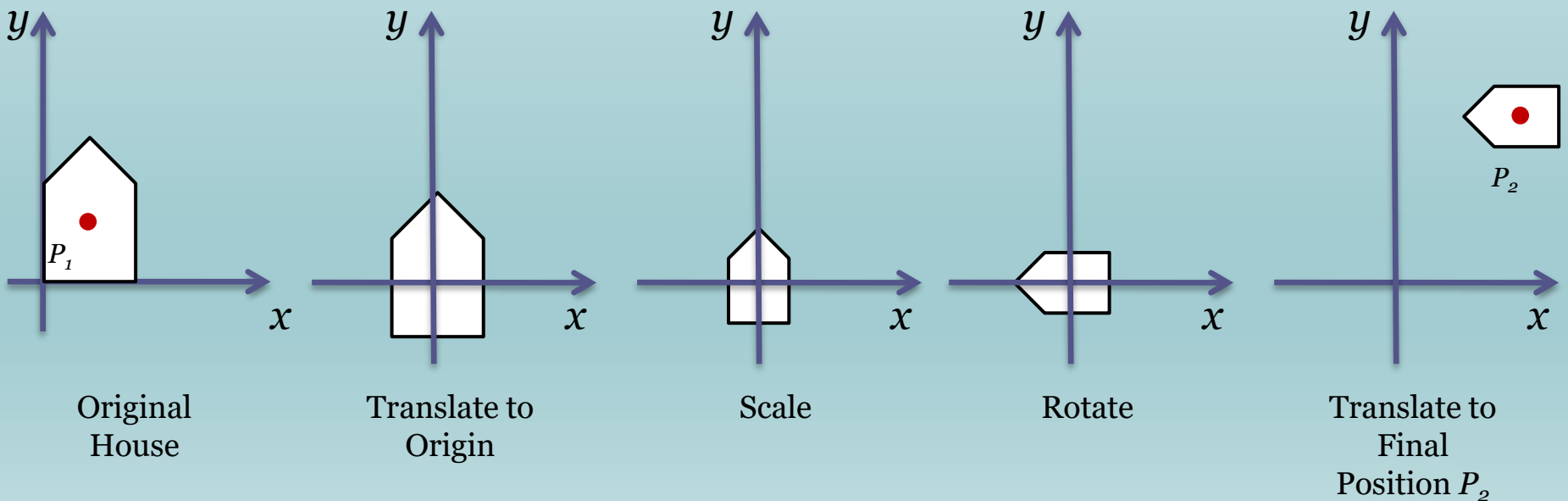
$$\mathbf{AB} \neq \mathbf{BA}$$

A rotation followed by a translation is not the same as a translation followed by a rotation

Applying Transformation Matrix (1 / 3)

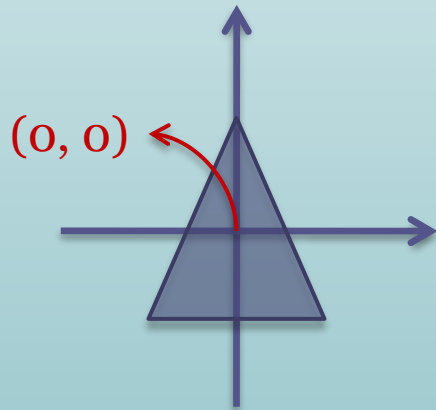
- General form:

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

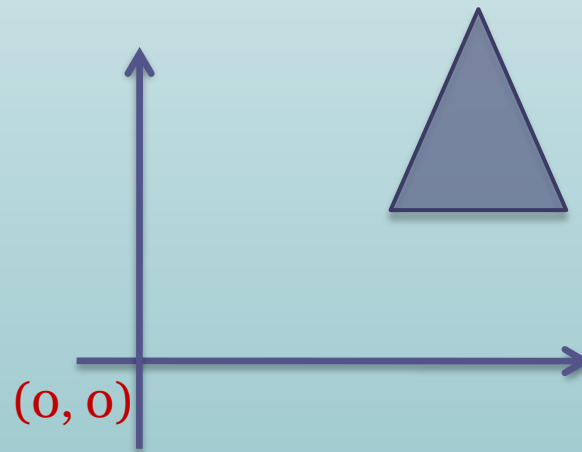


Applying Transformation Matrix (2/3)

- Having our object in its local coordinate system



Sprite's Local
Coordinate System



World Coordinate
System

Applying Transformation Matrix (3/3)

- We will apply

$$\begin{bmatrix} posX' \\ posY' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} posX \\ posY \\ 1 \end{bmatrix}$$

Applying Transformation Matrix (3/3)

- We will apply

$$\begin{bmatrix} posX' \\ posY' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} posX \\ posY \\ 1 \end{bmatrix}$$

- In this formulation, we see how the graphics transforms each vertex ($posX$, $posY$) that belongs to a mesh from it's local space to a global-world space

Applying Transformation Matrix (3/3)

- We will apply

$$\begin{bmatrix} posX' \\ posY' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} posX \\ posY \\ 1 \end{bmatrix}$$

- Application in Alpha Engine:
 - AEMtx33 trans, rot, scale; //to store the resultant matrices
 - AEMtx33Trans(&trans, d_x , d_y); // dx , dy are the position of the game object
 - AEMtx33Rot(&rot, α); // **Alpha** is the orientation angle of the game object
 - AEMtx33Scale(&scale, S_x , S_y); // **sx**, **sy** are the scale of the game object

Applying Transformation Matrix (3/3)

- Application in Alpha Engine:
 - Concatenation of matrices (matrix multiplication)
 - `AEMtx33 result, m1, m2;`
 - `AEMtx33Concat(&result, m1, m2);`//this means: $\text{result} = m1 * m2$
 - Sending the result matrix to Alpha Engine graphics system
 - `AEGfxSetTransform(&result.m);`//m is a matrix data representation