2D Geometric Transformations

Fall 2018

Seungyong Lee

POSTECH

Overview

- Geometric transformations
  - changing the positions of points
  - translation, scaling, rotation, and so on
  - animating objects and camera

Overview (2)

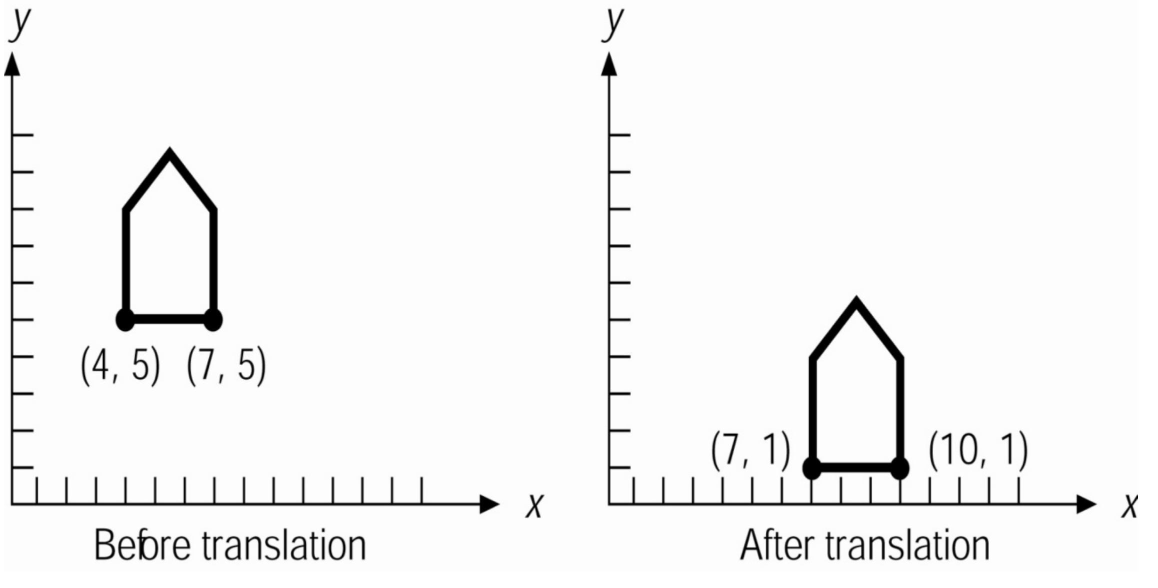
- Chapter summary
  - 2D primitive transformations
  - homogeneous coordinate system
  - other basic 2D transformations
  - general transformations
  - transformation process
- Related materials
  - Angel: Chapter 3
  - H&B: Sections 5.1 - 5.6

Primitive Transformations

- Translation

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

$$P' = P + T$$



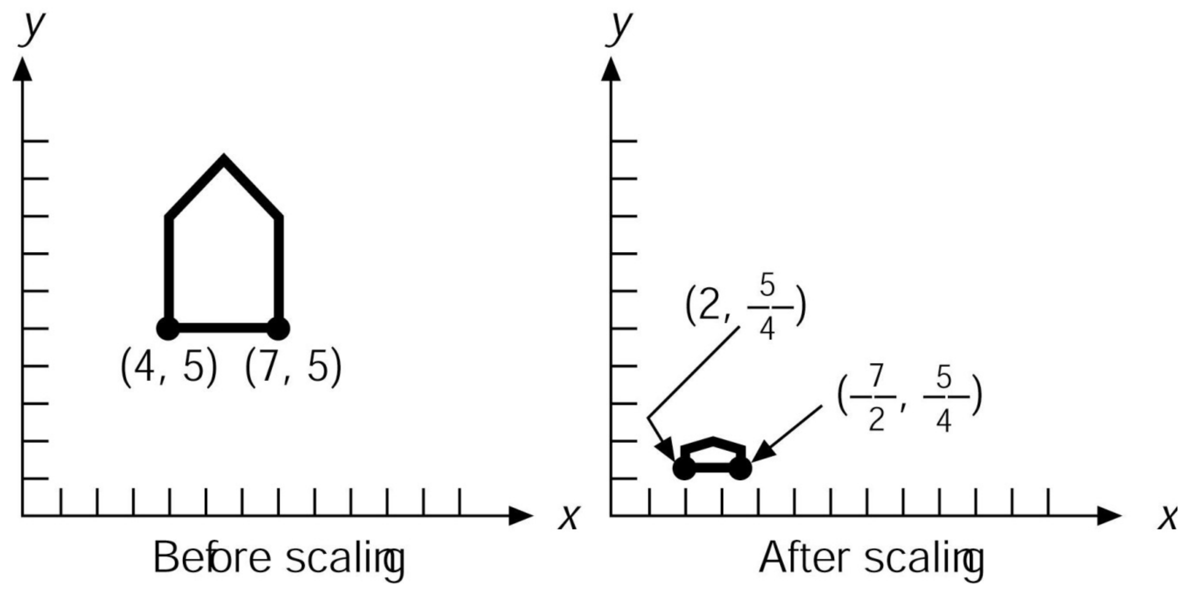
Primitive Transformations (2)

- Scaling: uniform, nonuniform

$$x' = s_x x,\ y' = s_y y$$

$$P' = SP$$

$$P = \begin{bmatrix} x \\ y \end{bmatrix}, P' = \begin{bmatrix} x' \\ y' \end{bmatrix}, S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$



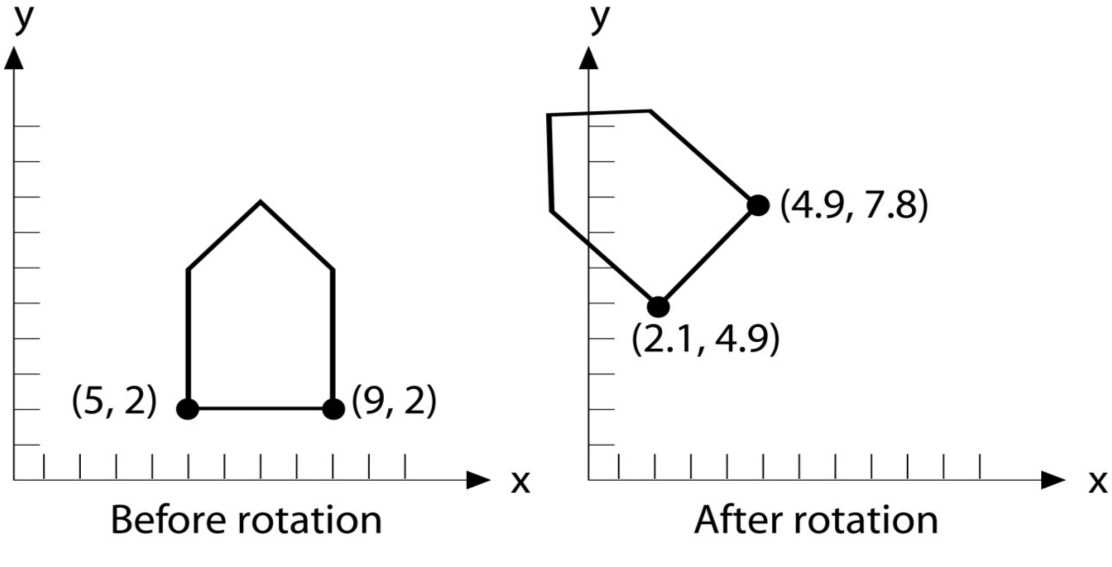
Primitive Transformations (3)

- Rotation

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

$$P' = RP$$

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



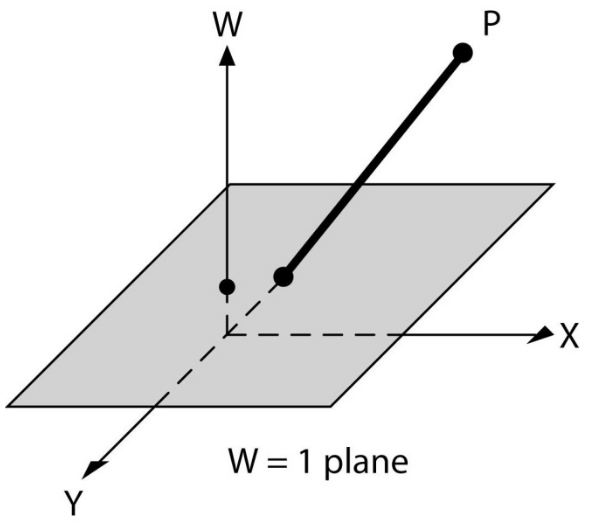
Homogeneous Coordinate System

- Matrix representation of a transformation
  - translation: P' = P + T
  - scaling: P' = SP
  - rotation: P' = RP
- Concatenation of transformations
  - two translations
    - P'' = P' + T2 = (P + T1) + T2 = P + (T1 + T2)
  - two rotations
    - P'' = R2P' = R2(R1P) = (R2R1)P
  - translation and rotation
    - P'' = RP' = R(P + T) = RP + RT

Homogeneous Coordinate System (2)

- Homogeneous coordinates
  - Cartesian → homogeneous: (x, y) → (x, y, 1)
  - $(x, y, w) = (\alpha x, \alpha y, \alpha w),\ \alpha \neq 0$
  - homogeneous → Cartesian: (x, y, w) → (x/w, y/w)
  - w = 0? → point at infinity
  - translation can be represented by a matrix multiplication

$$P' = TP, \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Homogeneous Coordinate System (3)

- Scaling and rotation in homogeneous coordinates

$$\boldsymbol{P' = SP}, \text{ where } \boldsymbol{P} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \boldsymbol{P'} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}, \boldsymbol{S} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{P' = RP}, \text{ where } \boldsymbol{P} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \boldsymbol{P'} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}, \boldsymbol{R} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

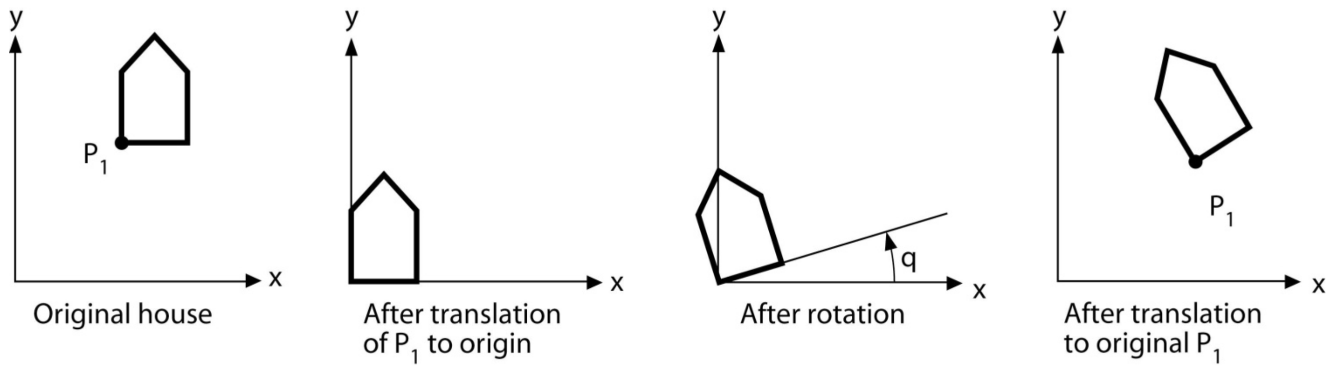
  – now every primitive transformation can be represented by a 3 × 3 matrix!!!

Homogeneous Coordinate System (4)

- Concatenations in homogeneous coordinates
  - two translations
    - $P'' = T_2\, P' = T_2(T_1P) = (T_2T_1)P$
  - two rotations
    - $P'' = R_2P' = R_2(R_1P) = (R_2R_1)P$
  - translation and rotation
    - $P'' = RP' = R(TP) = (RT)P$
  - any concatenation of transformations can be reduced to a single 3×3 matrix multiplication

Other Basic Transformations

- Primitive transformations can be concatenated to obtain other basic transformations
- Rotation about an arbitrary point
  - T(x1, y1) R(θ) T(–x1, –y1)



- Scaling about an arbitrary point
  - T(x1, y1) S(sx, sy) T(–x1, –y1)

Other Basic Transformations (2)

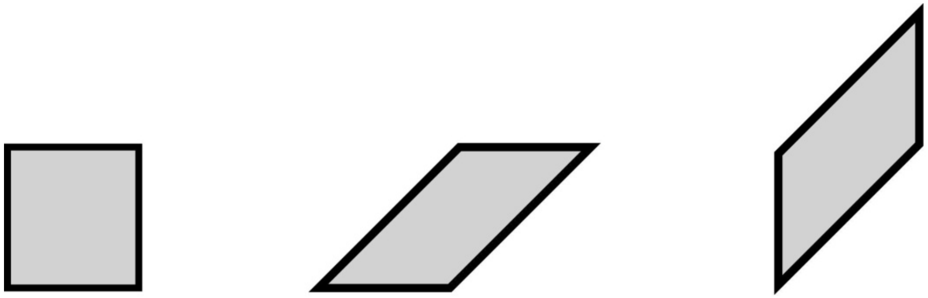
- General scaling directions
  - $R(-\theta)\, S(s_1, s_2)\, R(\theta)$
- Reflection
  - about x or y axis
    - nonuniform scaling: $S(1, -1)$ or $S(-1, 1)$
  - about the diagonal line $y = x$
    - rotation and coordinate-axis reflection

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ or } \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Other Basic Transformations (3)

- Shear transformation
  - x-direction shear relative to x axis
  - y-direction shear
  - can be represented by a concatenation of primitive transformations?
    - rotation → scaling → rotation → scaling

$$SH_x = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow SH_x \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + ay \\ y \\ 1 \end{bmatrix}$$

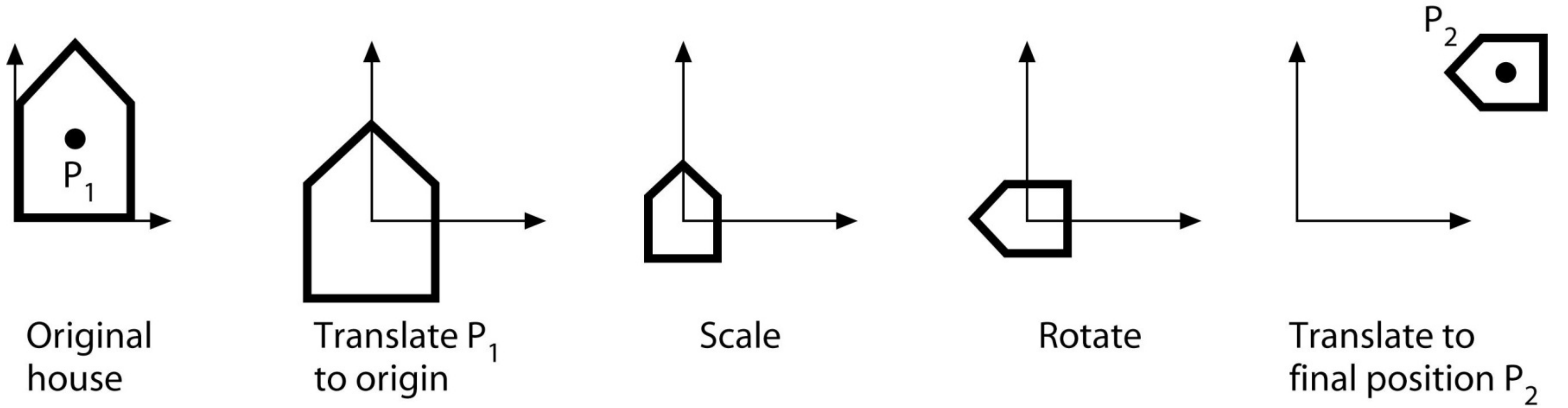


The unit cube sheared in the x direction

The unit cube sheared in the y direction

General Transformations

- Composition of primitive transformations
  - T(x2, y2) R(θ) S(sx, sy) T(–x1, –y1)



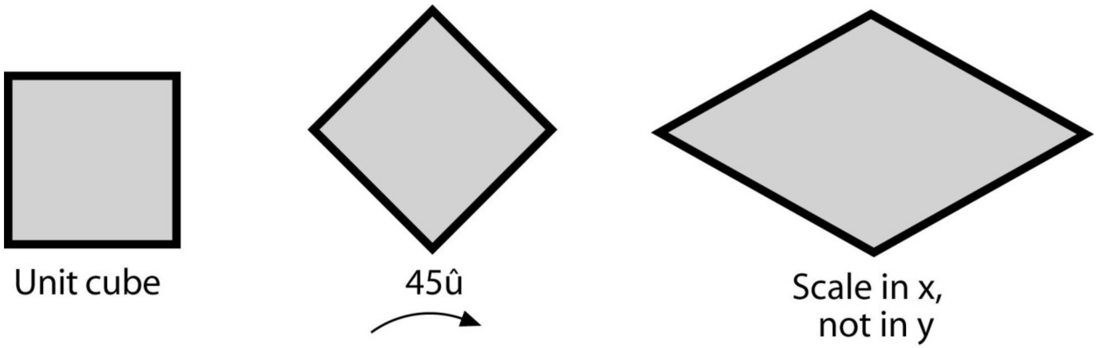
- Transformation matrix
  - translation, scaling, rotation, shear

$$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

General Transformations (2)

- Rigid-body transformation
  - (r_{11}, r_{12}) and (r_{21}, r_{22}) are orthonormal
- Affine transformation

$$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$



- $M_1 M_2 =? M_2 M_1$
  - the order is important in a composite transformation

Transformation Process

- Vertex coordinate change approach
  - algorithm

    divide the desired transformation into primitive ones

    compute the composite transformation matrix

    for each vertex of an object do

      Cartesian coordinates $\rightarrow$ homogeneous coordinates

      compute new coordinates by one matrix multiplication

      homogeneous coordinates $\rightarrow$ Cartesian coordinates

    draw the object
  - efficient computation

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \rightarrow \begin{array}{l} x = r_{11}x + r_{12}y + t_x \\ y = r_{21}x + r_{22}y + t_y \end{array}$$

Transformation Process (2)

- OpenGL transformation process
  - modeling coordinates
  - need not change the vertex coordinates
  - manipulation of the transformation matrix
  - current transformation matrix
  - transformation functions
    - construct a matrix and multiply it by the current matrix
    - glLoadIdentity();
    - glTranslatef(dx, dy, dz);
    - glRotatef(angle, vx, vy, vz);
    - glScalef(sx, sy, sz);

Transformation Process (3)

- Details of OpenGL transformation process
  - model-view matrix
  - order of the transformations?
  - matrix stack (*)
  - transforming several objects?

Summary

- Primitive transformations
  - translation, rotation, scaling
- Homogeneous coordinates
  - translation by a matrix multiplication
- General transformations
  - composite transformation by matrix multiplications
  - can always be represented by a 3×3 matrix
- Transformations in OpenGL
  - model-view matrix