CS 4731/543: Computer Graphics
Lecture 3 (Part I): Introduction to Transforms, 2D transforms

Emmanuel Agu

Introduction to Transformations

- Transformation changes an objects:
  - Position (translation)
  - Size (scaling)
  - Orientation (rotation)
  - Shapes (shear)
- We will introduce first in 2D or *(x,y)*, build intuition
- Later, talk about 3D and 4D?
- Transform object by applying sequence of matrix multiplications to object vertices

Why Matrices?

- All transformations can be performed using matrix/vector multiplication
- Allows pre-multiplication of all matrices
- Note: point (x,y) needs to be represented as (x,y,1), also called **Homogeneous coordinates**

Point Representation

- We use a column matrix (2x1 matrix) to represent a 2D point

$$\begin{pmatrix} x \\ y \end{pmatrix}$$

- General form of transformation of a point *(x,y)* to *(x',y')* can be written as:

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

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

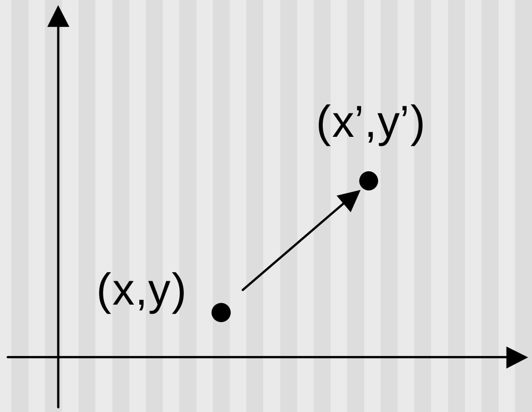
or

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

Translation

- To reposition a point along a straight line
- Given point *(x,y)* and translation distance (t_x, t_y)
- The new point: *(x',y')*

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



or

$$P' = P + T \quad \text{where} \quad P' = \begin{pmatrix} x' \\ y' \end{pmatrix} \quad P = \begin{pmatrix} x \\ y \end{pmatrix} \quad T = \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

3x3 2D Translation Matrix

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

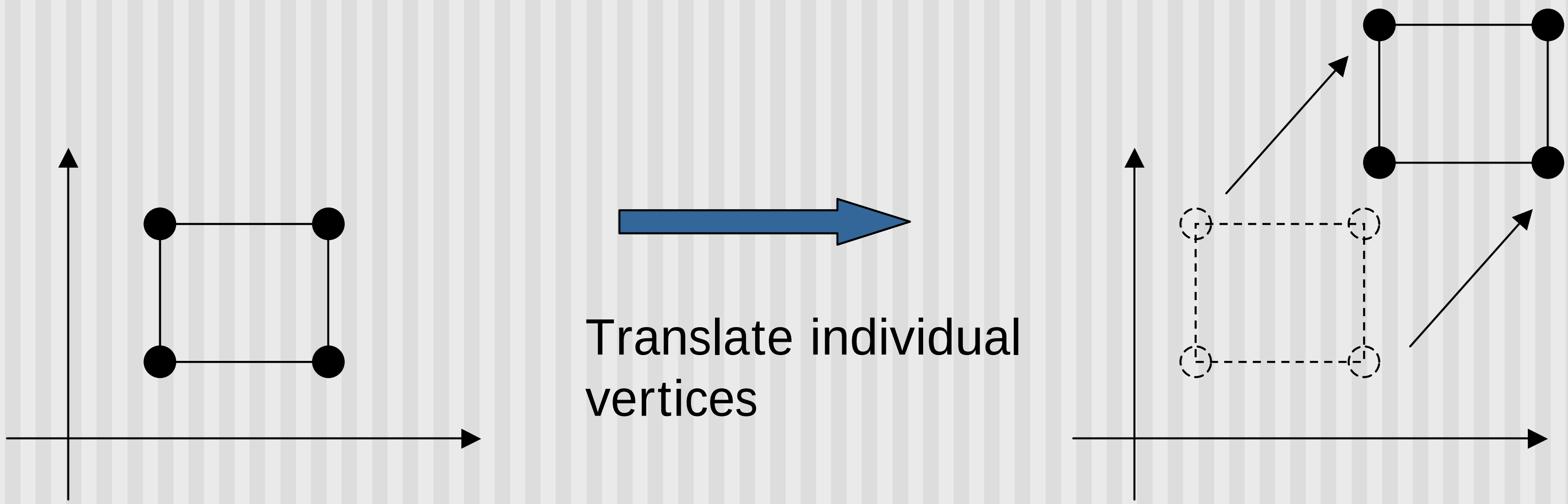
use 3x1 vector

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

- Note: it becomes a matrix-vector multiplication

Translation of Objects

- How to translate an object with multiple vertices?

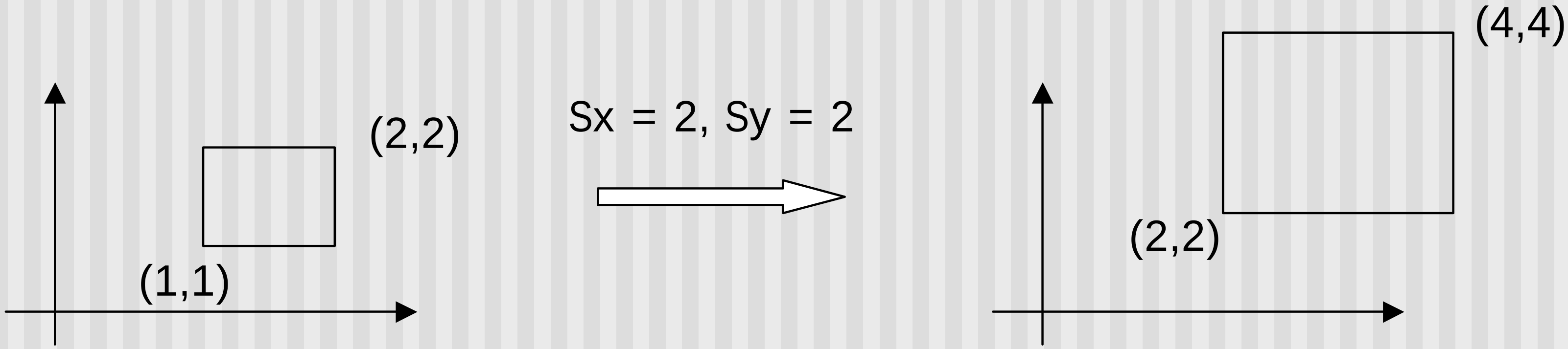


2D Scaling

- Scale: Alter object size by scaling factor (s_x, s_y). i.e

$$x' = x \cdot Sx$$
$$y' = y \cdot Sy$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} Sx & 0 \\ 0 & Sy \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

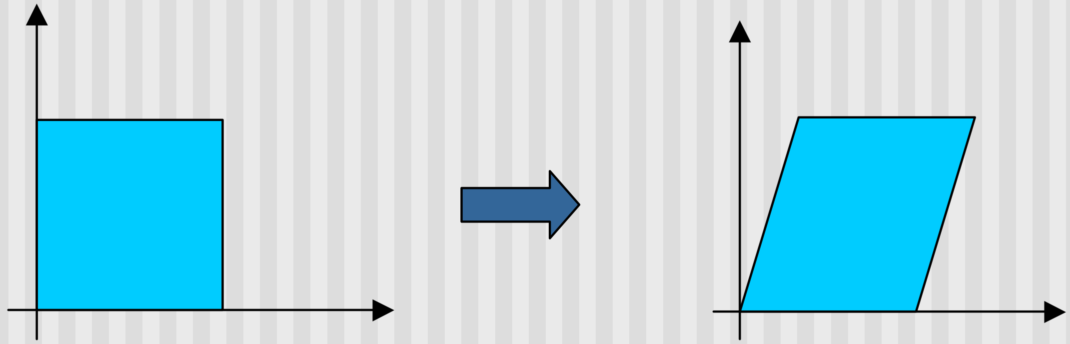


3x3 2D Scaling Matrix

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} Sx & 0 \\ 0 & Sy \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

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

Shearing



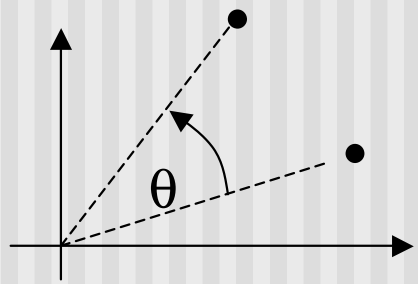
- Y coordinates are unaffected, but x cordinates are translated linearly with y
- That is:
  - **y' = y**
  - **x' = x + y * h**

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

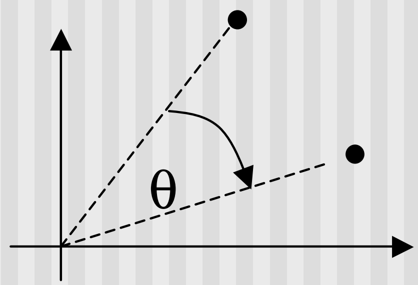
- h is fraction of y to be added to x

2D Rotation

- Default rotation center is origin *(0,0)*



$\theta > 0$: Rotate counter clockwise

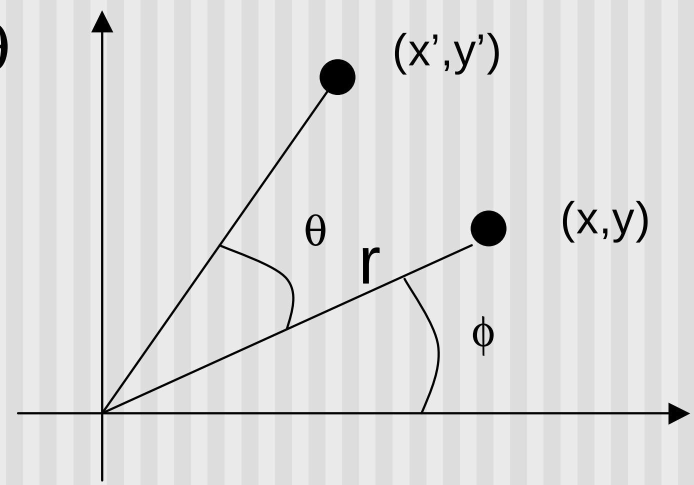


$\theta < 0$: Rotate clockwise

Rotation

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

$\longrightarrow$ (x’, y’)



How to compute (x’, y’) ?

$x = r\cos(f) \qquad y = r\sin(f)$

$x' = r\cos(f + q) \qquad y = r\sin(f + q)$

Rotation

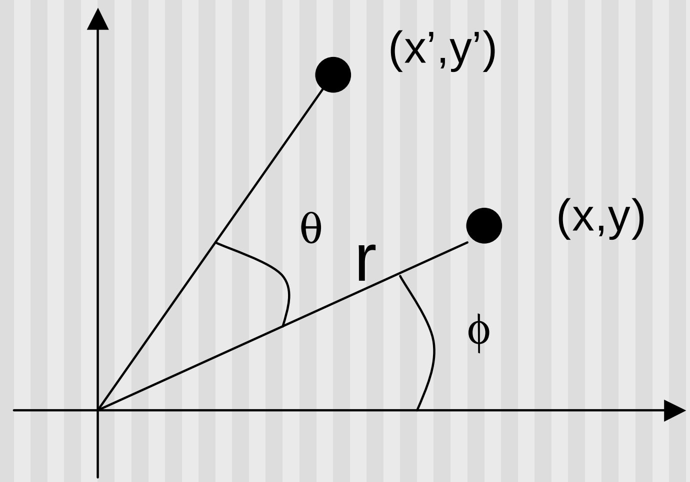
Using trig identities

$$\cos(q + f) = \cos q \cos f - \sin q \sin f$$

$$\sin(q + f) = \sin q \cos f + \cos q \sin f$$

x' = x cos(q) – y sin(q)

y' = y cos(q) + x sin(q)



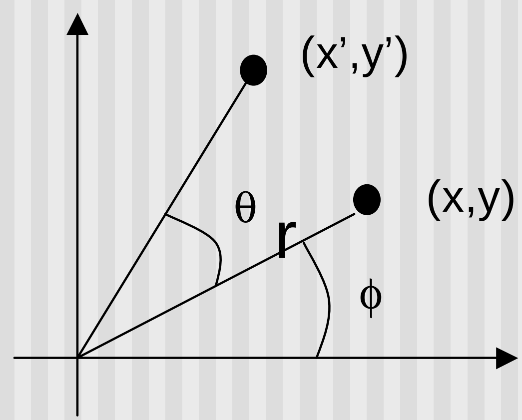
Matrix form?

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

3 x 3?

3x3 2D Rotation Matrix

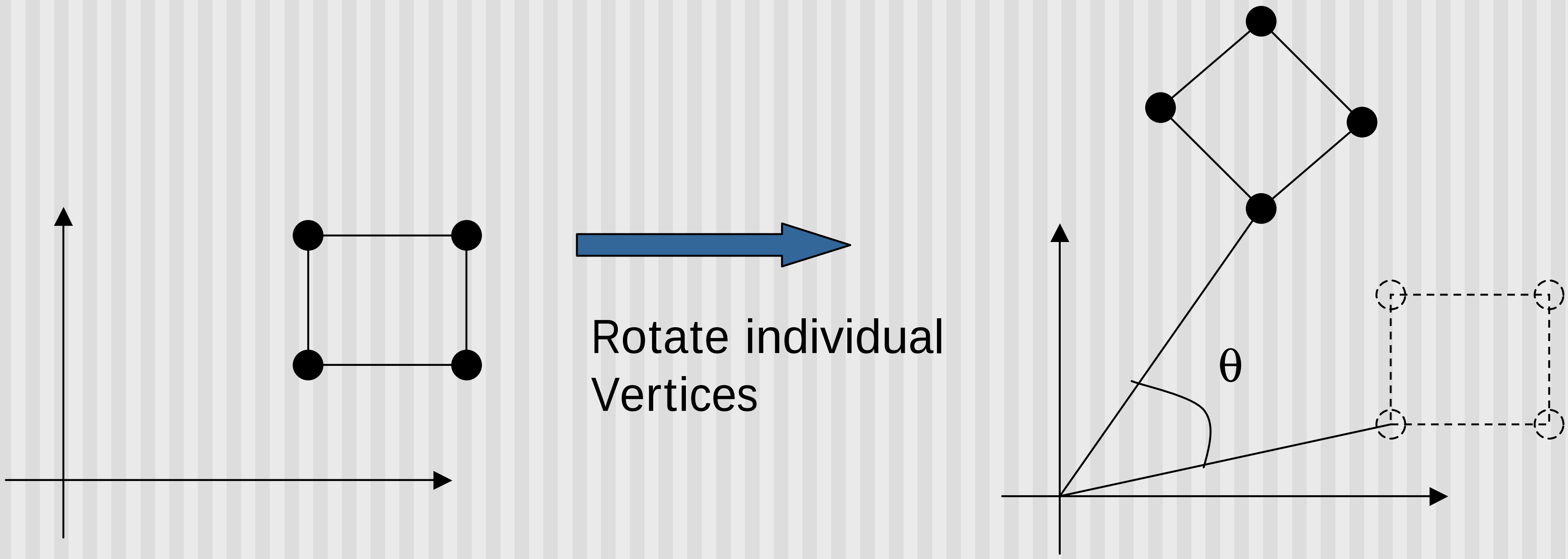
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos(q) & -\sin(q) \\ \sin(q) & \cos(q) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$



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

Rotation

- How to rotate an object with multiple vertices?



Arbitrary Rotation Center

- To rotate about arbitrary point P = (Px, Py) by θ:
  - Translate object by T(-Px, -Py) so that P coincides with origin
  - Rotate the object by R(θ)
  - Translate object back: T(Px, Py)
- In matrix form: T(Px,Py) R(θ) T(-Px,-Py) * P

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & Px \\ 0 & 1 & Py \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(q) & -\sin(q) & 0 \\ \sin(q) & \cos(q) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -Px \\ 0 & 1 & -Py \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

▪Similar for arbitrary scaling anchor,

Composing Transformation

- Composing transformation – applying several transforms in succession to form one overall transformation
- Example:

  M1 X M2 X M3 X P

  where M1, M2, M3 are transform matrices applied to P
- Be careful with the order
- For example:
  - Translate by (5,0) then rotate 60 degrees is NOT same as
  - Rotate by 60 degrees then translate by (5,0)

OpenGL Transformations

- Designed for 3D
- For 2D, simply ignore *z* dimension
- Translation:
  - glTranslated(*tx, ty, tz)*
  - glTranslated*(tx, ty, 0)* => for 2D
- Rotation:
  - glRotated(*angle, Vx, Vy, Vz*)
  - glRotated(*angle, 0, 0, 1*) => for 2D
- Scaling:
  - glScaled(sx, sy, sz)
  - glScaled(sx, sy, 0) => for 2D

References

- Hill, chapter 5.2