2D transformations and homogeneous coordinates

POONAM DHAND

RIDHIMA DUTTA

Map of the lecture

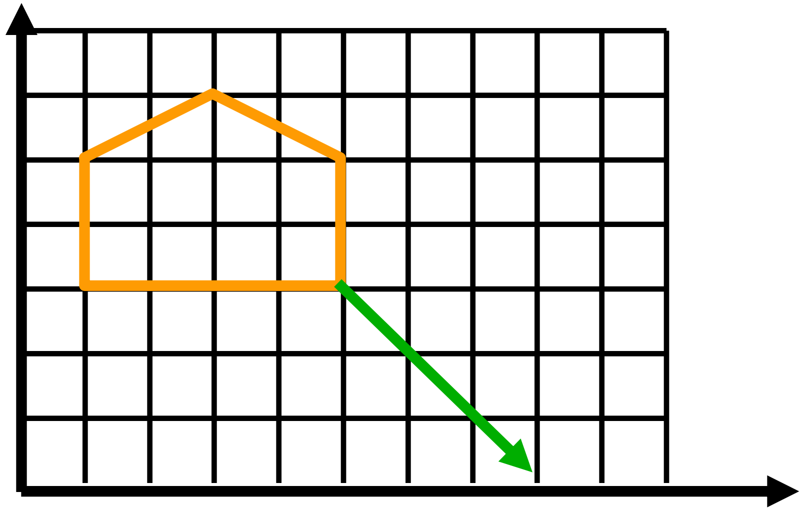
- Transformations in 2D:
  - vector/matrix notation
  - example: translation, scaling, rotation
- Homogeneous coordinates:
  - consistent notation
  - several other good points (later)
- Composition of transformations
- Transformations for the window system

Transformations in 2D

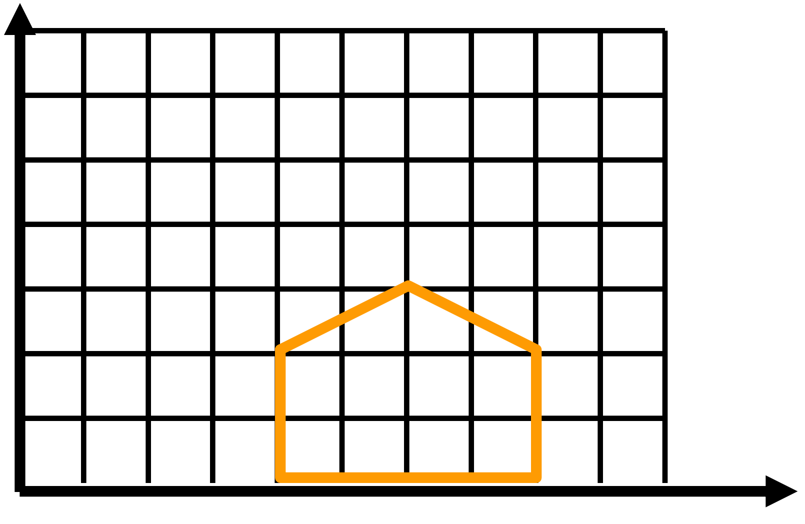
- In the application model:
  - a 2D description of an object (vertices)
  - a transformation to apply
- Each vertex is modified:
  - $x' = f(x,y)$
  - $y' = g(x,y)$
- Express the modification

Translations

- Each vertex is modified:
  - $x' = x + t_x$
  - $y' = y + t_y$



Before



After

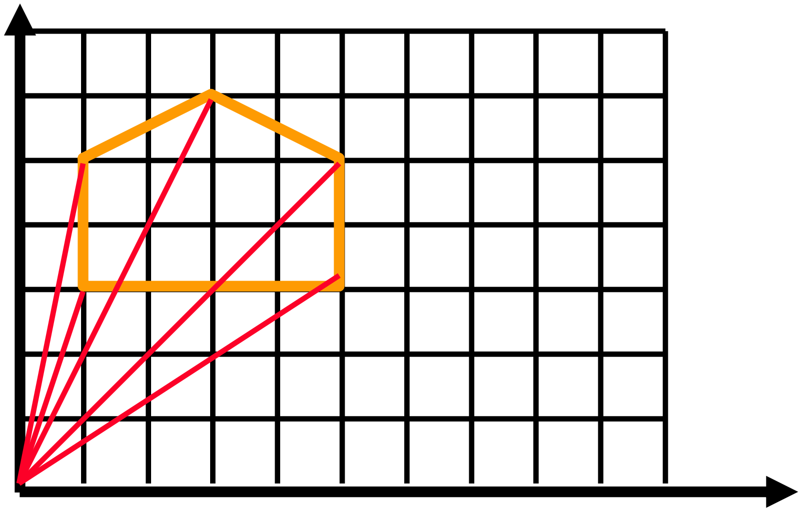
Translations: vector notation

- Use vector for the notation:
  - makes things simpler
- A point is a vector: $\begin{bmatrix} x \\ y \end{bmatrix}$
- A translation is merely a vector sum:

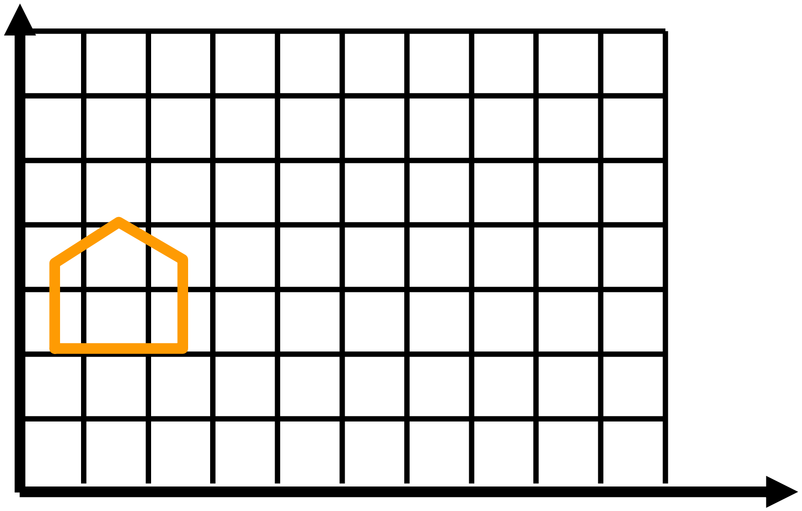
$$P' = P + T$$

Scaling in 2D

- Coordinates multiplied by the scaling factor:
  - $x' = s_x\, x$
  - $y' = s_y\, y$



Before



After

Scaling in 2D, matrix notation

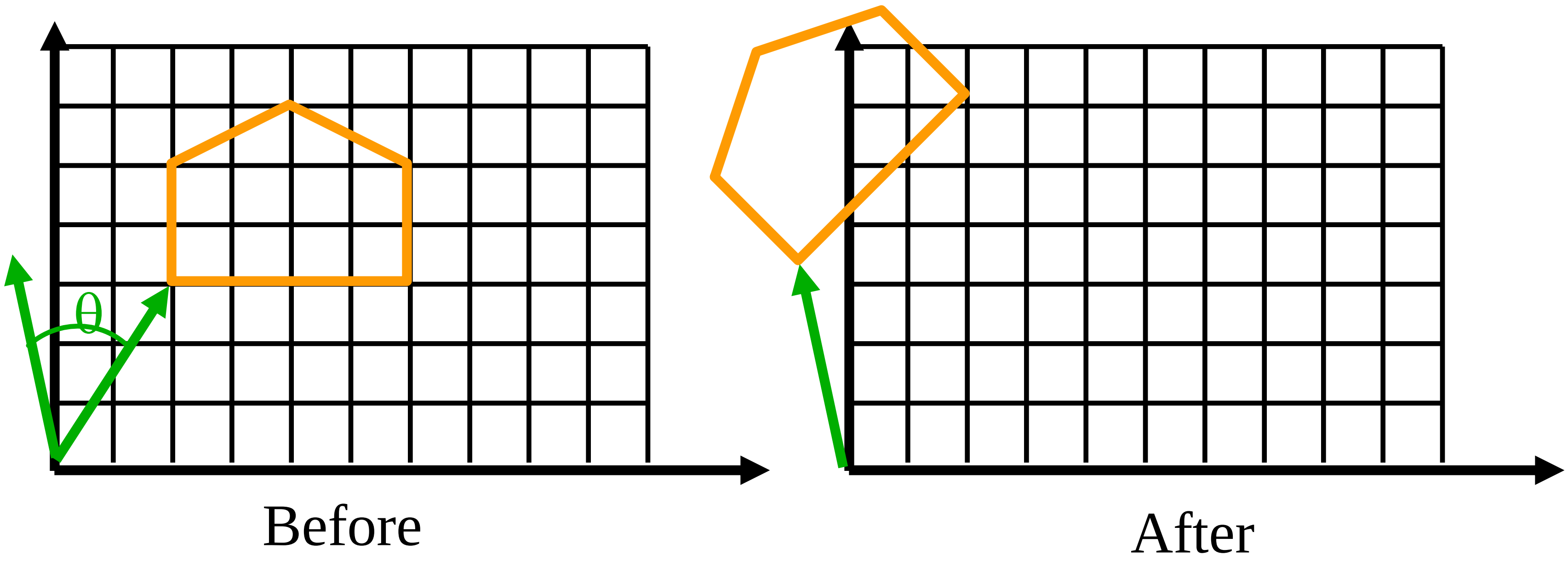
- Scaling is a matrix multiplication:

$$P' = \mathbf{S}P$$

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

Rotating in 2D

- New coordinates depend on *both* x and y
  - $x' = \cos\theta\, x - \sin\theta\, y$
  - $y' = \sin\theta\, x + \cos\theta\, y$



Rotating in 2D, matrix notation

- A rotation is a matrix multiplication:

$$P' = \mathbf{R}P$$

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

2D transformations, summary

- Vector-matrix notation simplifies writing:
  - translation is a vector sum
  - rotation and scaling are matrix-vector multiplication
- I would like a consistent notation:
  - that expresses all three identically
  - that expresses combination of these also identically
- How to do this?

Homogeneous coordinates

- Introduced in mathematics:
  - for projections and drawings
  - used in artillery, architecture
  - used to be classified material (in the 1850s)
- Add a third coordinate, w
- A 2D point is a 3 coordinates vector:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Homogeneous coordinates (2)

- Two points are equal if and only if:
  $x'/w' = x/w$ and $y'/w' = y/w$
- $w=0$: points at infinity
  - useful for projections and curve drawing
- Homogenize = divide by w.
- Homogenized points: $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

Translations with homogeneous

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

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

$$\begin{cases} \dfrac{x'}{w'} = \dfrac{x}{w} + t_x \\ \dfrac{y'}{w'} = \dfrac{y}{w} + t_y \end{cases}$$

Scaling with homogeneous

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$\begin{cases} x' = s_x x \\ y' = s_y y \\ w' = w \end{cases}$$

$$\begin{cases} \dfrac{x'}{w'} = s_x \dfrac{x}{w} \\ \dfrac{y'}{w'} = s_y \dfrac{y}{w} \end{cases}$$

Rotation with homogeneous

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$\begin{cases} x' = \cos\theta x - \sin\theta y \\ y' = \sin\theta x + \cos\theta y \\ w' = w \end{cases}$$

$$\begin{cases} \frac{x'}{w'} = \cos\theta \frac{x}{w} - \sin\theta \frac{y}{w} \\ \frac{y'}{w'} = \sin\theta \frac{x}{w} + \cos\theta \frac{y}{w} \end{cases}$$

Composition of transformations

- To compose transformations, multiply the matrices:
  - composition of a rotation and a translation:

$$\mathbf{M} = \mathbf{RT}$$

- all transformations can be expressed as matrices
  - even transformations that are not translations, rotations and scaling

Rotation around a point Q

- Rotation about a point Q:
  - translate Q to origin ($\mathbf{T}_Q$),
  - rotate about origin ($\mathbf{R}_\Theta$)
  - translate back to Q ($\mathbf{-}\ \mathrm{T}_Q$).

$$\longrightarrow \quad P' = (-\mathbf{T}_Q)\mathbf{R}_\Theta \mathbf{T}_Q\, P$$

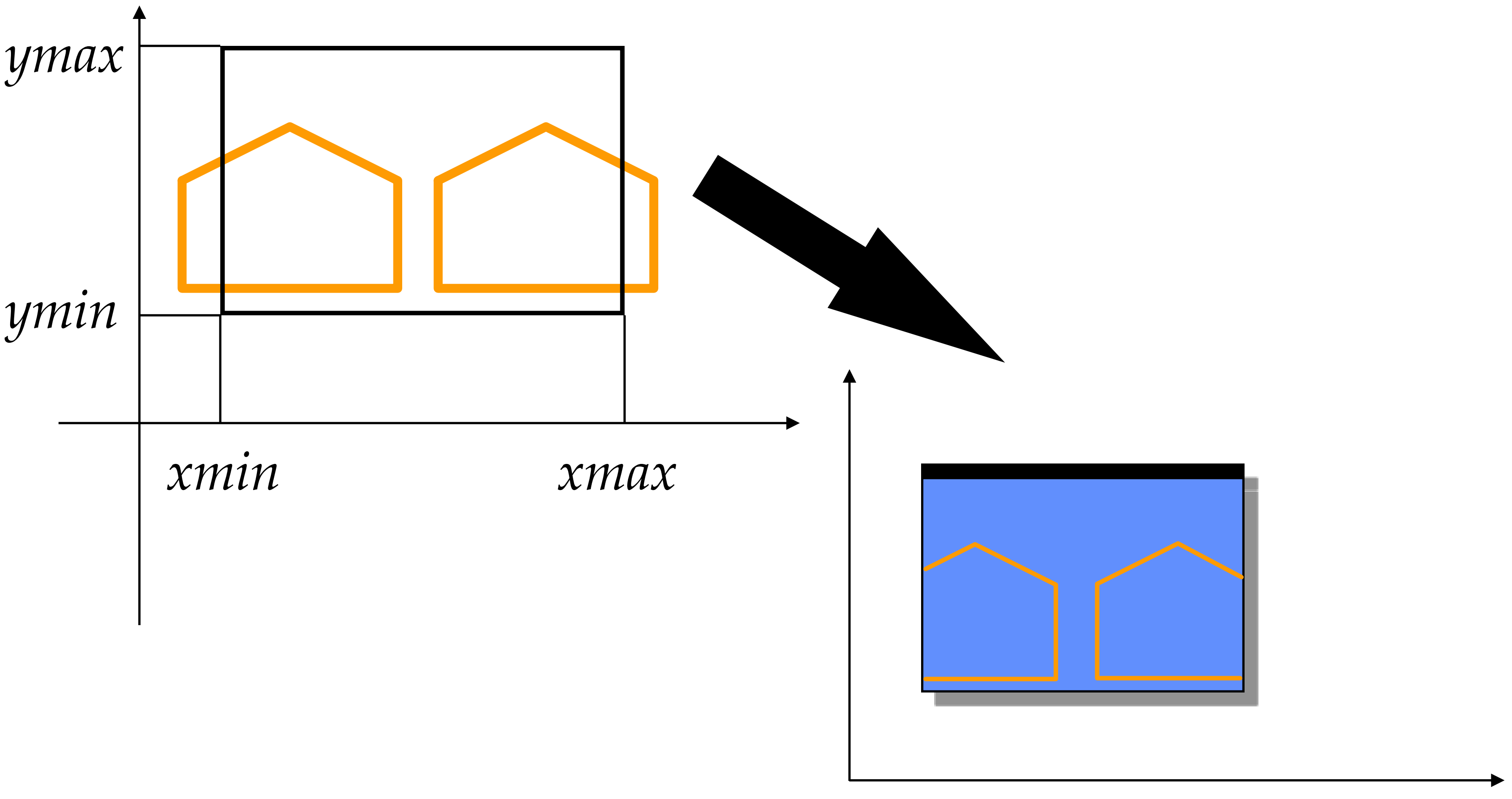
Beware!

- Matrix multiplication is *not* commutative
- The order of the transformations is vital
  - Rotation followed by translation is *very* different from translation followed by rotation
  - careful with the order of the matrices!
- Small commutativity:
  - rotation commute with rotation, translation with translation…

From World to Window

- Inside the application:
  - application model
  - coordinates related to the model
  - possibly floating point
- On the screen:
  - pixel coordinates
  - integer
  - restricted viewport: *umin/umax, vmin/vmax*

From Model to Viewport



From Model to Viewport

- Model is *(xmin,ymin)-(xmax,ymax)*
- Viewport is *(umin,vmin)-(umax,vmax)*
- Translate by *(-xmin,-ymin)*
- Scale by $\left(\frac{umax\text{-}umin}{xmax\text{-}xmin}, \frac{vmax\text{-}vmin}{ymax\text{-}ymin}\right)$
- Translate by *(umin,vmin)*

M = T'ST

From Model to Viewport

Pixel Coordinates

$$\begin{bmatrix} u \\ v \\ w' \end{bmatrix} = \mathbf{M} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Model Coordinates

Mouse position: inverse problem

- Mouse click: coordinates in pixels
- We want the equivalent in World Coord
  - because the user has selected an object
  - to draw something
  - for interaction
- How can we convert from window coordinates to model coordinates?

Mouse click: inverse problem

- Simply inverse the matrix:

$$M^{-1} = (T'ST)^{-1}$$

Model Coordinates

Pixels coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = M^{-1} \begin{bmatrix} u \\ v \\ w' \end{bmatrix}$$

2D transformations: conclusion

- Simple, consistent matrix notation
  - using homogeneous coordinates
  - all transformations expressed as matrices
- Used by the window system:
  - for conversion from model to window
  - for conversion from window to model
- Used by the application:
  - for modeling transformations