# 2D Transformations



University of Victoria | Computer Science

# 2D Linear Transformations

- Each 2D linear map can be represented by a unique 2×2 matrix

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

- Concatenation of mappings corresponds to multiplication of matrices

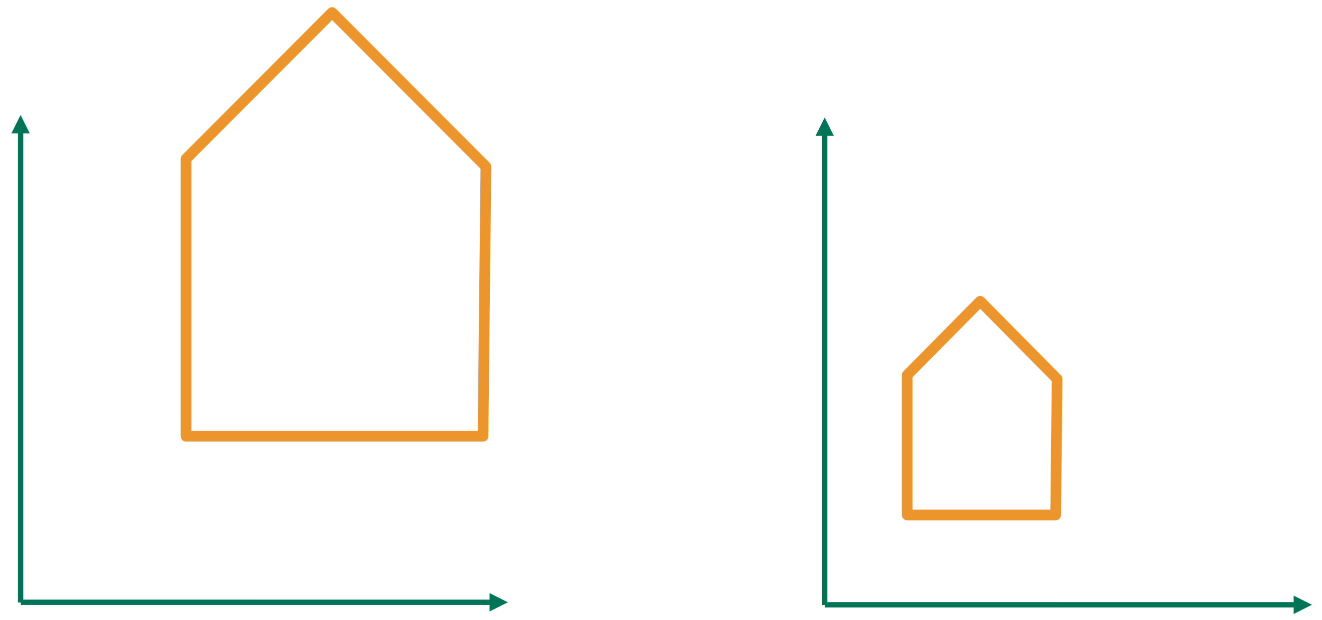
$$L_2(L_1(\mathbf{x})) = \mathbf{L}_2 \, \mathbf{L}_1 \, \mathbf{x}$$

```
L2 * L1 * x;
```

- Linear transformations are very common in computer graphics!

# 2D Scaling

- Scaling $\begin{pmatrix} x' \\ y' \end{pmatrix} = \underbrace{\begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix}}_{\mathbf{S}(s_x, s_y)} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$

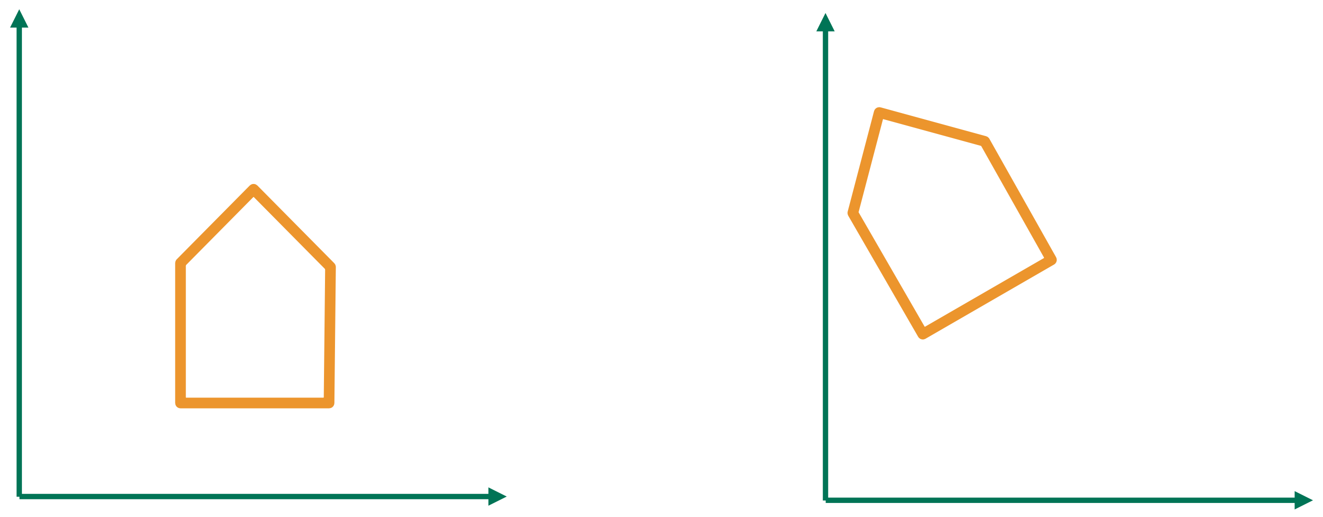


$\mathbf{S}(0.5, 0.5)$

Image Copyright: Mark Pauly

# 2D Rotation

- Rotation $\begin{pmatrix} x' \\ y' \end{pmatrix} = \underbrace{\begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}}_{\mathbf{R}(\alpha)} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$



$\mathbf{R}(20^\circ)$

Special case: $\mathbf{R}(90) = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$

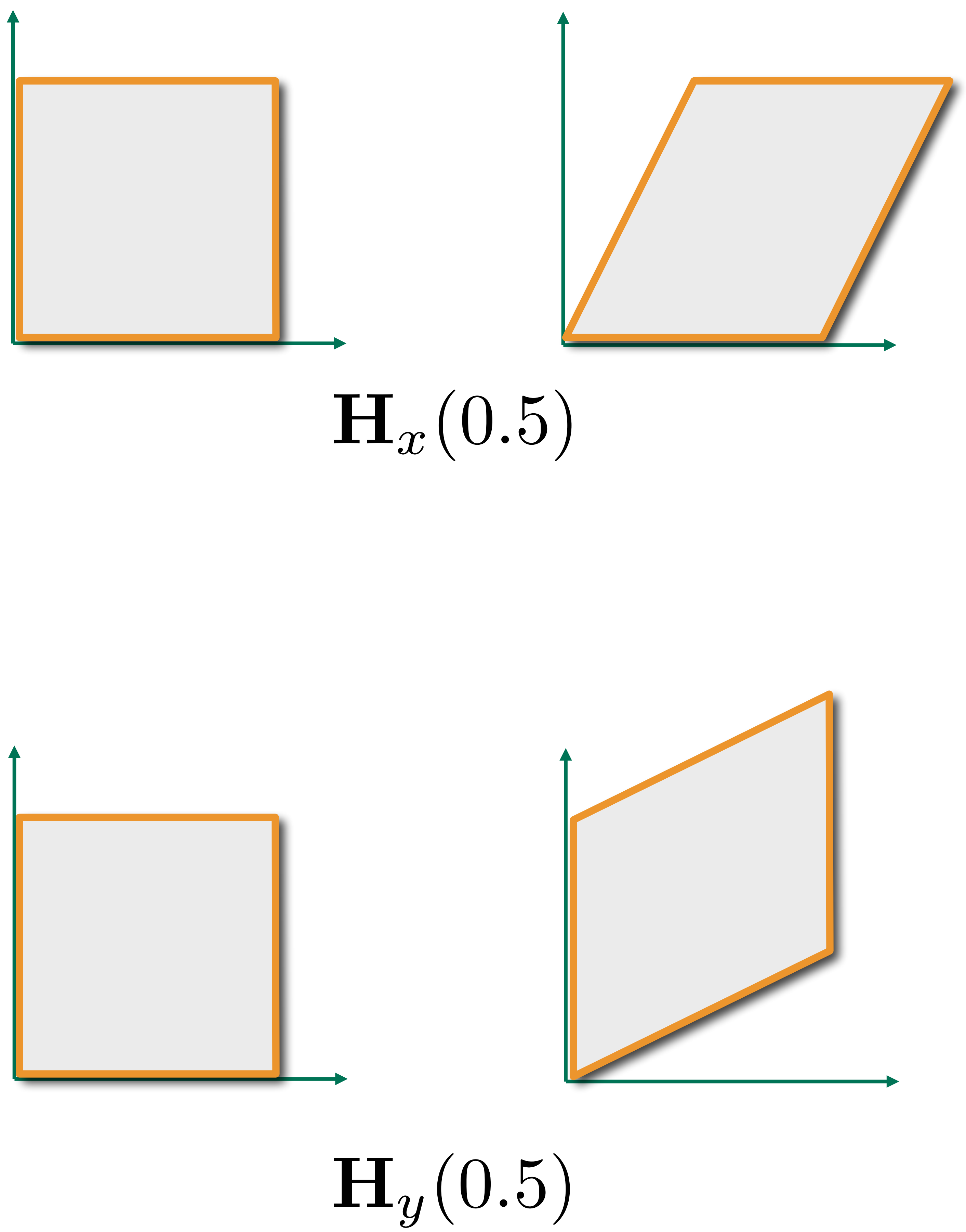
# 2D Shearing

- Shear along x-axis

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix}}_{\mathbf{H}_x(a)} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

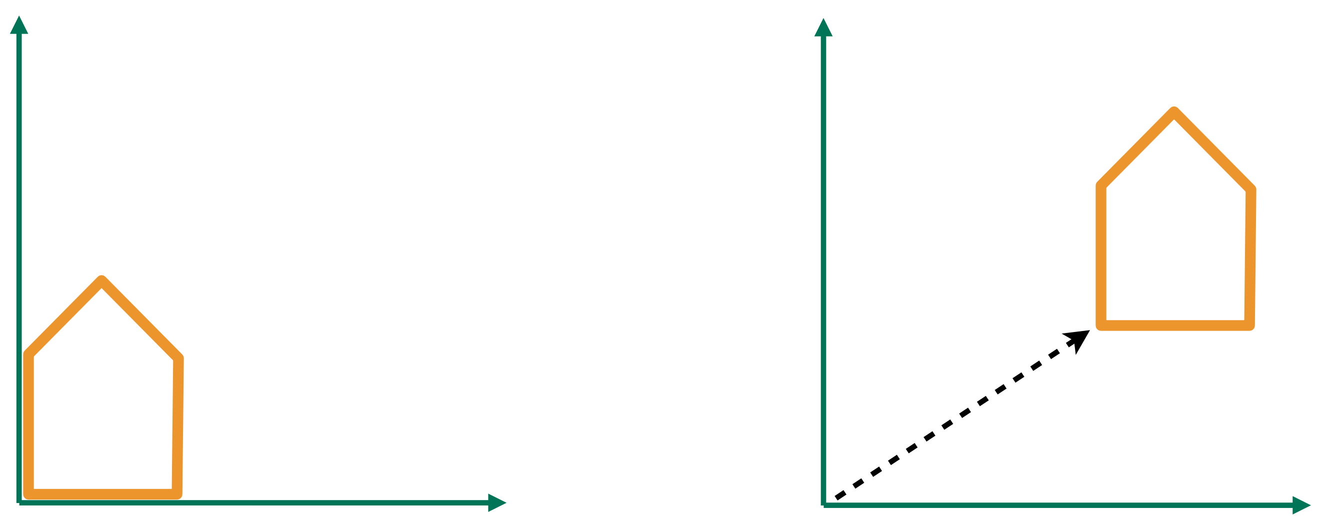
- Shear along y-axis

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ b & 1 \end{pmatrix}}_{\mathbf{H}_y(b)} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$



# 2D Translation

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



- Matrix representation? $\begin{pmatrix} x' \\ y' \end{pmatrix} = \mathbf{T}(t_x, t_y) \cdot \begin{pmatrix} x \\ y \end{pmatrix}$



Image Copyright: Mark Pauly

# Affine Transformations

- Translation is not linear, but it is *affine*
  - Origin is no longer a fixed point
- Affine map = linear map + translation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} = \mathbf{Lx} + \mathbf{t}$$

- Is there a matrix representation for affine transformations?
  - We would like to handle all transformations in a unified framework -> simpler to code and easier to optimize!

# Homogenous Coordinates

- Add a third coordinate (*w-coordinate*)
  - 2D point = $(x, y, 1)^T$
  - 2D vector = $(x, y, 0)^T$

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

- Matrix representation of translations

# Homogenous Coordinates

- Valid operation if the resulting *w*-coordinate is 1 or 0
  - vector + vector = vector
  - point - point = vector
  - point + vector = point
  - point + point = ???

# Homogenous Coordinates

- Geometric interpretation: 2 hyperplanes in $\mathbf{R}^3$

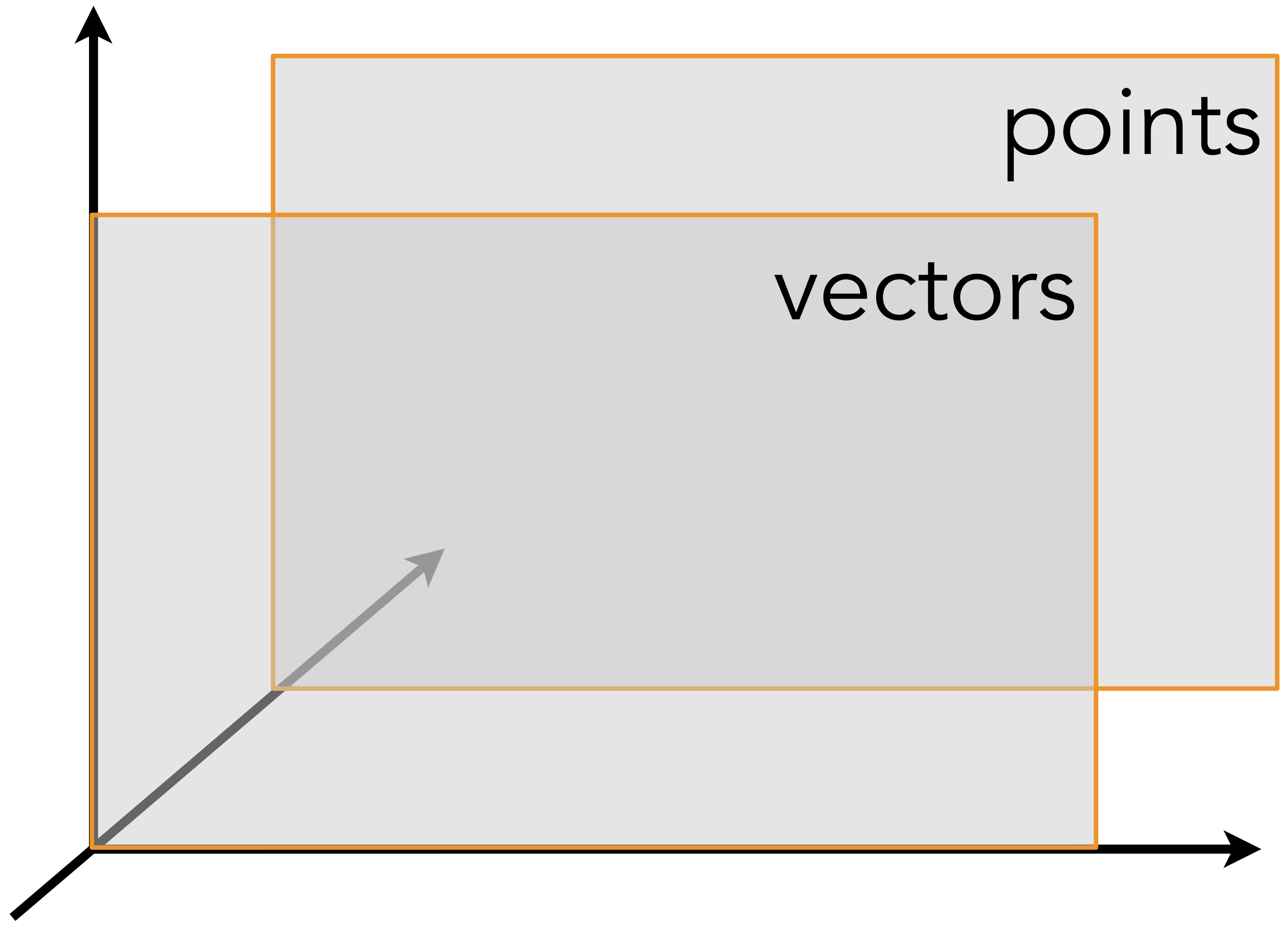



Image Copyright: Mark Pauly

# Affine Transformations

- Affine map = linear map + translation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

- Using homogenous coordinates:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

# 2D Transformations

- Scale

$$\mathbf{S}(s_x, s_y) = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Rotation

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

- Translation

$$\mathbf{T}(t_x, t_y) = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

# Concatenation of Transformations

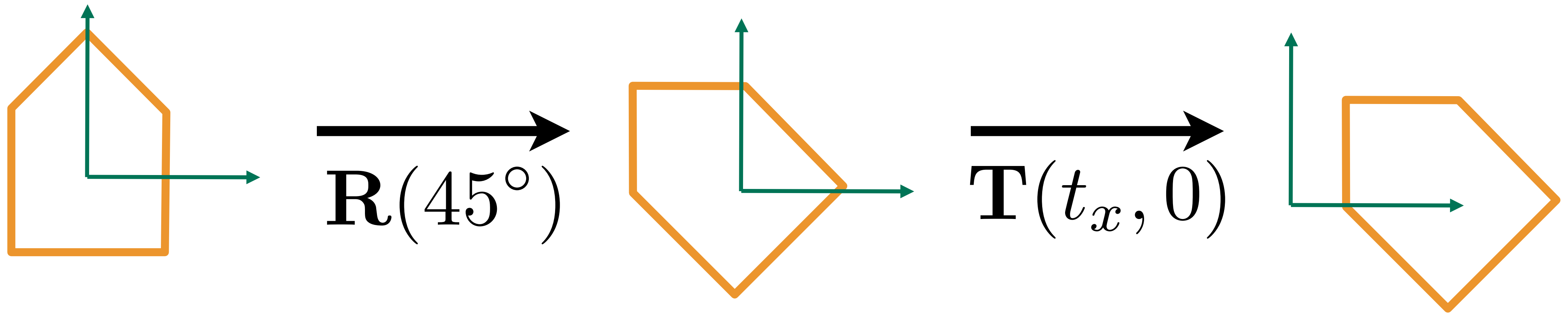
- Sequence of affine maps $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{A}_3$, ...
  - Concatenation by matrix multiplication

$$A_n(\dots A_2(A_1(\mathbf{x}))) \;=\; \mathbf{A}_n \cdots \mathbf{A}_2 \cdot \mathbf{A}_1 \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

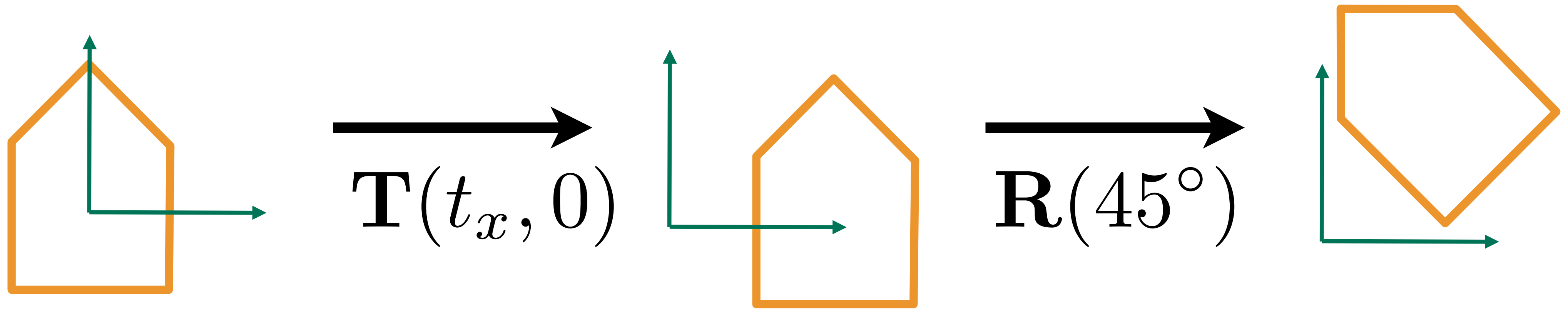
  - Very important for performance!
  - Matrix multiplication not commutative, ordering is important!

# Rotation and Translation

- Matrix multiplication is not commutative!
  - First rotation, then translation

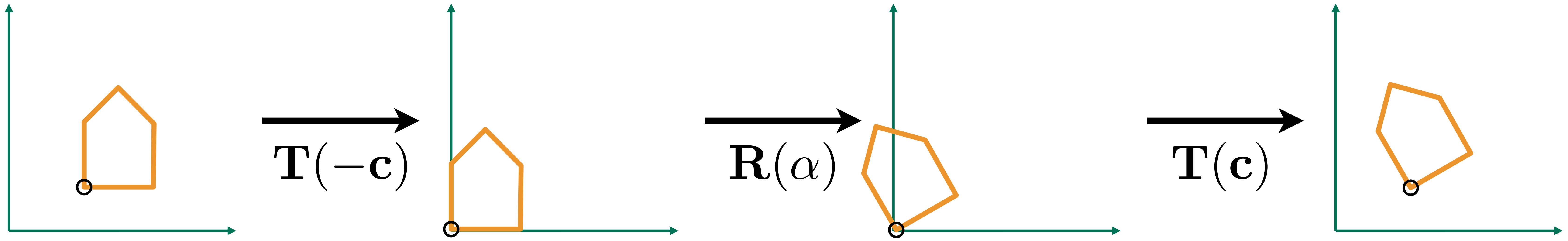



  - First translation, then rotation



# 2D Rotation

- How to rotate around a given point **c**?
  1. Translate **c** to origin
  2. Rotate
  3. Translate back




- Matrix representation?

$$\mathbf{T}(\mathbf{c}) \cdot \mathbf{R}(\alpha) \cdot \mathbf{T}(-\mathbf{c})$$

# Transform Object or Camera?

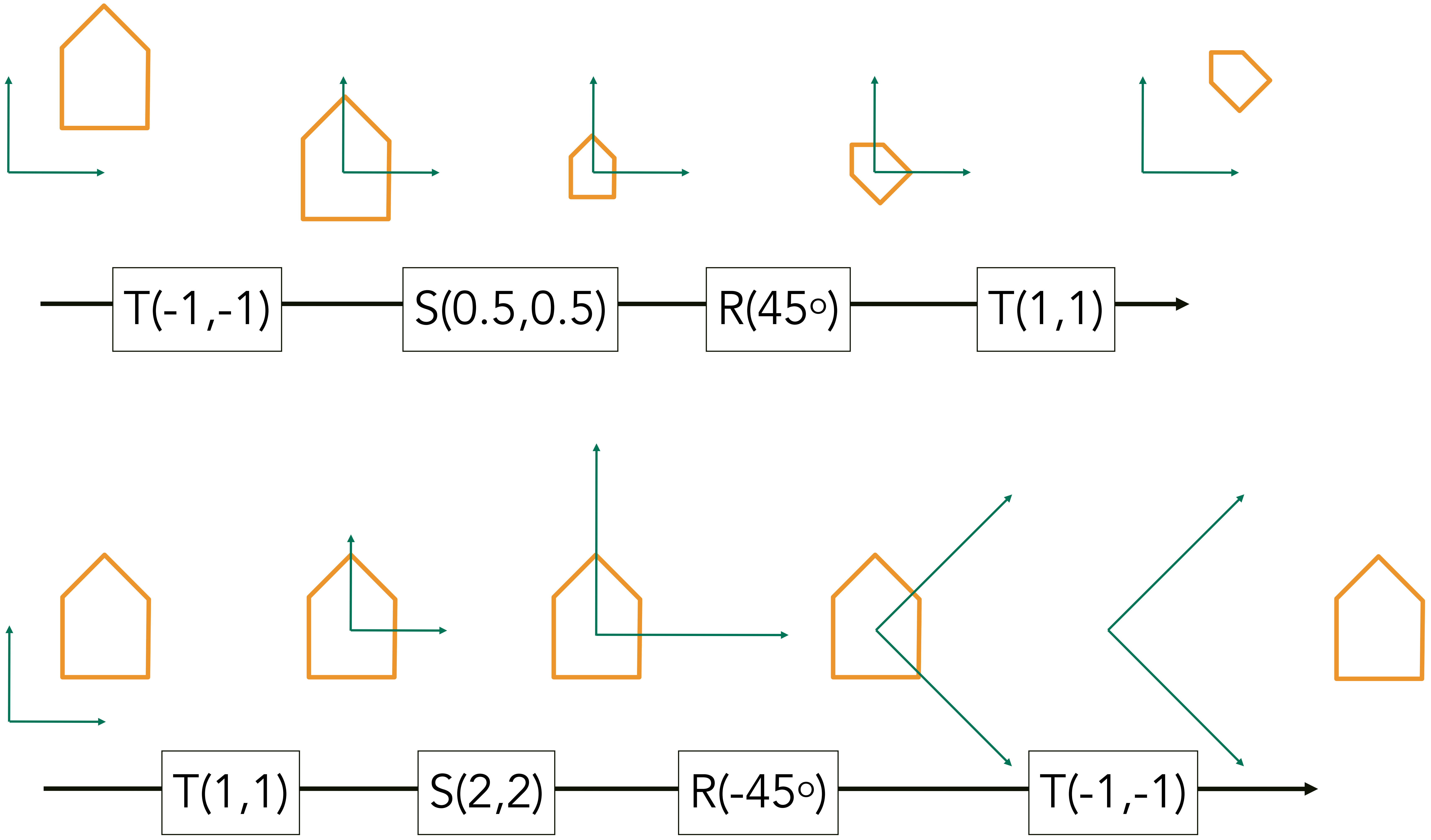
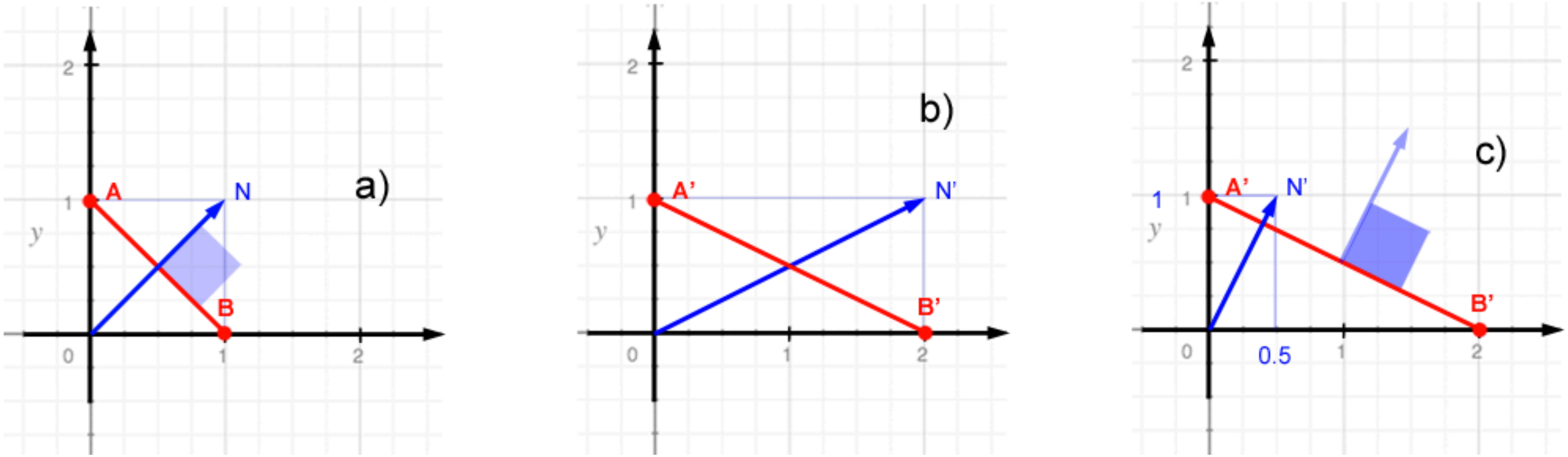



Image Copyright: Mark Pauly

# A note on transforming normals

- If you transform a point **v** with a matrix M: **v′** = **Mv** …
- the transformed normal **n′** at the point **v** is **n′ = M$^{-T}$ n**




http://www.scratchapixel.com/lessons/mathematics-physics-for-computer-graphics/geometry/transforming-normals

# References

**Fundamentals of Computer Graphics, Fourth Edition**
4th Edition **by Steve Marschner, Peter Shirley**

Chapter 6