Lecture 4:

Transforms

Computer Graphics
CMU 15-462/15-662, Spring 2016

Notes

- **The web page is up:**

  http://www.cs.cmu.edu/~15462/

- **Assignment 1 is out:**

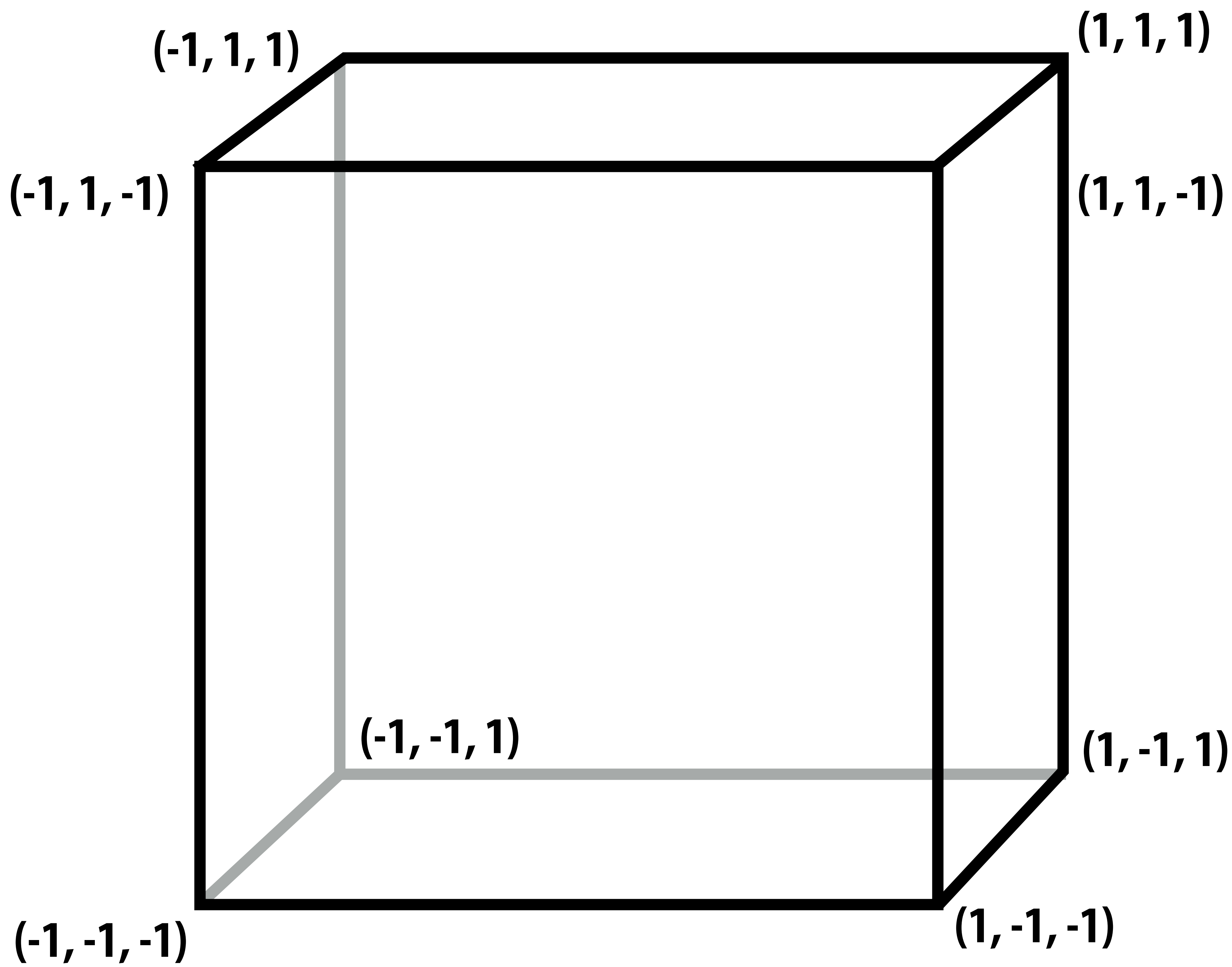
  http://462cmu.github.io/asst1_drawsvg/

- **If you are not on piazza, please add yourself or talk to me or a TA!**

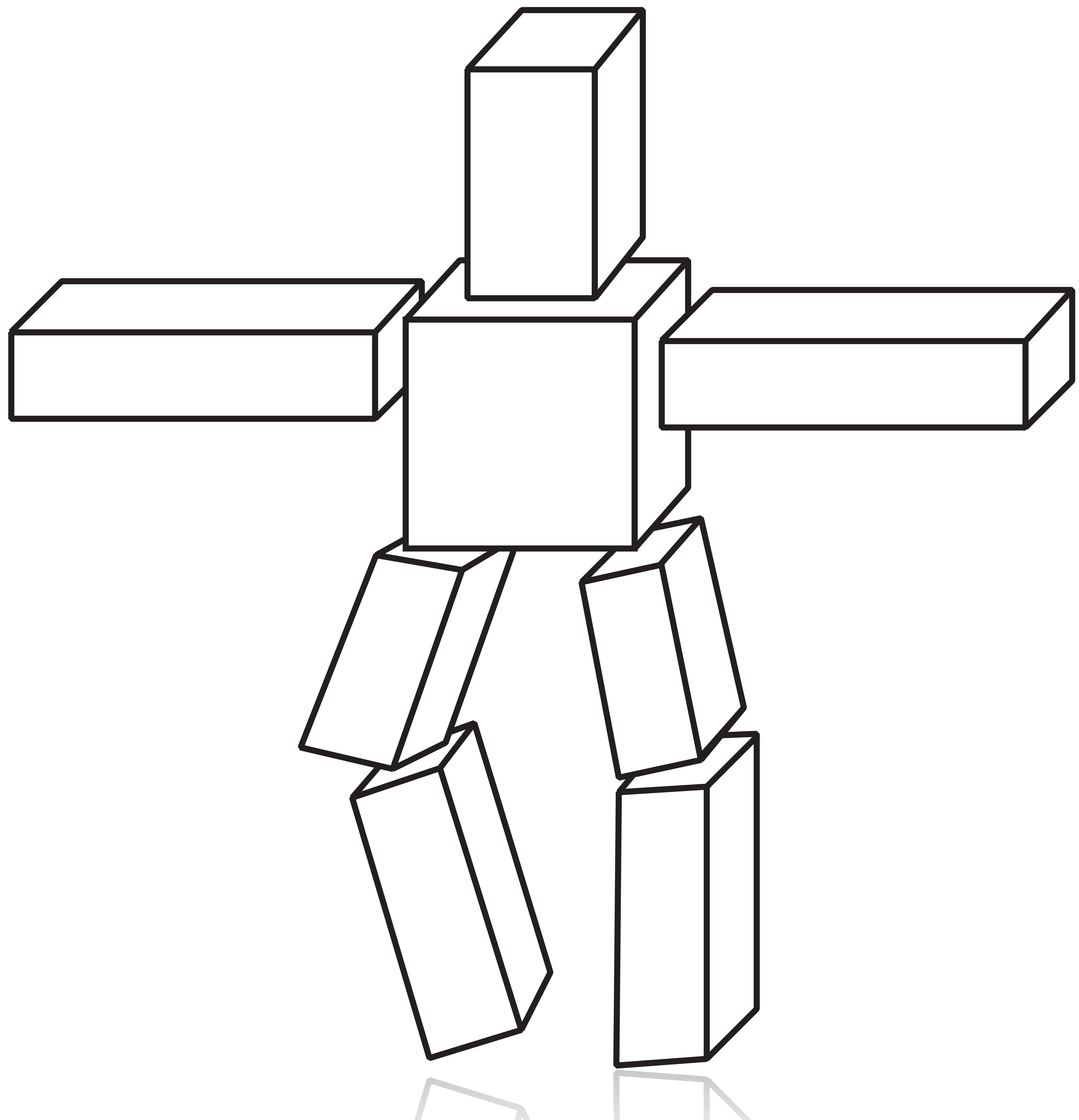
  https://piazza.com/class/ijeillqemgr2l2

Review from last time

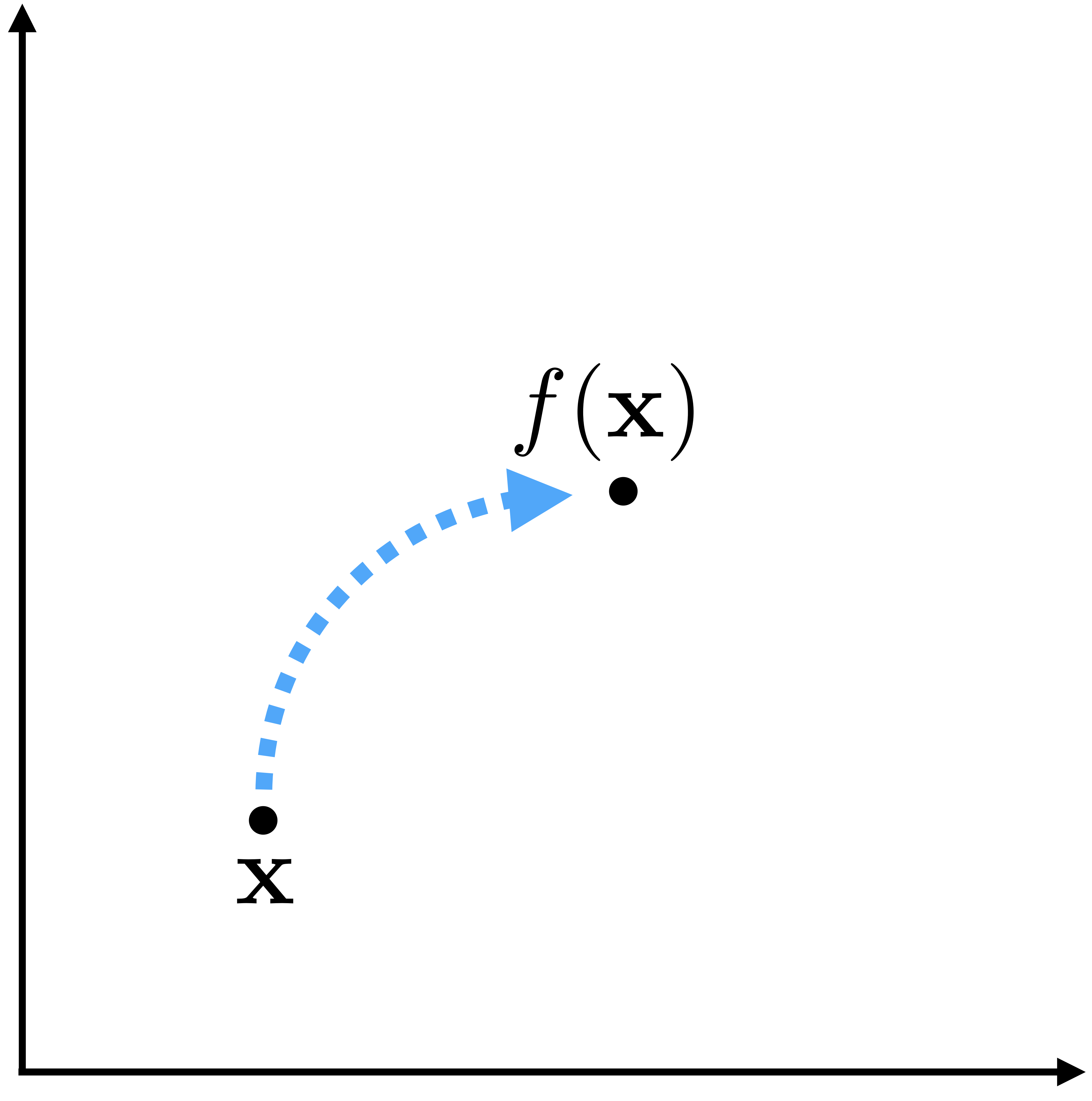
Cube



Cube man



f transforms $\mathbf{x}$ to $f(\mathbf{x})$



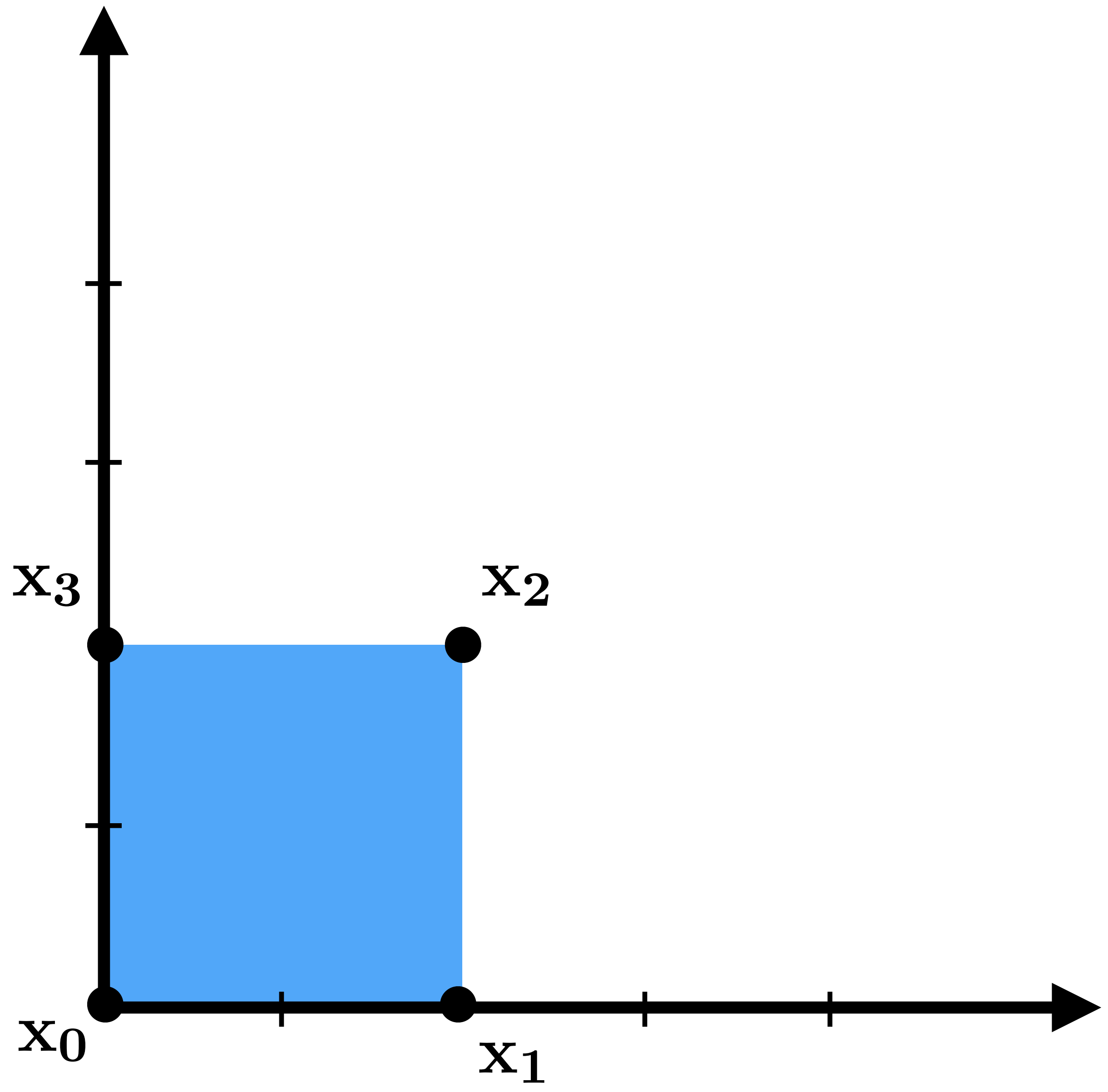
Linear transforms

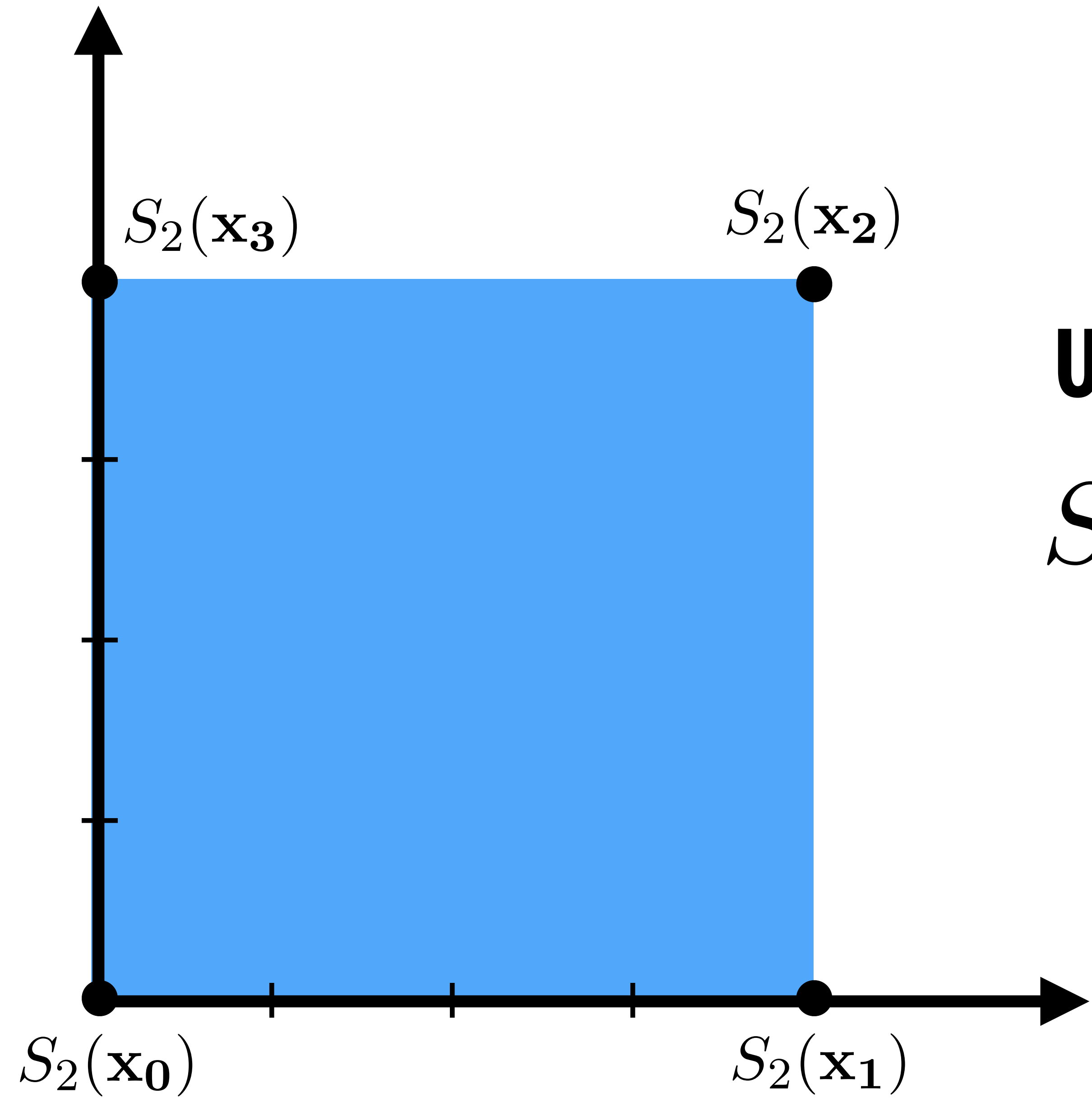
Transform f is linear if and only if:

$$f(\mathbf{x} + \mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$$

$$f(a\mathbf{x}) = af(\mathbf{x})$$

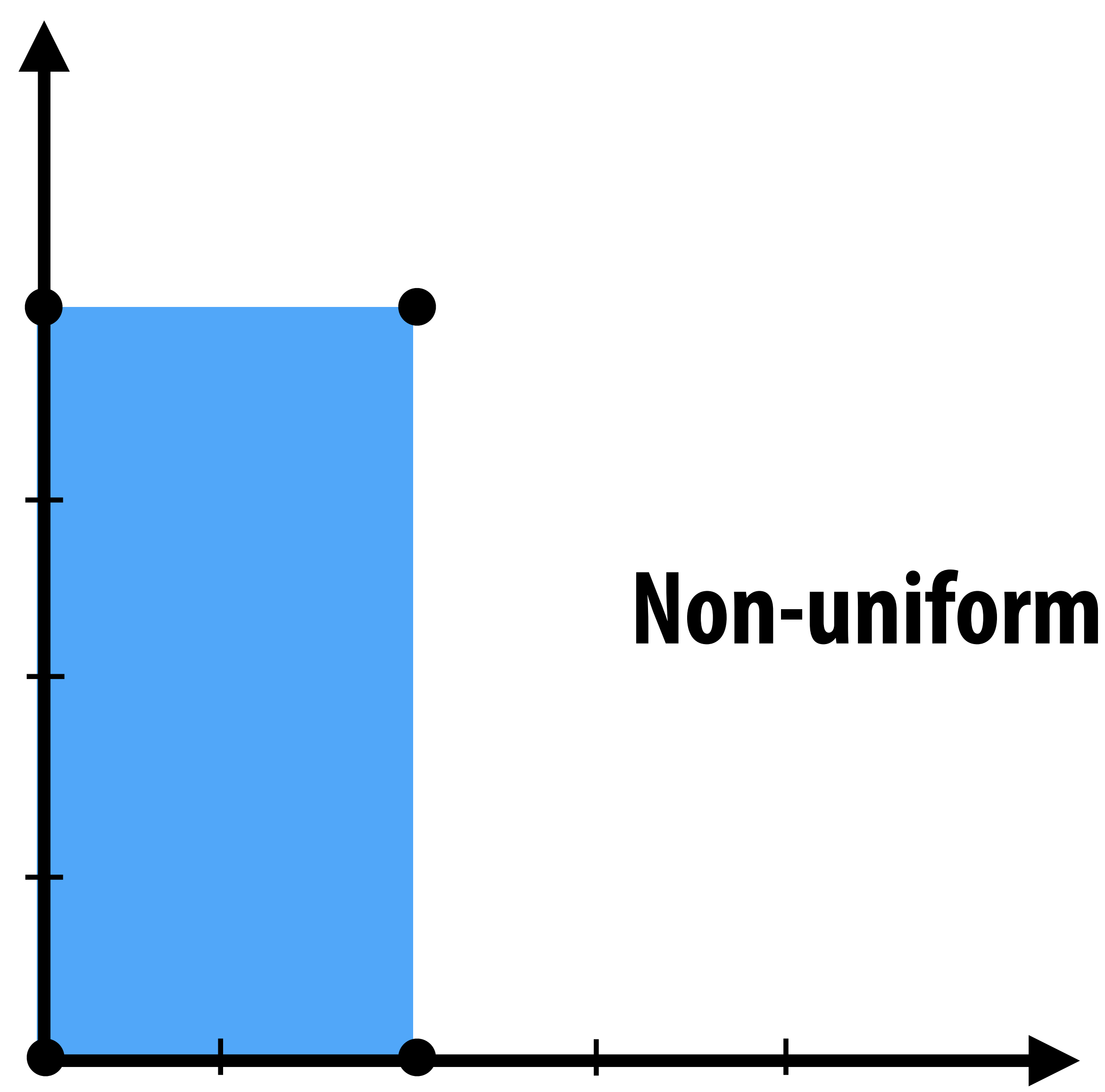
Scale





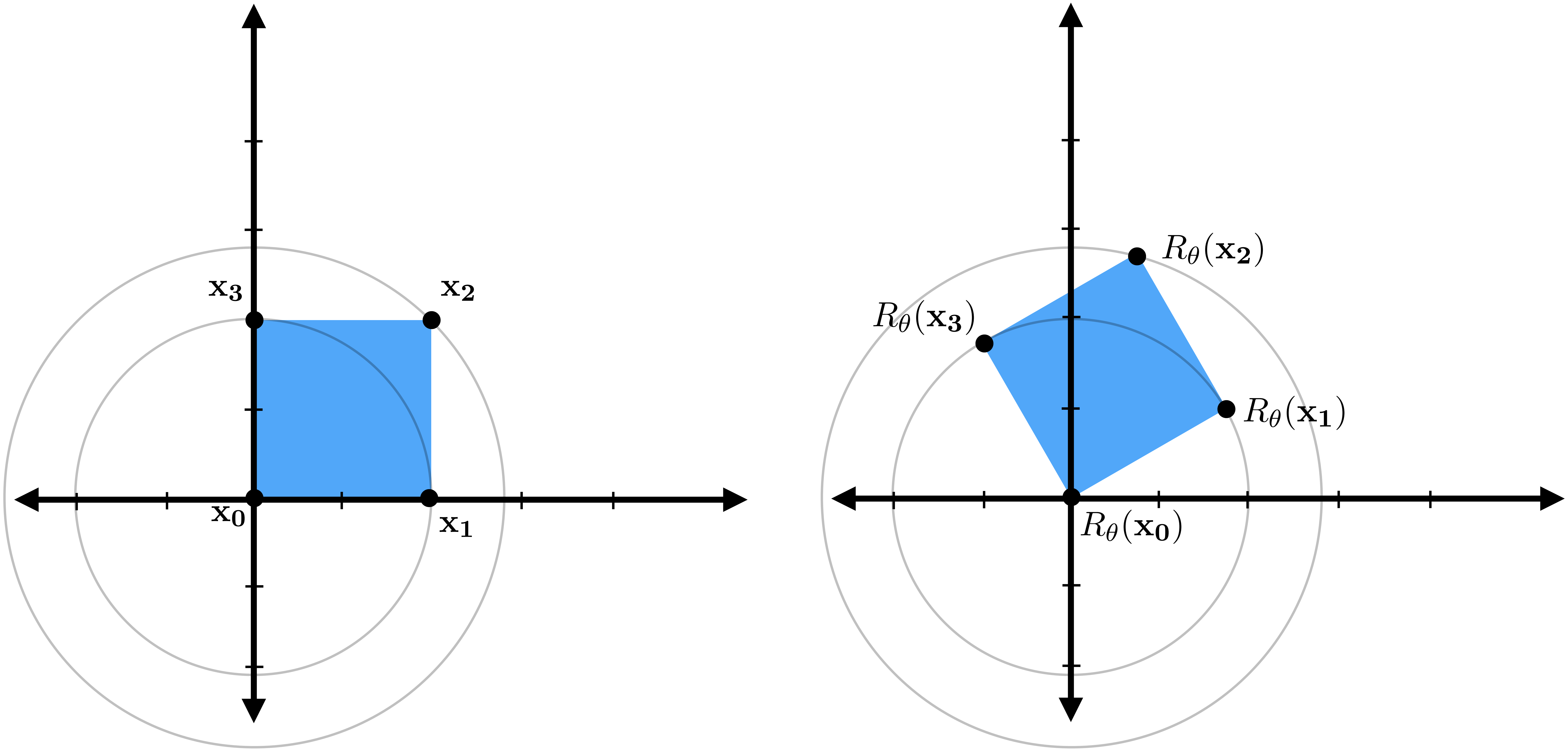
Uniform scale:

$$S_a(\mathbf{x}) = a\mathbf{x}$$



Non-uniform scale??

Rotation as Circular Motion

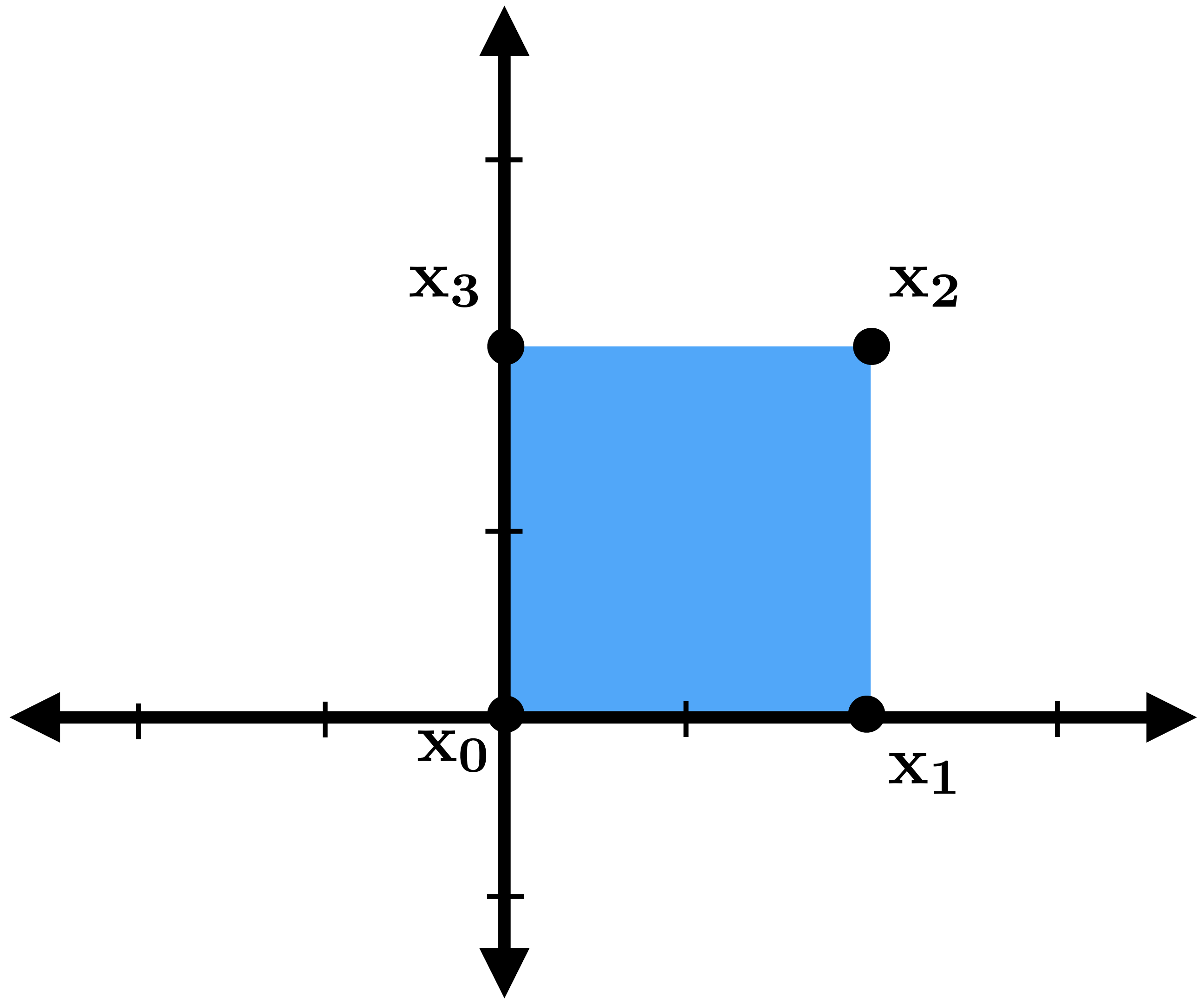


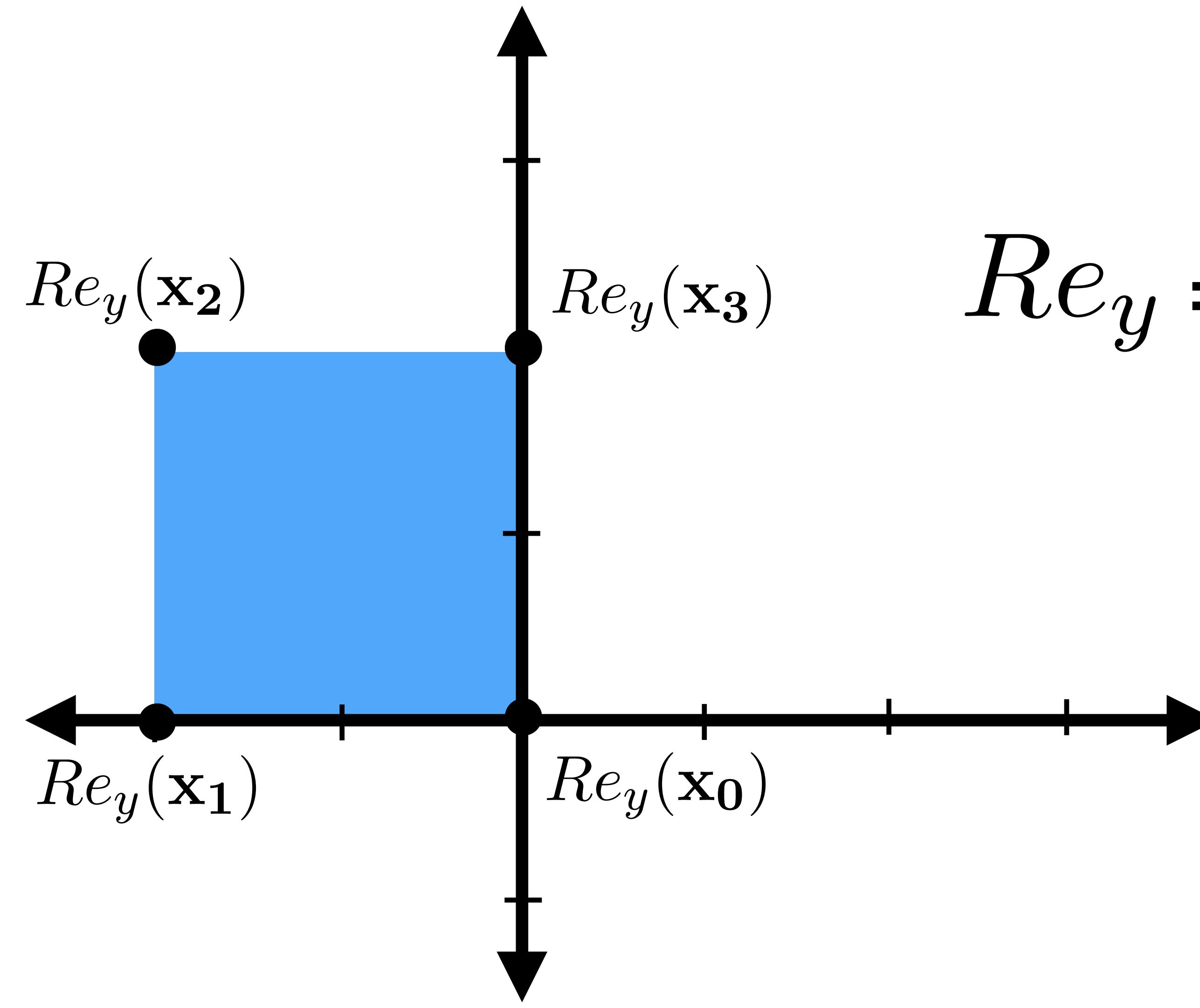
R_θ **= rotate counter-clockwise by** θ

As angle changes, points move along circular trajectories.

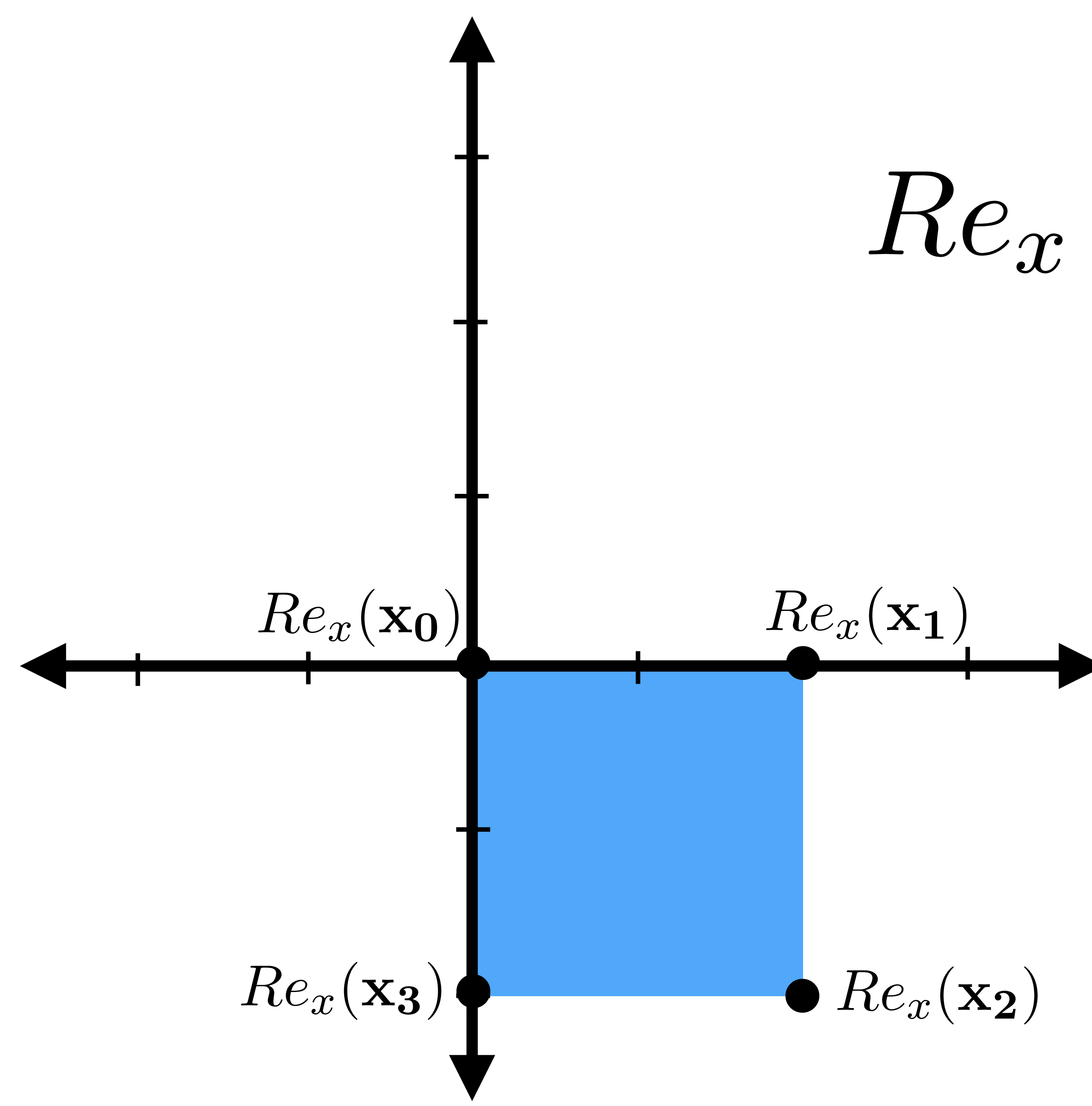
Hence, rotations preserve length of vectors: $|R_\theta(\mathbf{x})| = |\mathbf{x}|$

Reflection



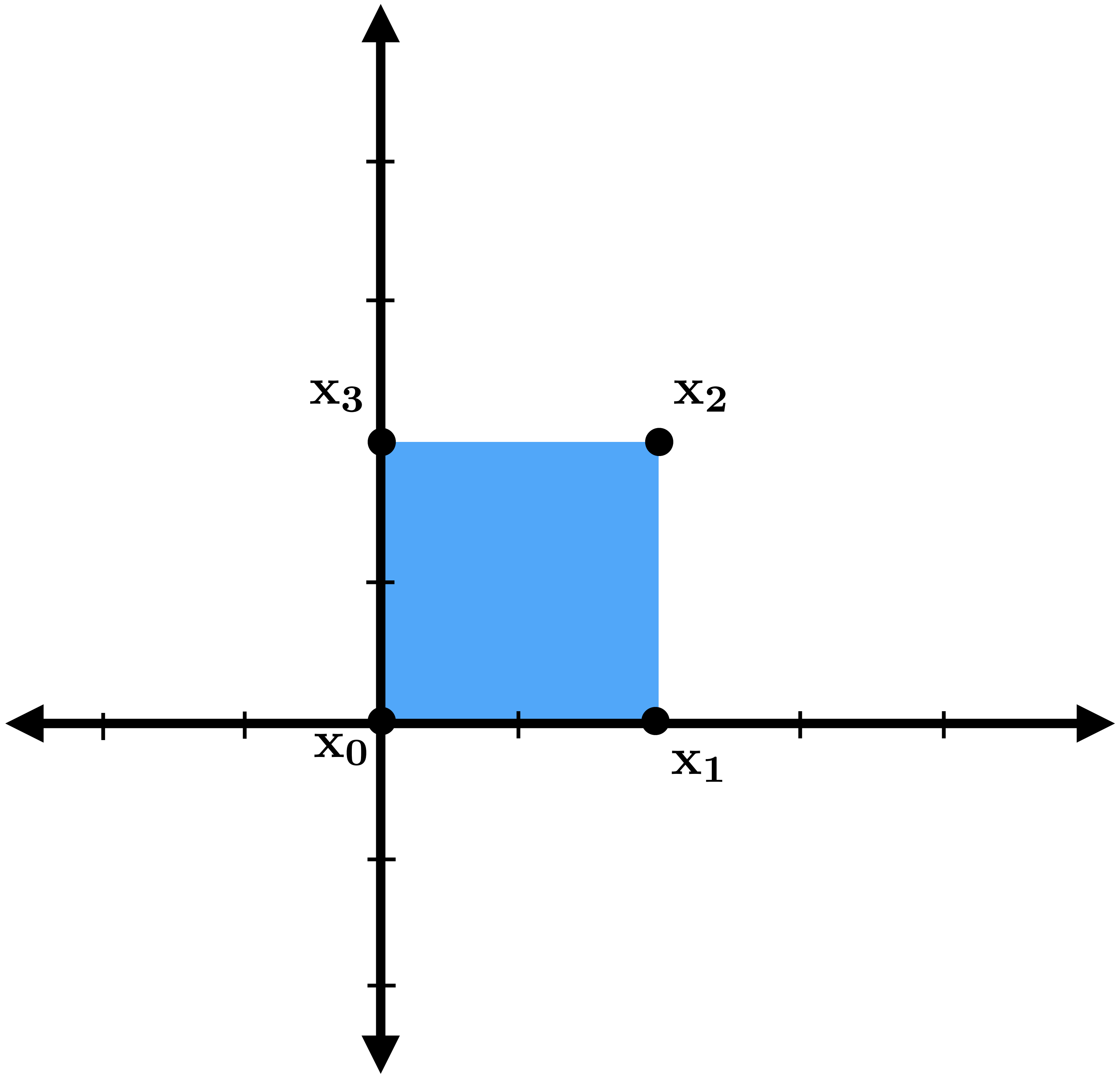


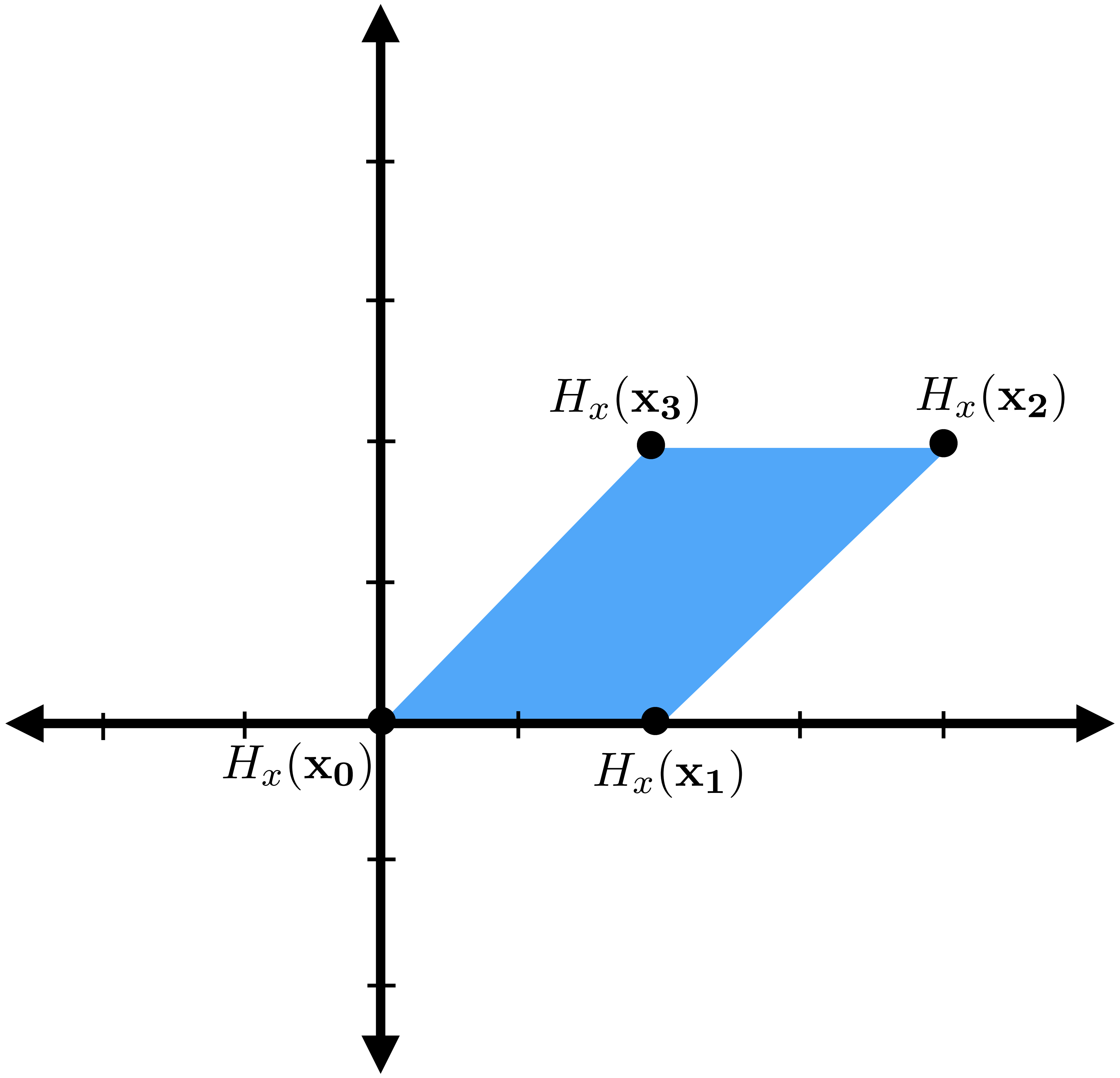
Re_y = **reflection about *y***



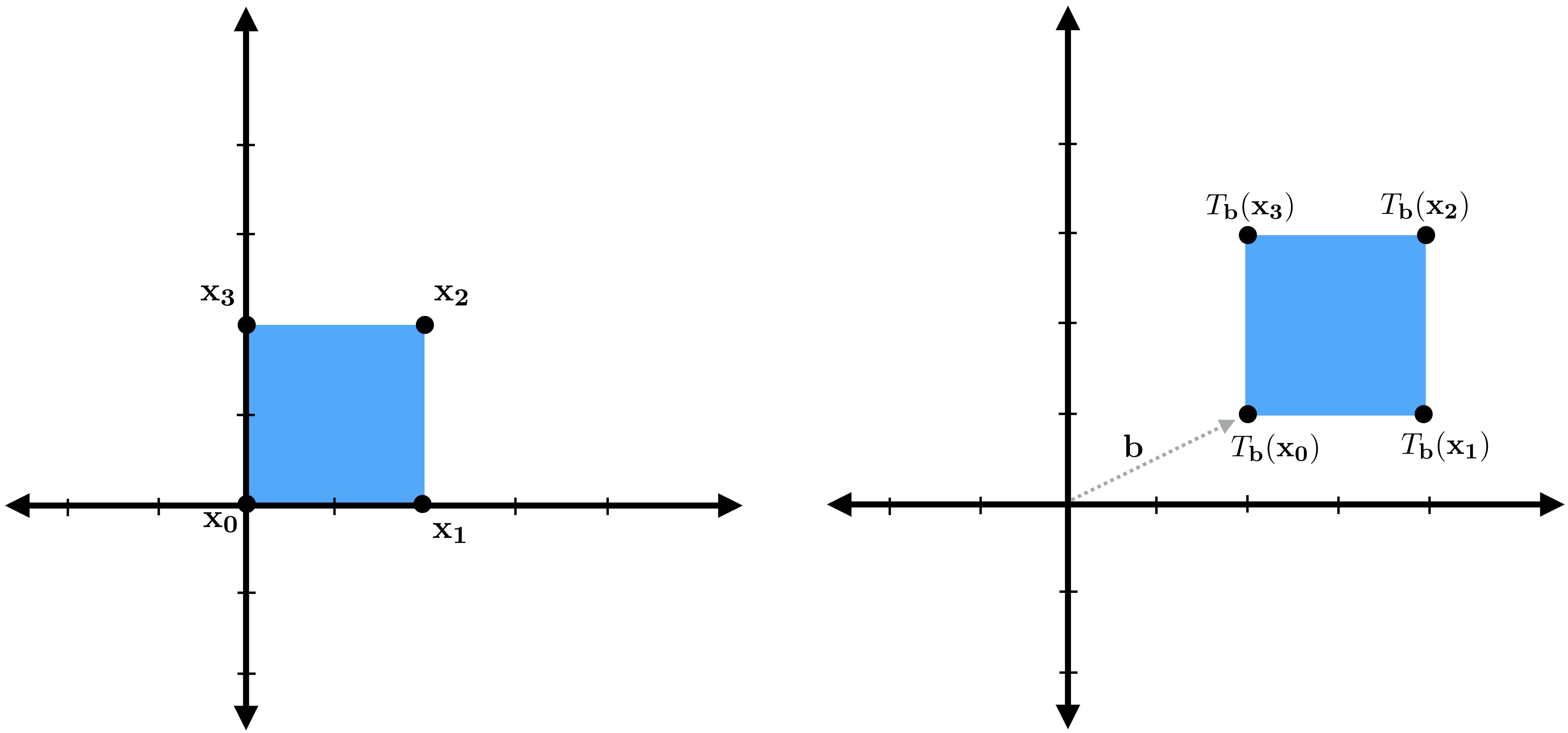
Re_x = **reflection about *x***

Shear (in x direction)





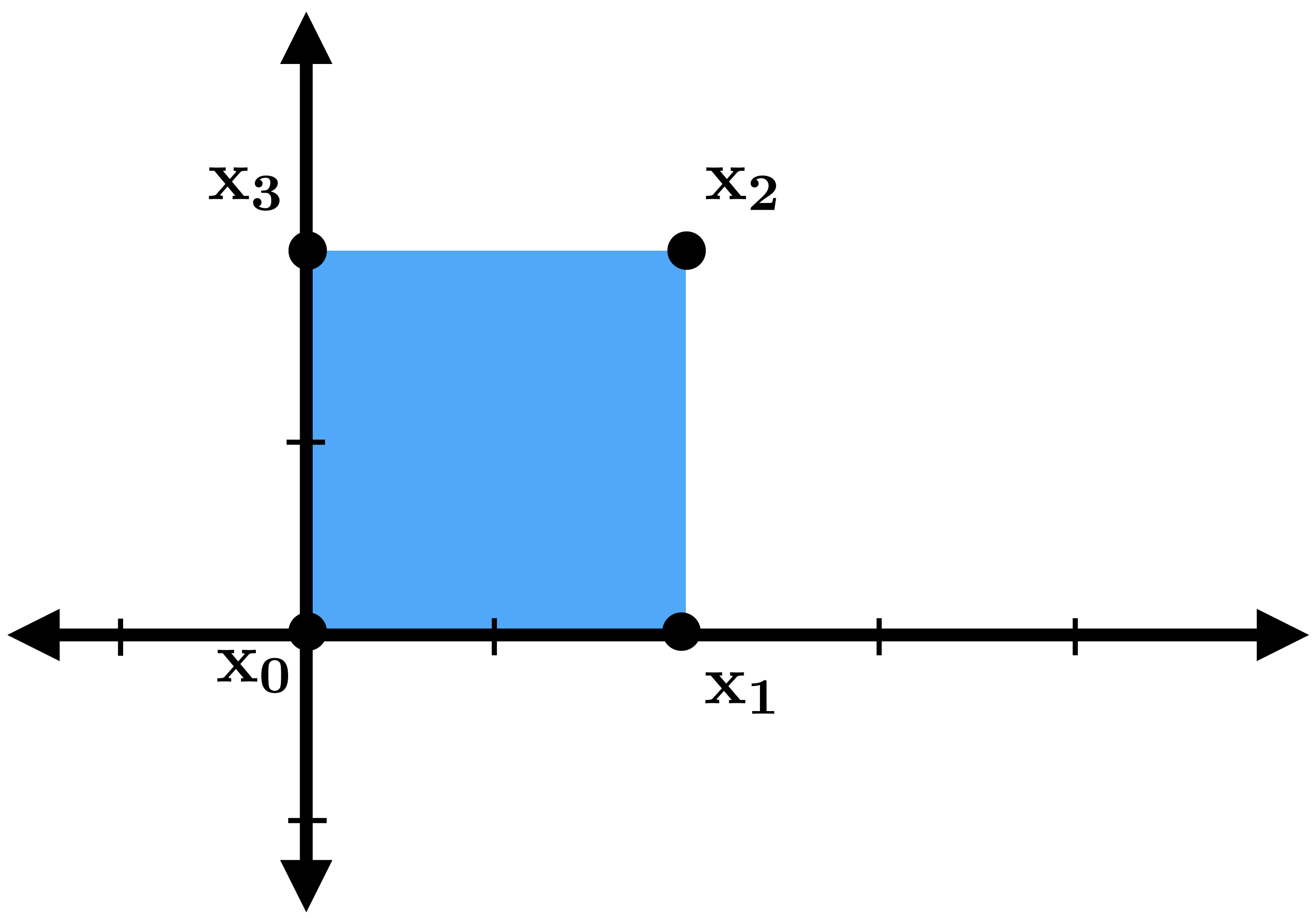
Translation

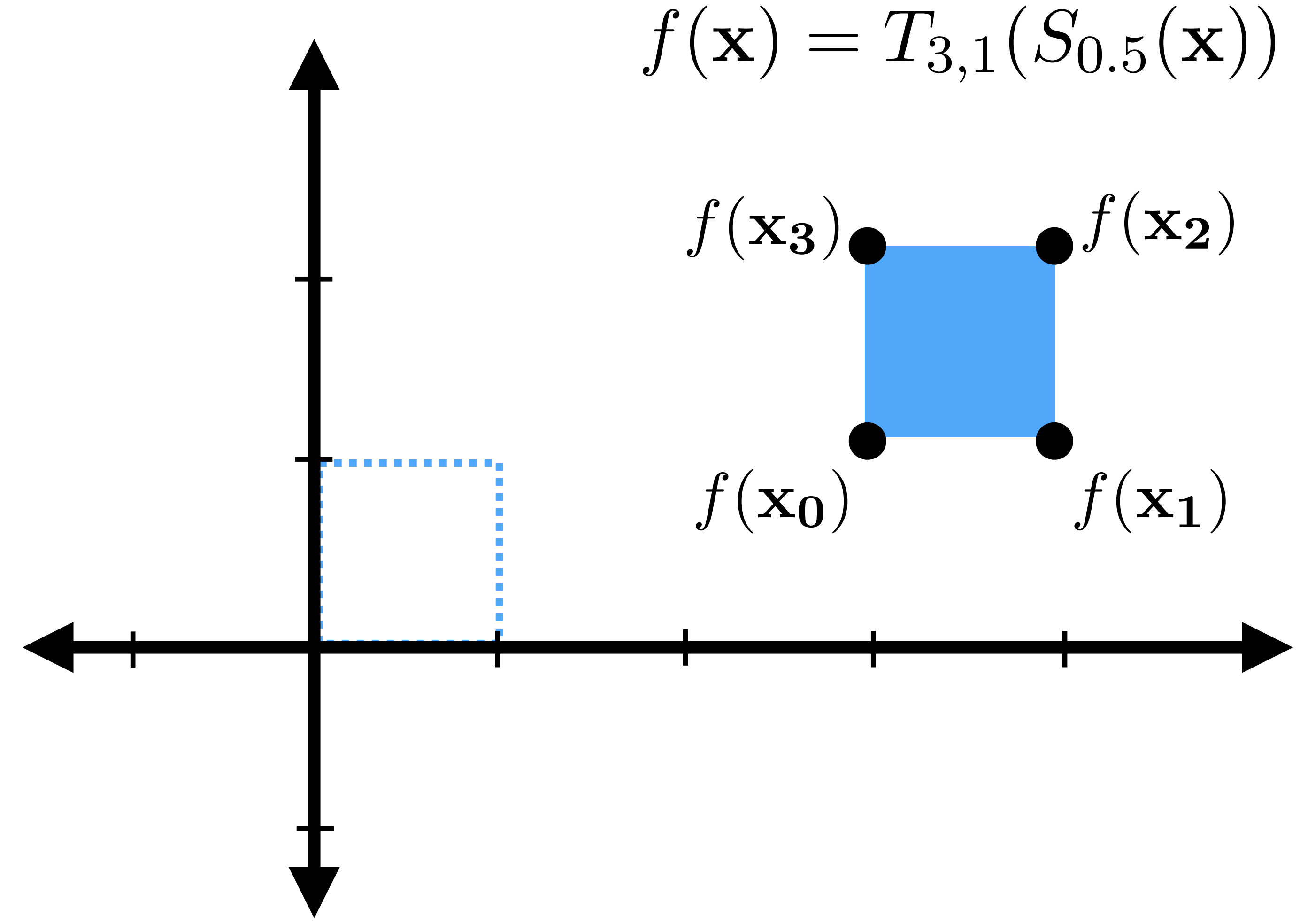


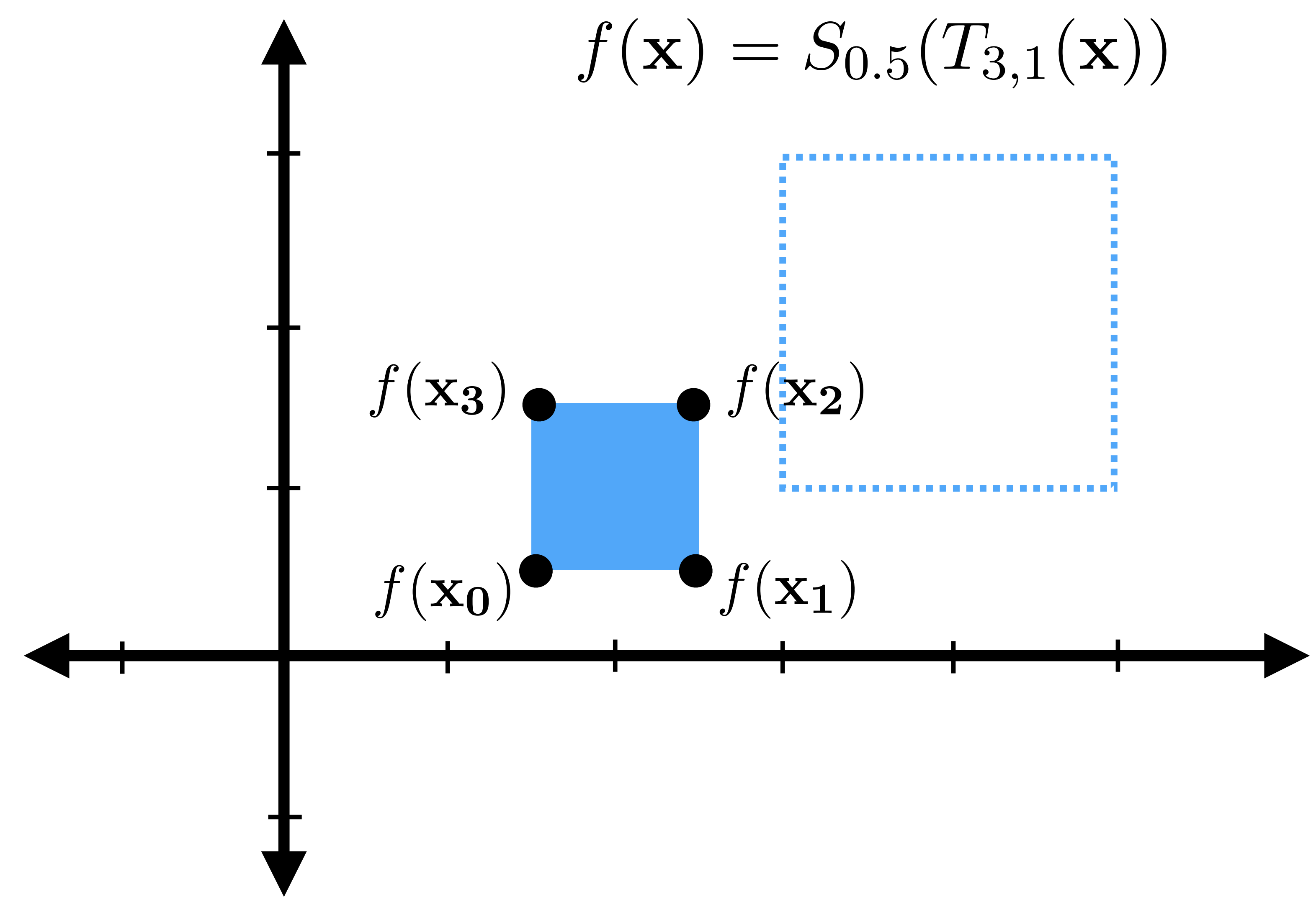
$T_\mathbf{b}(\mathbf{x})$ **= translate by** b

$$T_\mathbf{b}(\mathbf{x}) = \mathbf{x} + \mathbf{b}$$

Compose basic transforms to construct more complex transforms



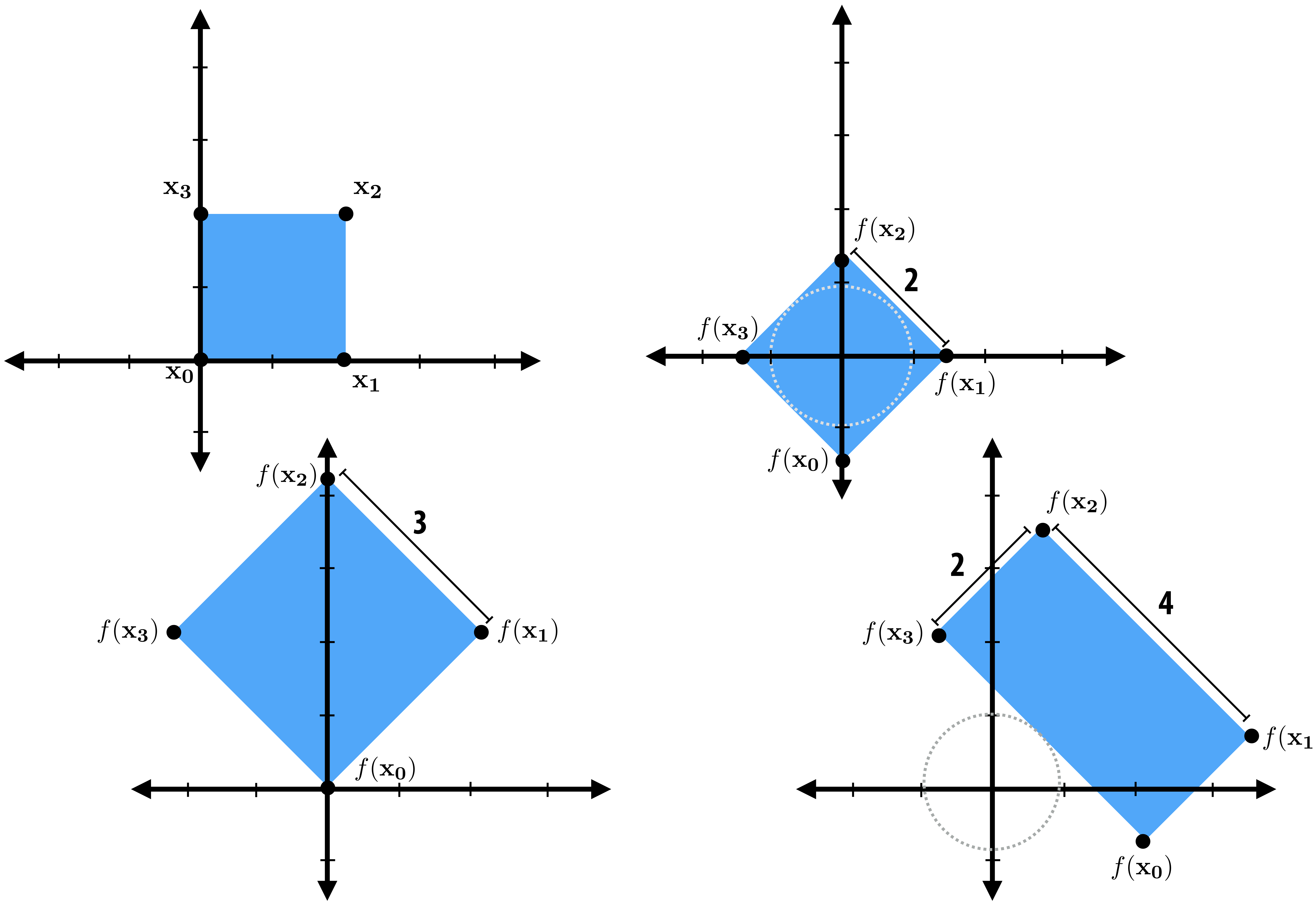




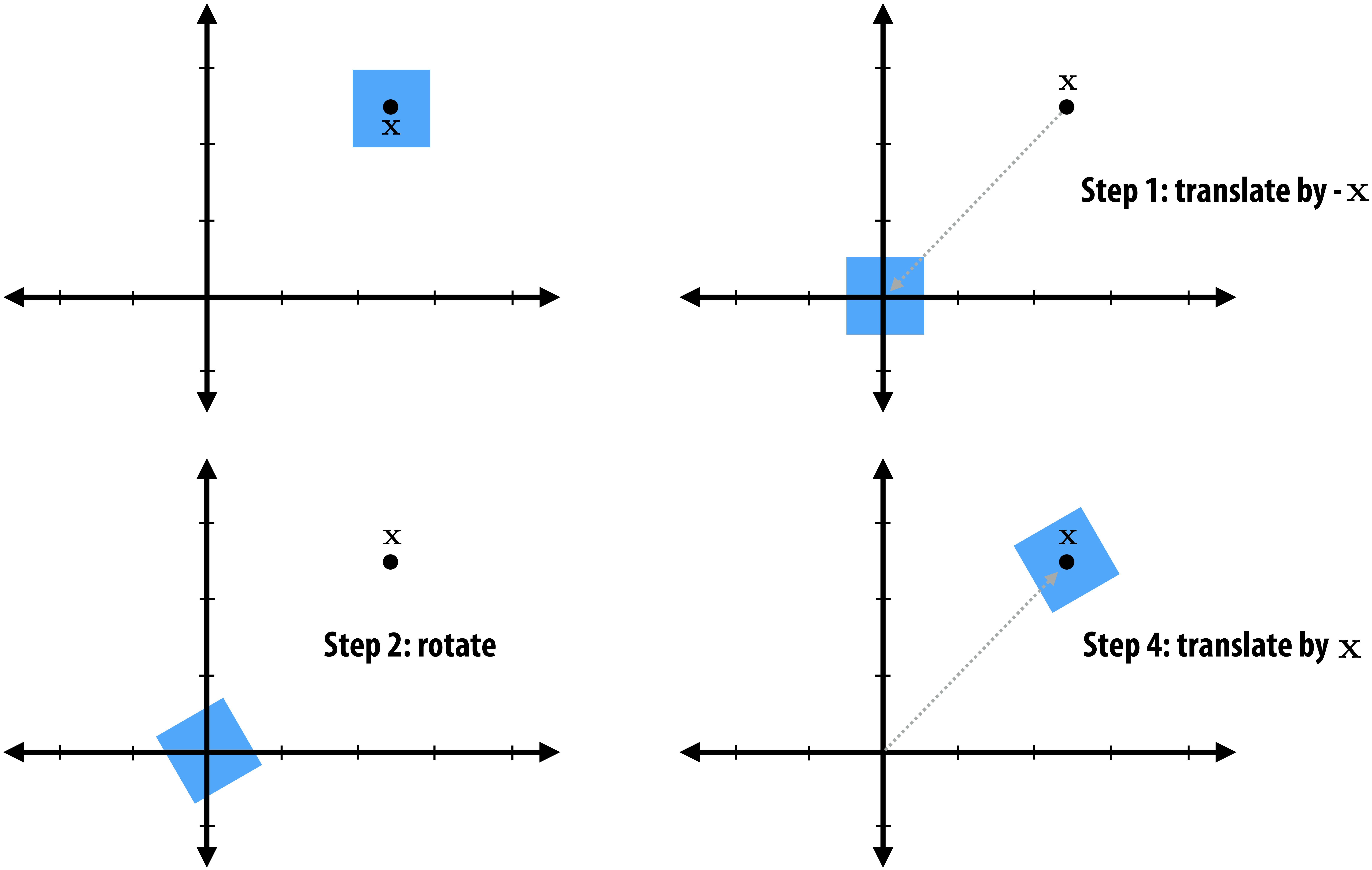
Note: order of composition matters

Top-right: scale, then translate
Bottom-right: translate, then scale

How would you perform these transformations?



Common pattern: rotation about point x



Summary of basic transforms

Linear:

$$f(\mathbf{x}+\mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$$
$$f(a\mathbf{x}) = af(\mathbf{x})$$

Scale
Rotation
Reflection
Shear

Not linear:

Translation

Affine:

Composition of linear transform + translation (all examples on previous two slides)

$$f(\mathbf{x}) = g(\mathbf{x}) + \mathbf{b}$$

Not affine: perspective projection (will discuss later)

Euclidean: (Isometries)

Preserve distance between points (preserves length)

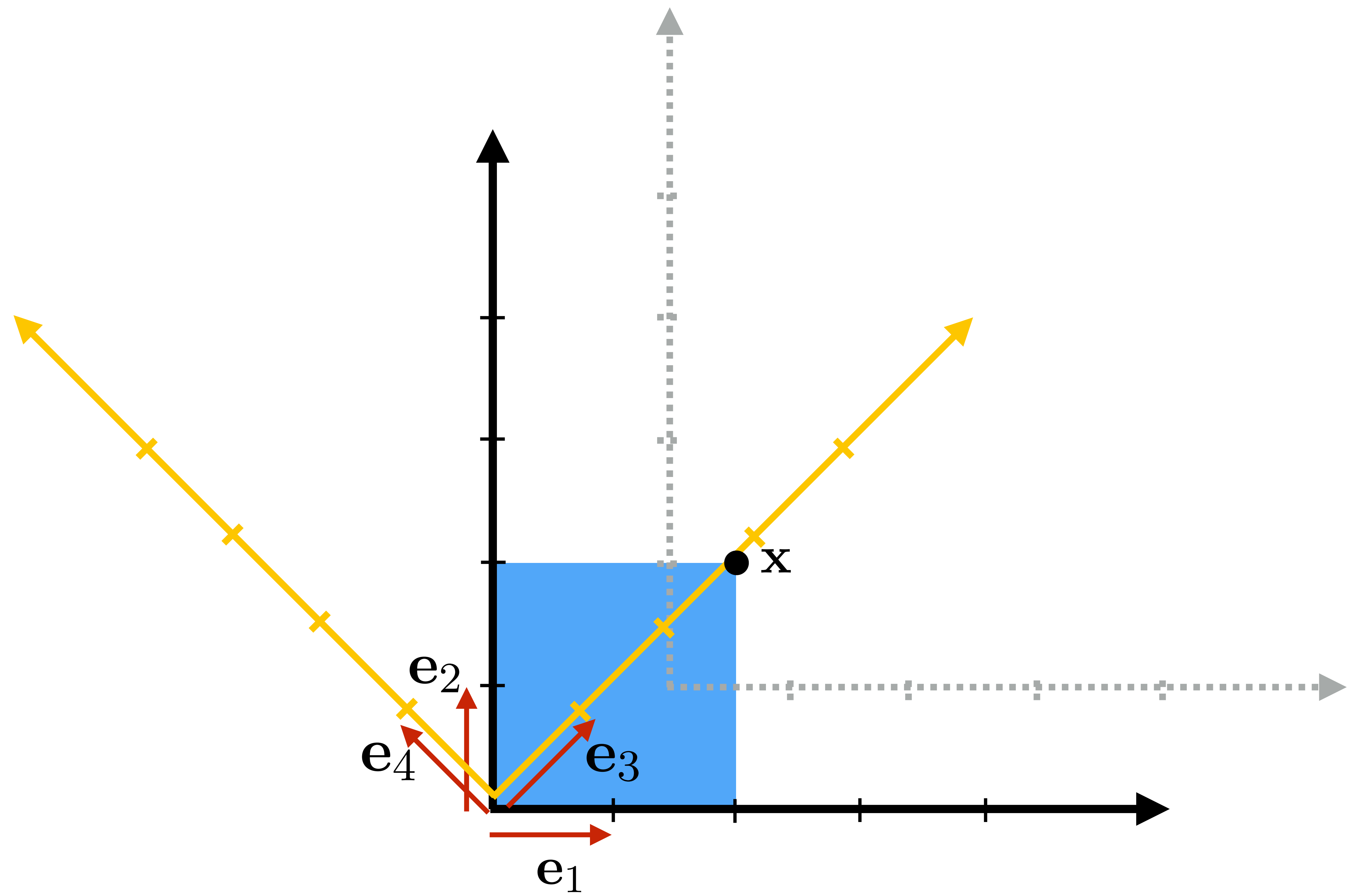
$$|f(\mathbf{x}) - f(\mathbf{y})| = |\mathbf{x} - \mathbf{y}|$$

Translation
Rotation
Reflection

"Rigid body" transforms are Euclidean transforms that also preserve "winding" (does not include reflection)

Representing Transforms

Review: representing points in a coordinate space



Consider coordinate space defined by orthogonal vectors $\mathbf{e}_1$ **and** $\mathbf{e}_2$

$$\mathbf{x} = 2\mathbf{e}_1 + 2\mathbf{e}_2$$

$$\mathbf{x} = \begin{bmatrix} 2 & 2 \end{bmatrix}$$

$\mathbf{x} = \begin{bmatrix} 0.5 & 1 \end{bmatrix}$ **in coordinate space defined by** $\mathbf{e}_1$ **and** $\mathbf{e}_2$**, with origin at (1.5, 1)**

$\mathbf{x} = \begin{bmatrix} \sqrt{8} & 0 \end{bmatrix}$ **in coordinate space defined by** $\mathbf{e}_3$ **and** $\mathbf{e}_4$**, with origin at (0, 0)**

Review: 2D matrix multiplication

$$\begin{bmatrix} ax + by \\ cx + dy \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

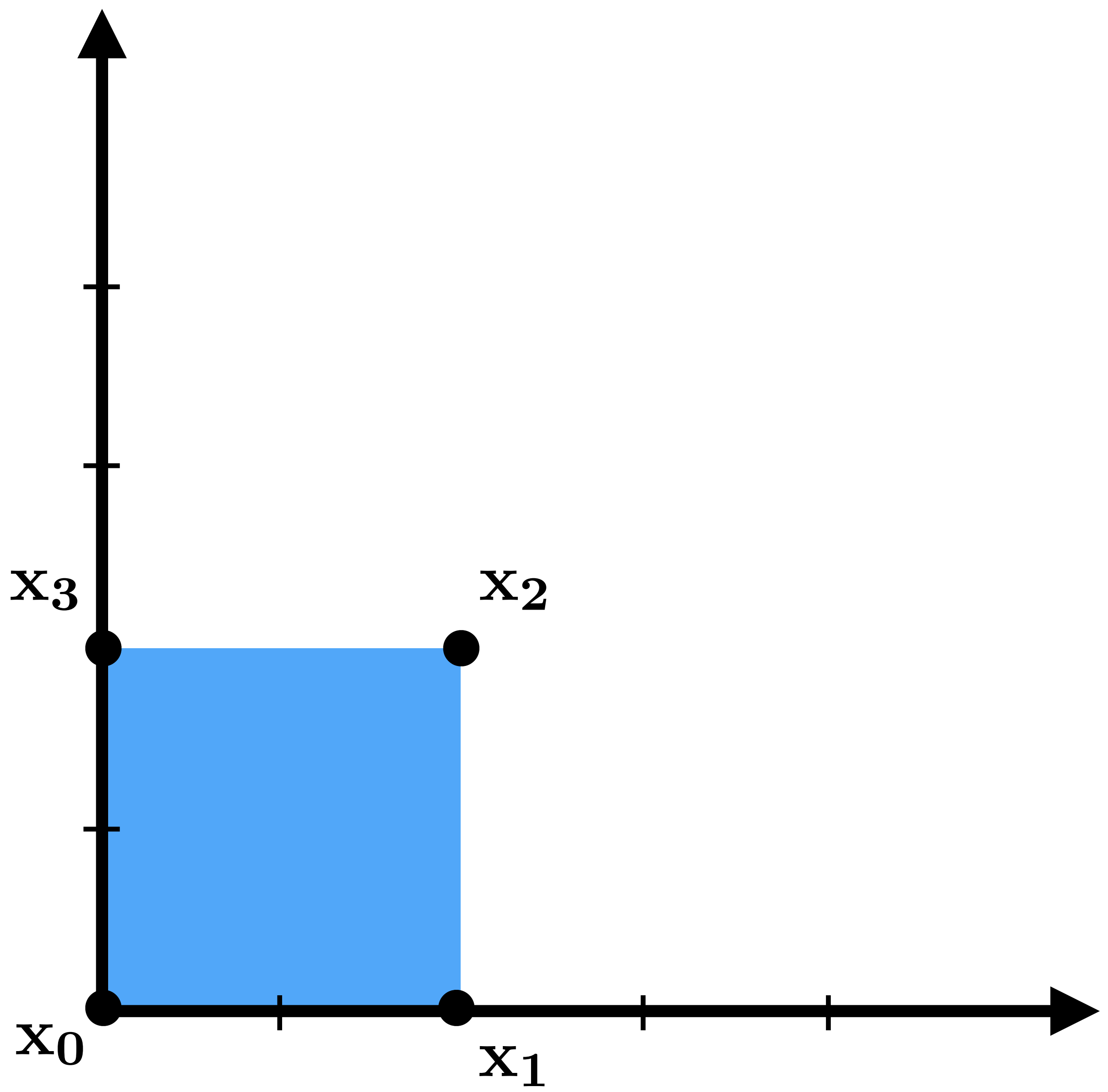
$$x \begin{bmatrix} a \\ c \end{bmatrix} + y \begin{bmatrix} b \\ d \end{bmatrix} =$$

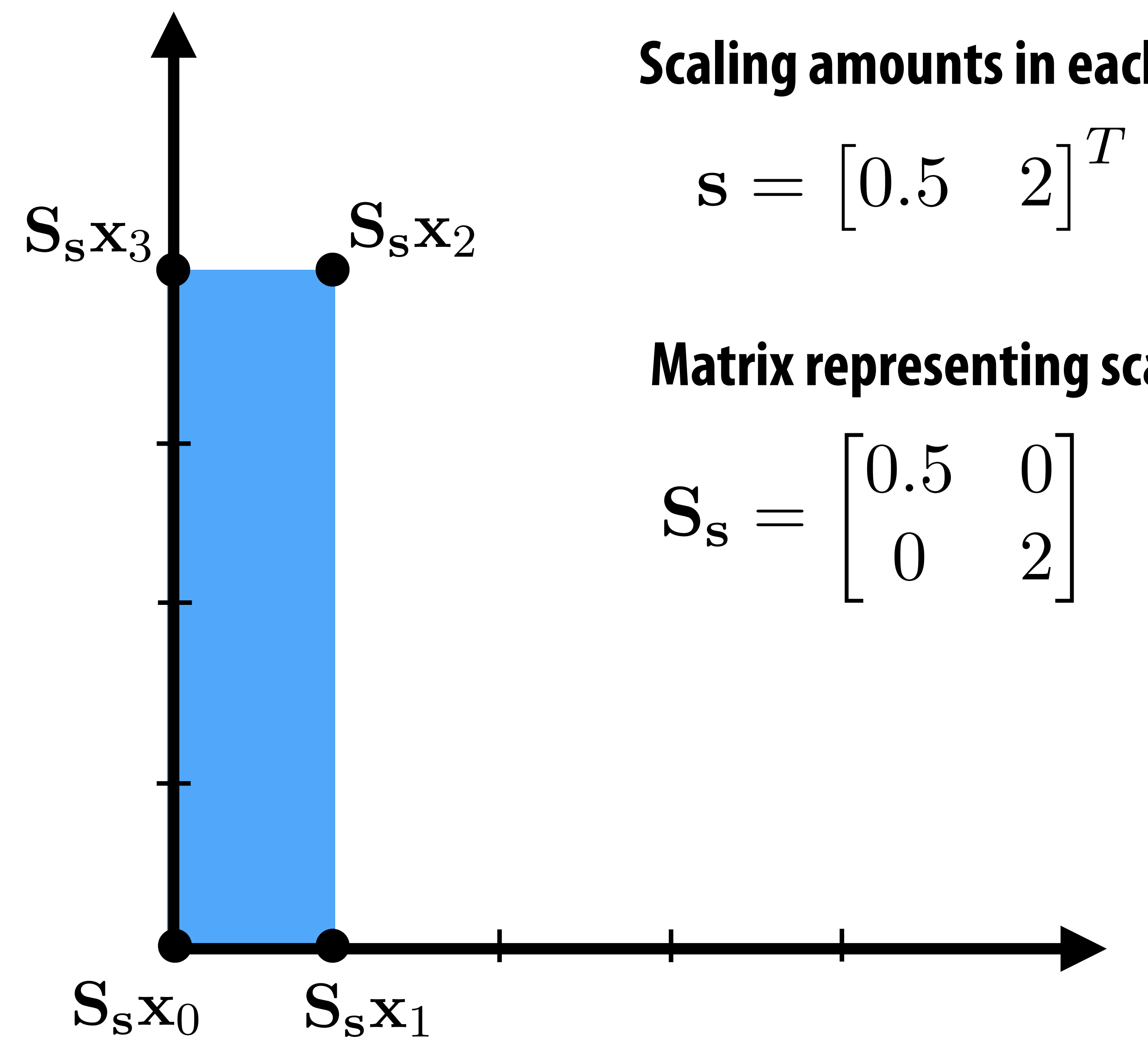
Linear transforms in 2D can be represented as 2x2 matrices

- Scale
- Shear
- Rotation
- Translation?

Linear transforms in 2D can be represented as 2x2 matrices

Consider non-uniform scale: $\mathbf{S_s} = \begin{bmatrix} \mathbf{s}_x & 0 \\ 0 & \mathbf{s}_y \end{bmatrix}$





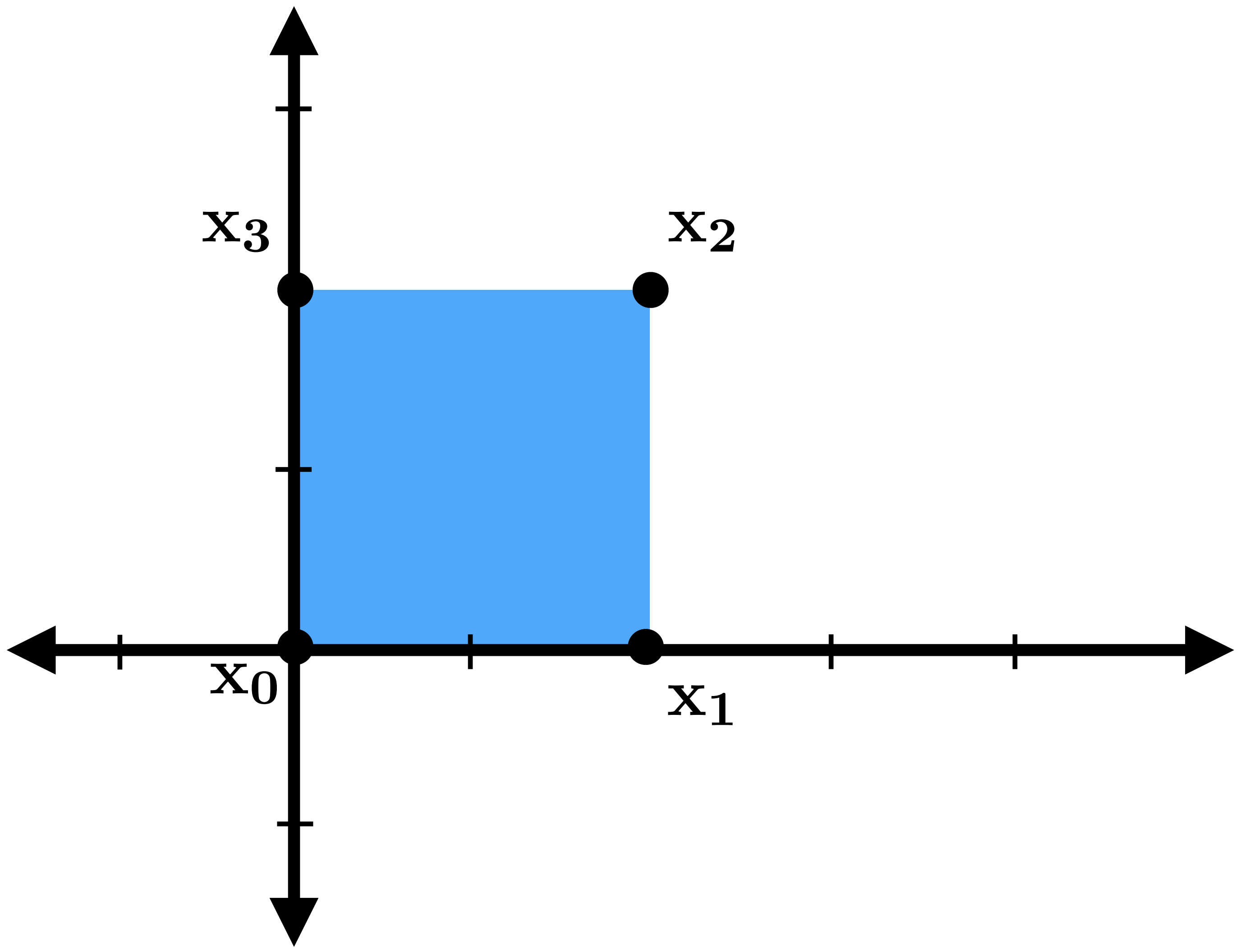
Scaling amounts in each direction:

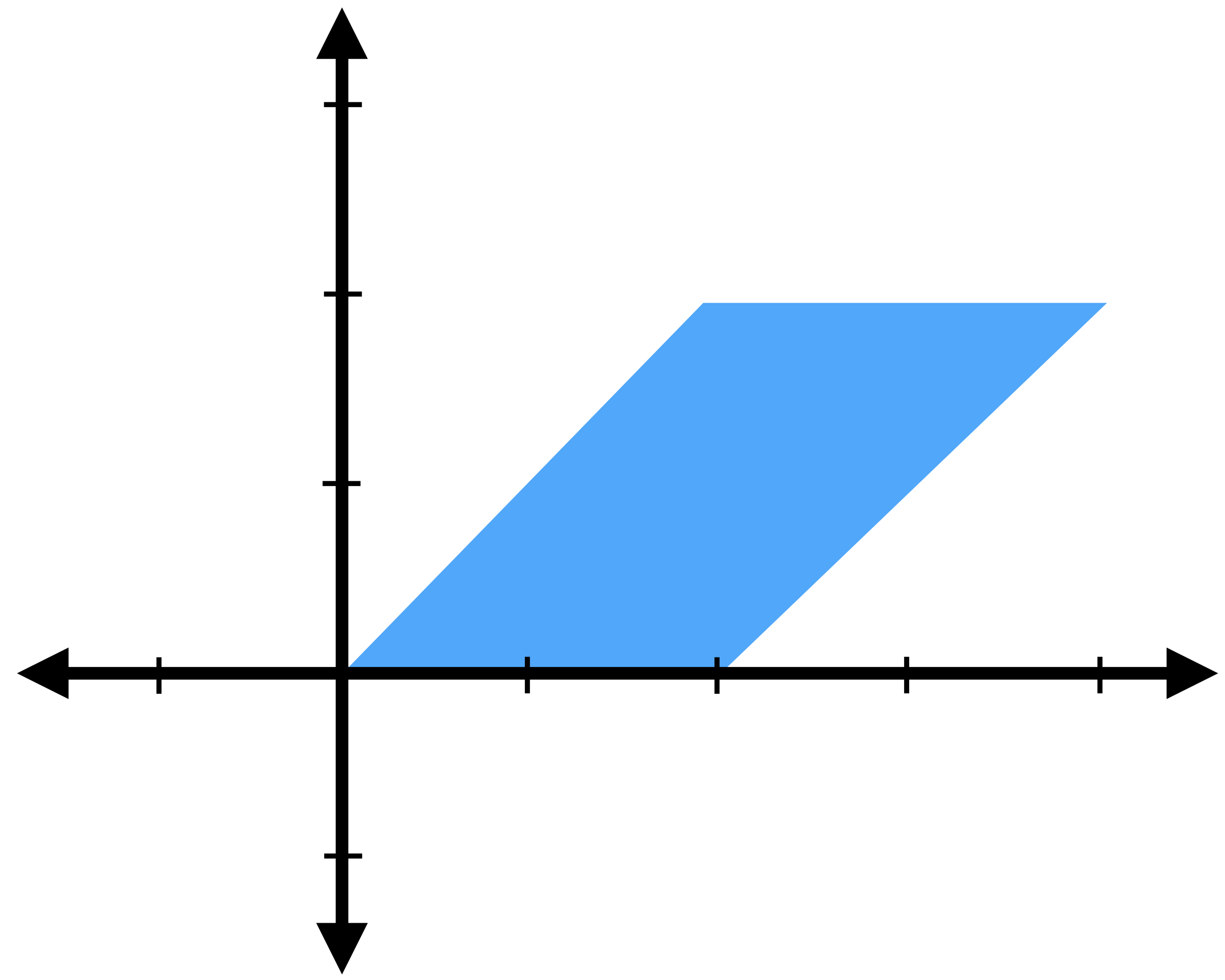
$$\mathbf{s} = \begin{bmatrix} 0.5 & 2 \end{bmatrix}^T$$

Matrix representing scale transform:

$$\mathbf{S_s} = \begin{bmatrix} 0.5 & 0 \\ 0 & 2 \end{bmatrix}$$

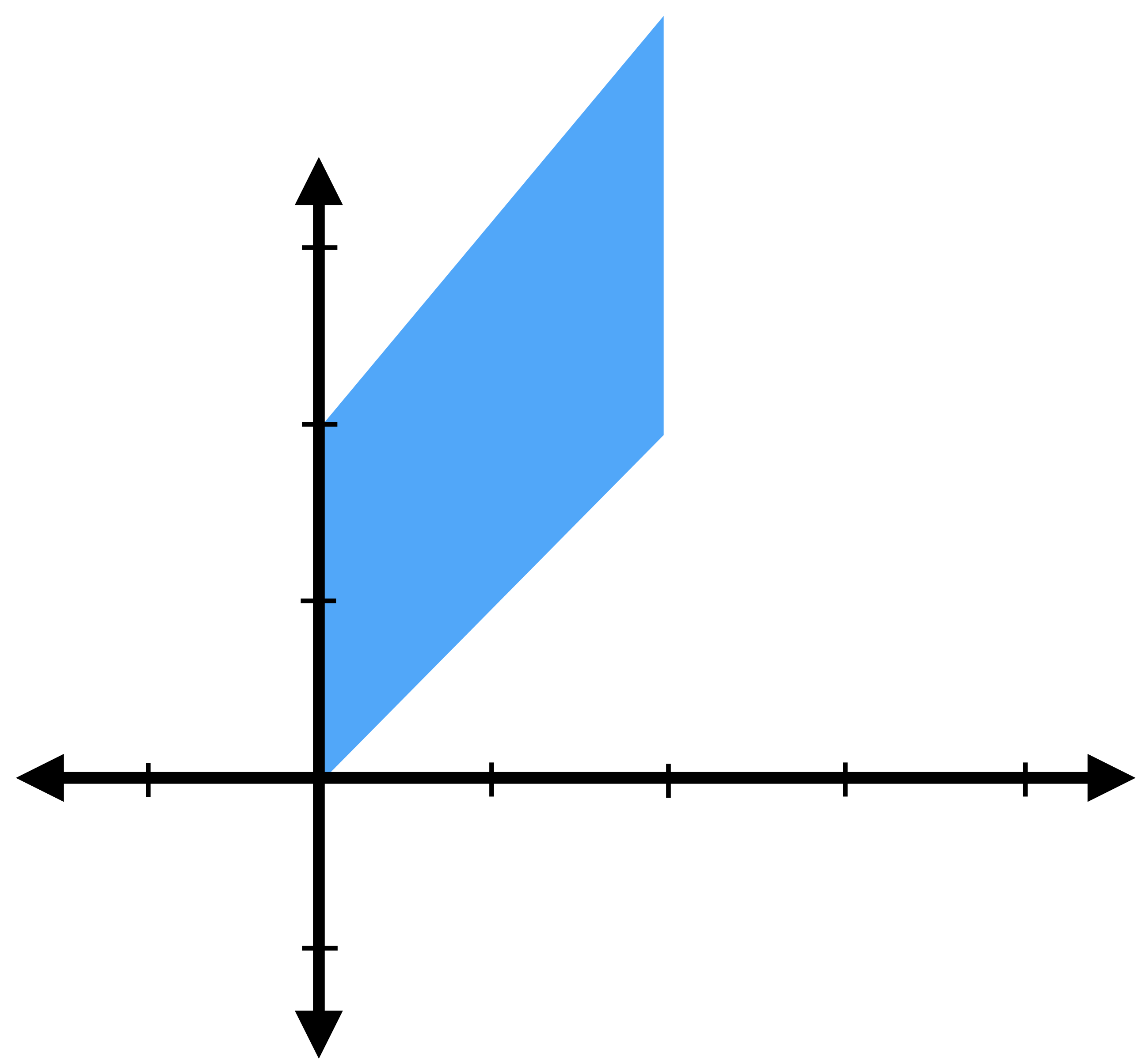
Shear





Shear in x:

$$\mathbf{H}_{xs} = \begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix}$$



Shear in y:

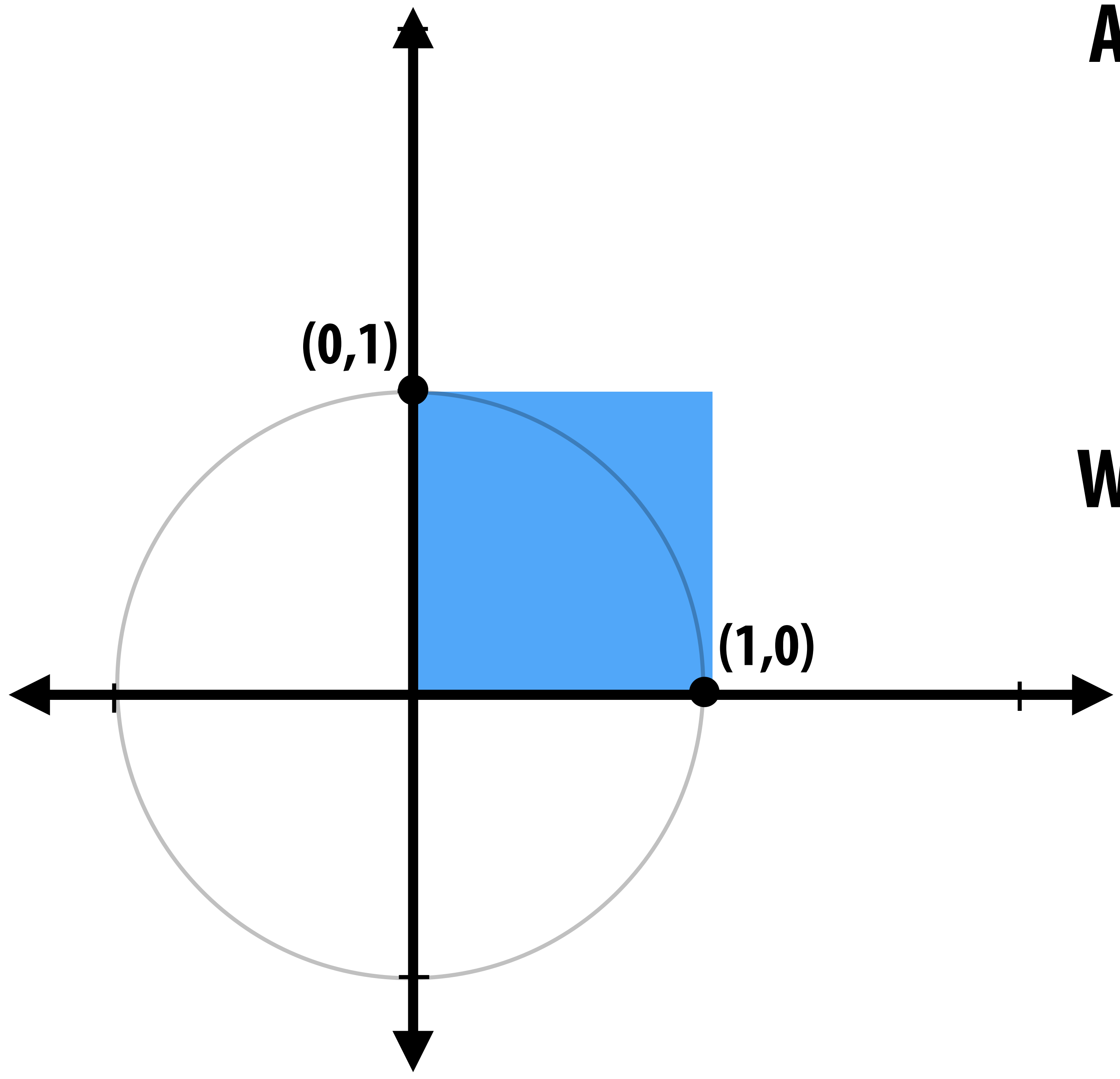
$$\mathbf{H}_{ys} = \begin{bmatrix} 1 & 0 \\ s & 1 \end{bmatrix}$$

Rotation matrix (2D)

Question: what happens to (1, 0) and (0,1) after rotation by θ?

Reminder: rotation moves points along circular trajectories.

(Recall that $\cos\theta$ and $\sin\theta$ are the coordinates of a point on the unit circle.)



Answer:

$$R_\theta(1,0) = (\cos(\theta), \sin(\theta))$$

$$R_\theta(0,1) = (\cos(\theta + \pi/2), \sin(\theta + \pi/2))$$

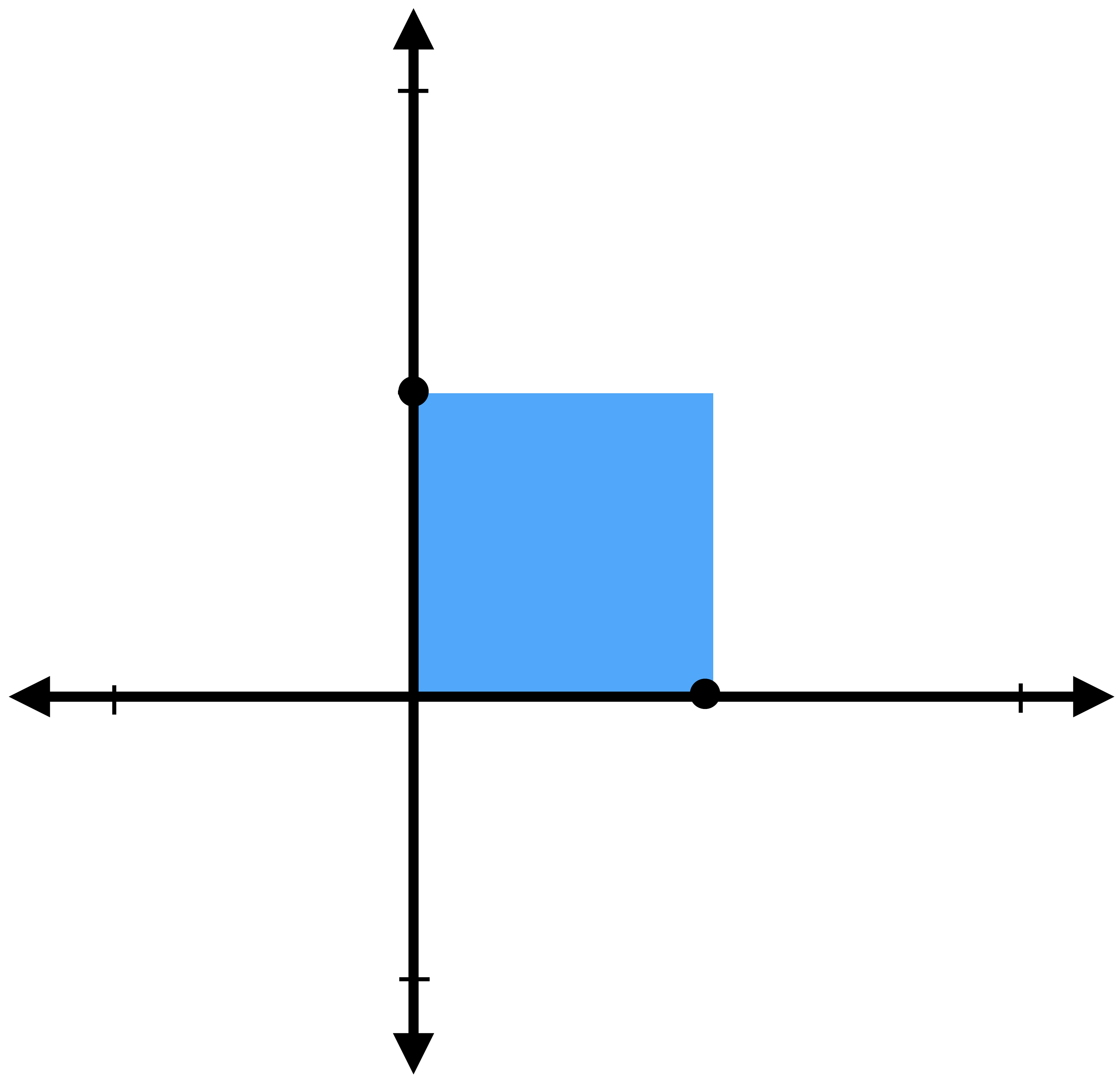
Which means the matrix must look like:

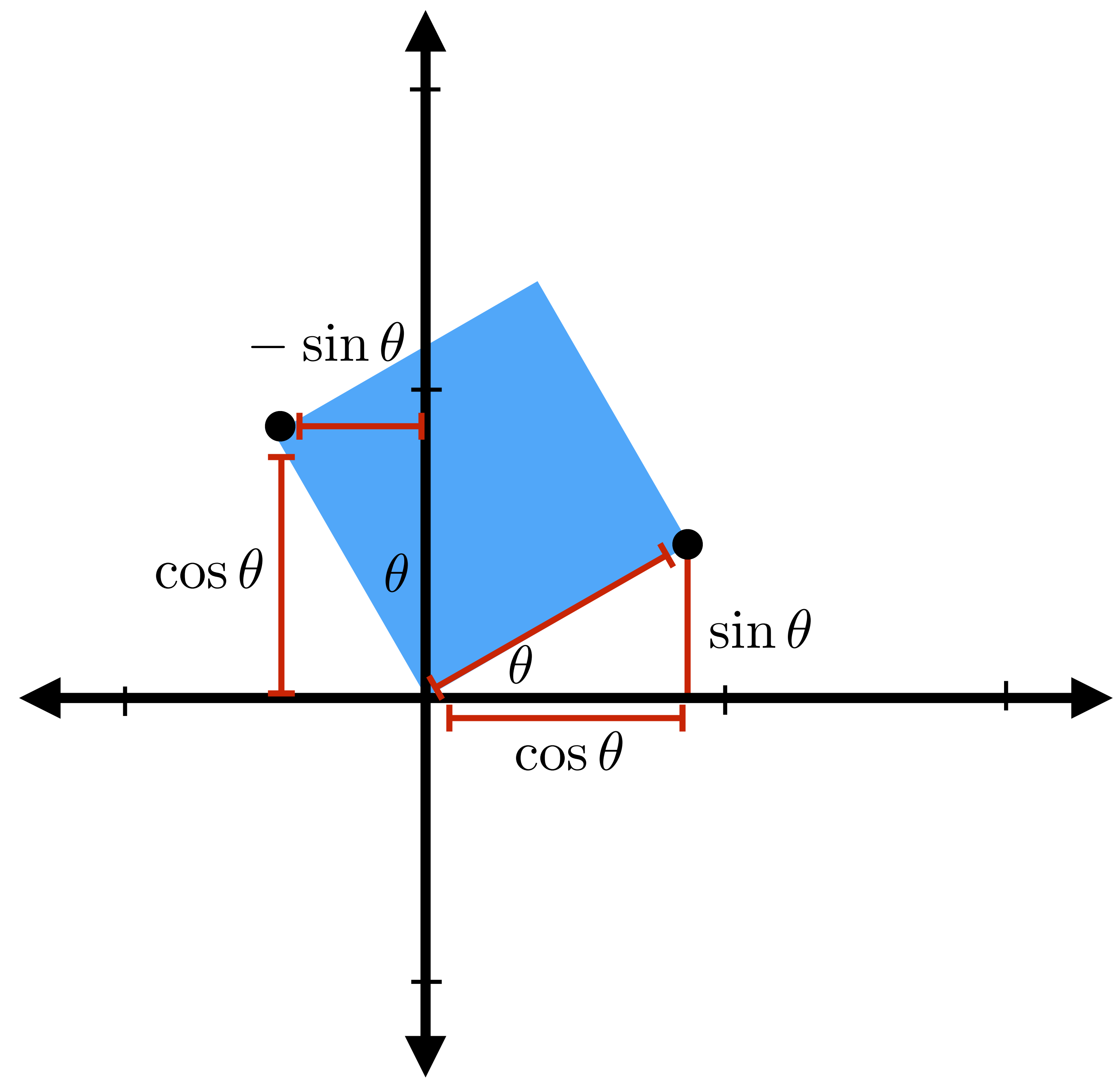
$$R_\theta = \begin{bmatrix} \cos(\theta) & \cos(\theta + \pi/2) \\ \sin(\theta) & \sin(\theta + \pi/2) \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

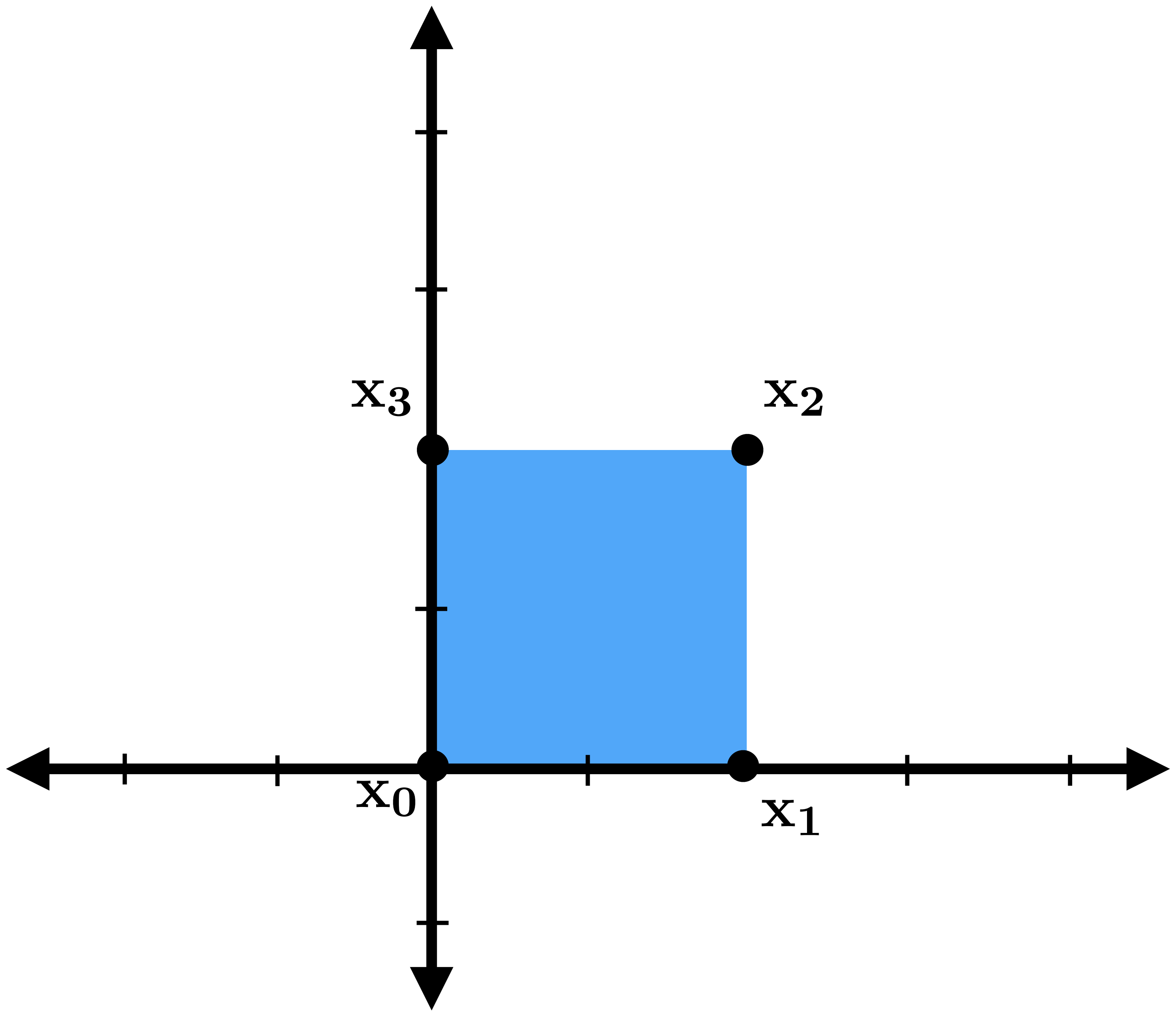
Rotation matrix (2D): another way...

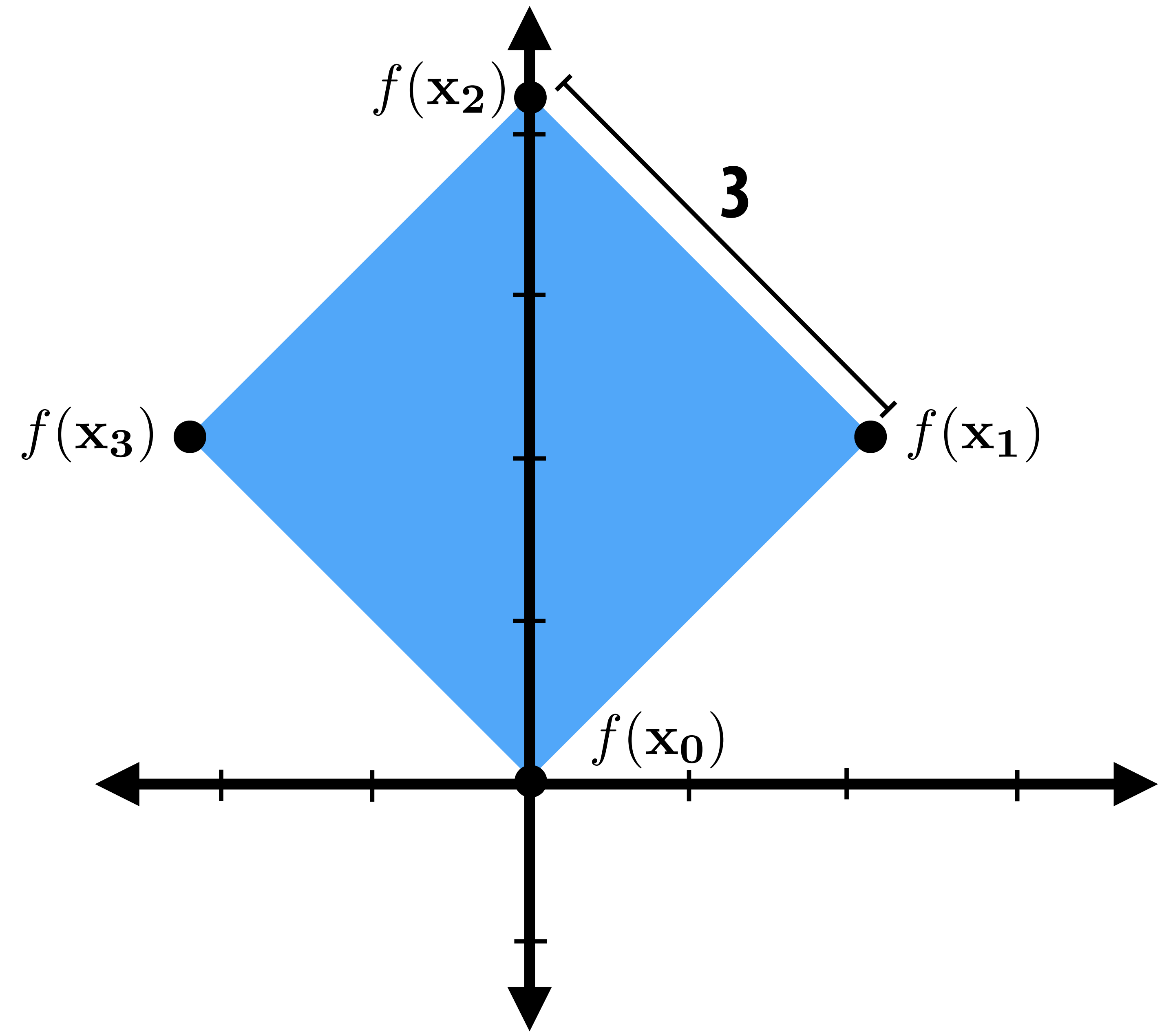
$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$





How do we compose linear transforms?





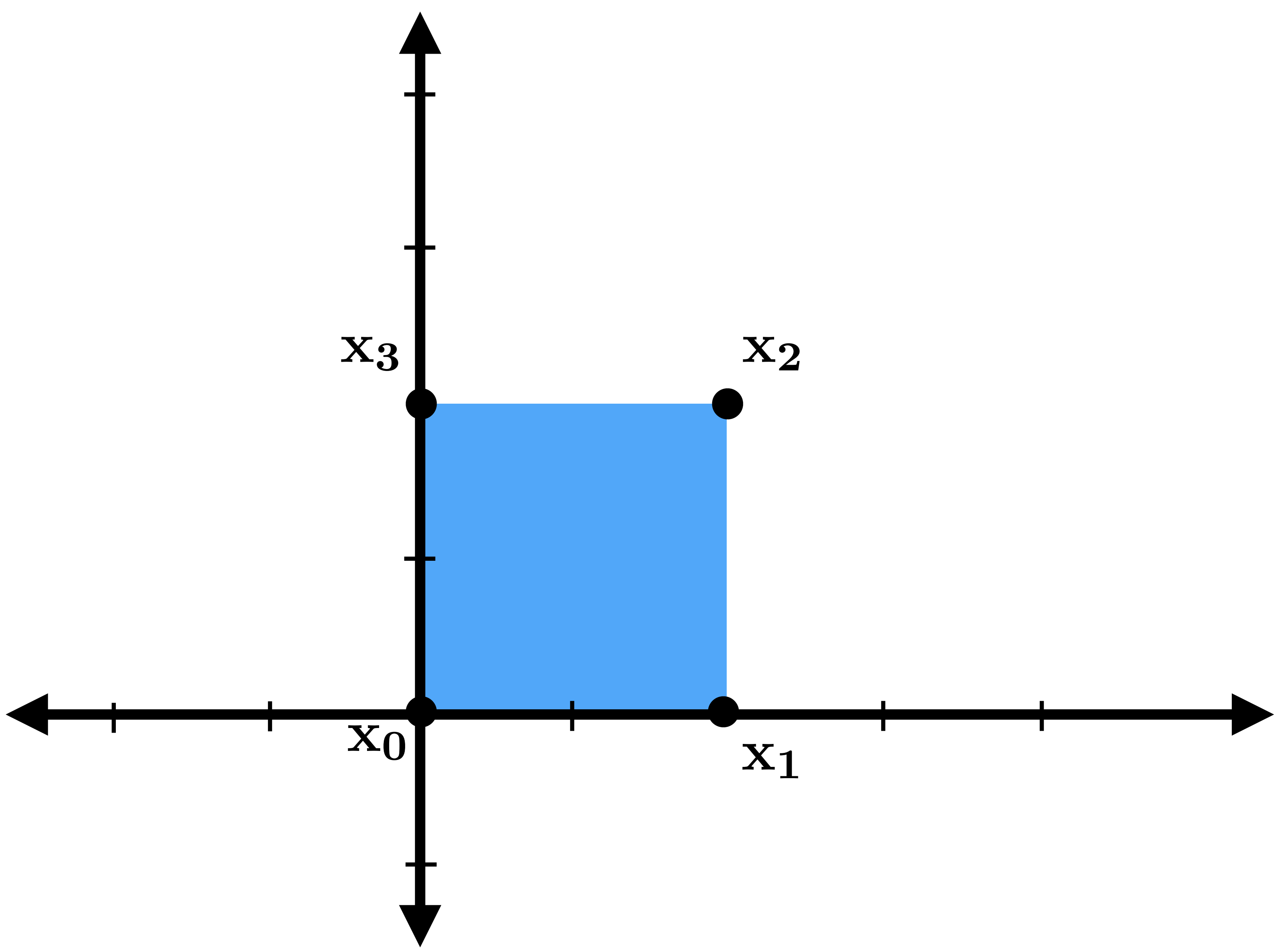
Compose linear transforms via matrix multiplication.
This example: uniform scale, followed by rotation

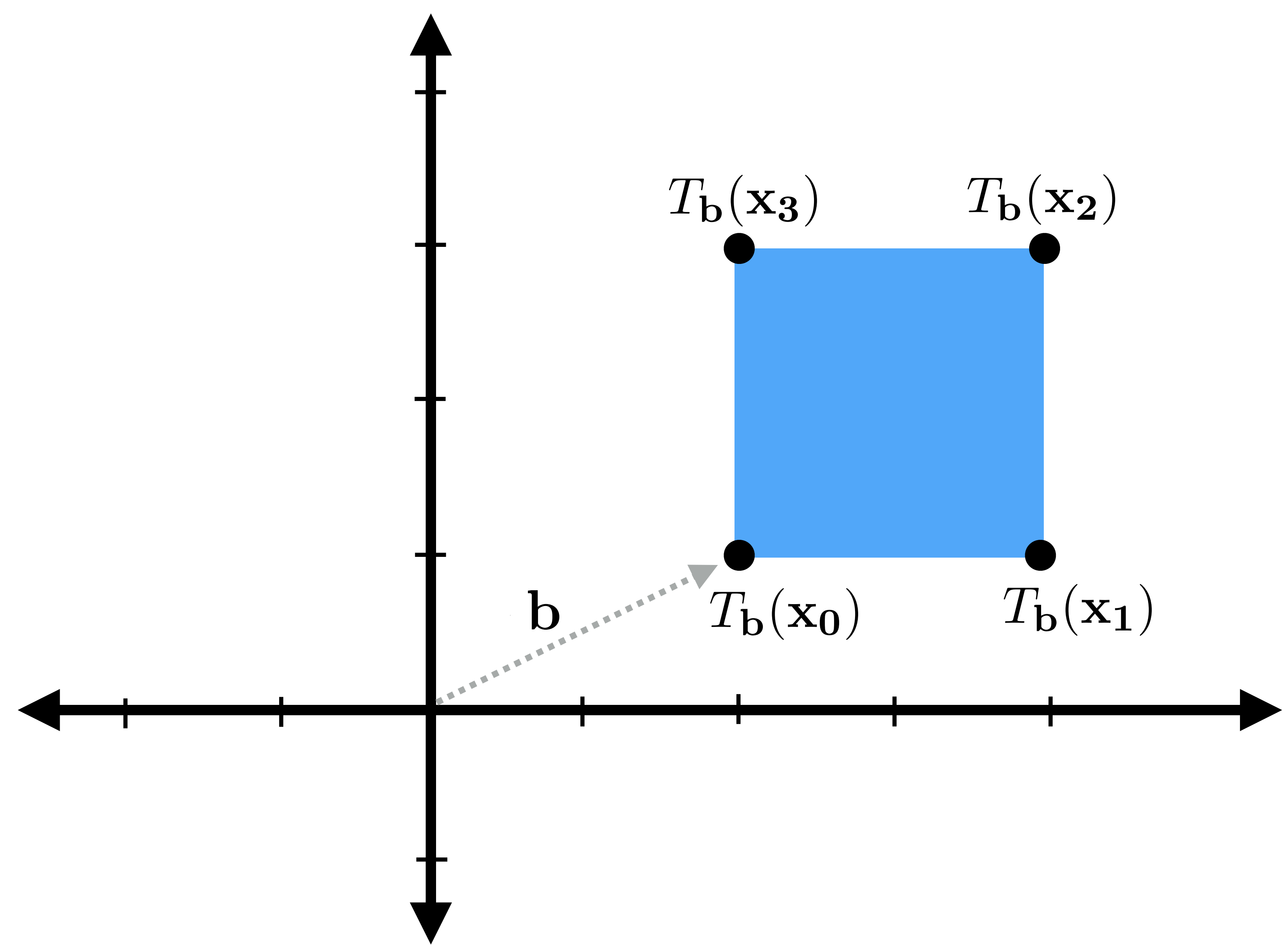
$$f(\mathbf{x}) = R_{\pi/4}\mathbf{S}_{[1.5,1.5]}\mathbf{x}$$

Enables simple, efficient implementation: reduce complex chain of transforms to a single matrix multiplication.

Translation?

$$T_{\mathbf{b}}(\mathbf{x}) = \mathbf{x} + \mathbf{b}$$





Recall: translation is not a linear transform

→ Output coefficients are not a linear combination of input coefficients

→ Translation operation cannot be represented by a 2x2 matrix

$$\mathbf{x_{out}}_x = \mathbf{x}_x + \mathbf{b}_x$$

$$\mathbf{x_{out}}_y = \mathbf{x}_y + \mathbf{b}_y$$

Translation math

2D homogeneous coordinates (2D-H)

Key idea: represent 2D points in 3D coordinate space

So the point (x, y) **is represented as the 3-vector:** $\begin{bmatrix} x & y & 1 \end{bmatrix}^T$

And transforms are represented a 3x3 matrices that transform these vectors.

For example: here are 2D scale and rotation transforms written in 2D homogeneous form:

$$\mathbf{S_s} = \begin{bmatrix} \mathbf{S}_x & 0 & 0 \\ 0 & \mathbf{S}_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{R}_\theta = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Observe:

In these examples, the last row just propagates third coordinate of input to output.

Expressing transformations in 2D-H coords

Translation expressed as 3x3 matrix multiplication:

$$\mathbf{T_b} = \begin{bmatrix} 1 & 0 & \mathbf{b}_x \\ 0 & 1 & \mathbf{b}_y \\ 0 & 0 & 1 \end{bmatrix}$$

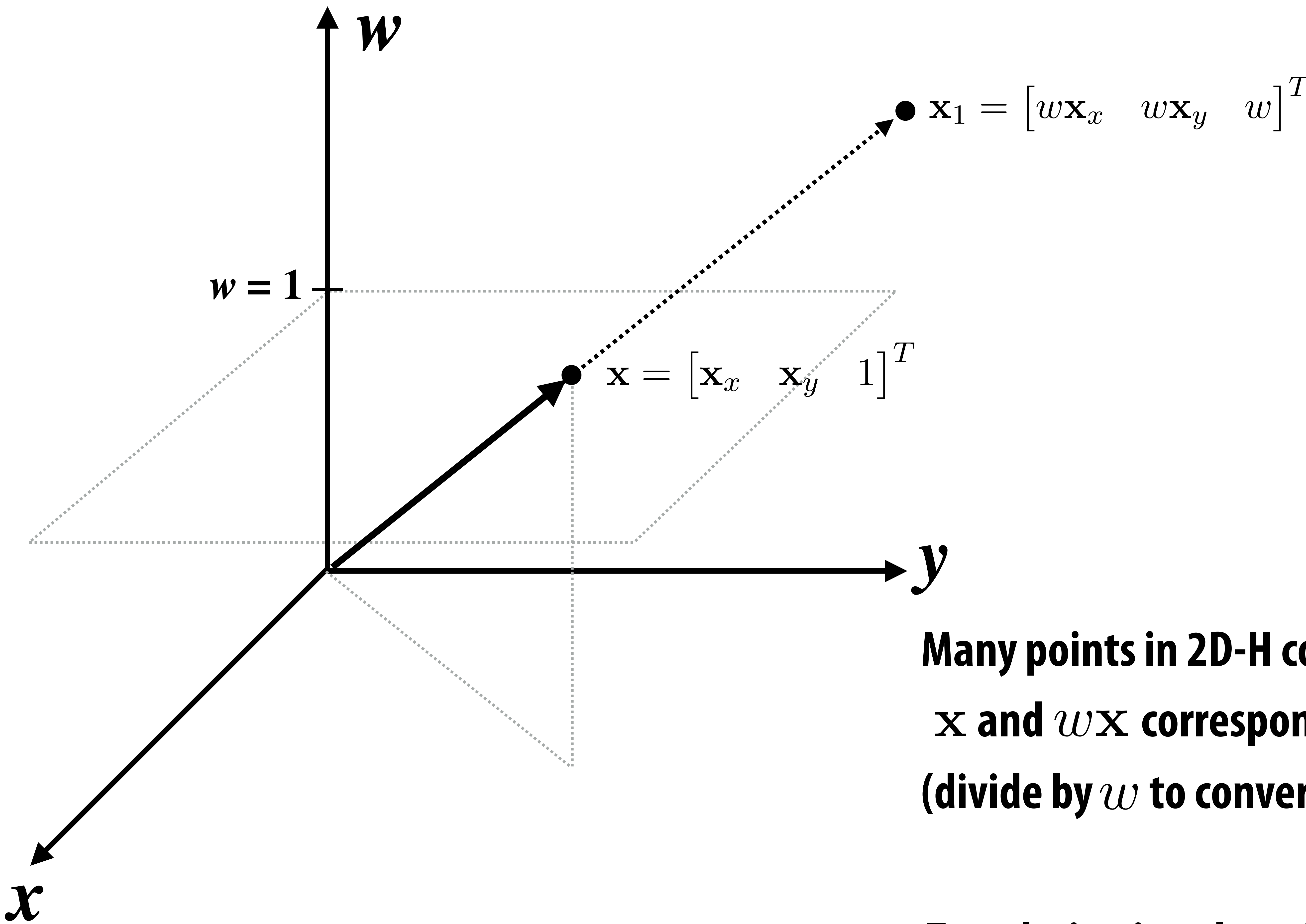
$$\mathbf{T_b x} = \begin{bmatrix} 1 & 0 & \mathbf{b}_x \\ 0 & 1 & \mathbf{b}_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_x \\ \mathbf{x}_y \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}_x + \mathbf{b}_x \\ \mathbf{x}_y + \mathbf{b}_y \\ 1 \end{bmatrix}$$

Homogeneous representation enables composition of affine transforms!

Example: rotation about point $\mathbf{b}$

$\mathbf{T_b R_\theta T_{-b}}$

Homogeneous coordinates: some intuition



Many points in 2D-H correspond to same point in 2D

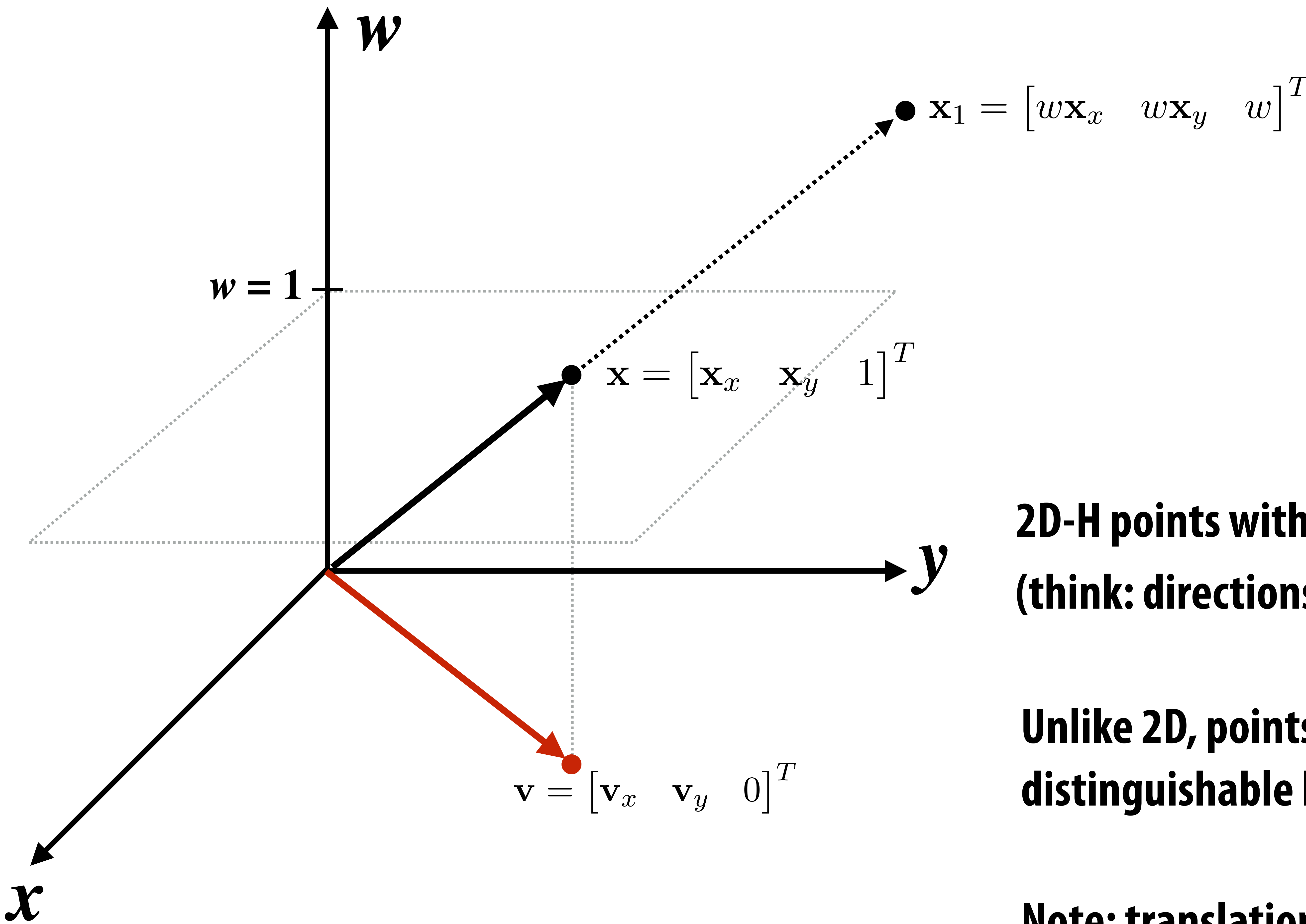
$\mathbf{x}$ and $w\mathbf{x}$ correspond to the same 2D point

(divide by w to convert 2D-H back to 2D)

Translation is a shear in x and y in 2D-H space

$$\mathbf{T_b}\mathbf{x} = \begin{bmatrix} 1 & 0 & \mathbf{b}_x \\ 0 & 1 & \mathbf{b}_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} w\mathbf{x}_x \\ w\mathbf{x}_y \\ w \end{bmatrix} = \begin{bmatrix} w\mathbf{x}_x + w\mathbf{b}_x \\ w\mathbf{x}_y + w\mathbf{b}_y \\ w \end{bmatrix}$$

Homogeneous coordinates: points vs. vectors



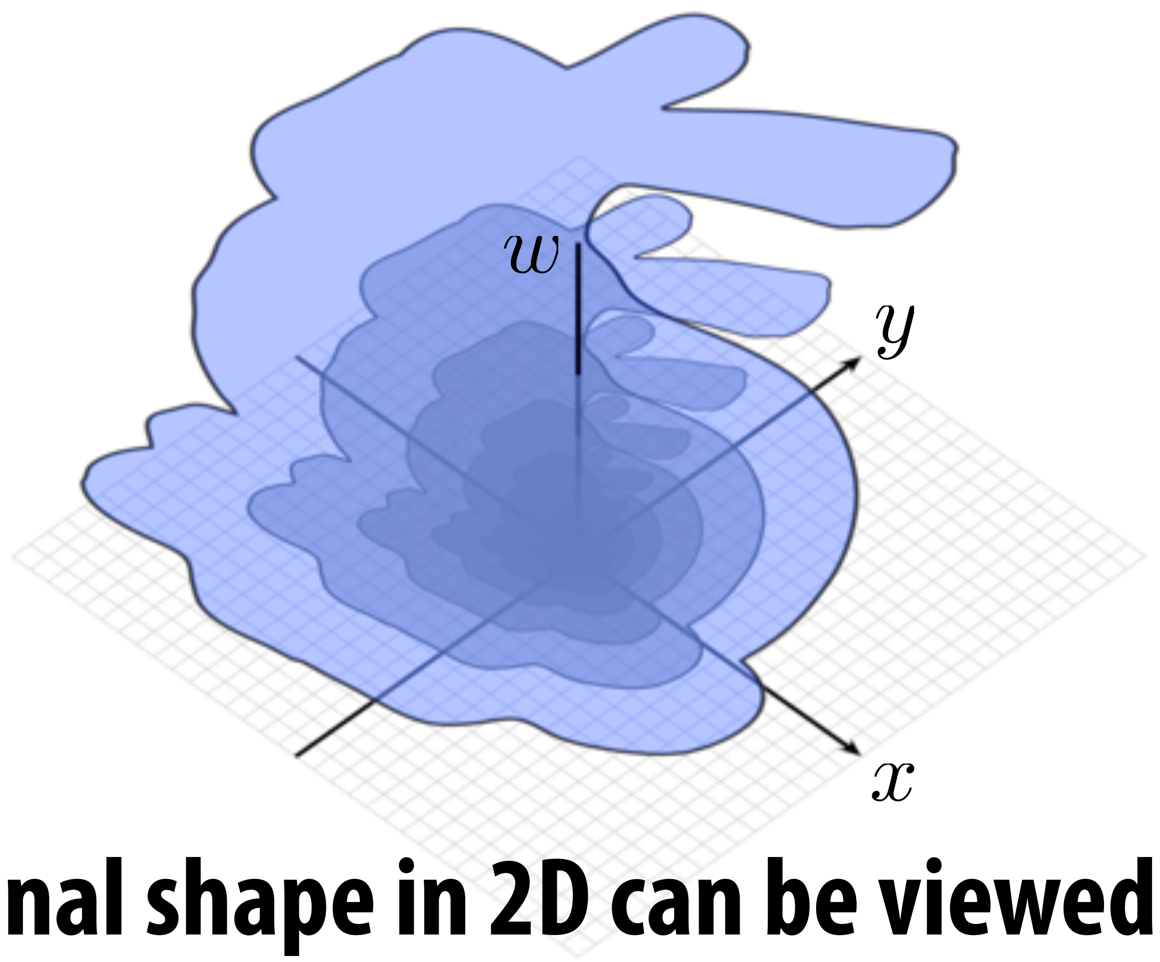
2D-H points with w=0 represent 2D vectors (think: directions are points at infinity)

Unlike 2D, points and directions are distinguishable by their representation in 2D-H

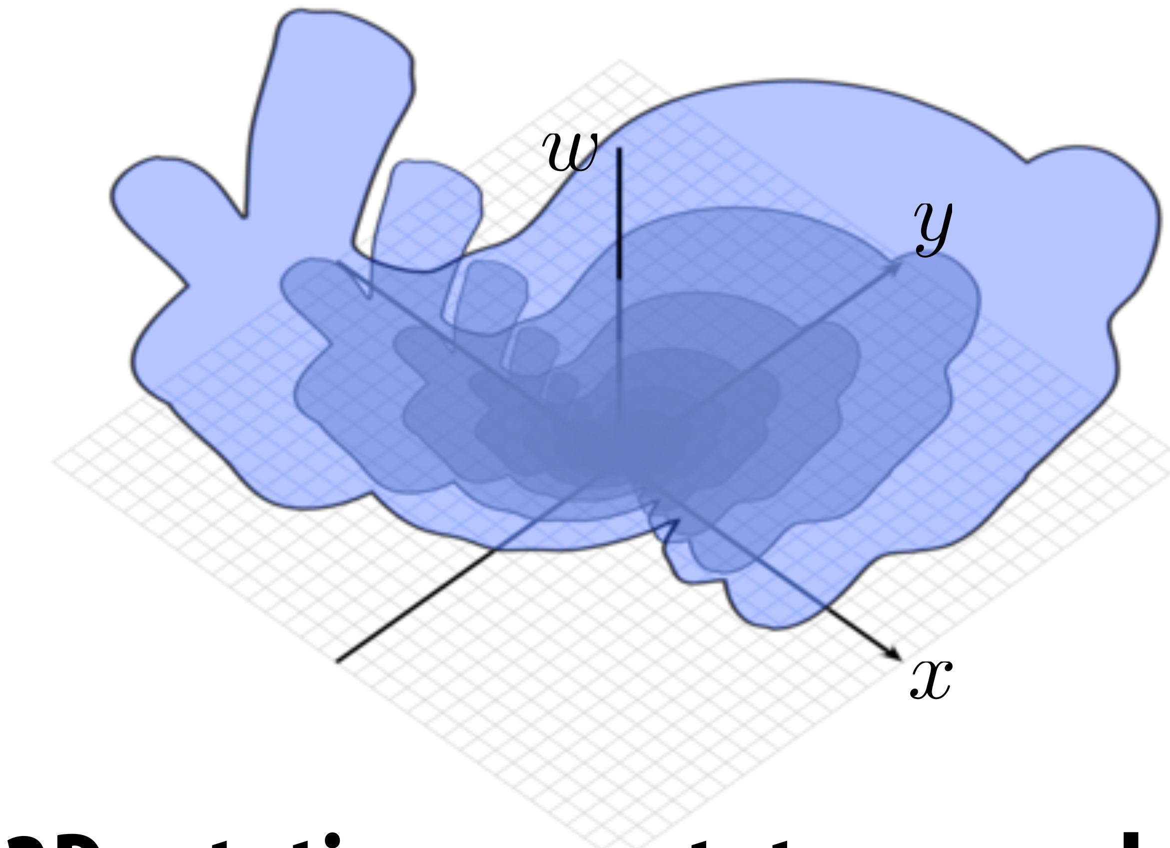
Note: translation does not modify directions:

$$\mathbf{T_b}\mathbf{v} = \begin{bmatrix} 1 & 0 & \mathbf{b}_x \\ 0 & 1 & \mathbf{b}_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{v}_x \\ \mathbf{v}_y \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{v}_x \\ \mathbf{v}_y \\ 0 \end{bmatrix}$$

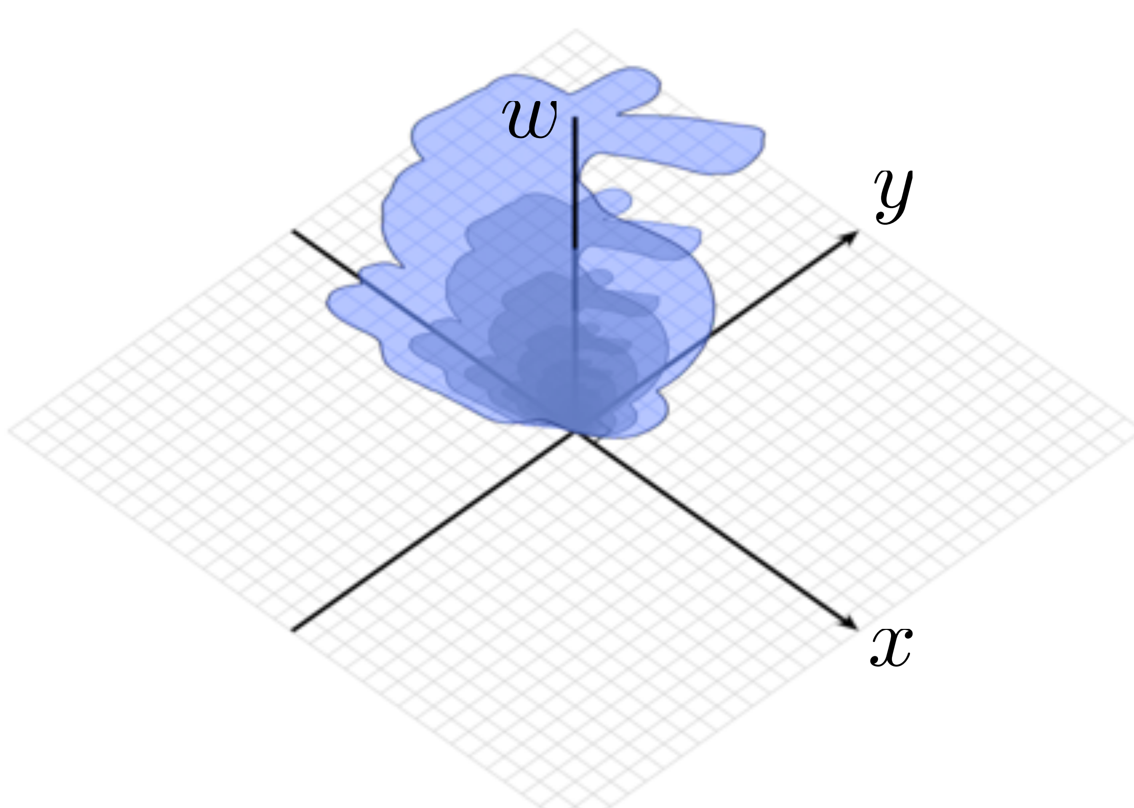
Visualizing 2D transformations in 2D-H



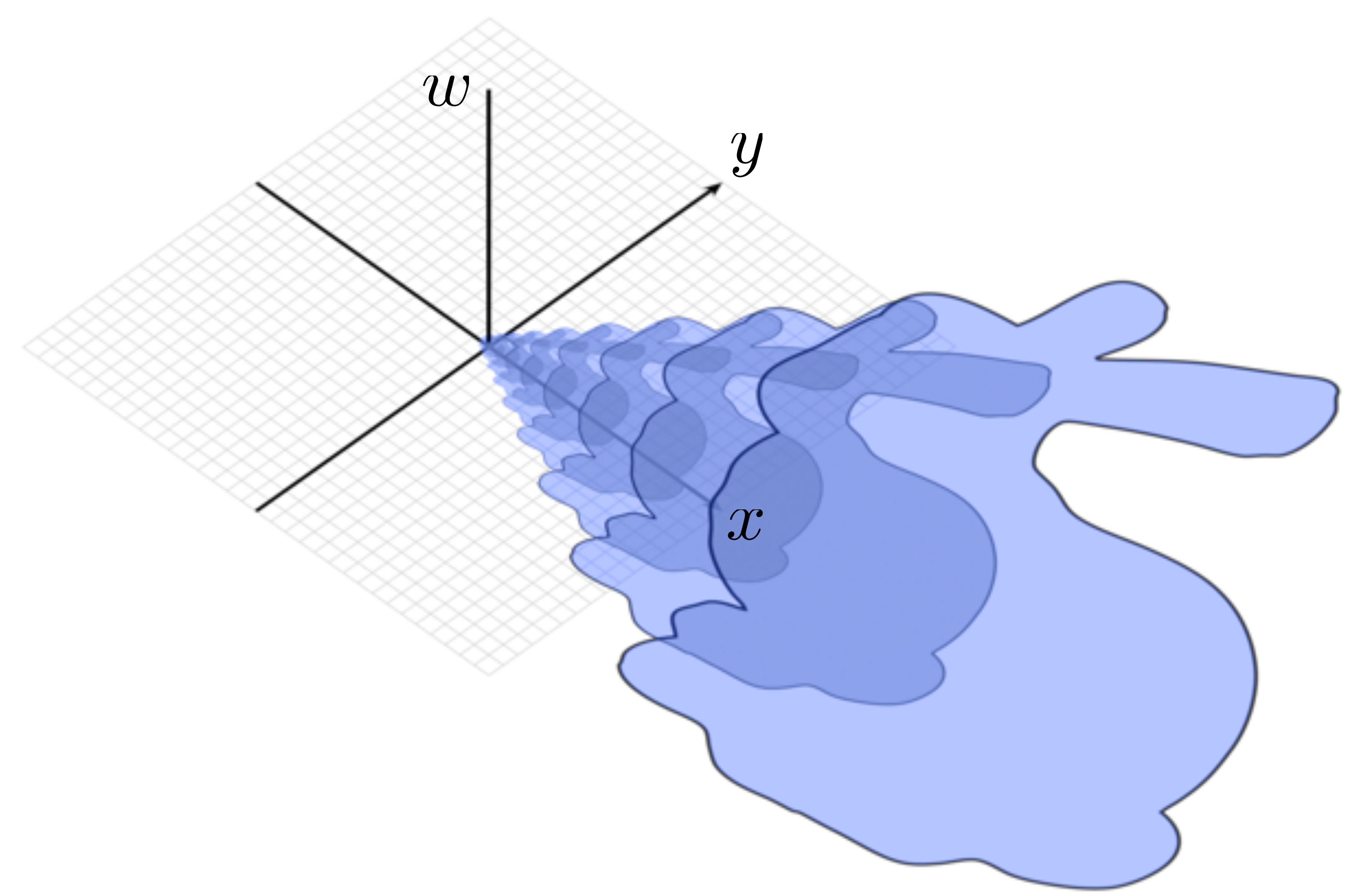
Original shape in 2D can be viewed as many copies, uniformly scaled by w.



2D rotation ↔ rotate around w



2D scale ↔ scale x and y; preserve w
(Question: what happens to 2D shape if you scale x, y, and w



2D translate ↔ shear in xy

Moving to 3D (and 3D-H)

Represent 3D transforms as 3x3 matrices and 3D-H transforms as 4x4 matrices

Scale:

3D

$$\mathbf{S_s} = \begin{bmatrix} \mathbf{S}_x & 0 & 0 \\ 0 & \mathbf{S}_y & 0 \\ 0 & 0 & \mathbf{S}_z \end{bmatrix}$$

3D-H

$$\mathbf{S_s} = \begin{bmatrix} \mathbf{S}_x & 0 & 0 & 0 \\ 0 & \mathbf{S}_y & 0 & 0 \\ 0 & 0 & \mathbf{S}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Shear (in x, based on y,z position):

$$\mathbf{H}_{x,\mathbf{d}} = \begin{bmatrix} 1 & \mathbf{d}_y & \mathbf{d}_z \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{H}_{x,\mathbf{d}} = \begin{bmatrix} 1 & \mathbf{d}_y & \mathbf{d}_z & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translate:

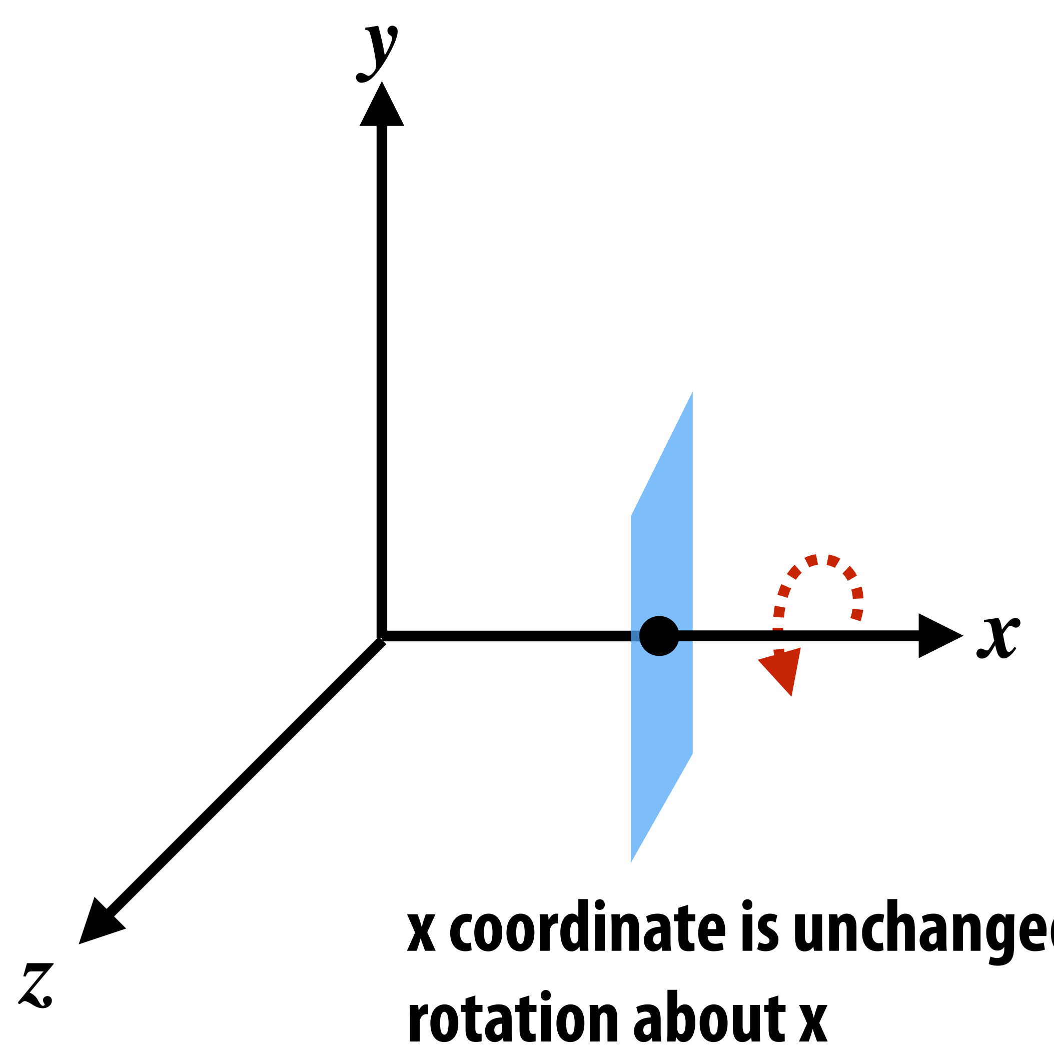
3D-H

$$\mathbf{T_b} = \begin{bmatrix} 1 & 0 & 0 & \mathbf{b}_x \\ 0 & 1 & 0 & \mathbf{b}_y \\ 0 & 0 & 1 & \mathbf{b}_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotations in 3D

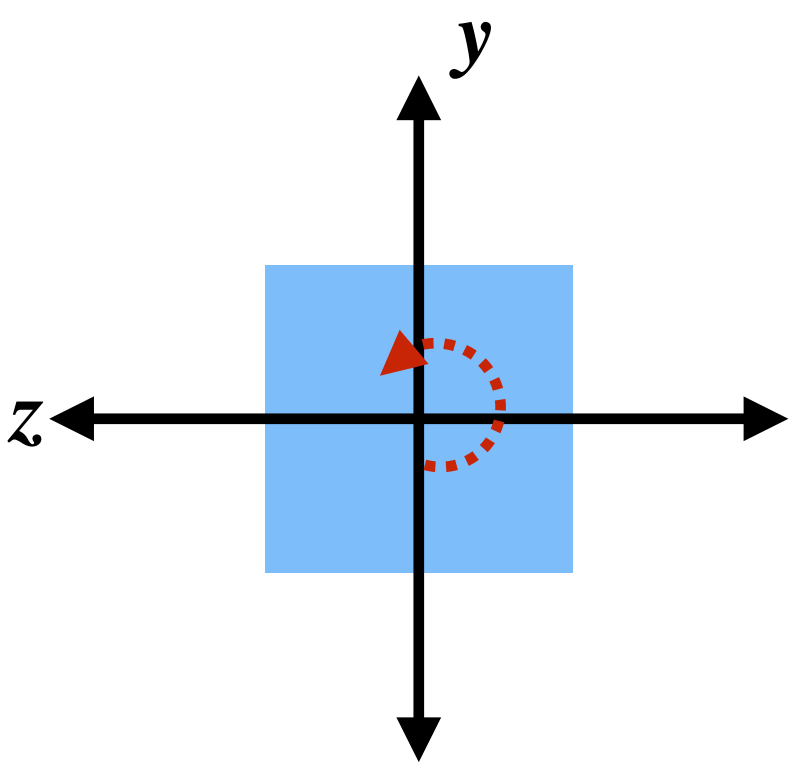
Rotation about x axis:

$$\mathbf{R}_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$



x coordinate is unchanged by rotation about x

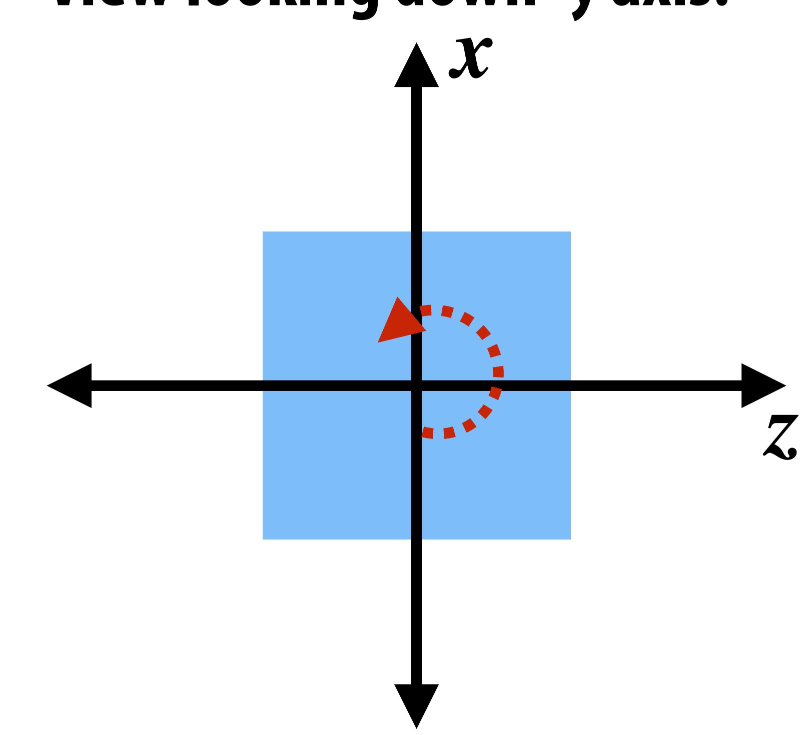
View looking down -x axis:



Rotation about y axis:

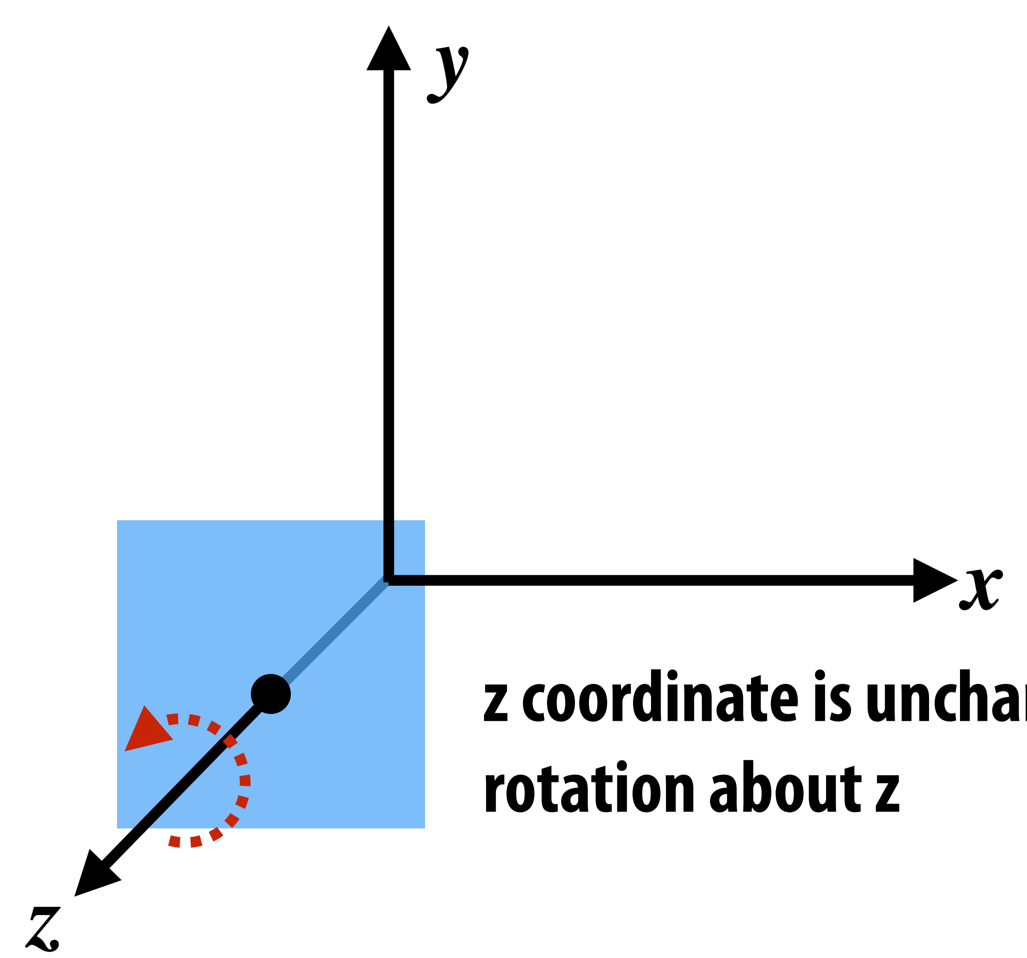
$$\mathbf{R}_{y,\theta} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

View looking down -y axis:



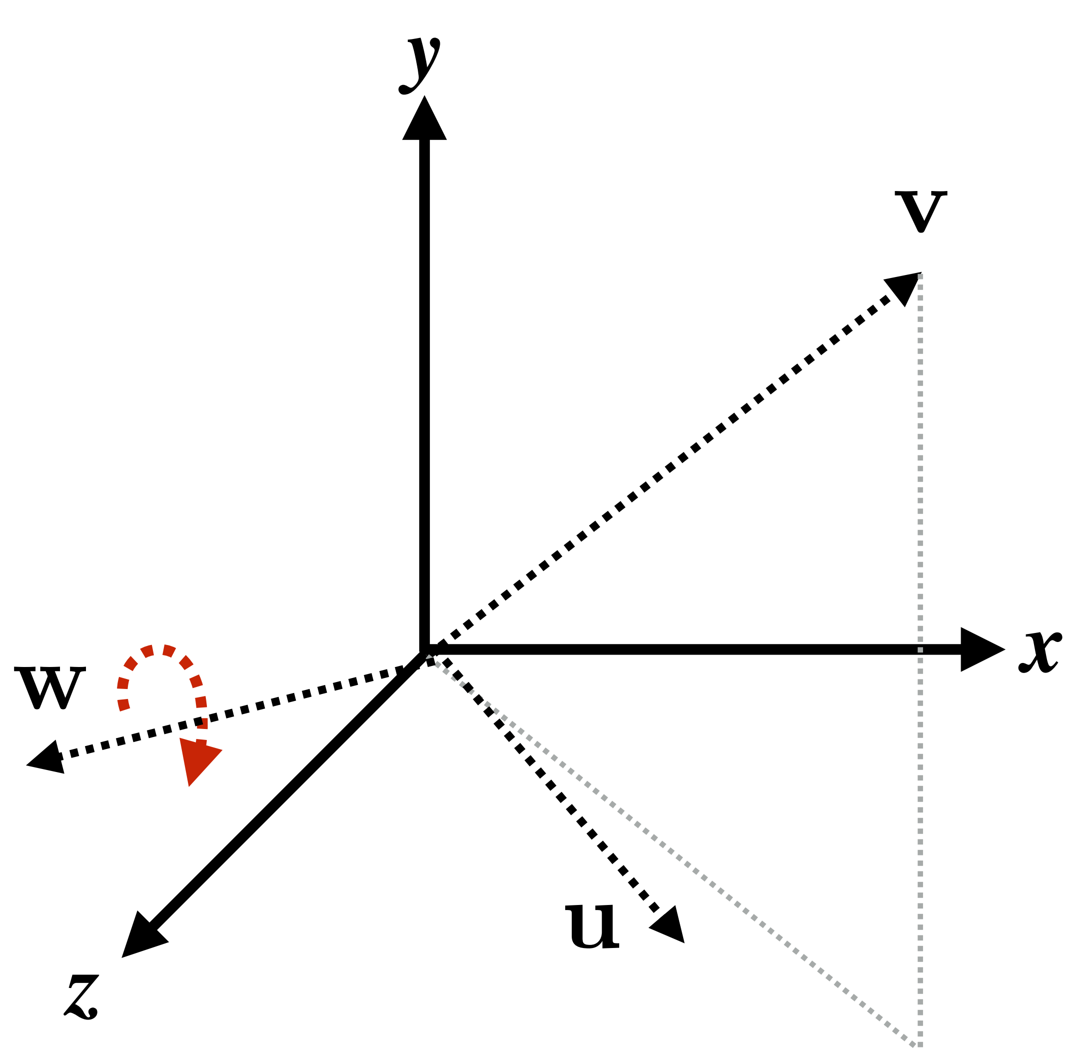
Rotation about z axis:

$$\mathbf{R}_{z,\theta} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



z coordinate is unchanged by rotation about z

Rotation about an arbitrary axis



To rotate by θ about w:

1. Form orthonormal basis around w (see u and v in figure)
2. Rotate to map w to [0 0 1] (change in coordinate space)

$$\mathbf{R}_{uvw} = \begin{bmatrix} \mathbf{u}_x & \mathbf{u}_y & \mathbf{u}_z \\ \mathbf{v}_x & \mathbf{v}_y & \mathbf{v}_z \\ \mathbf{w}_x & \mathbf{w}_y & \mathbf{w}_z \end{bmatrix}$$

$$\mathbf{R}_{uvw}\mathbf{u} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{R}_{uvw}\mathbf{v} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{R}_{uvw}\mathbf{w} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

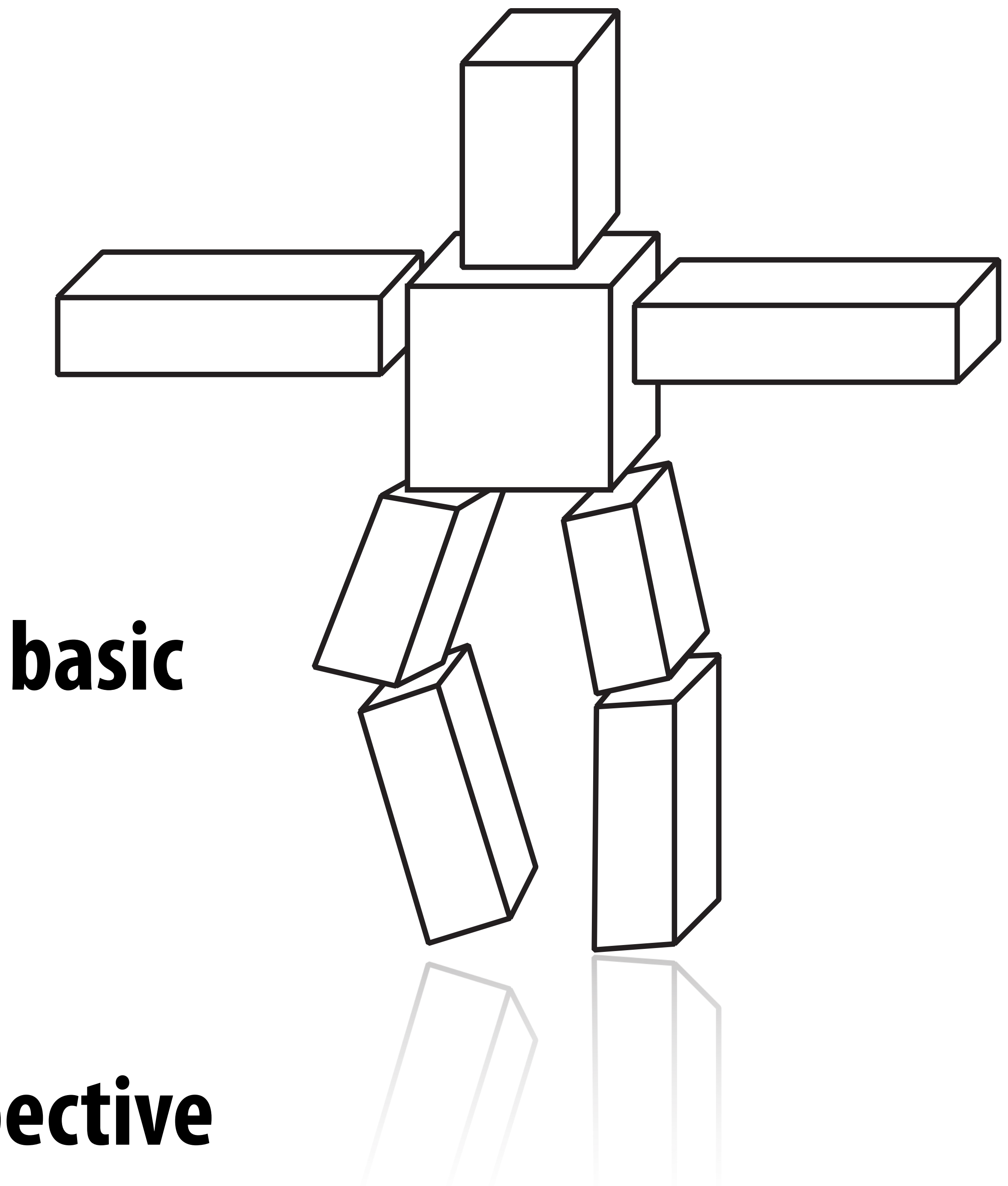
3. Perform rotation about z: $\mathrm{R}_{z,\theta}$
4. Rotate back to original coordinate space: $\mathrm{R}^{\mathbf{T}}_{\mathbf{uvw}}$

$$\mathbf{R}^{-1}_{uvw} = \mathbf{R}^{T}_{uvw} = \begin{bmatrix} \mathbf{u}_x & \mathbf{v}_x & \mathbf{w}_x \\ \mathbf{u}_y & \mathbf{v}_y & \mathbf{w}_y \\ \mathbf{u}_z & \mathbf{v}_x & \mathbf{w}_z \end{bmatrix}$$

$$\mathbf{R}_{\mathbf{w},\theta} = \mathbf{R}^{\mathbf{T}}_{\mathbf{uvw}}\mathbf{R}_{z,\theta}\mathbf{R}_{\mathbf{uvw}}$$

Tranformations summary

- Transformations can be interpreted as operations that move points in space
  - e.g., for modeling, animation
- Or as a change of coordinate system
  - e.g., screen and view transforms
- Construct complex transformations as compositions of basic transforms
- Homogeneous coordinate representation allows for expression of non-linear transforms (e.g., affine, perspective projection) as matrix operations (linear transforms) in higher-dimensional space
  - Matrix representation affords simple implementation and efficient composition



Further Reading

- **Basic transforms are nicely covered here (<u>Real Time Rendering</u> -- Chapter 4. by T. Akenine Moller, E. Haines, N. Hoffman)**

What you should know

- Create 2D and 3D transformation matrices to perform specific scale, shear, rotation, reflection, and translation operations
- Compose transformations to achieve compound effects
- Rotate an object about a fixed point
- Rotate an object about a given axis
- Create an orthonormal basis given a single vector
- Understand the equivalence of [x y 1] and [wx wy w] vectors
- Explain/illustrate how translations in 2D (x, y) are a shear operation in the homogeneous coordinate space (x, y, w)