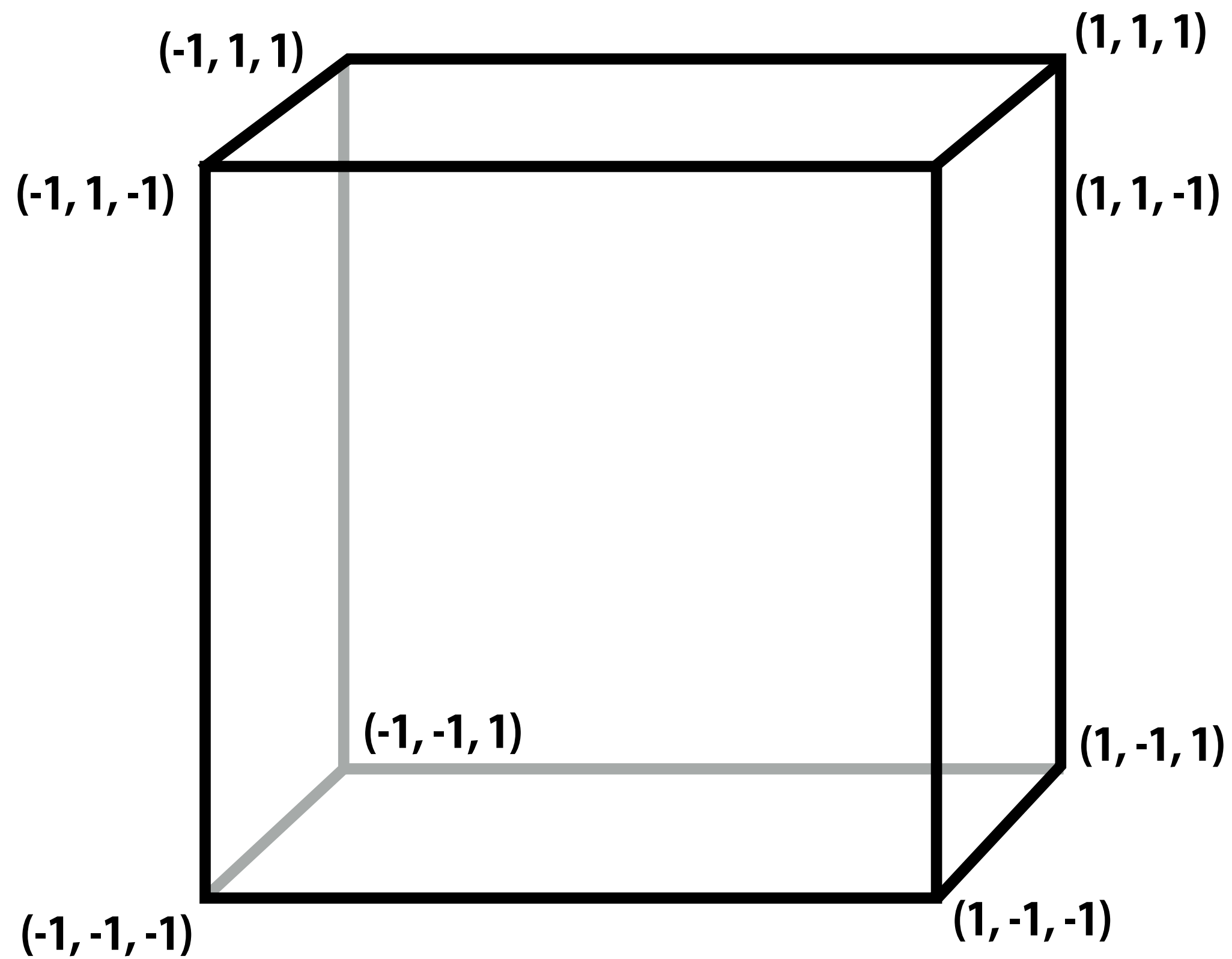


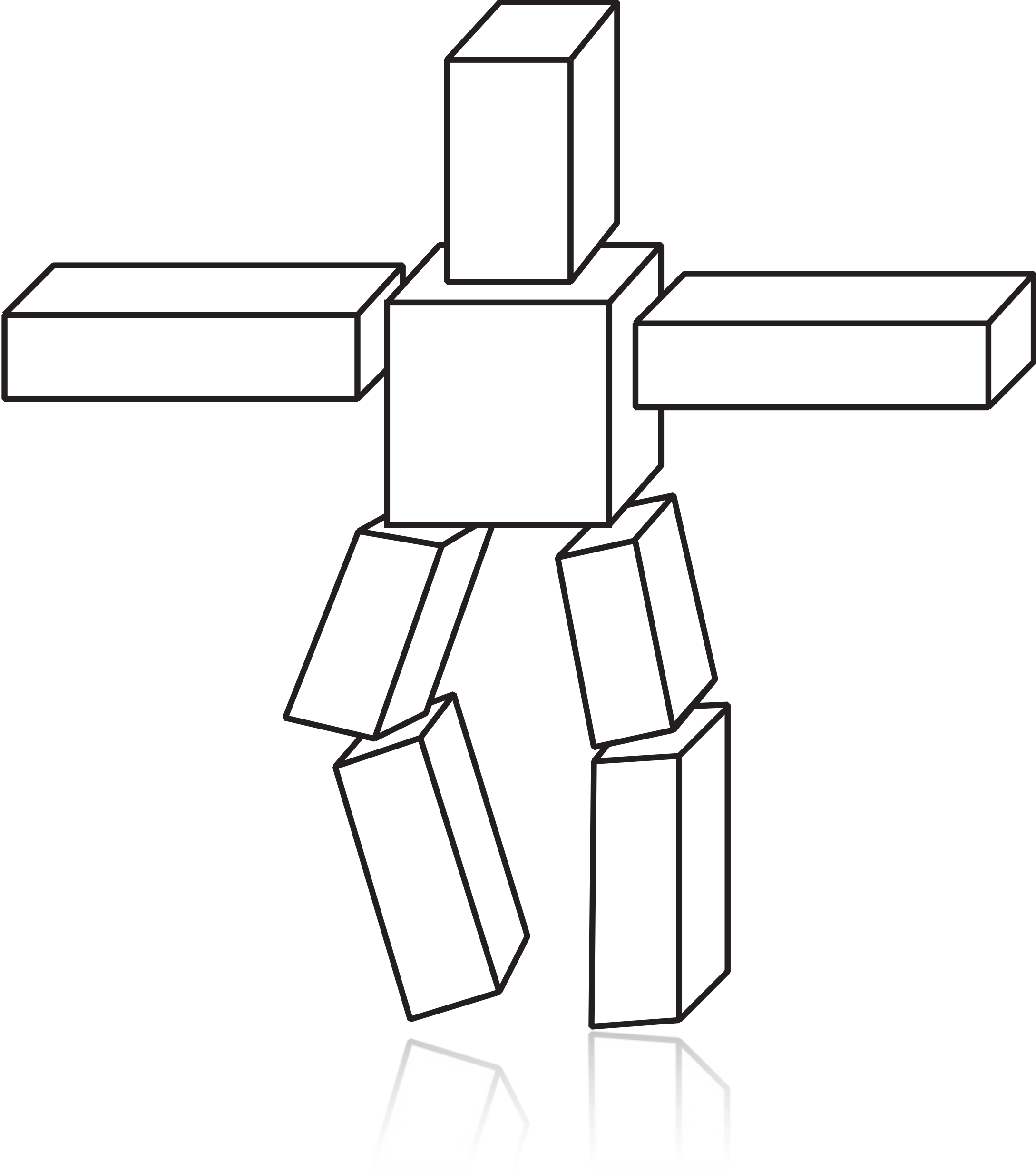
Transformations

Computer Graphics
CMU 15-462/15-662, Spring 2018

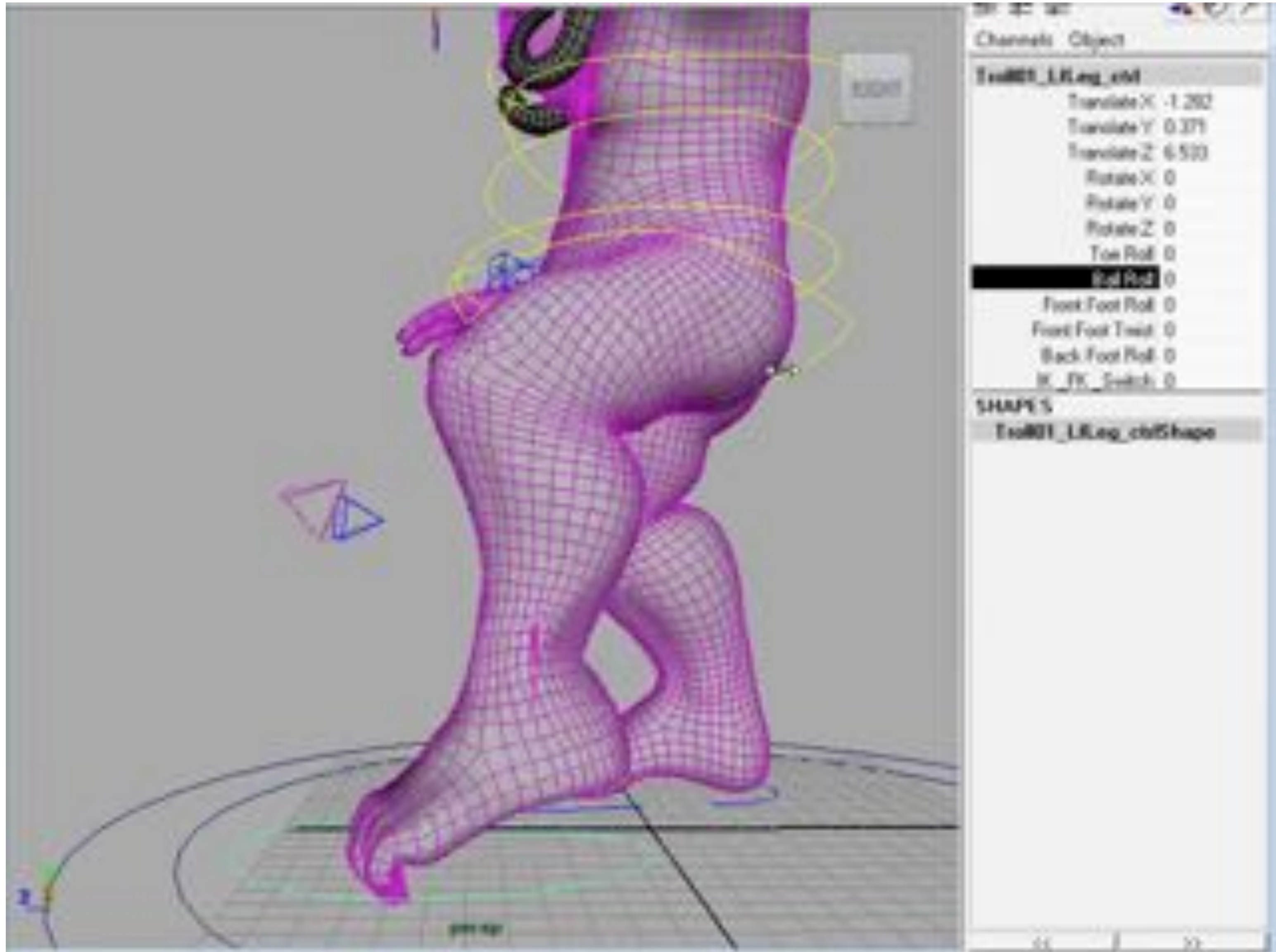
Cube



Cube man



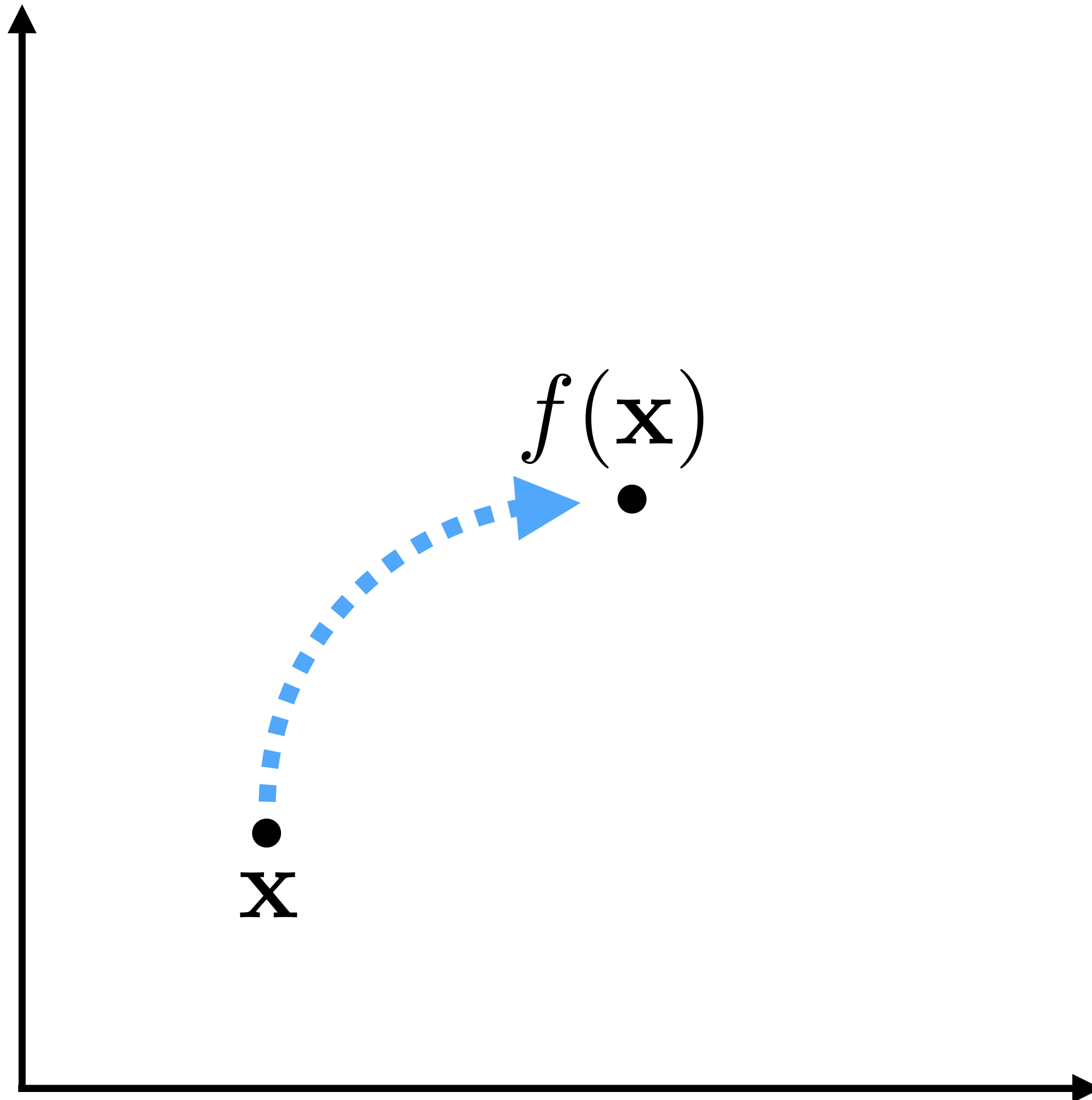
Transformations in Rigging



Transformations in Instancing



Basic idea: f transforms \mathbf{x} to $f(\mathbf{x})$



What can we do with linear transformations?

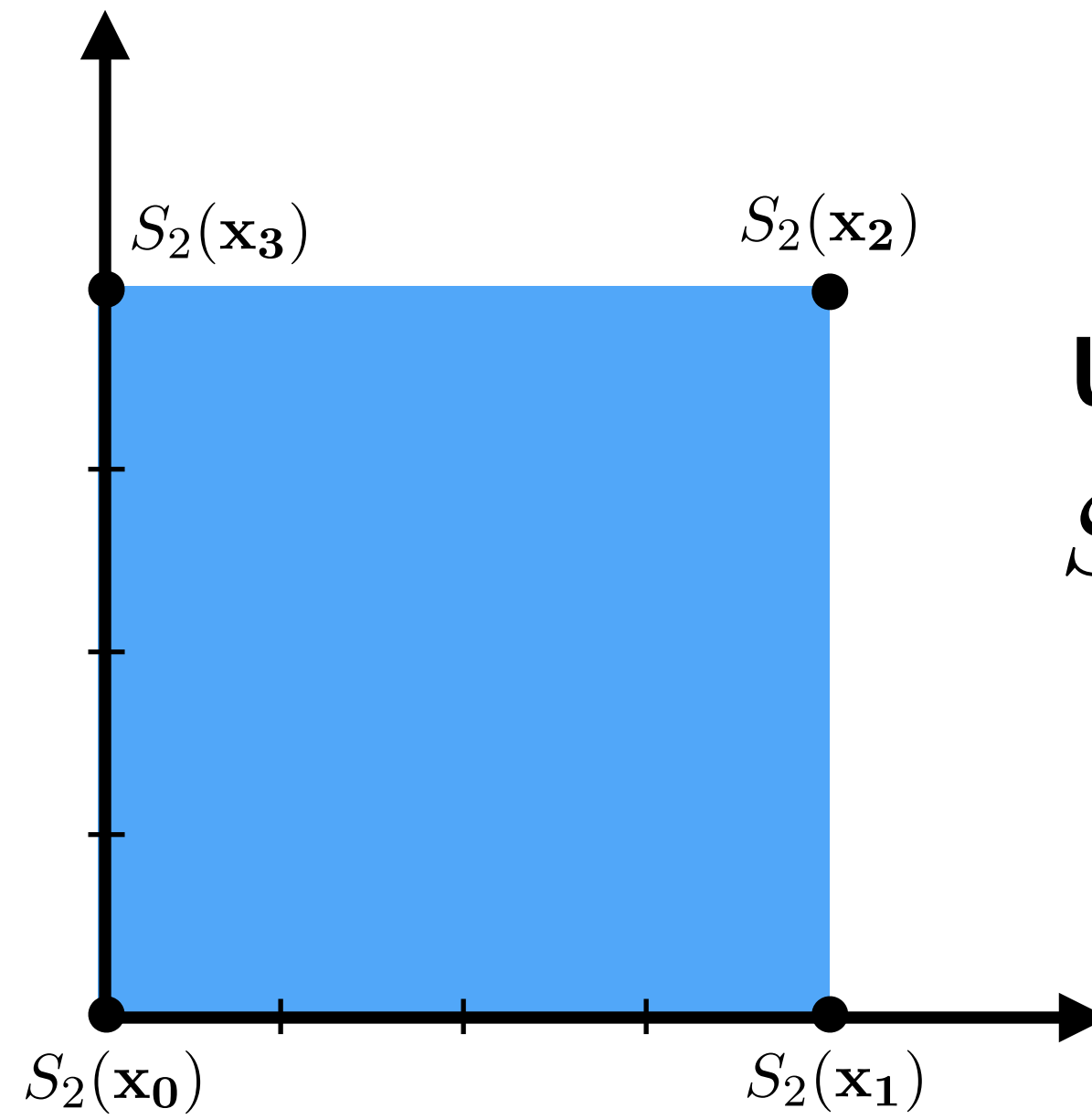
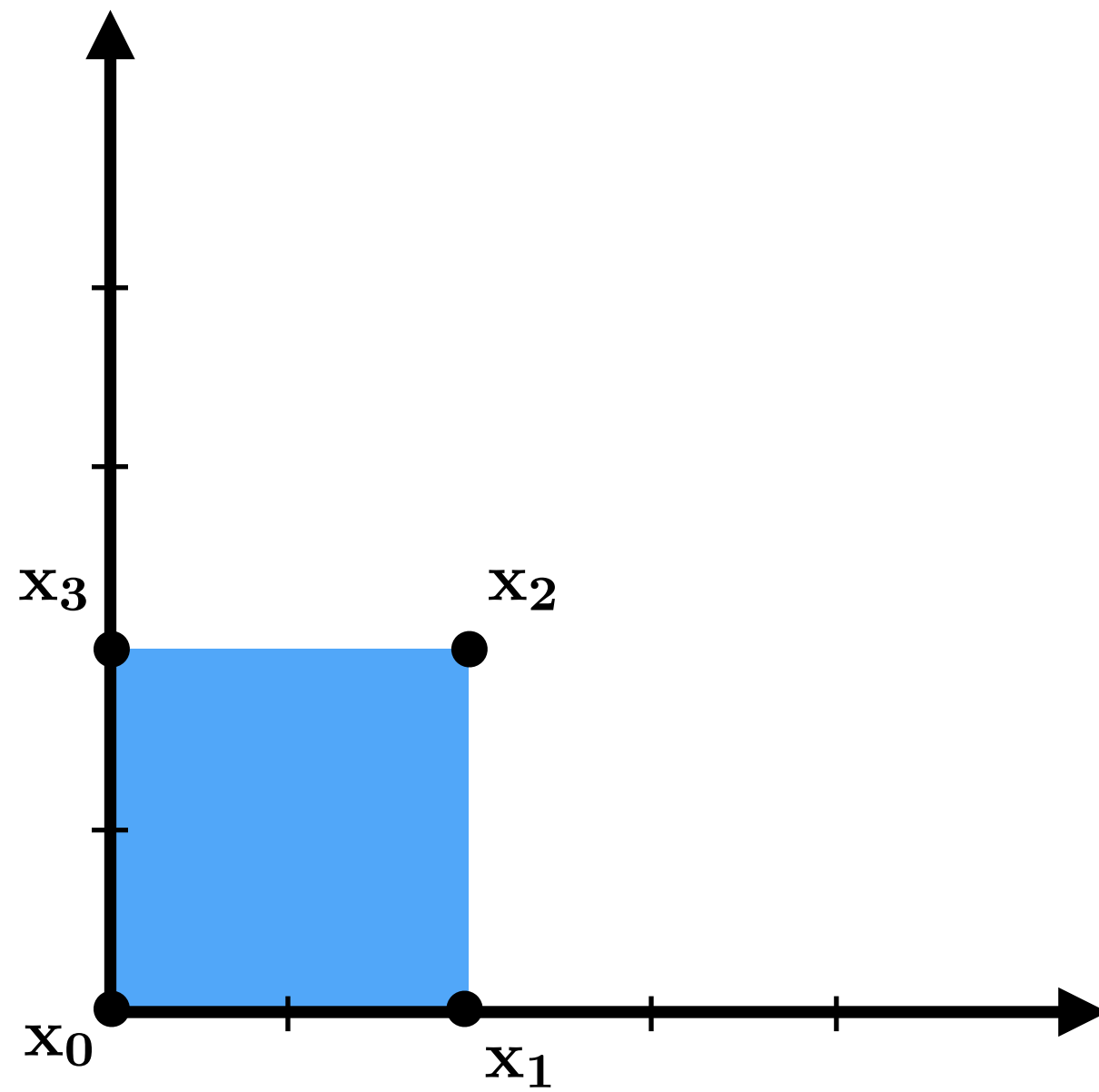
- **(What did linear mean?)**

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

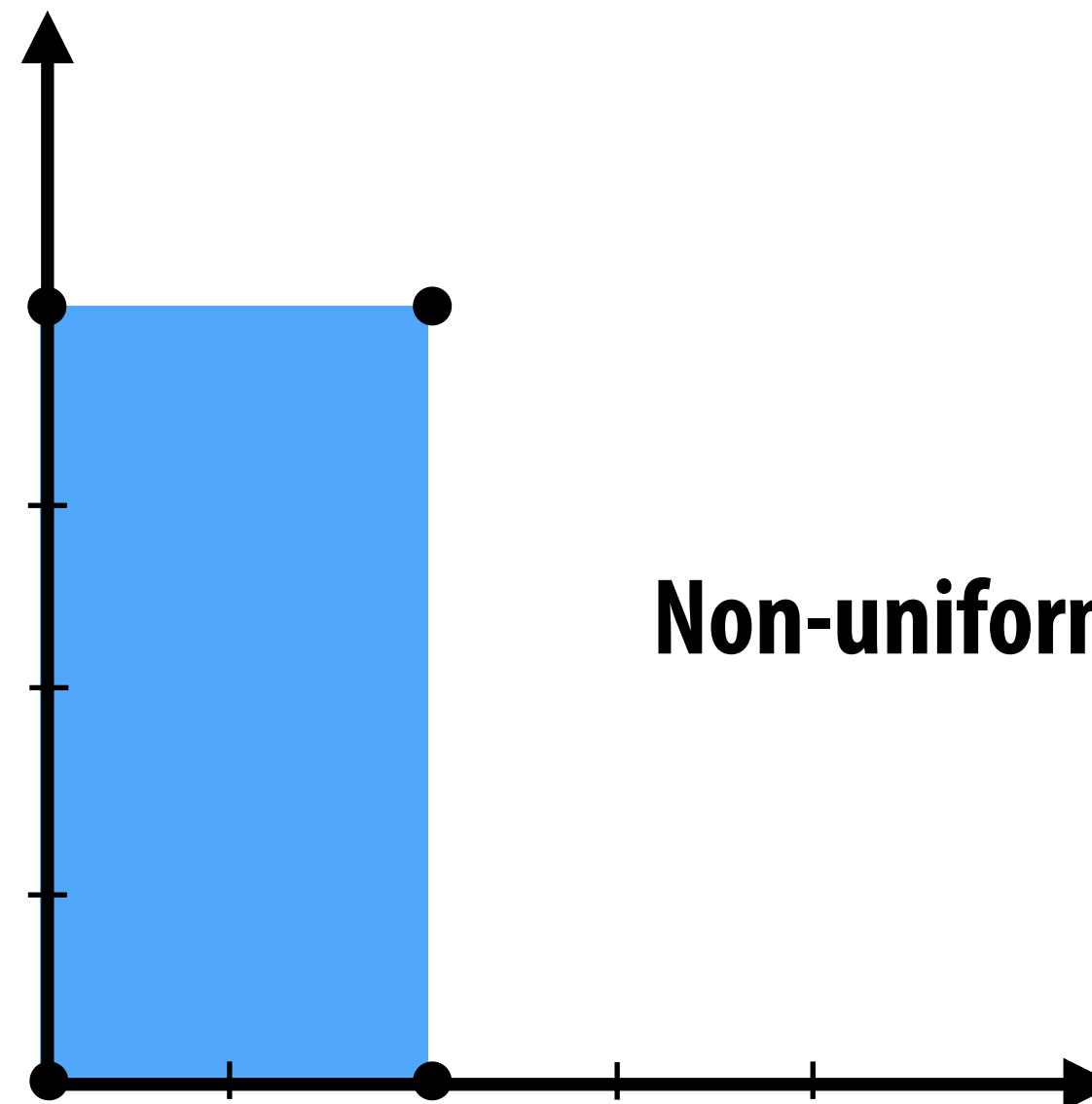
$$f(a\mathbf{x}) = af(\mathbf{x})$$

- **Cheap to compute**
- **Composition of linear transformations is linear**
 - **Leads to uniform representation of transformations**
 - **E.g., in graphics card (GPU) or graphics APIs**

Scale



Uniform scale:
 $S_a(\mathbf{x}) = a\mathbf{x}$



Non-uniform scale??

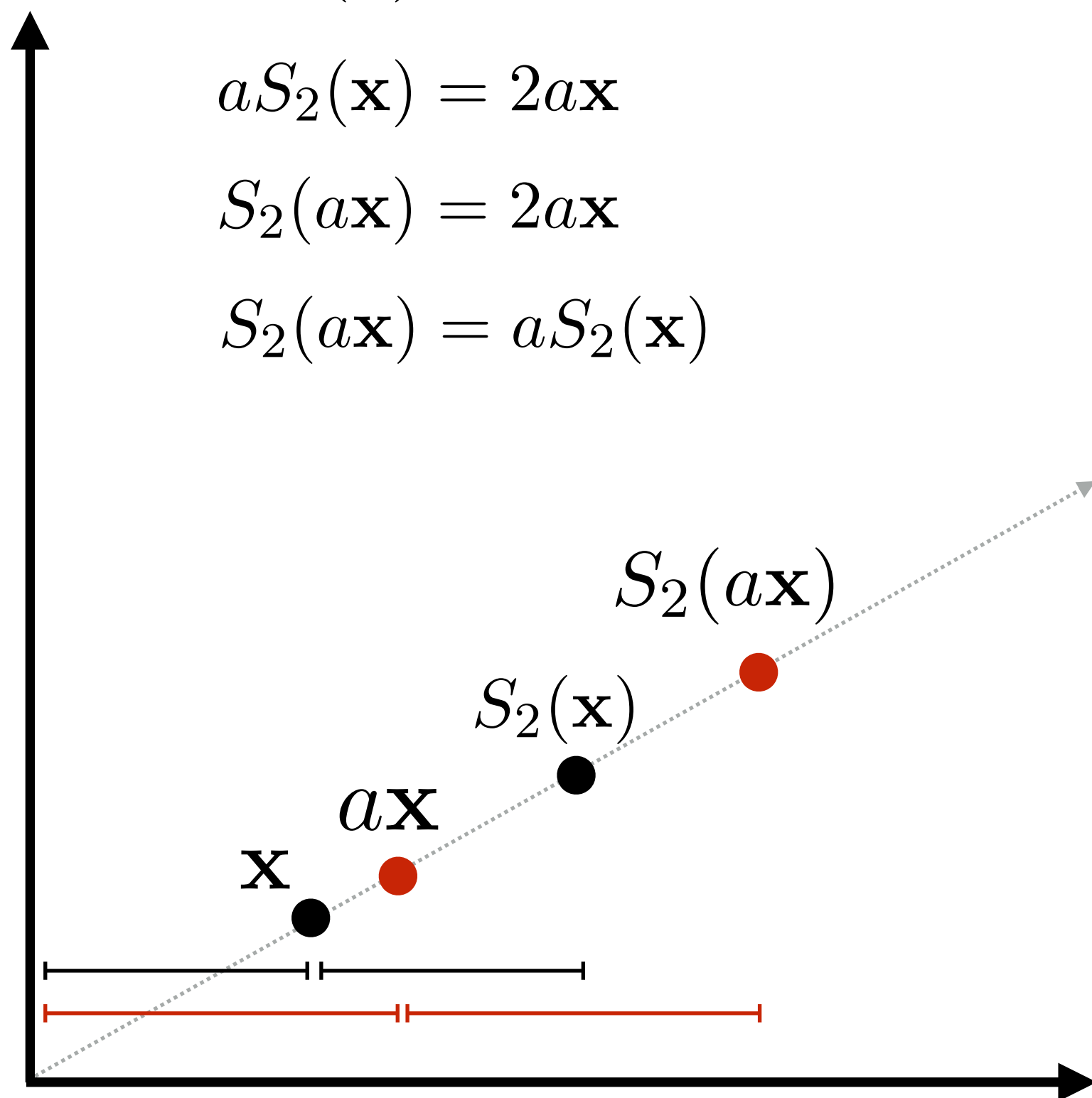
Is scale a linear transform?

$$S_2(\mathbf{x}) = 2\mathbf{x}$$

$$aS_2(\mathbf{x}) = 2a\mathbf{x}$$

$$S_2(a\mathbf{x}) = 2a\mathbf{x}$$

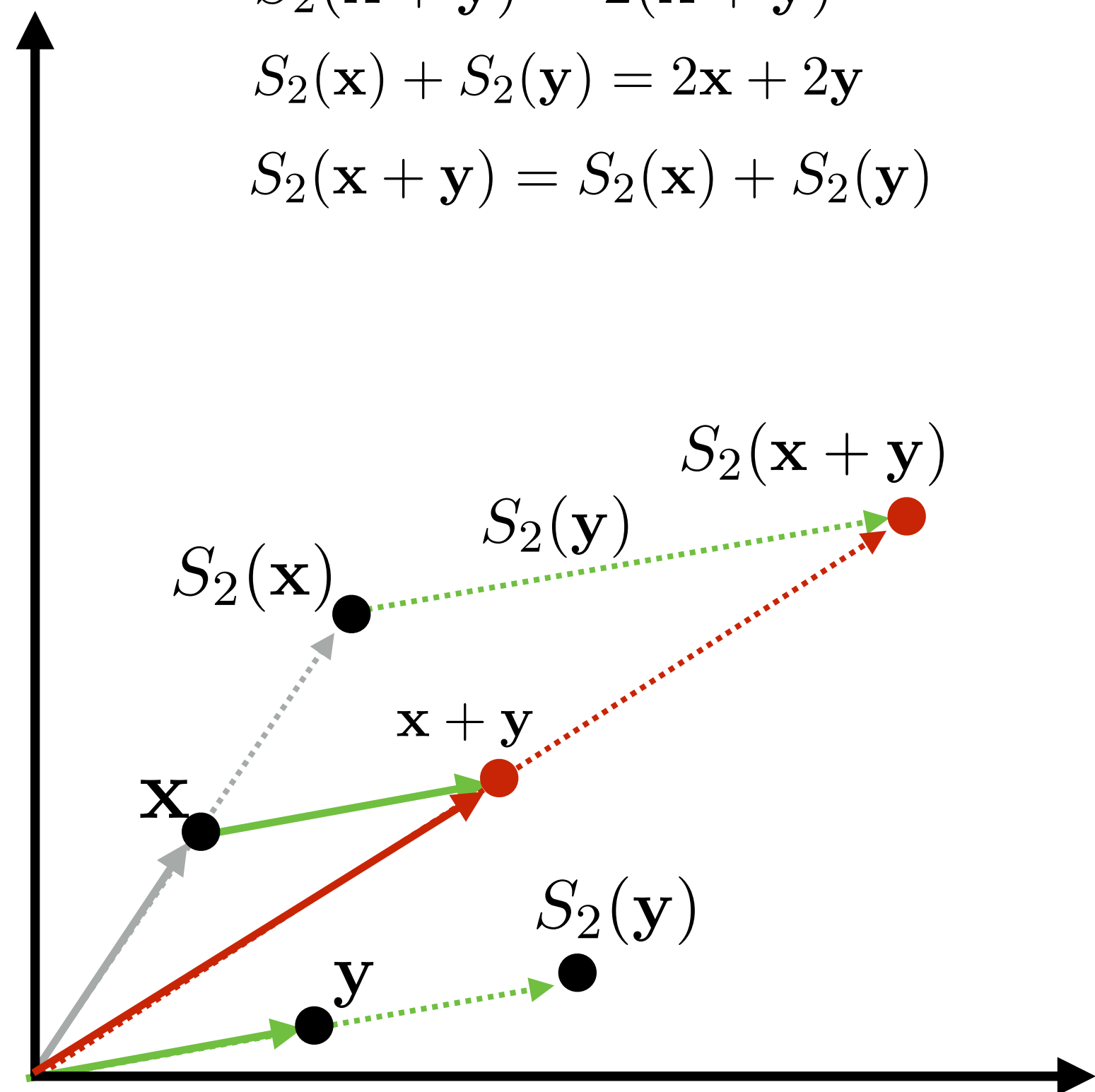
$$S_2(a\mathbf{x}) = aS_2(\mathbf{x})$$



$$S_2(\mathbf{x} + \mathbf{y}) = 2(\mathbf{x} + \mathbf{y})$$

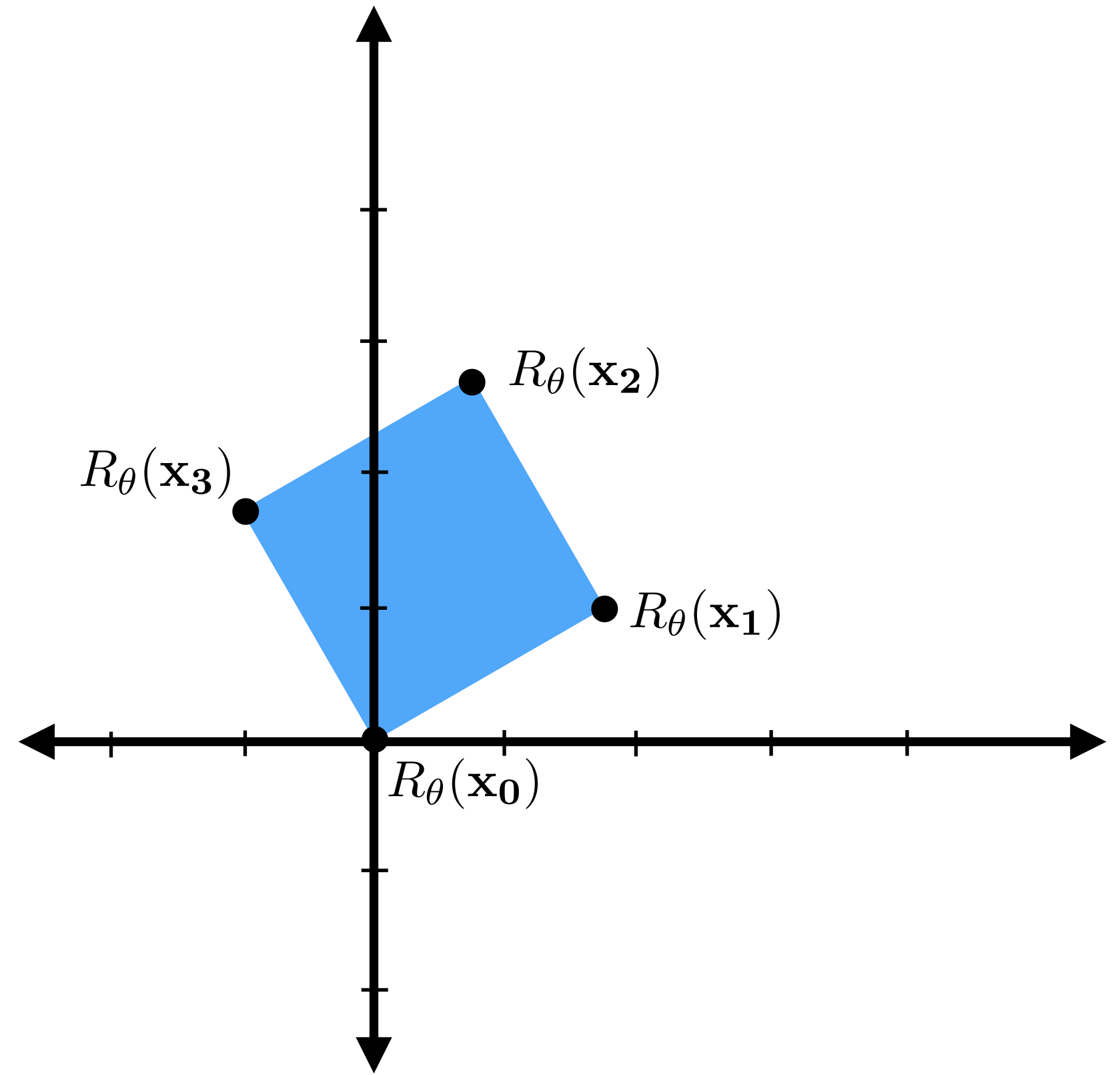
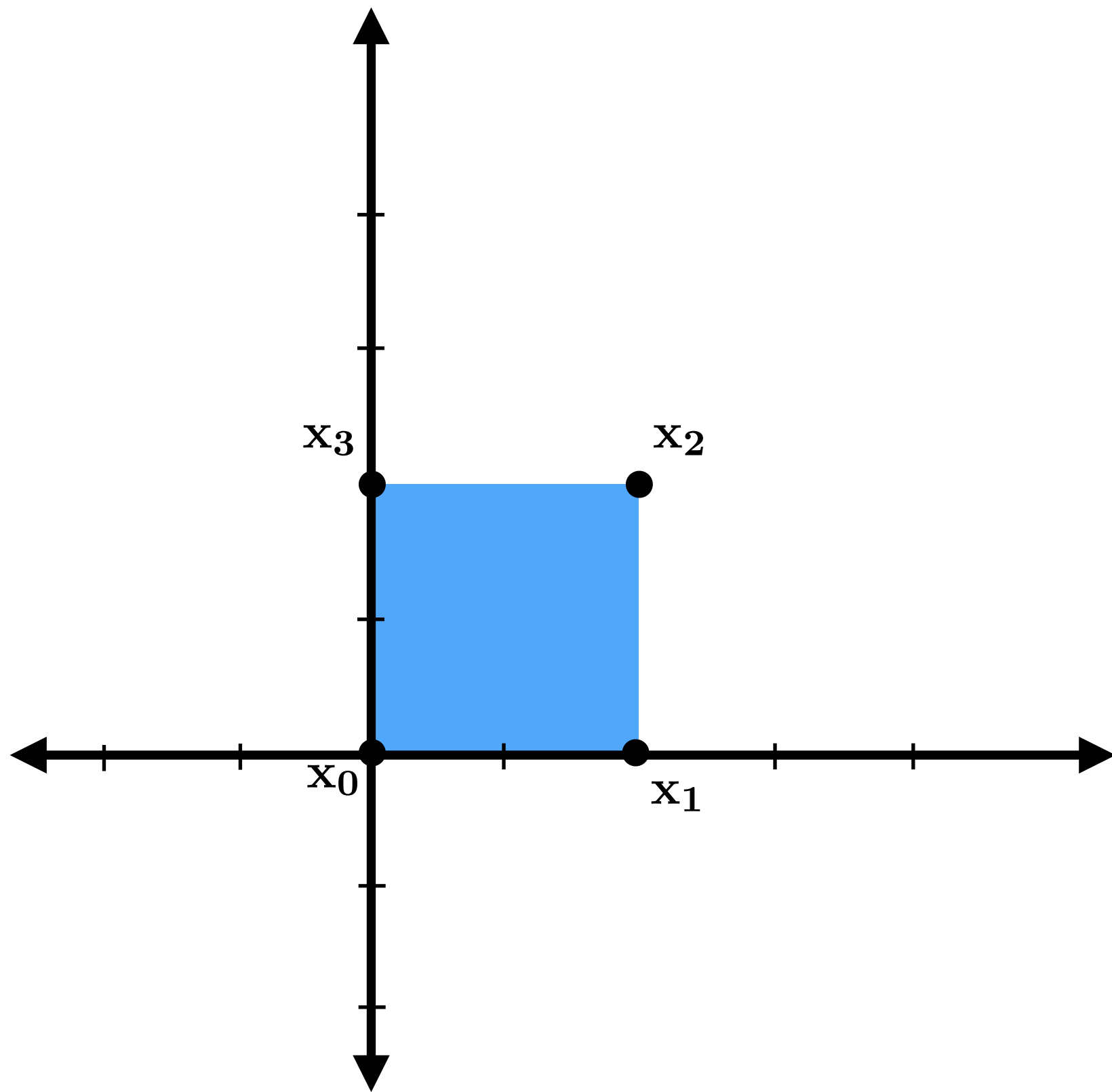
$$S_2(\mathbf{x}) + S_2(\mathbf{y}) = 2\mathbf{x} + 2\mathbf{y}$$

$$S_2(\mathbf{x} + \mathbf{y}) = S_2(\mathbf{x}) + S_2(\mathbf{y})$$



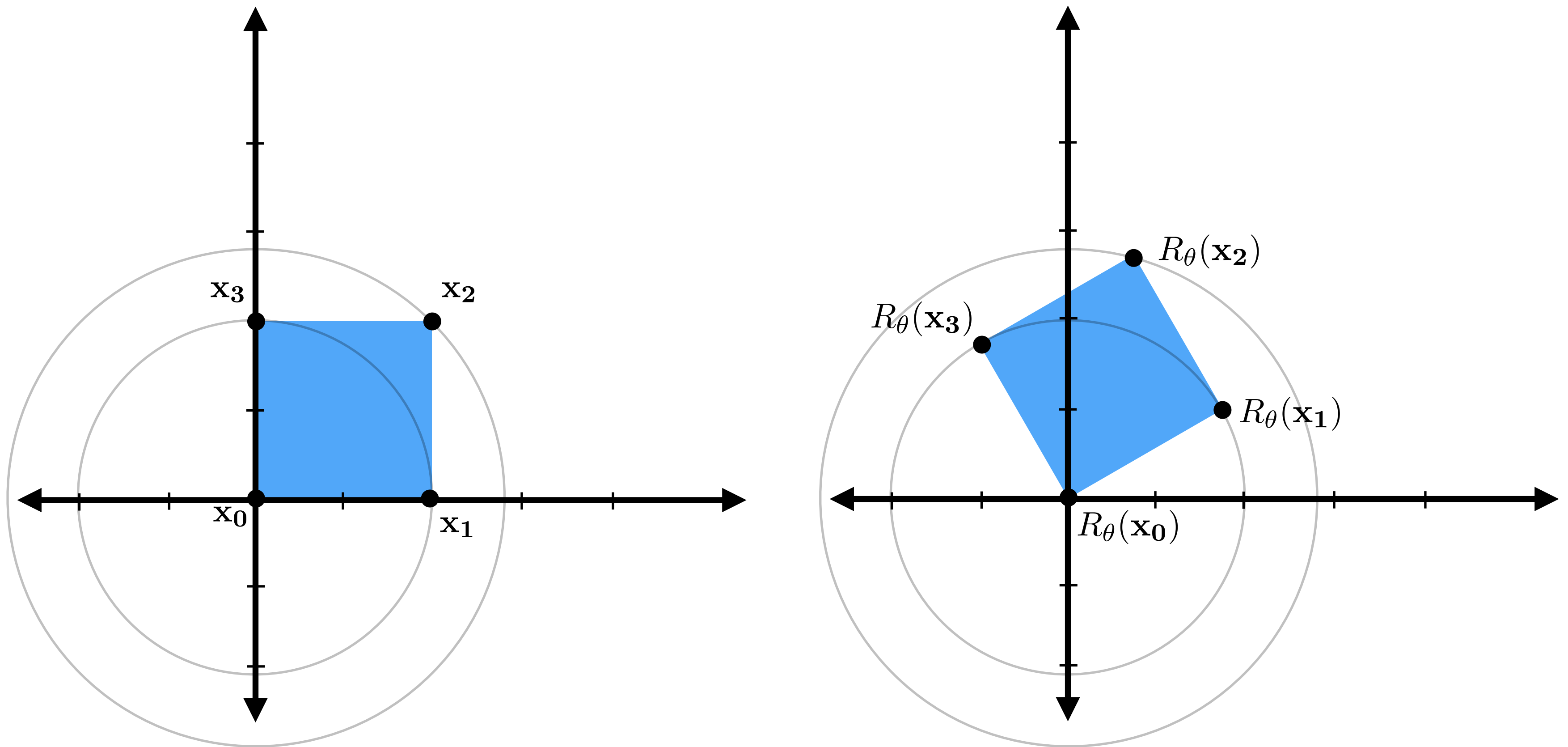
Yes!

Rotation



R_θ = rotate counter-clockwise by θ

Rotation as Circular Motion

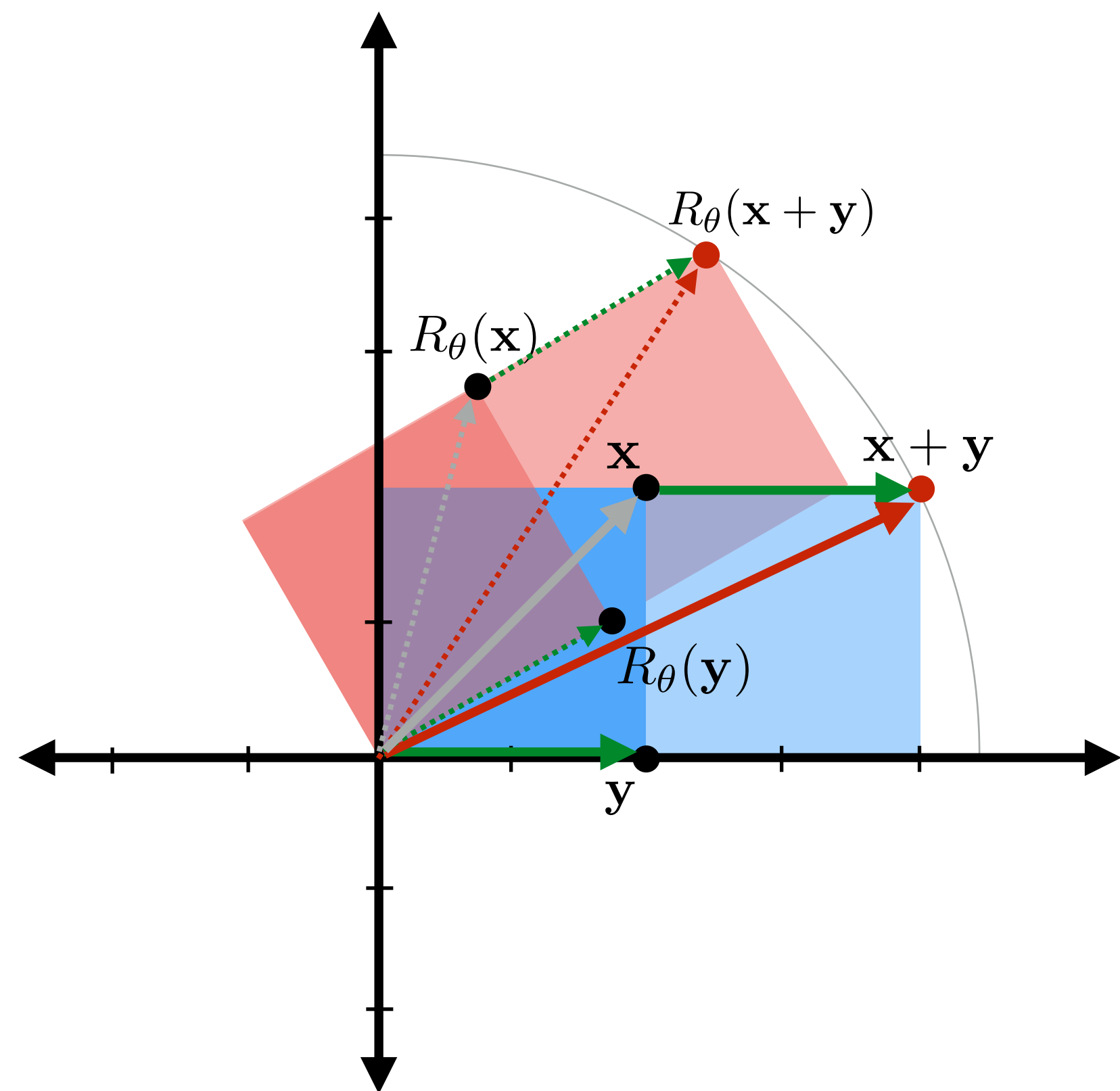
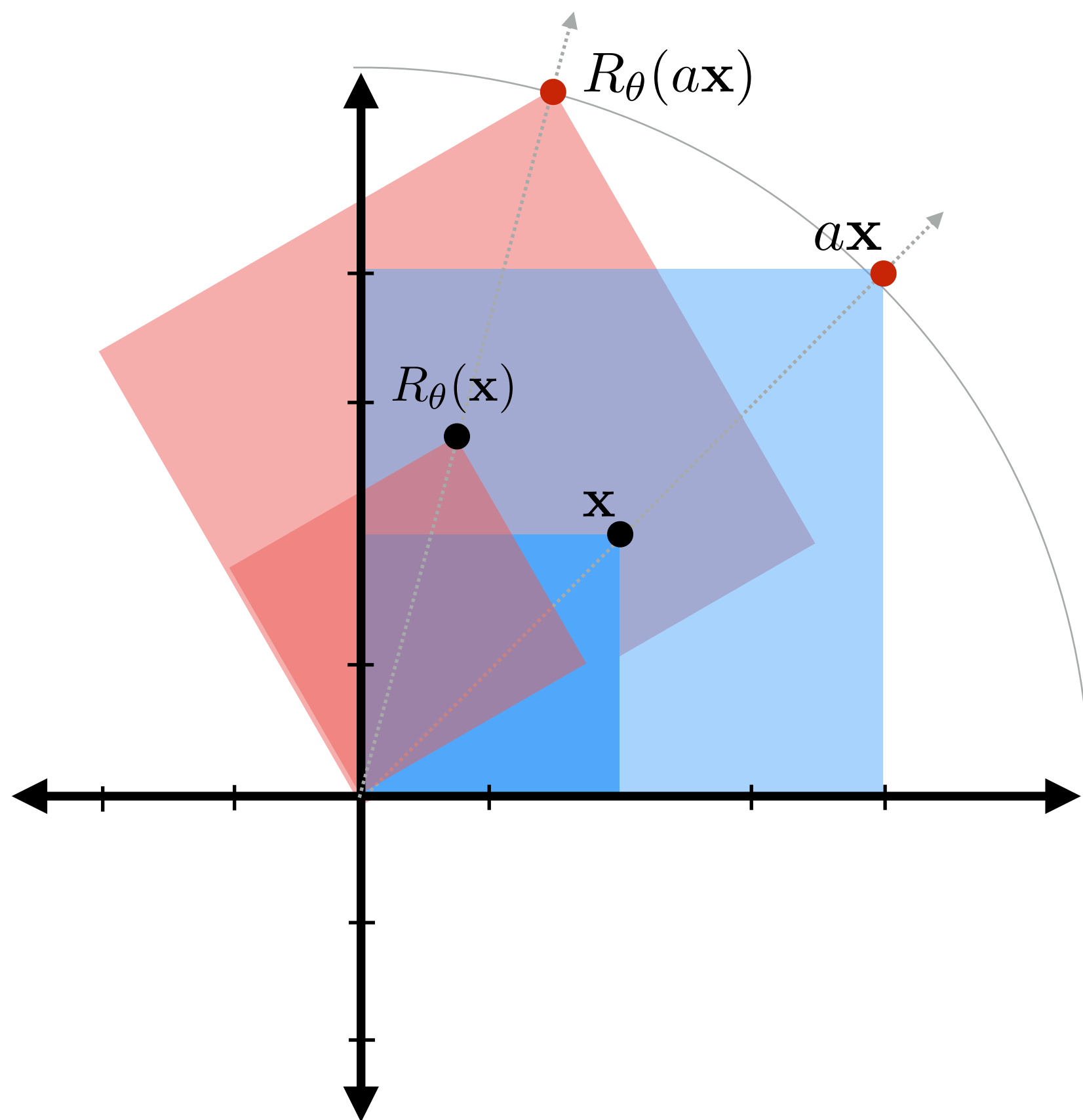


R_θ = rotate counter-clockwise by θ

As angle changes, points move along circular trajectories.

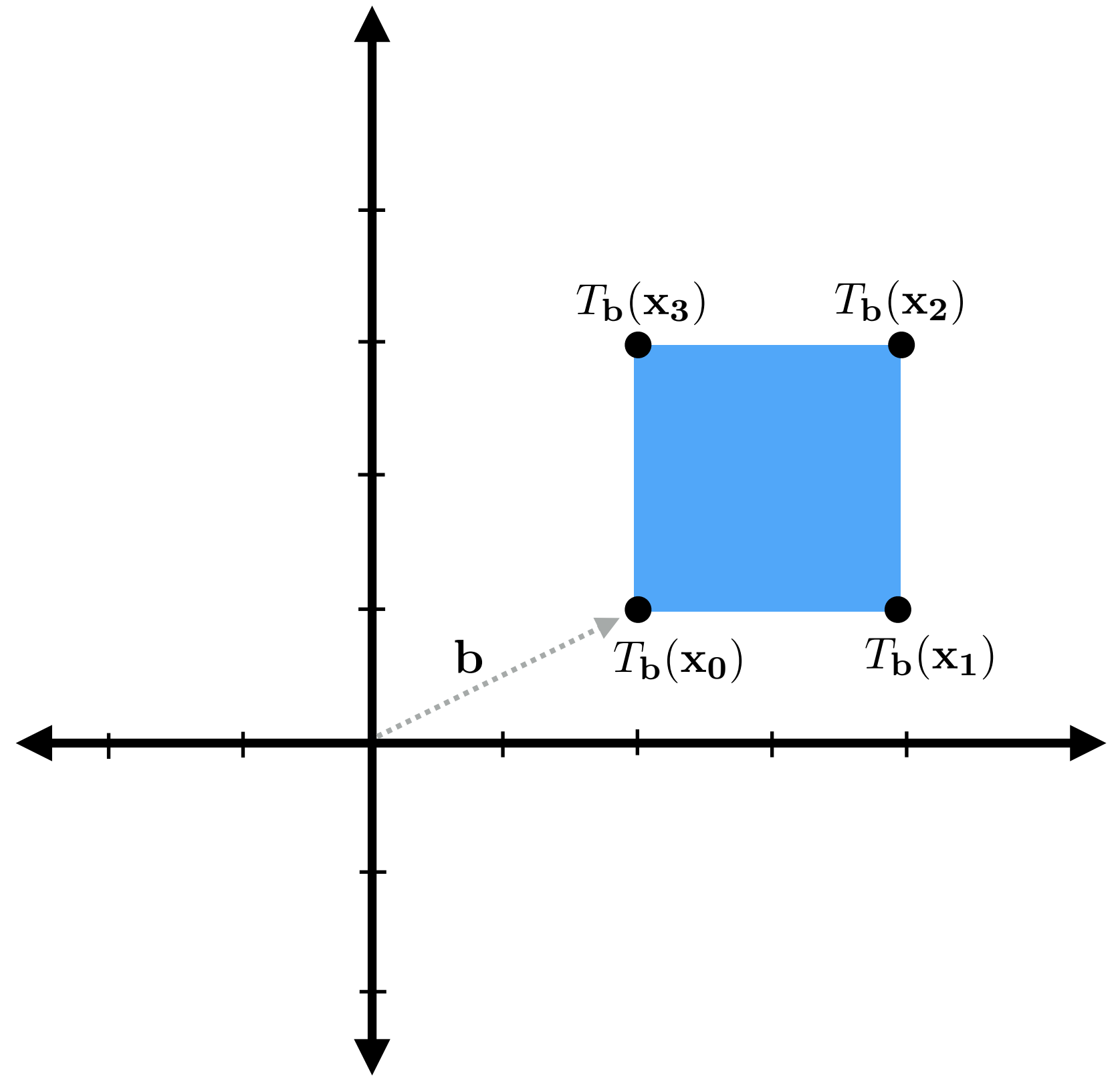
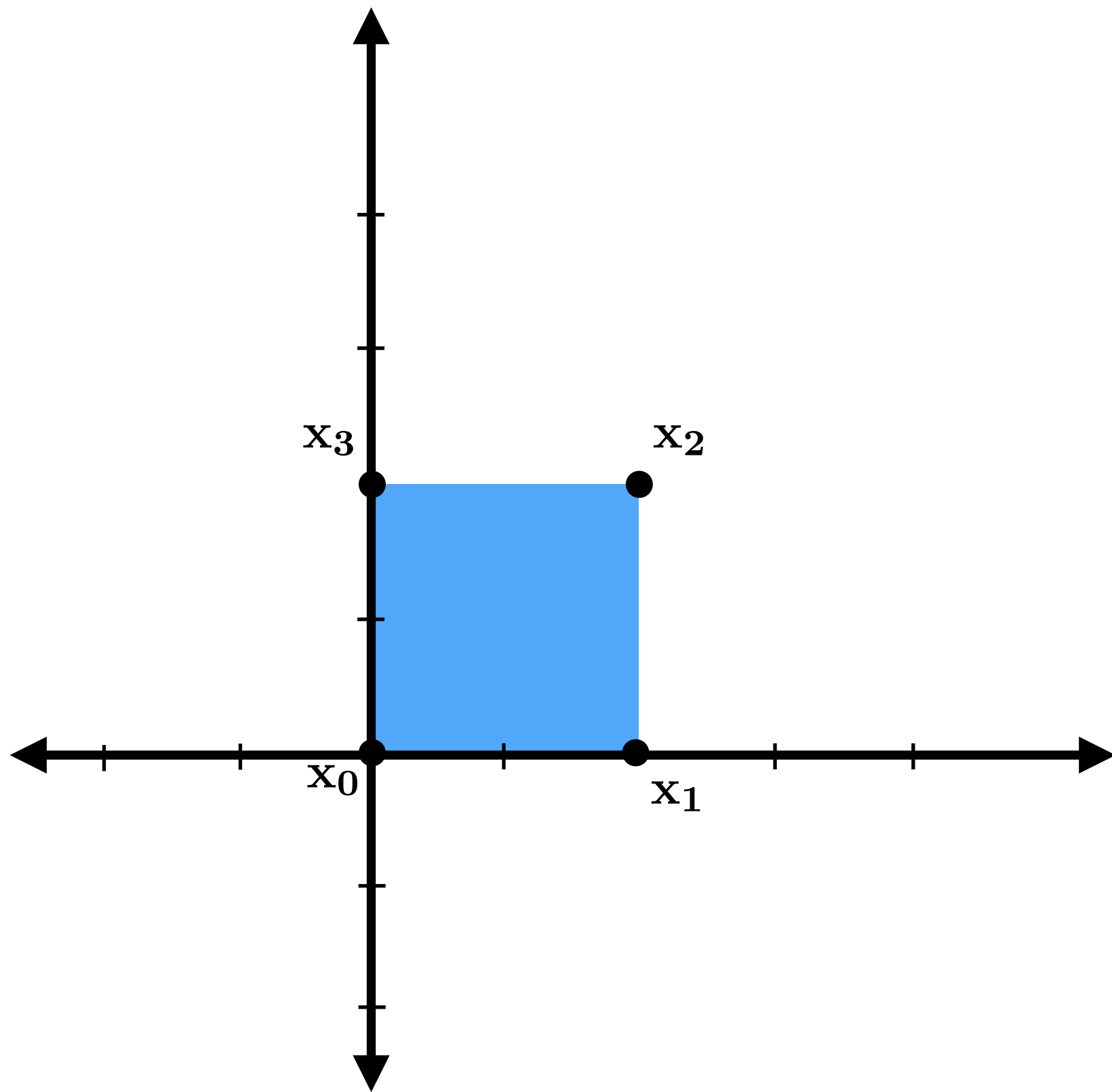
Hence, rotations preserve length of vectors: $|R_\theta(\mathbf{x})| = |\mathbf{x}|$

Is rotation linear?



Yes!

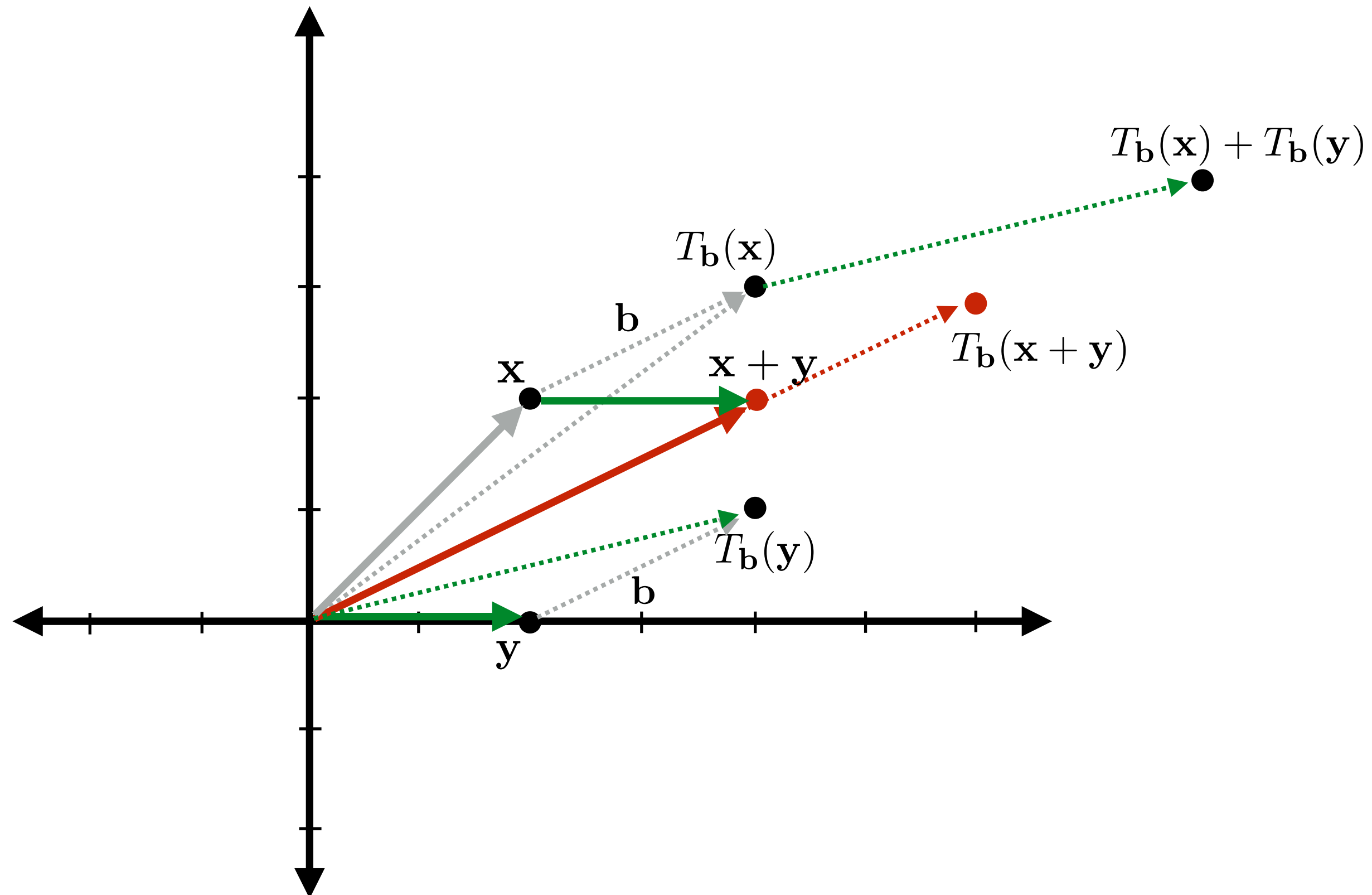
Translation



T_b — “translate by \mathbf{b} ”

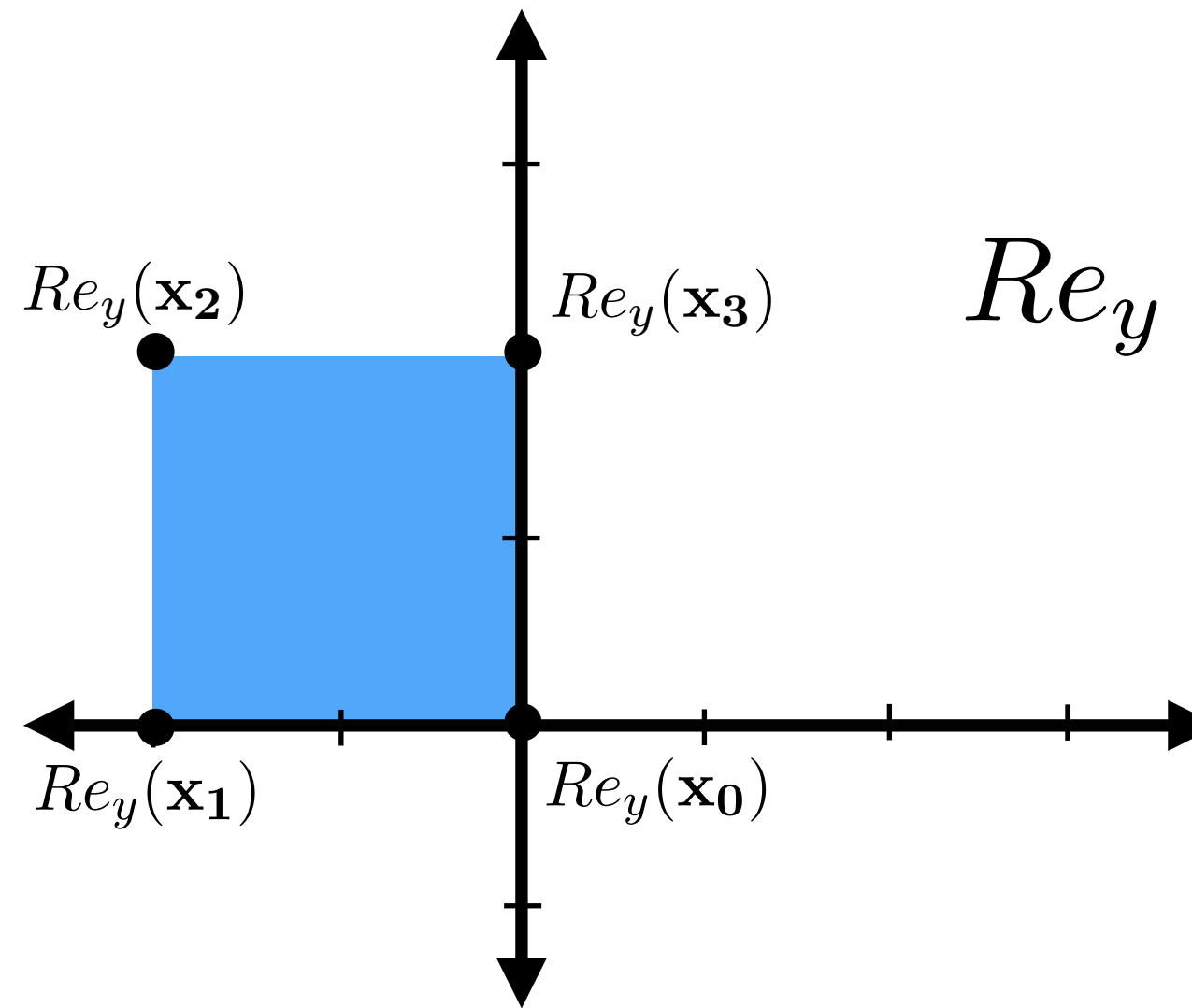
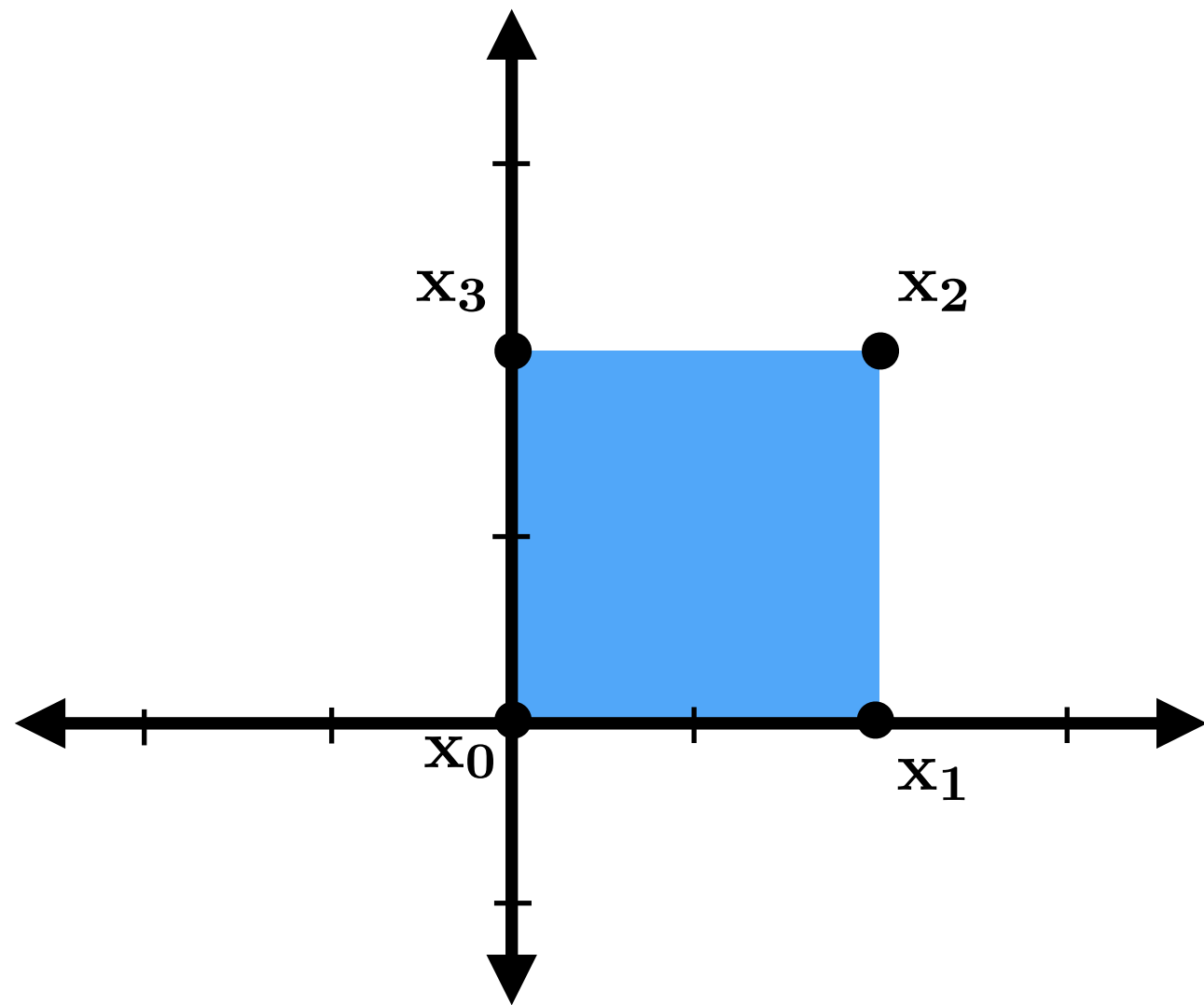
$$T_b(\mathbf{x}) = \mathbf{x} + \mathbf{b}$$

Is translation linear?

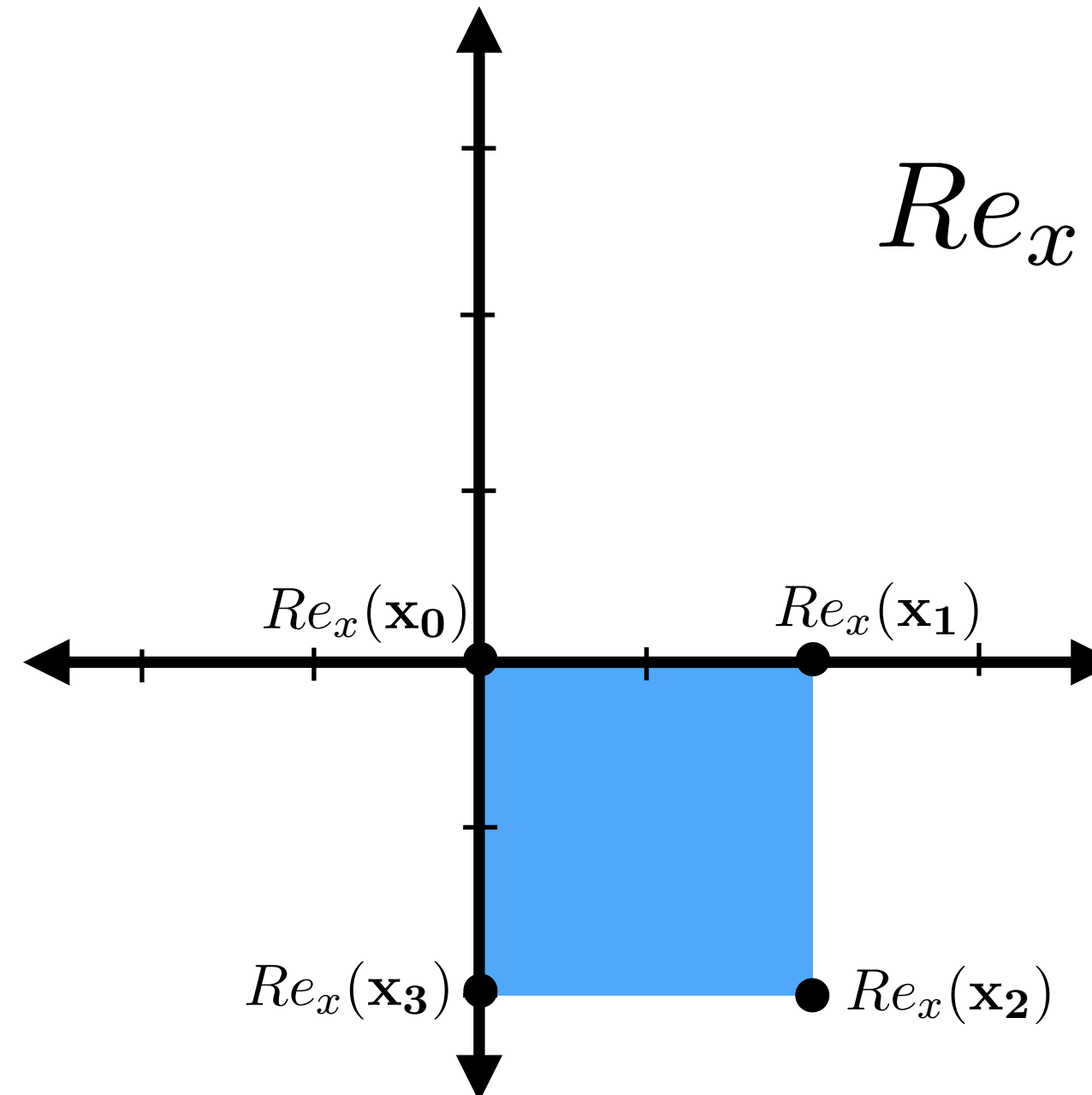


No. Translation is affine.

Reflection

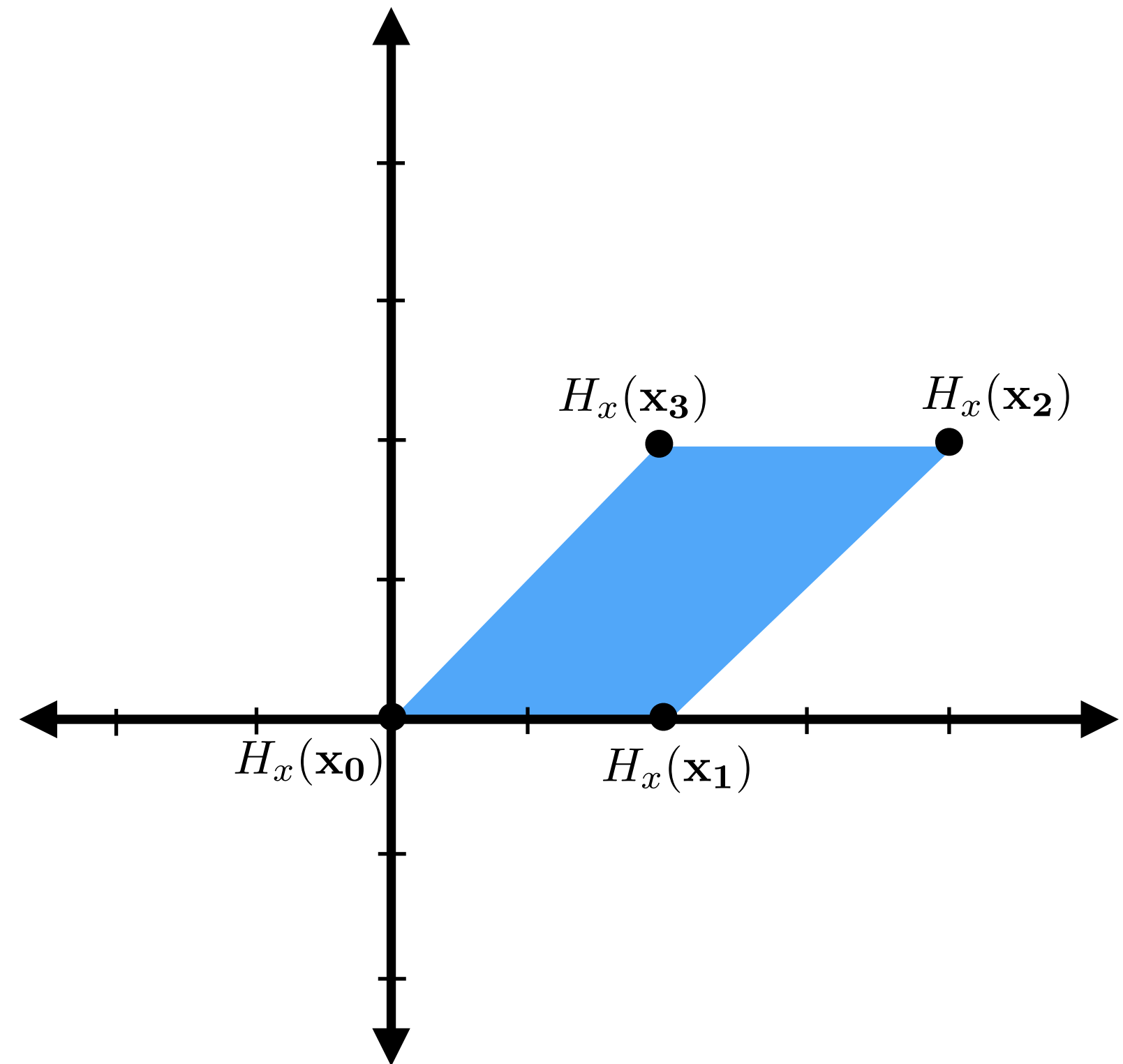
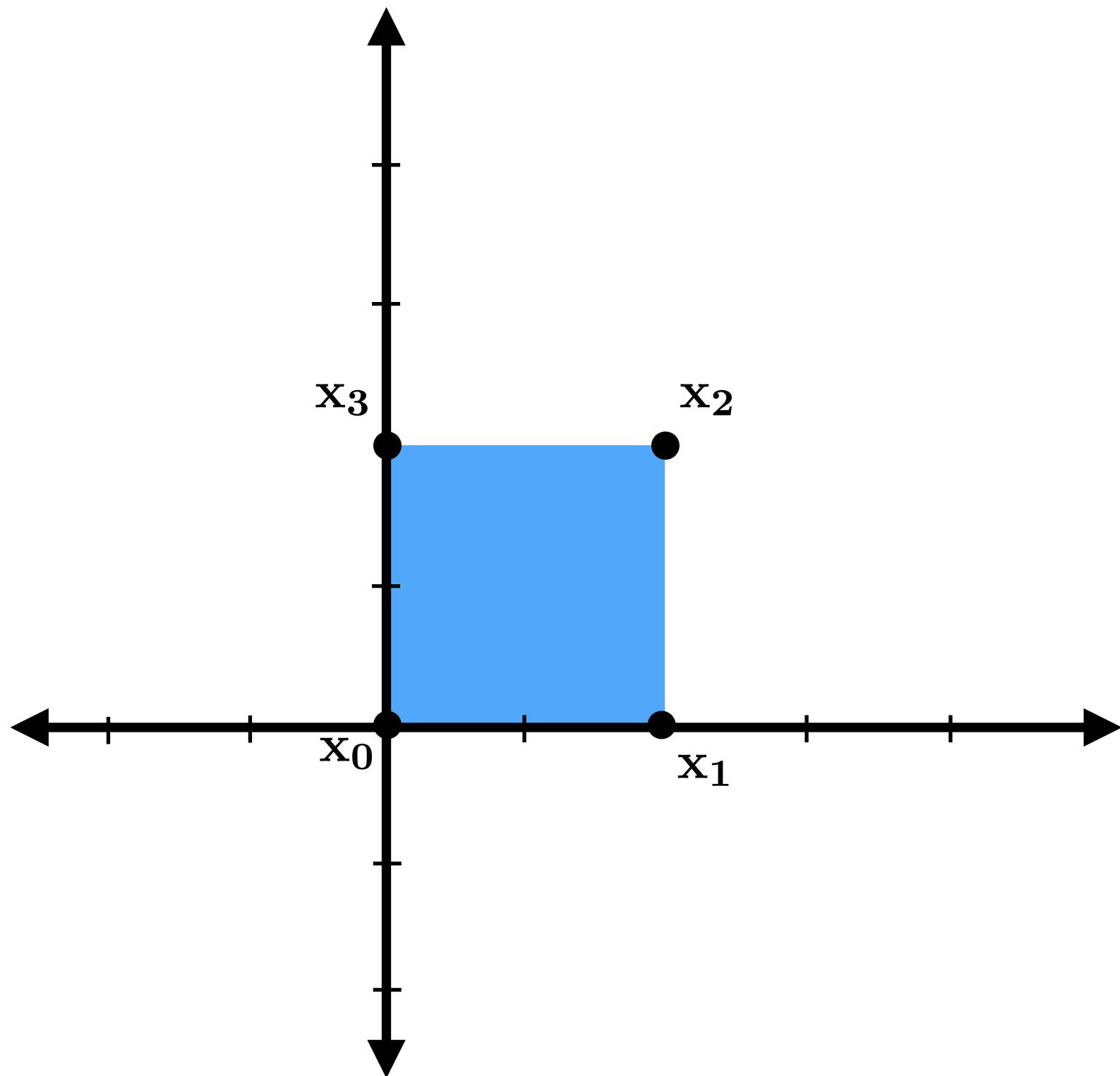


$Re_y =$ reflection about y

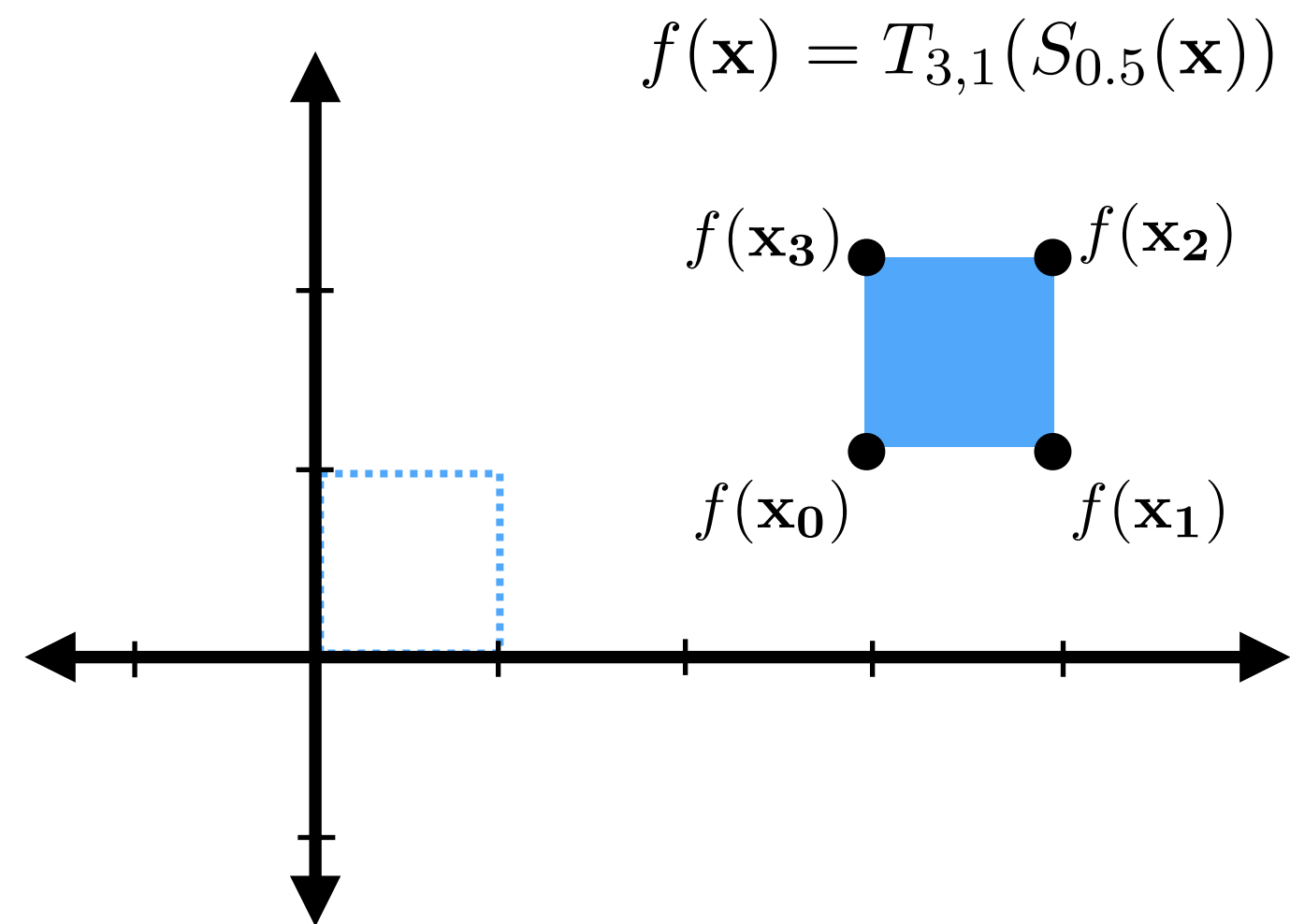
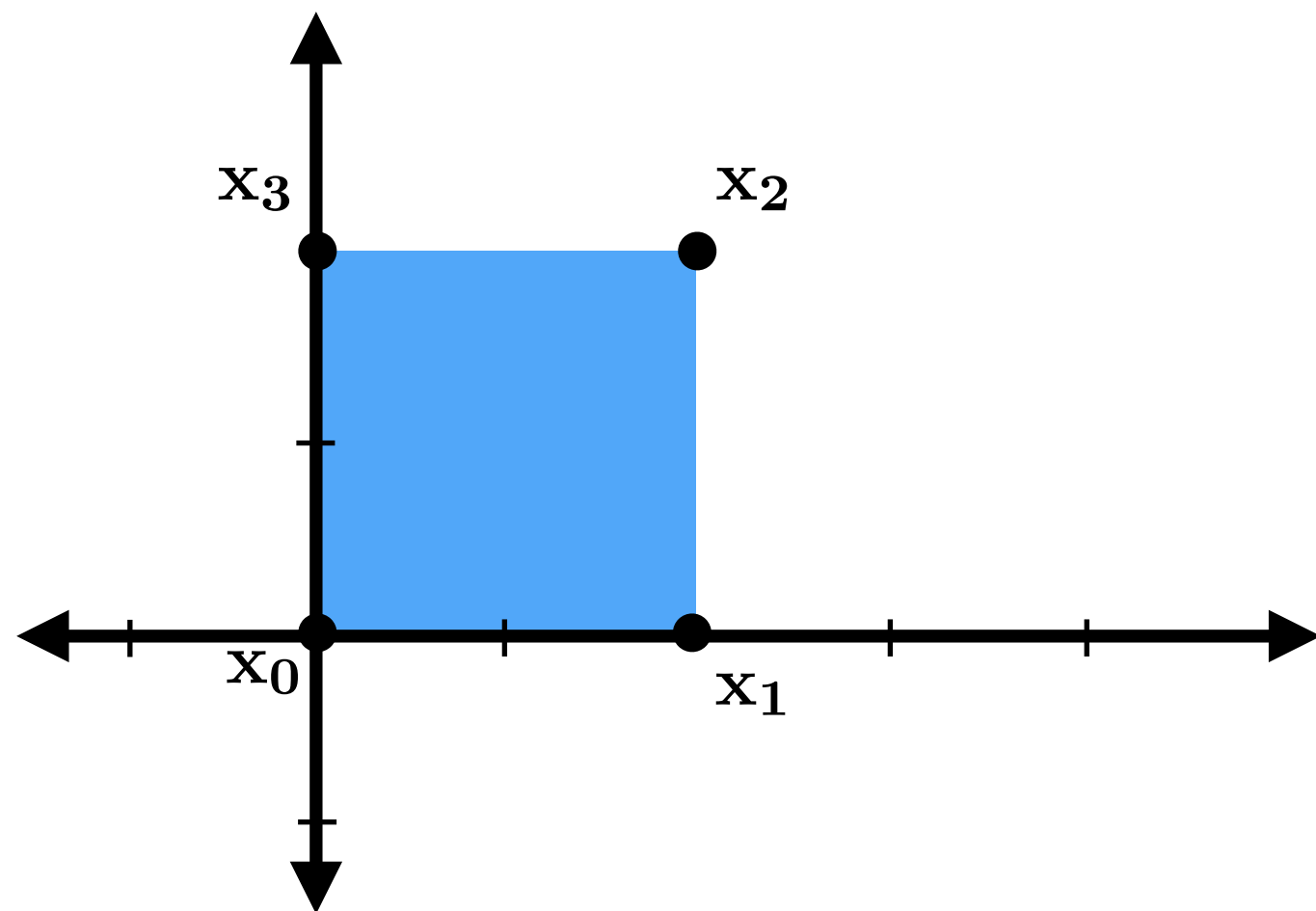


$Re_x =$ reflection about x

Shear (in x direction)



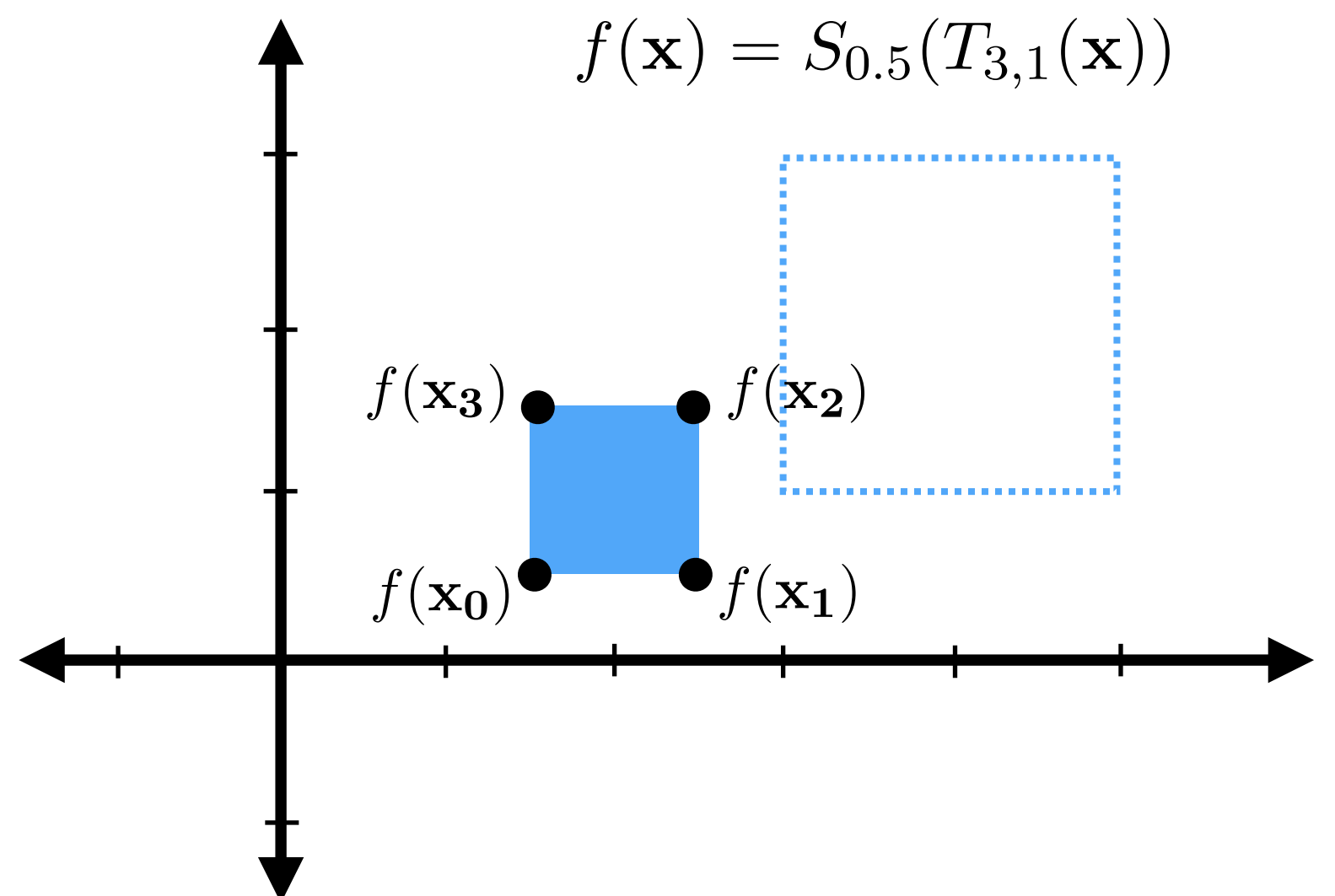
Compose basic transformations to construct more complicated ones



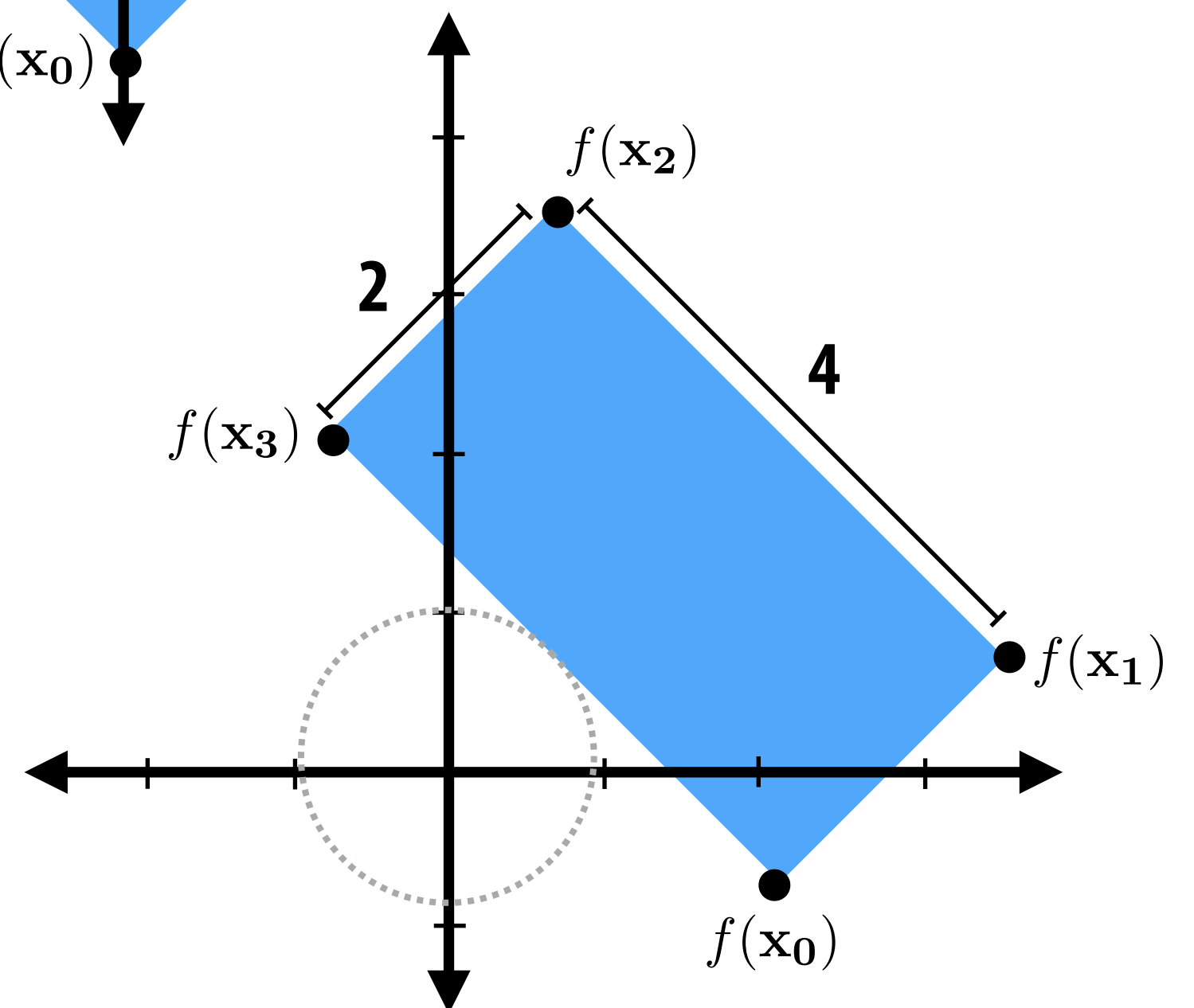
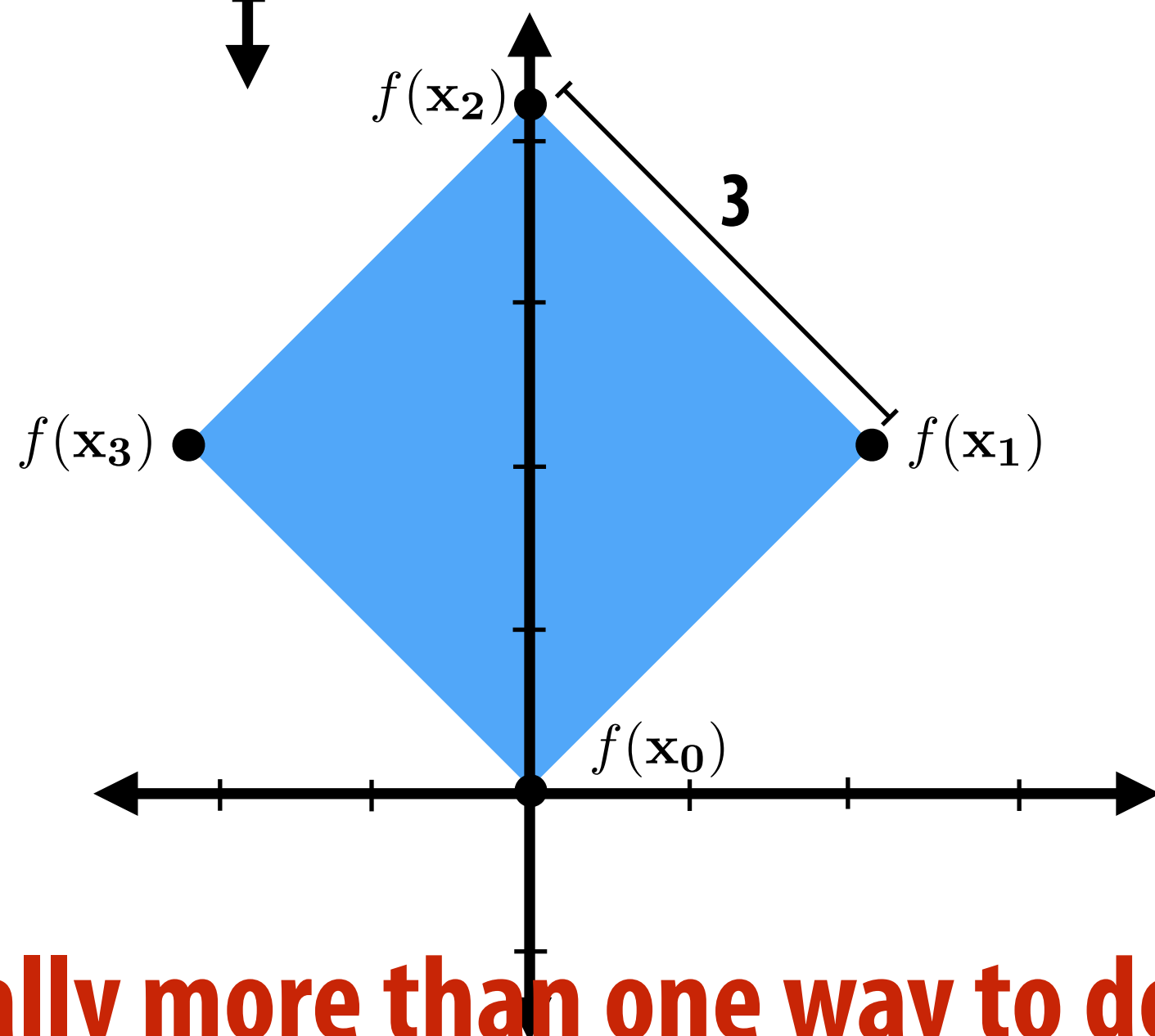
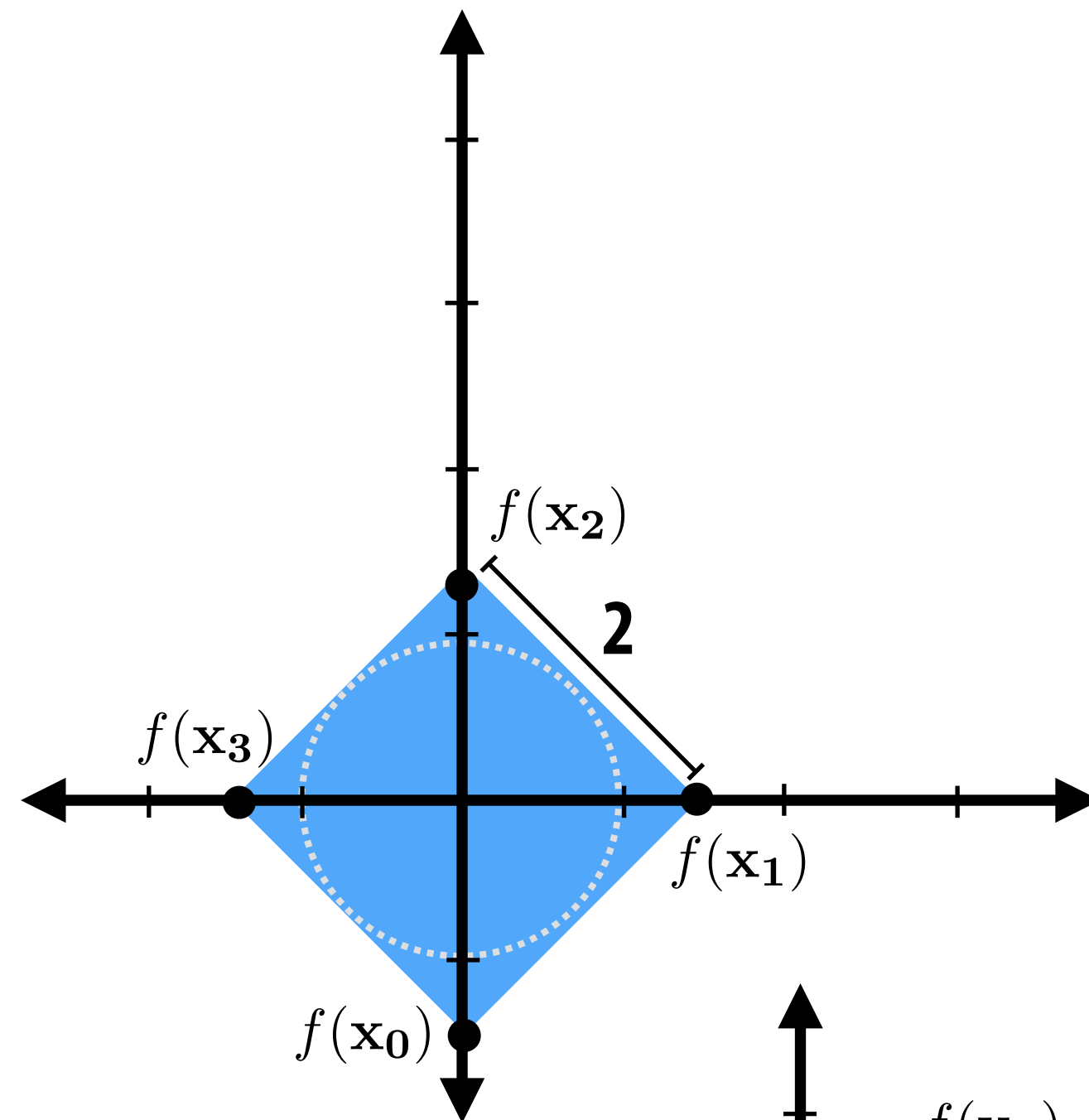
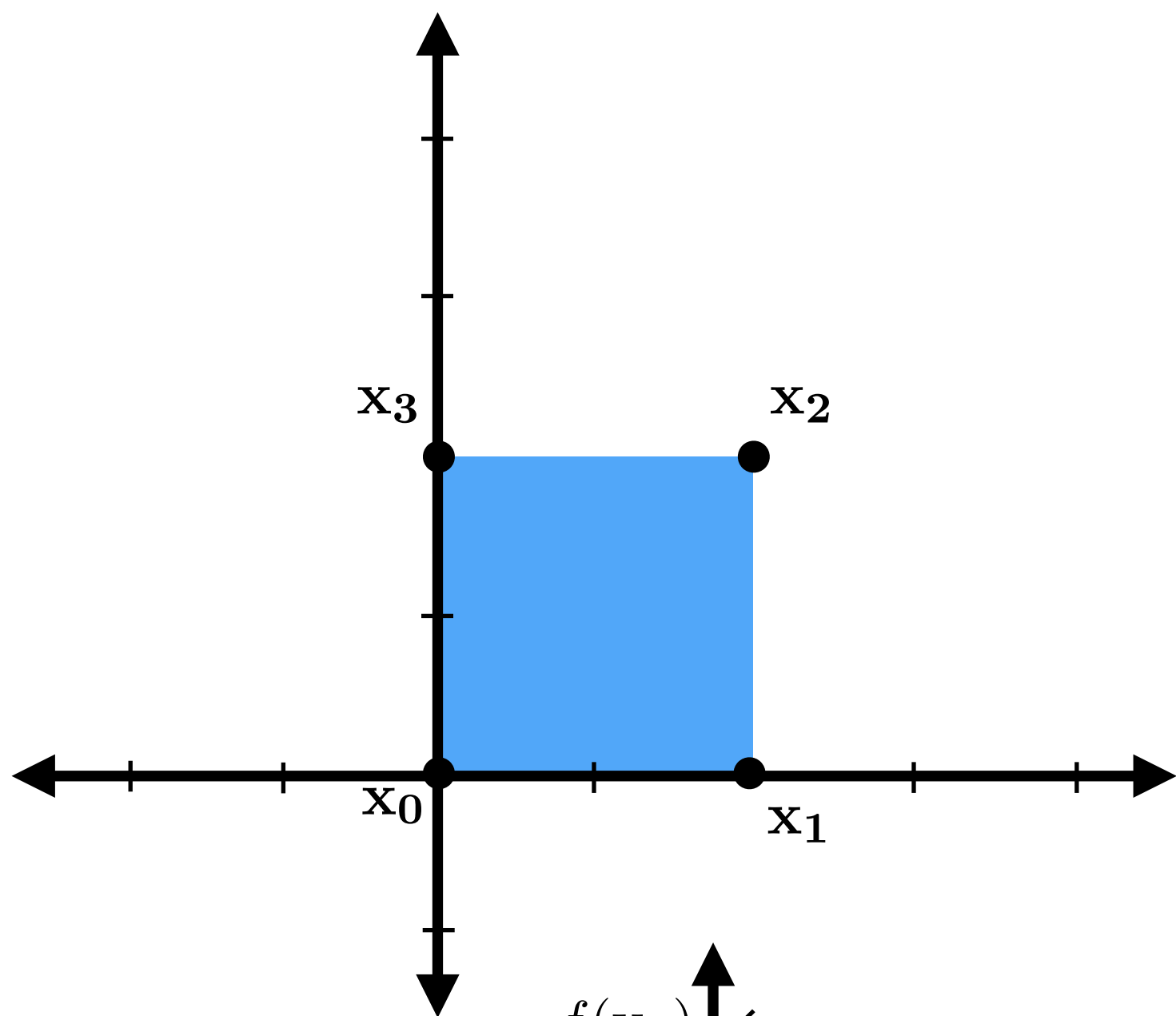
Note: order of composition matters

Top-right: scale, then translate

Bottom-right: translate, then scale

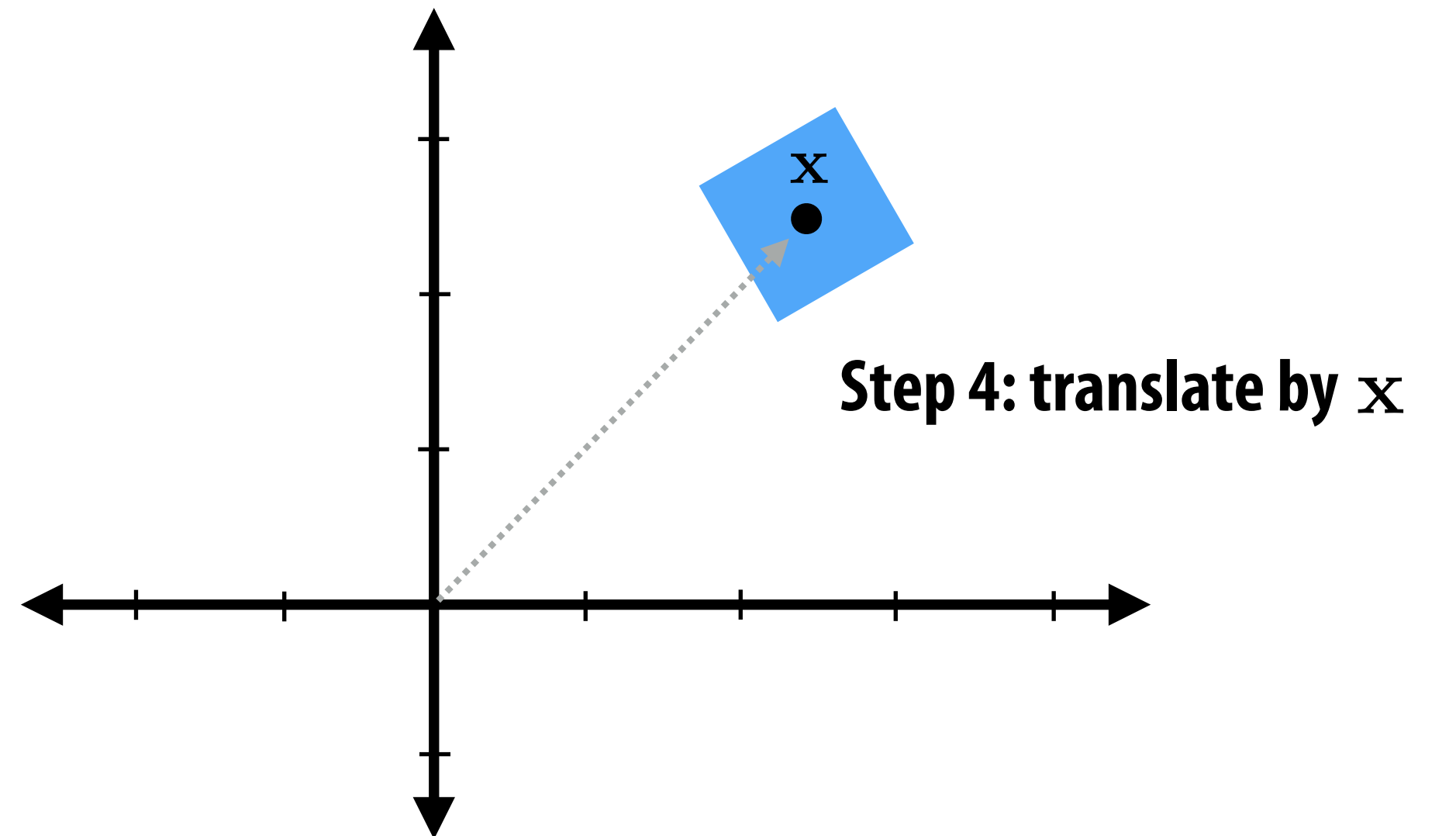
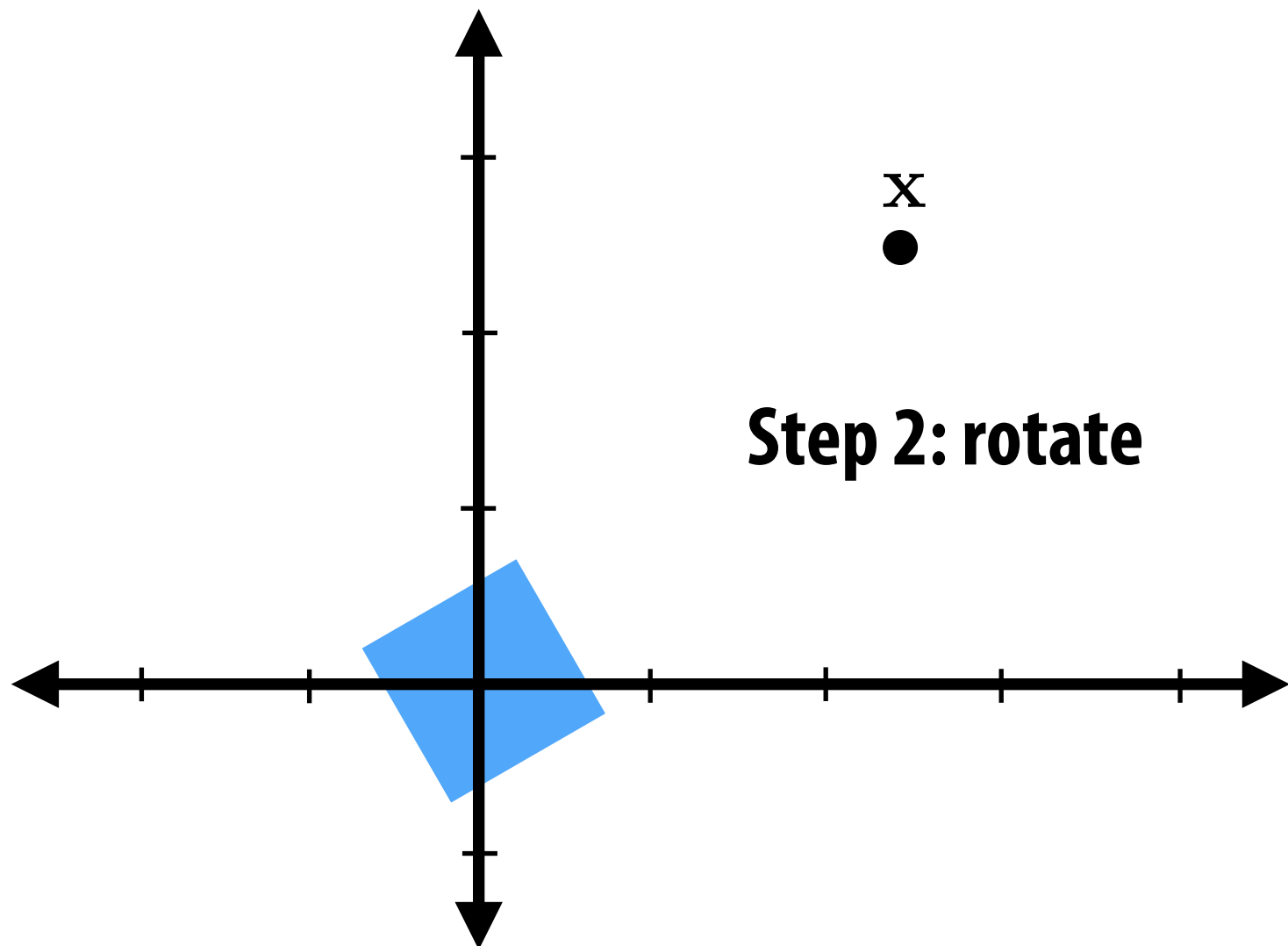
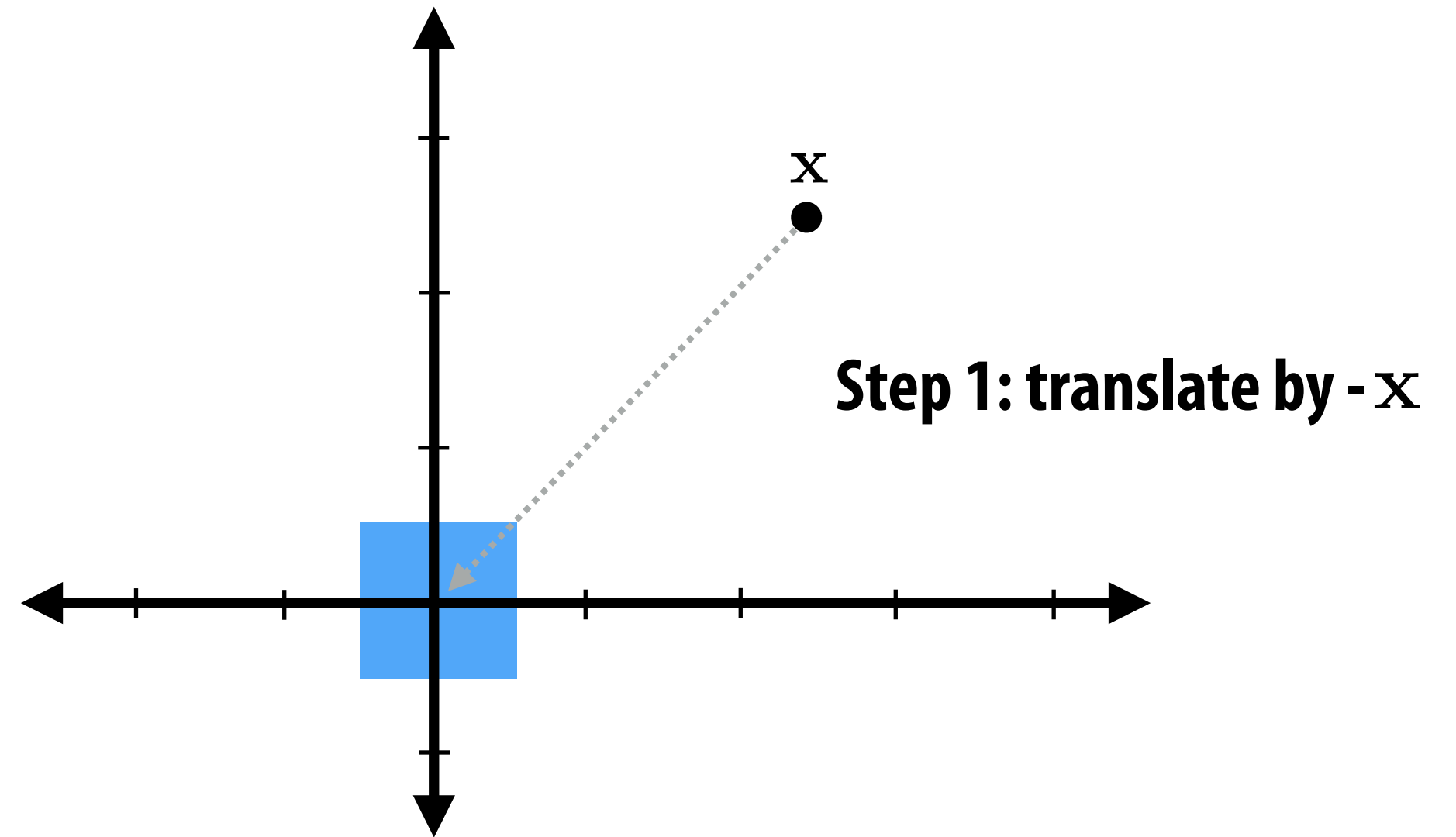
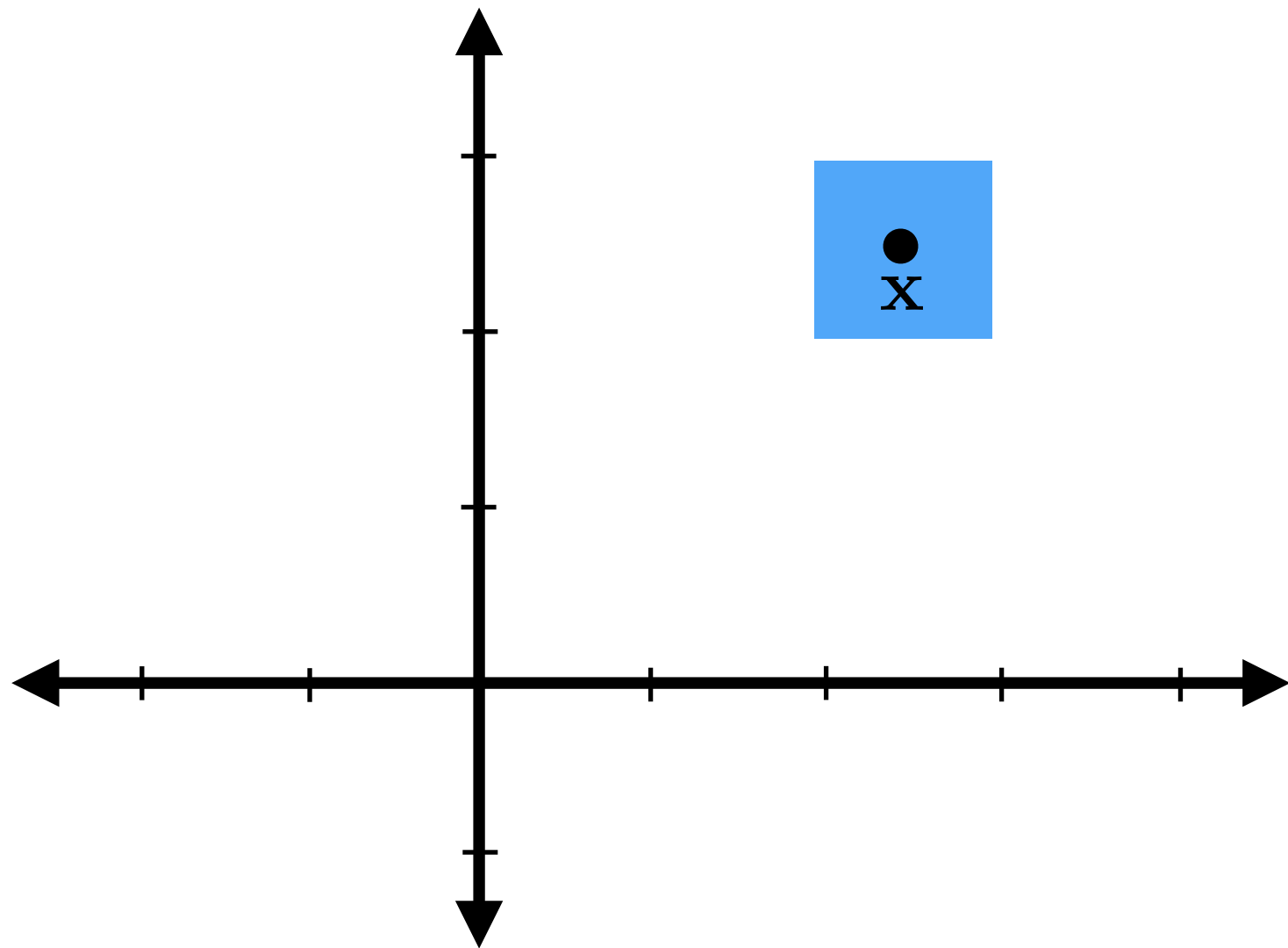


How would you perform these transformations?



Usually more than one way to do it!

Common task: rotate about a point x



Summary of basic transformations

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

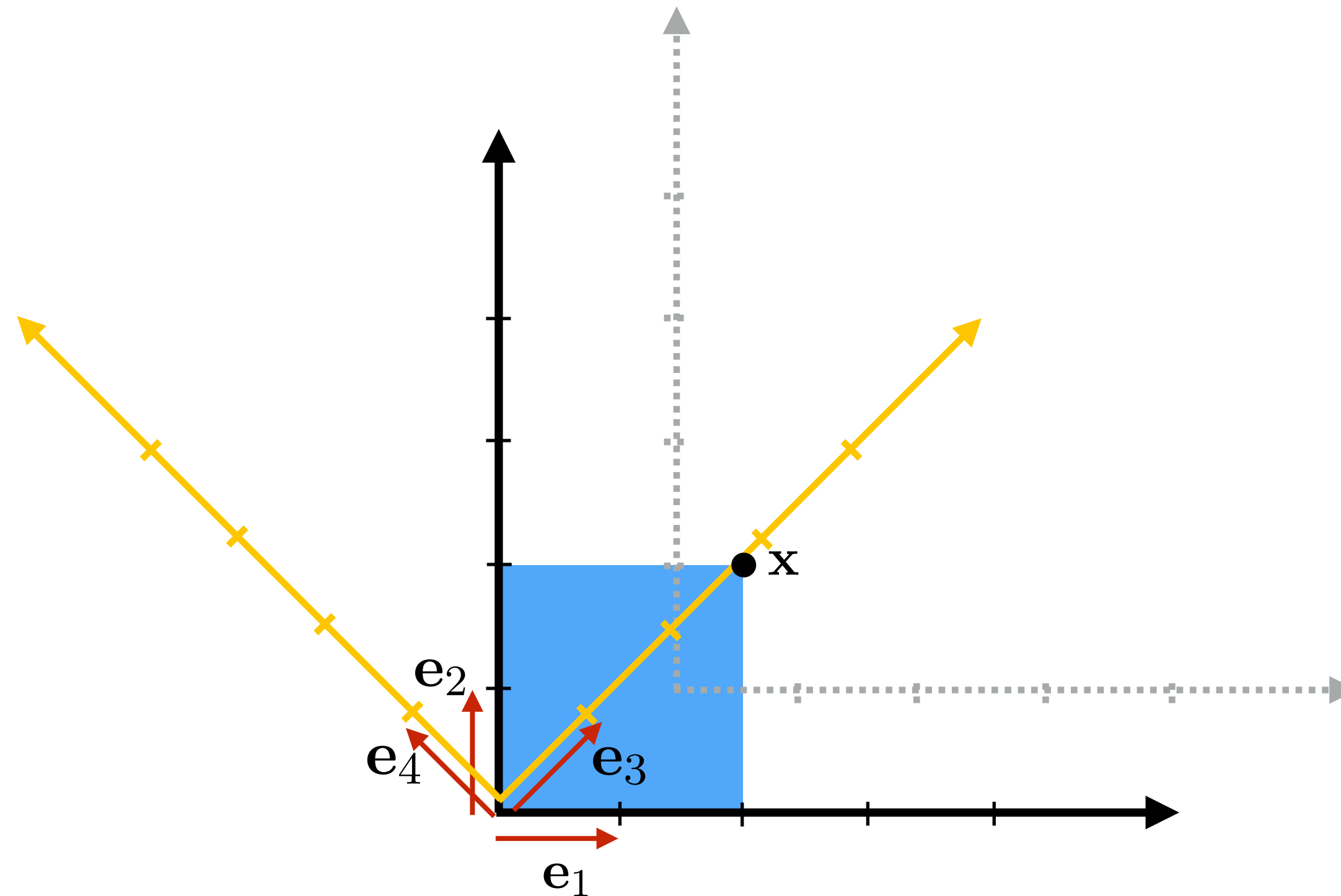
Rotation

Reflection

“Rigid body” transformations are distance-preserving motions that also preserve orientation (i.e., does not include reflection)

Representing Transformations in Coordinates

Review: representing points in a coordinate space



Consider coordinate space defined by orthogonal vectors e_1 and 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 e_1 and e_2 , with origin at $(1.5, 1)$

$\mathbf{x} = \begin{bmatrix} \sqrt{8} & 0 \end{bmatrix}$ in coordinate space defined by e_3 and e_4 , with origin at $(0, 0)$

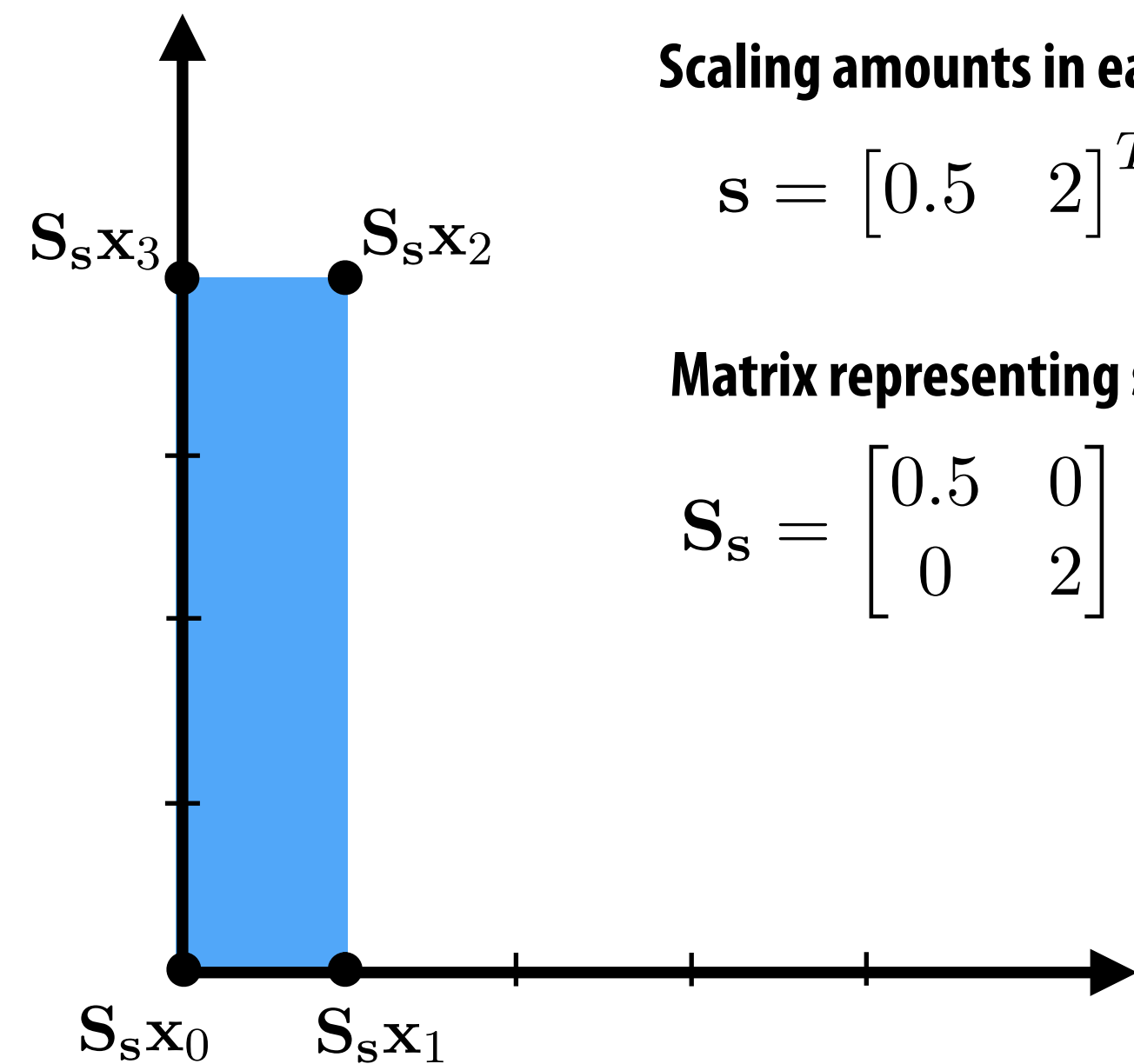
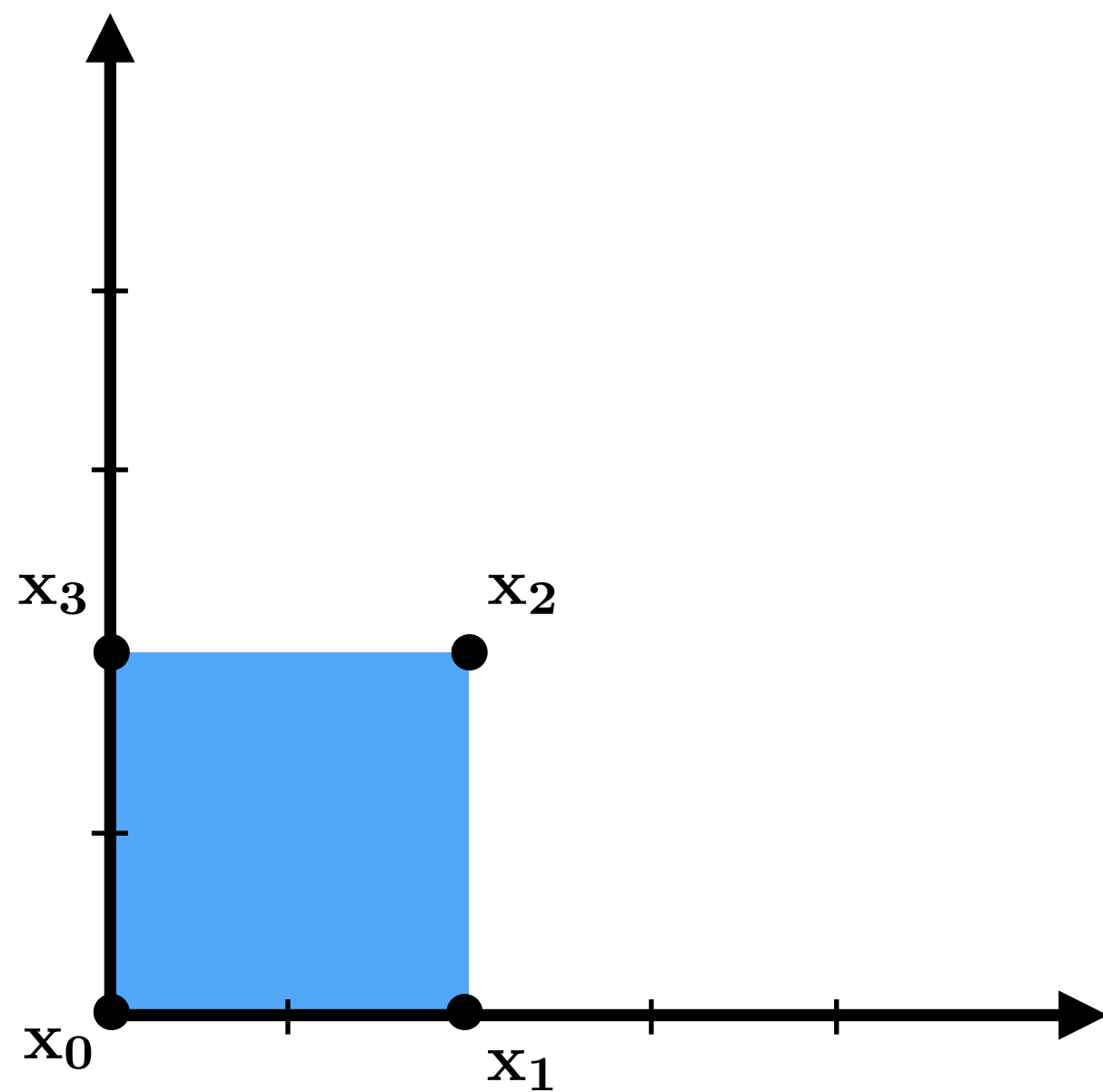
Review: 2D matrix multiplication

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

- **Matrix multiplication is linear combination of columns**
- **Encodes a linear map!**

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

Consider non-uniform scale: $S_s = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$



Scaling amounts in each direction:

$$s = [0.5 \quad 2]^T$$

Matrix representing scale transform:

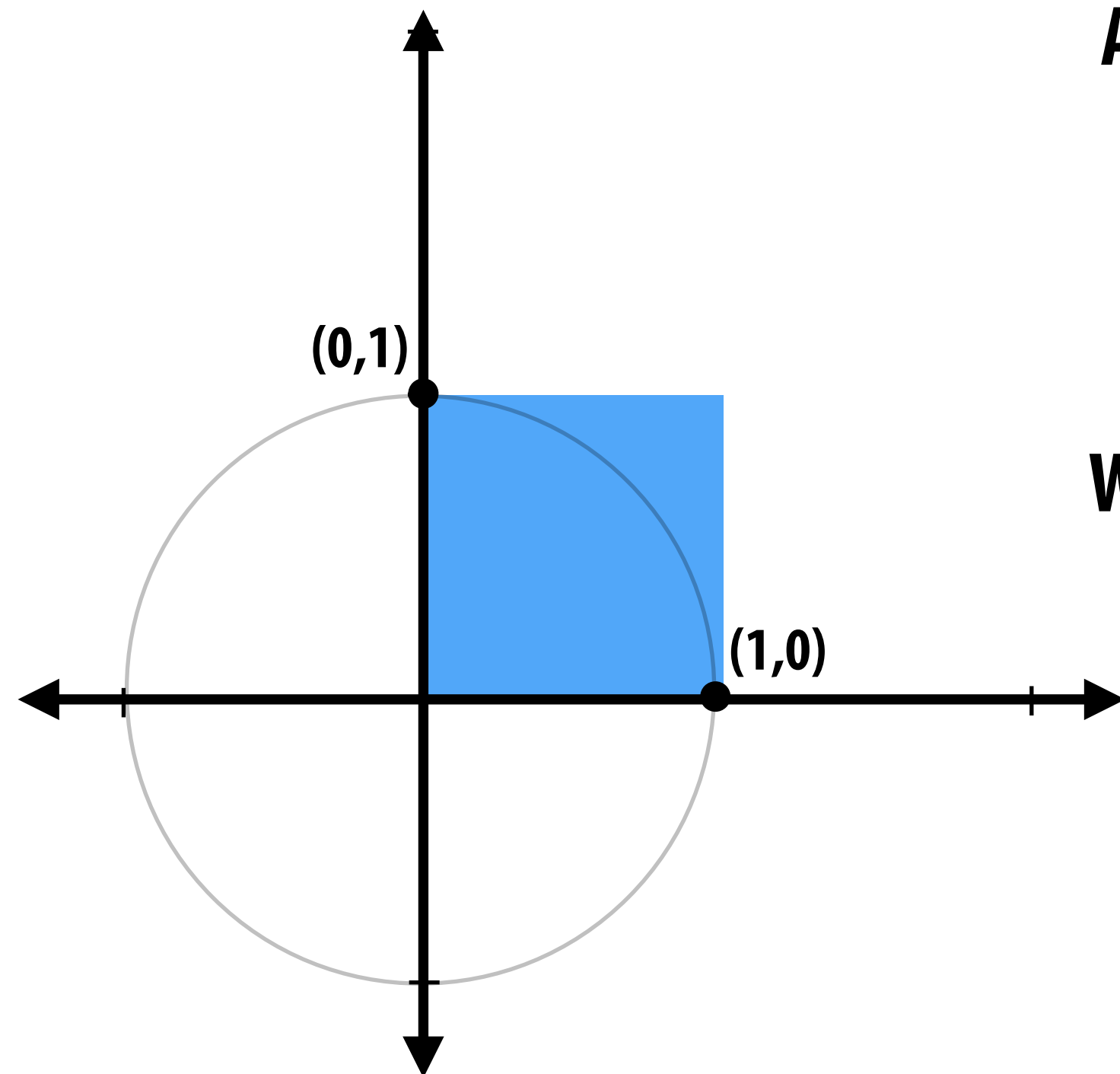
$$S_s = \begin{bmatrix} 0.5 & 0 \\ 0 & 2 \end{bmatrix}$$

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:

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

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

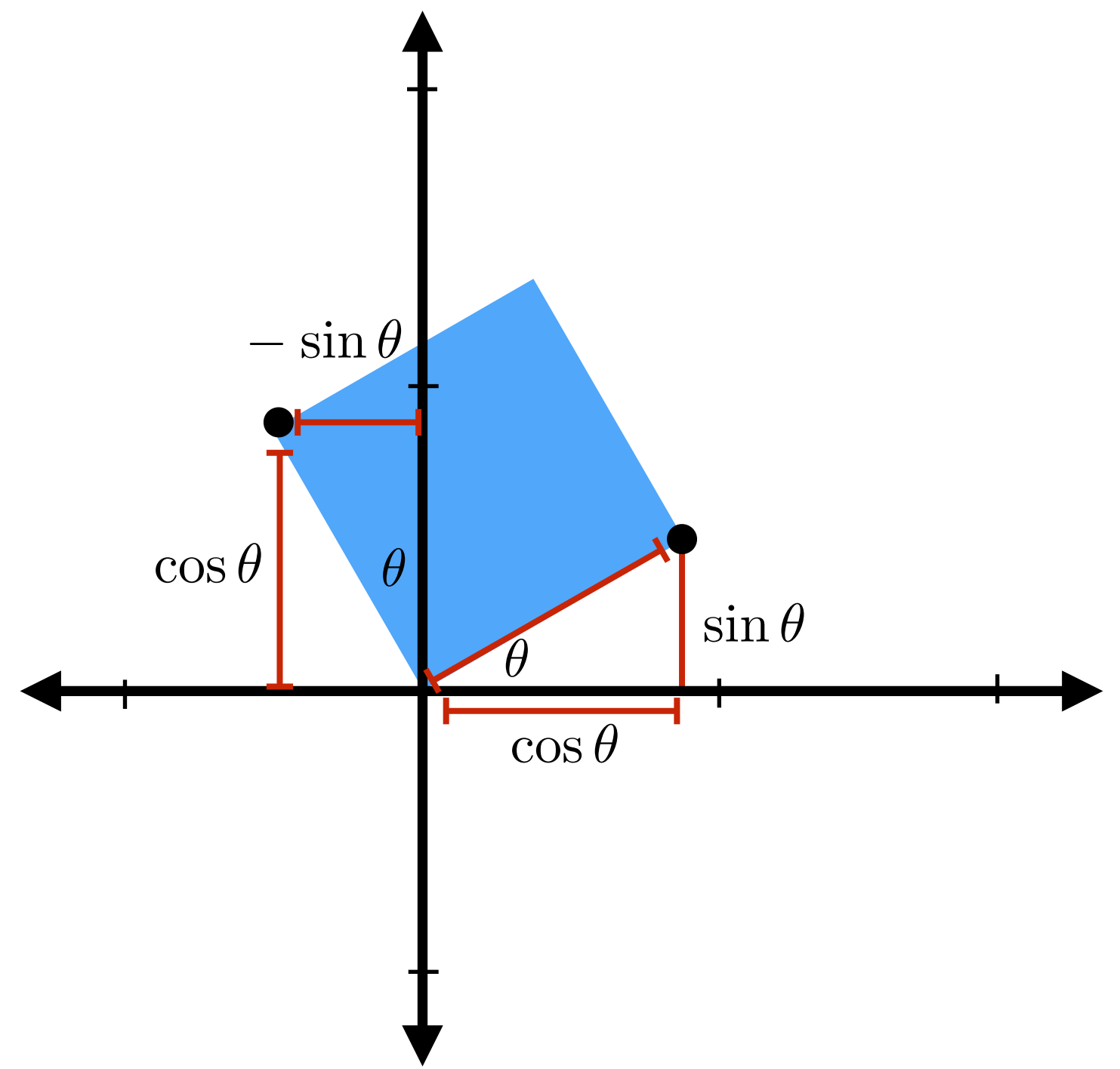
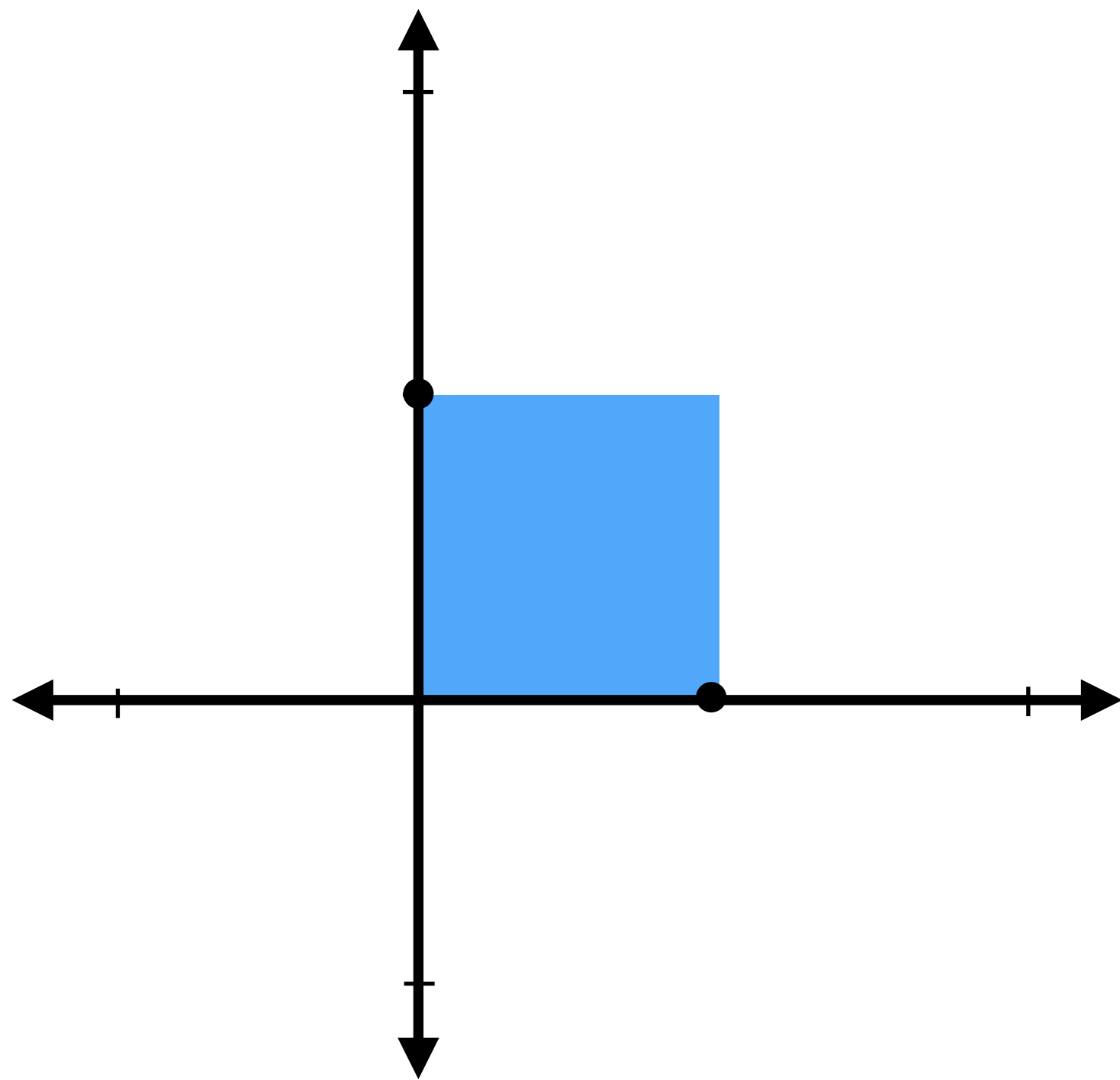
Which means the matrix must look like:

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

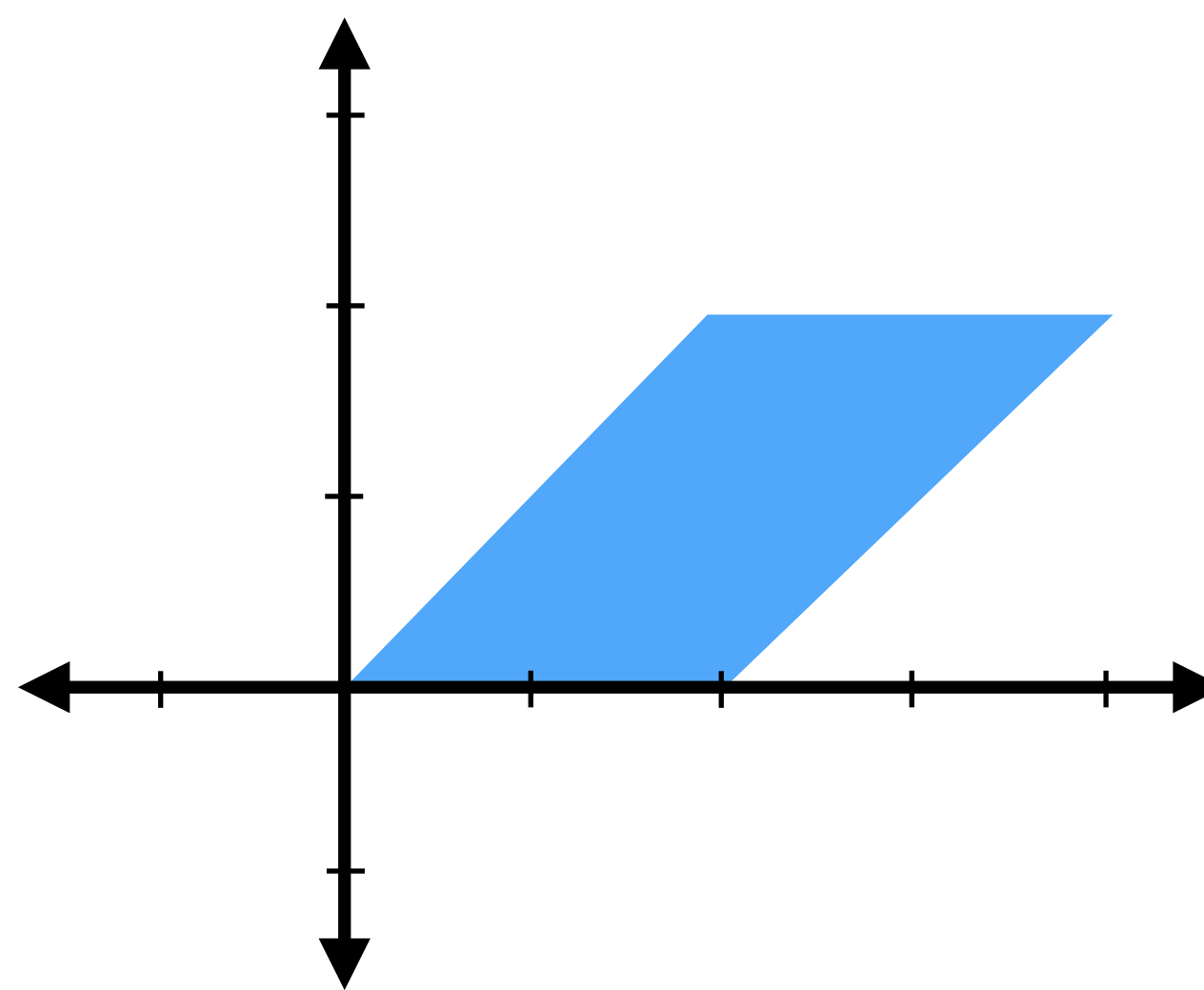
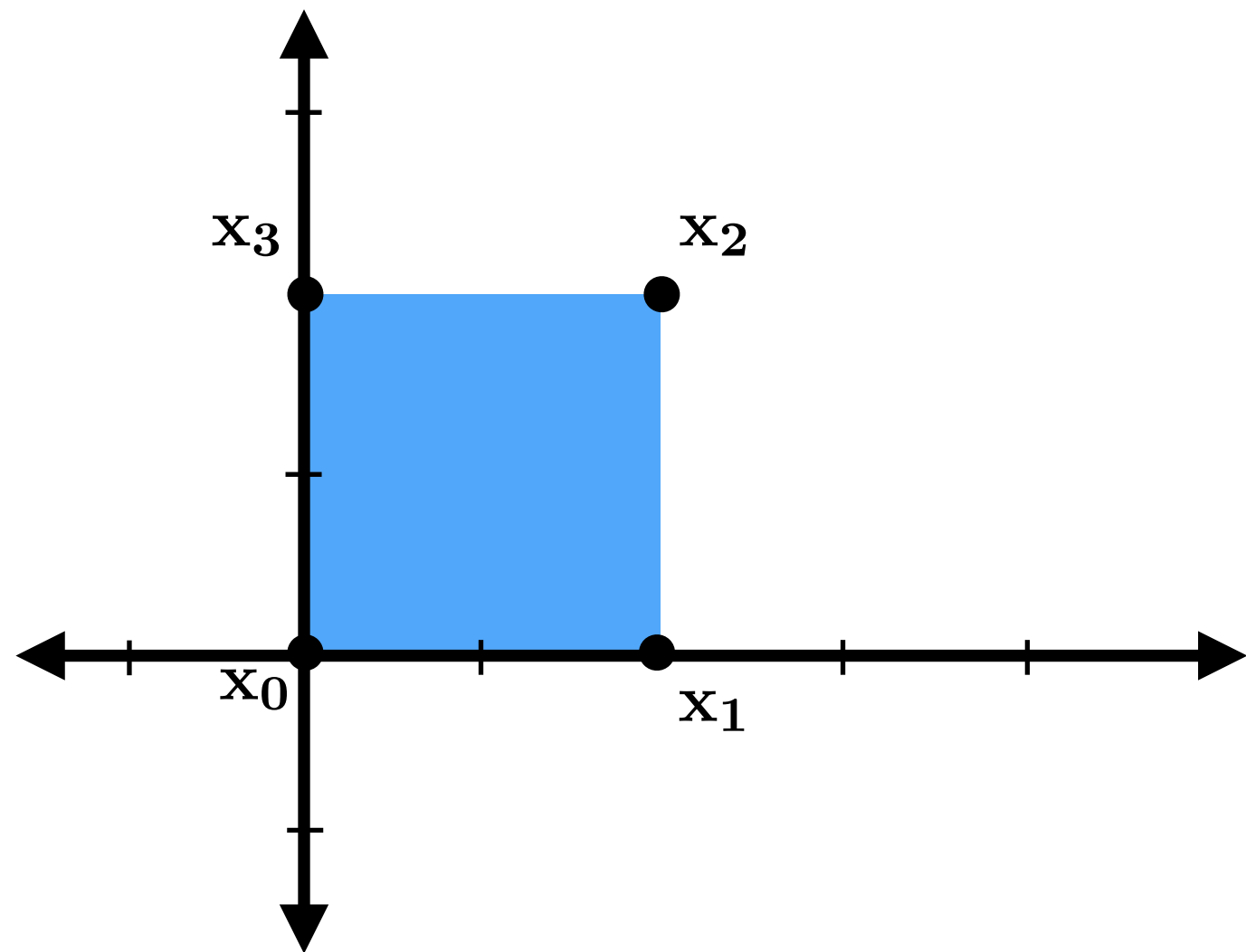
$$= \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

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

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



Shear

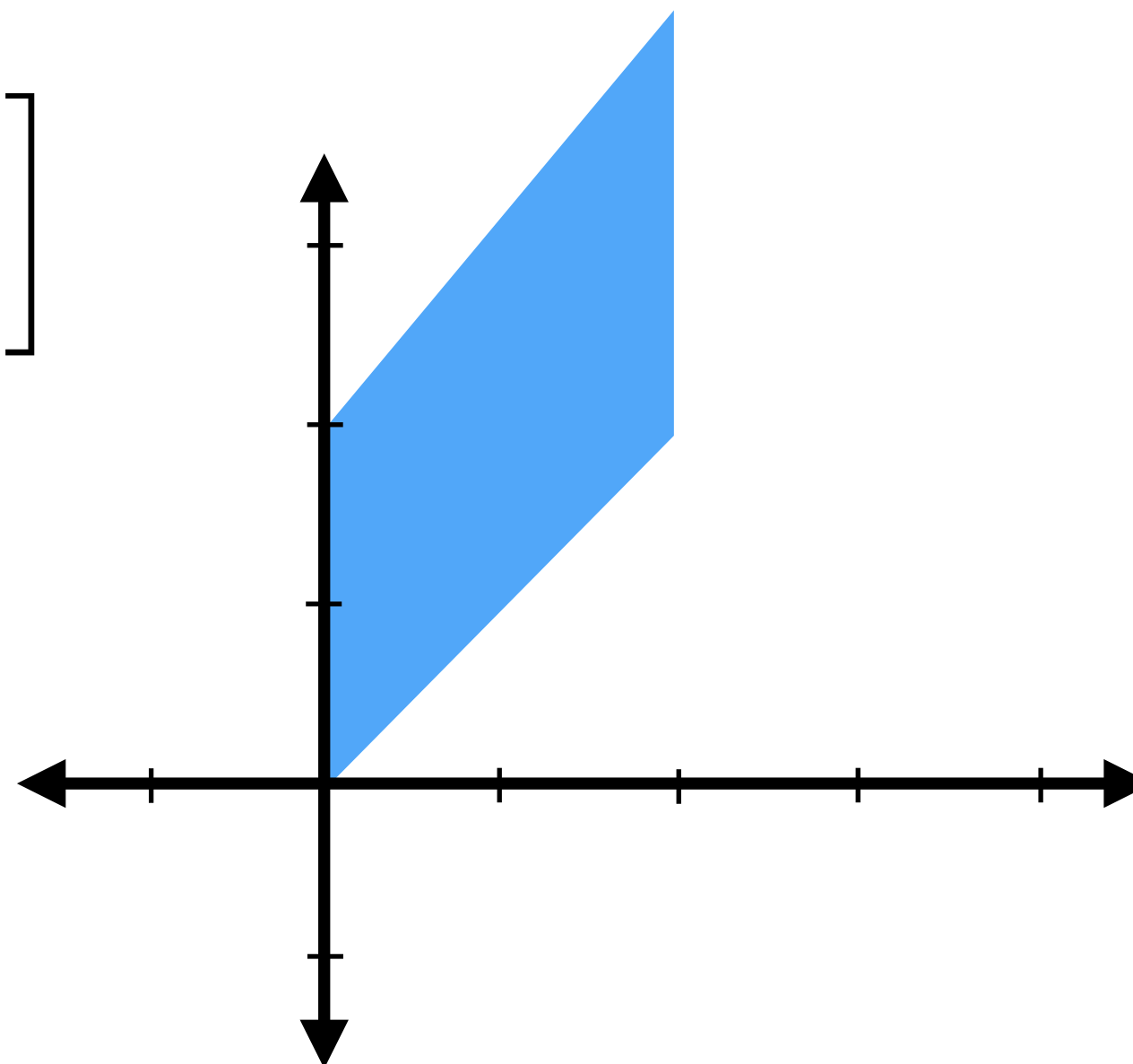
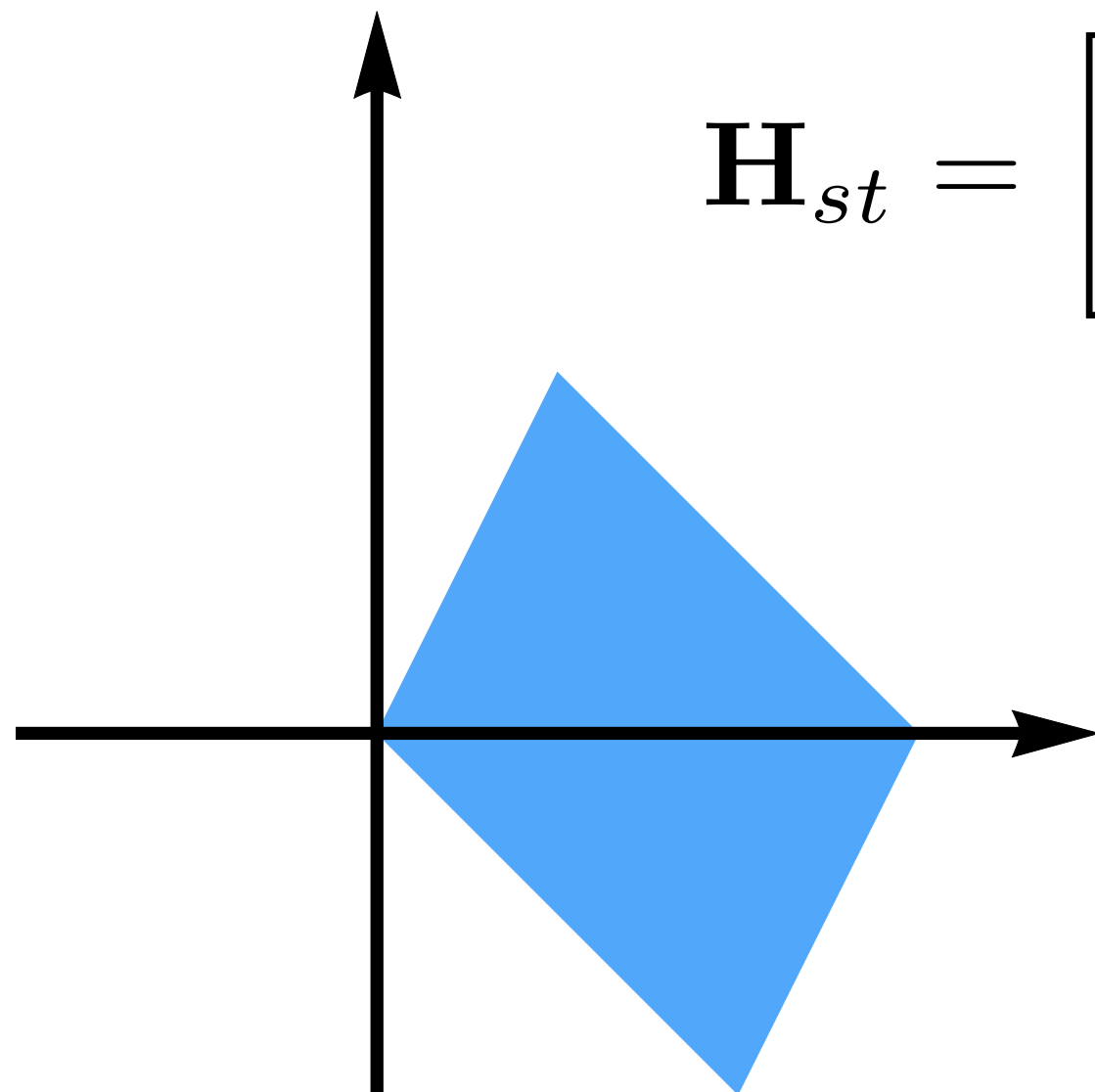


Shear in x:

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

Arbitrary shear:

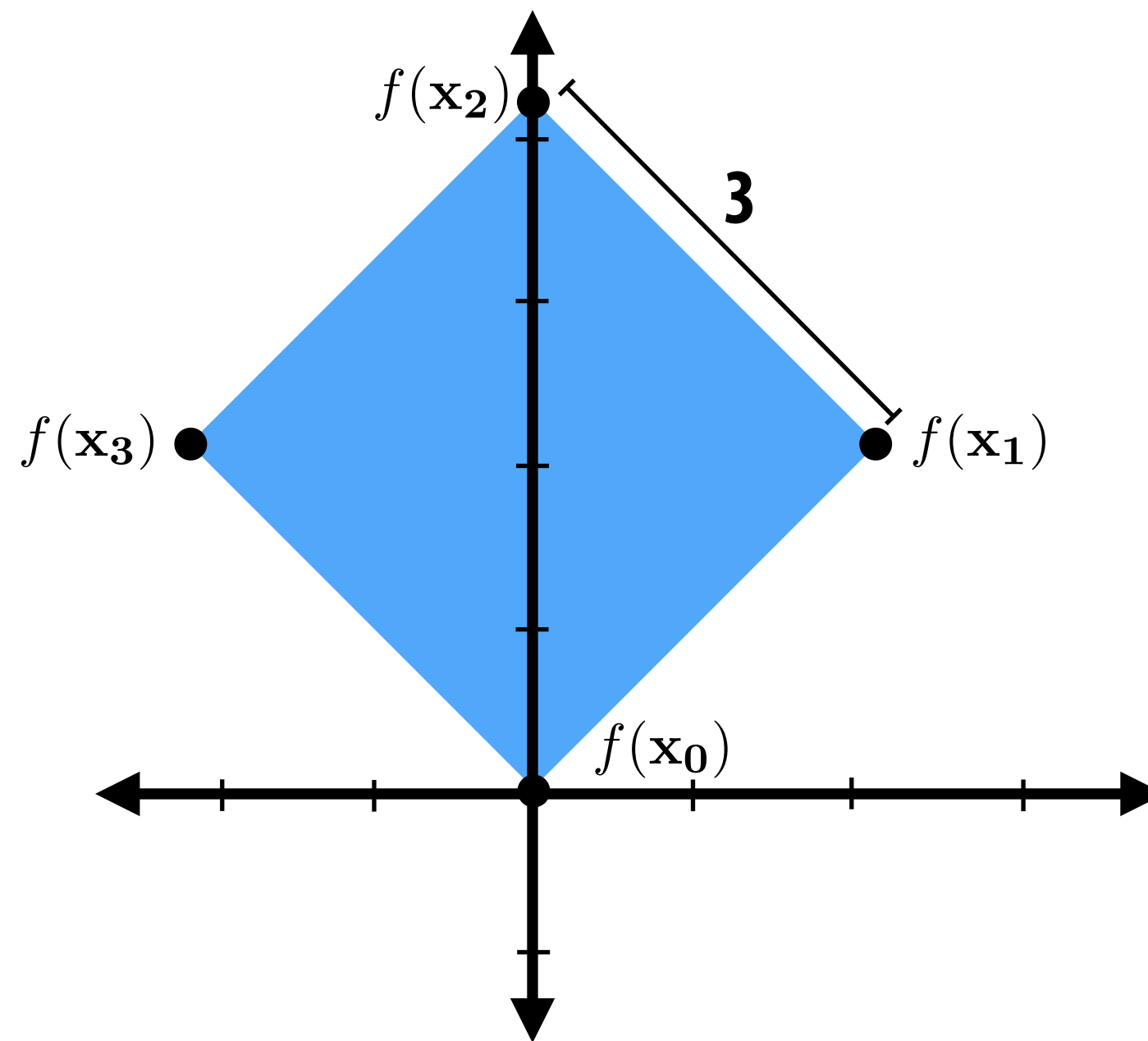
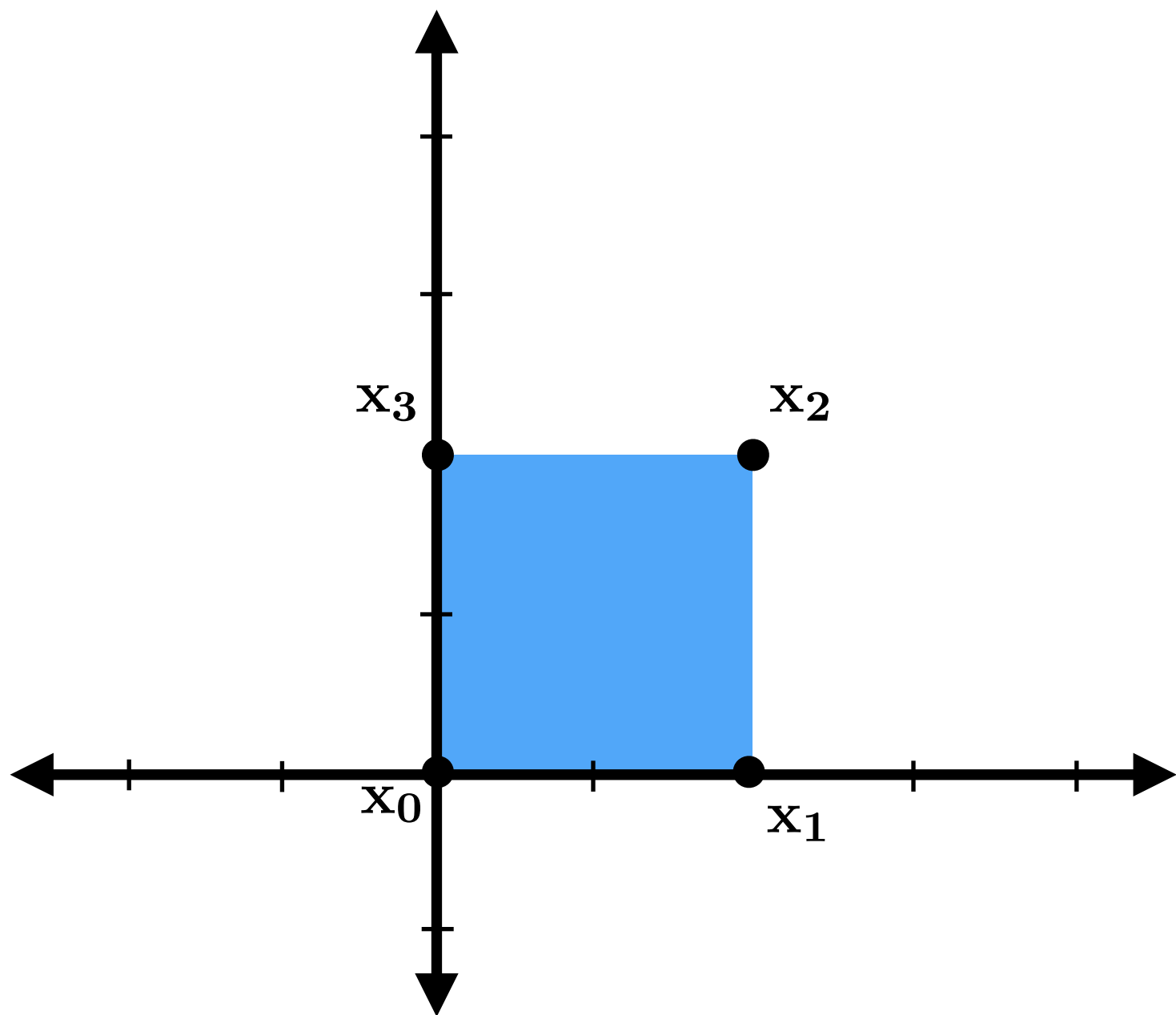
$$\mathbf{H}_{st} = \begin{bmatrix} 1 & s \\ t & 1 \end{bmatrix}$$



Shear in y:

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

How do we compose linear transformations?



Compose linear transformations via matrix multiplication.

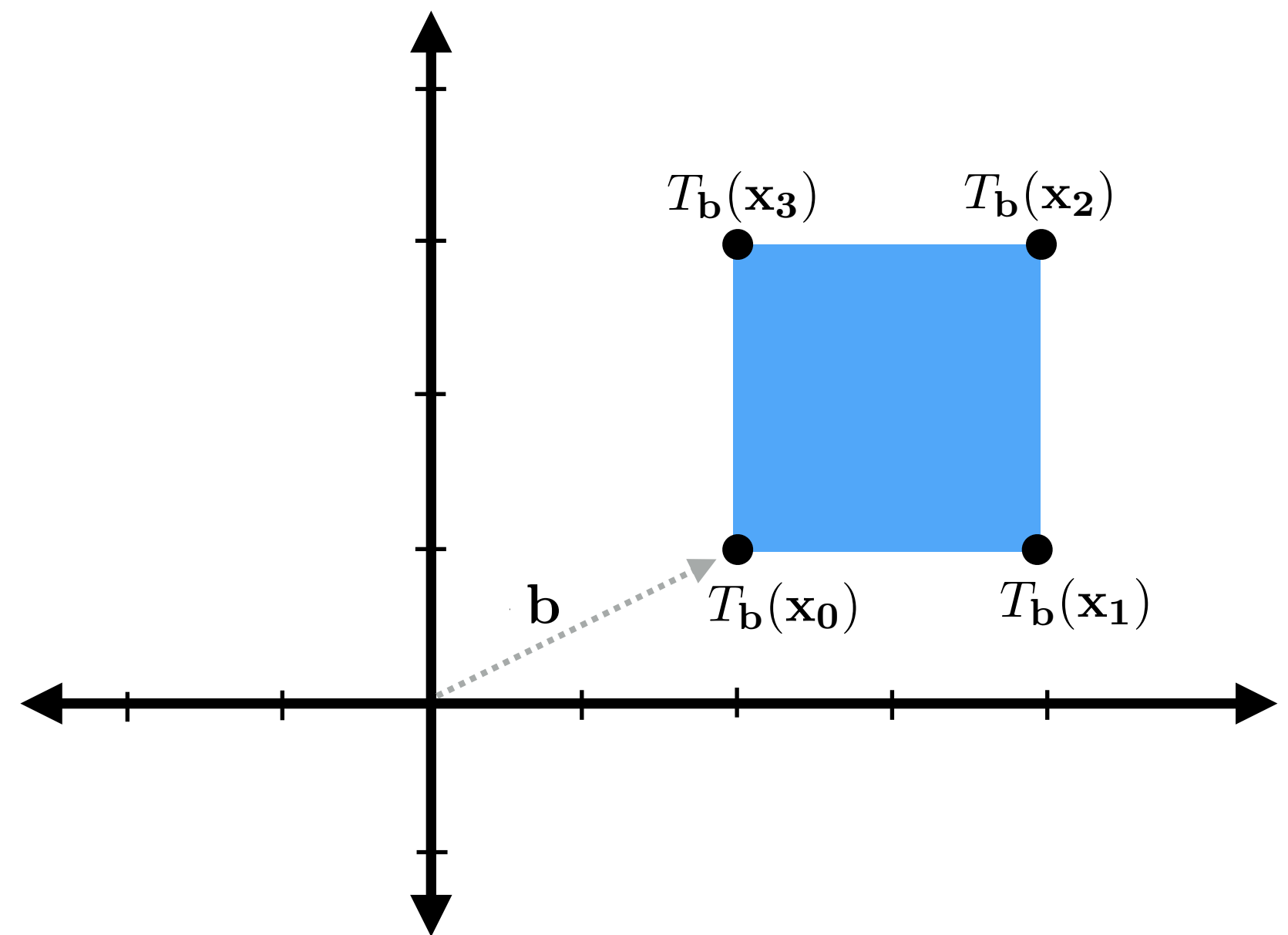
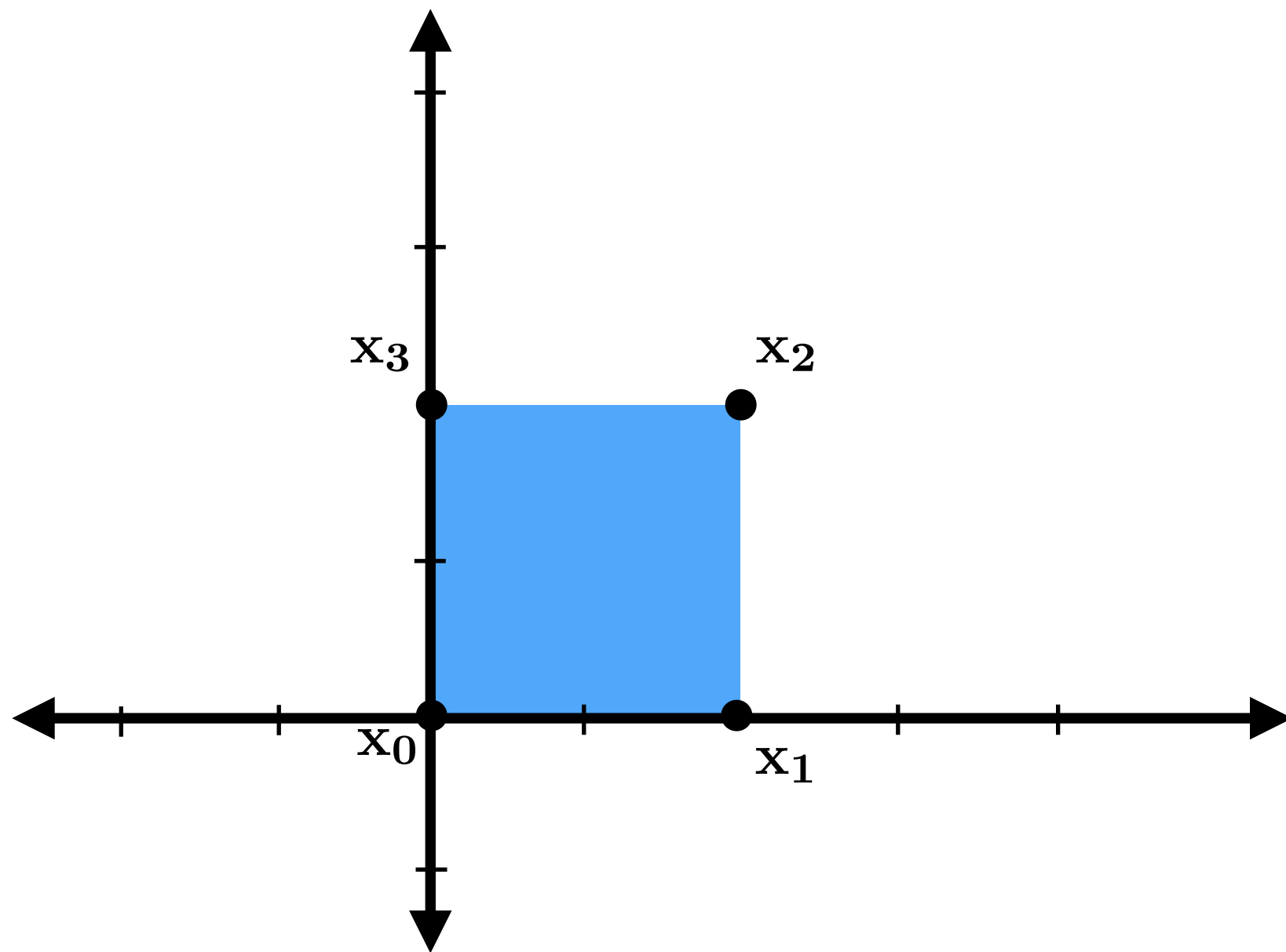
This example: uniform scale, followed by rotation

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

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

How do we deal with translation? (Not linear)

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



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**

$$x_{\text{out}x} = x_x + b_x$$

$$x_{\text{out}y} = x_y + b_y$$

Translation math

2D homogeneous coordinates (2D-H)

Interesting idea: represent 2D points with THREE values (“homogeneous coordinates”)

So the point (x, y) is represented as the 3-vector: $[x \quad y \quad 1]^T$

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

Recover final 2D coordinates by dividing by “extra” (third) coordinate

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

(More on this later...)

Example: Scale & Rotation in 2D-H Coords

- For transformations that are already linear, not much changes:

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

Notice that the last row/column doesn't do anything interesting. E.g., for scaling:

$$\mathbf{S}_s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{S}_x x \\ \mathbf{S}_y y \\ 1 \end{bmatrix}$$

Now we divide by the 3rd coordinate to get our final 2D coordinates (not too exciting!)

$$\begin{bmatrix} \mathbf{S}_x x \\ \mathbf{S}_y y \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} \mathbf{S}_x x / 1 \\ \mathbf{S}_y y / 1 \end{bmatrix} = \begin{bmatrix} \mathbf{S}_x x \\ \mathbf{S}_y y \end{bmatrix}$$

(Will get more interesting when we talk about perspective...)

Translation in 2D homogeneous coordinates

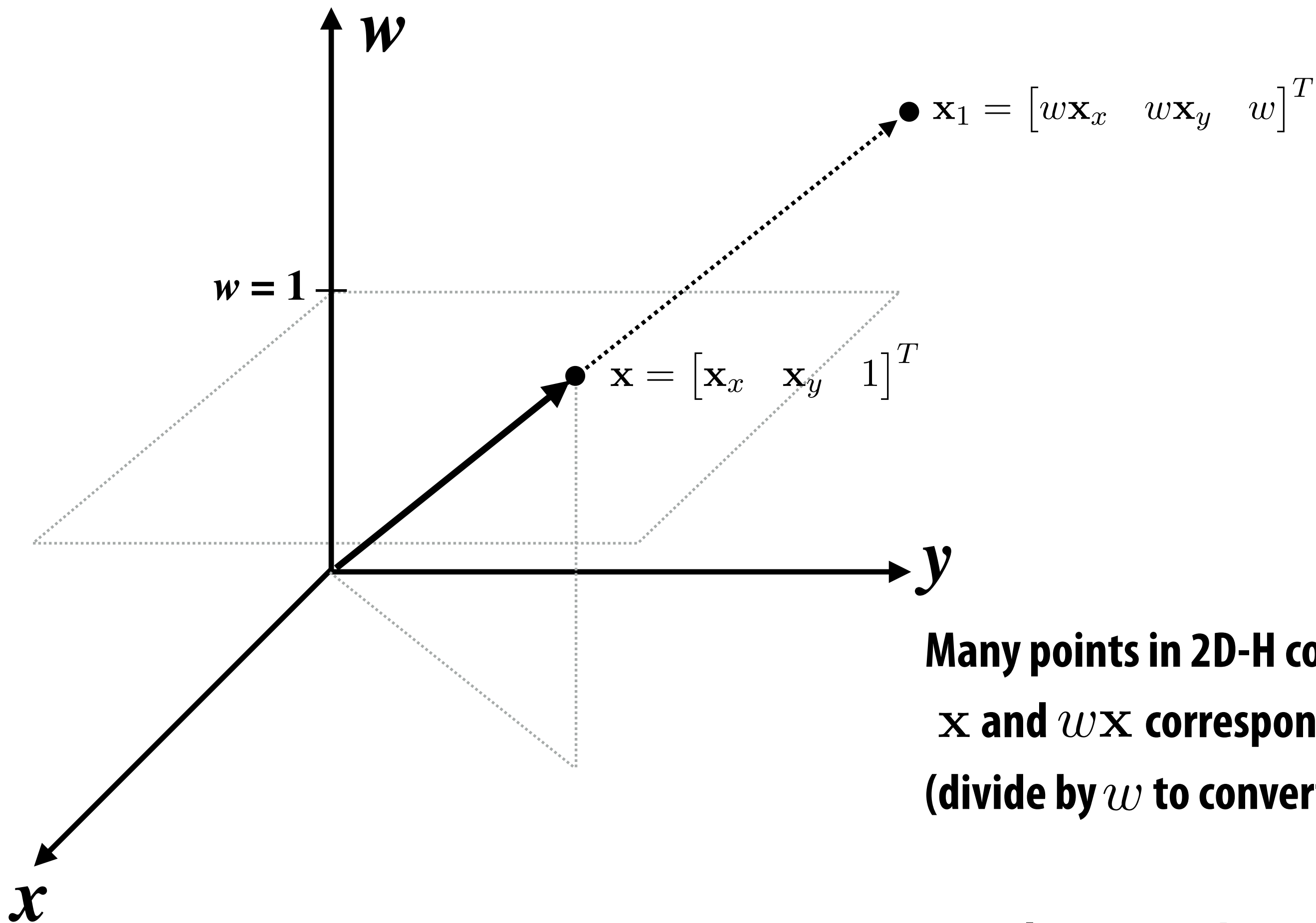
Translation expressed as 3x3 matrix multiplication:

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

$$\mathbf{T}_b \mathbf{x} = \begin{bmatrix} 1 & 0 & \mathbf{b}_x \\ 0 & 1 & \mathbf{b}_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_x \\ \mathbf{x}_y \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}_x + \mathbf{b}_x \\ \mathbf{x}_y + \mathbf{b}_y \\ 1 \end{bmatrix} \text{ (remember: linear combination of columns!)}$$

Cool: homogeneous coordinates let us encode translations as linear transformations!

Homogeneous coordinates: some intuition

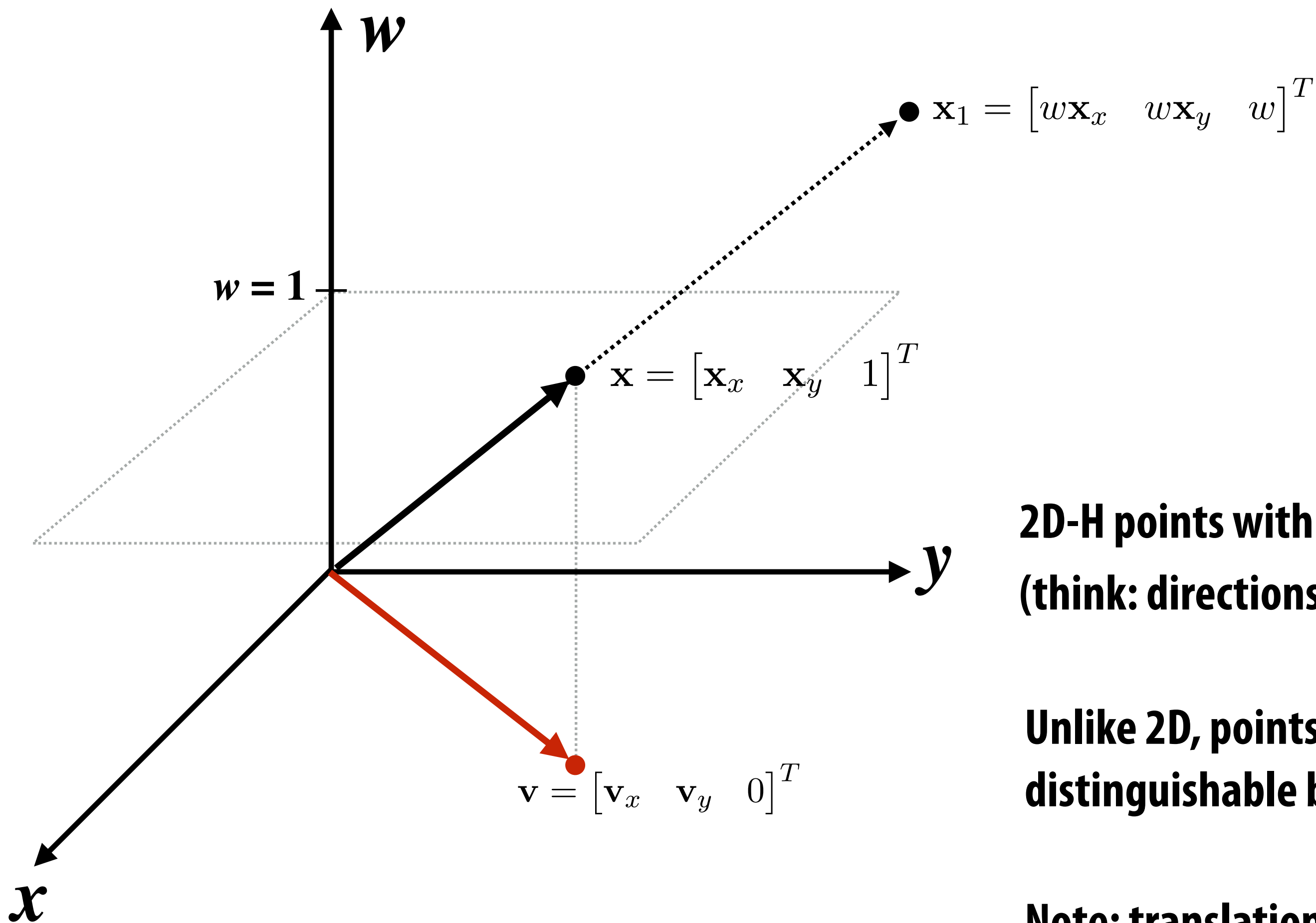


Many points in 2D-H correspond to same point in 2D
 \mathbf{x} and $w\mathbf{x}$ correspond to the same 2D point
(divide by w to convert 2D-H back to 2D)

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

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

Homogeneous coordinates: points vs. vectors



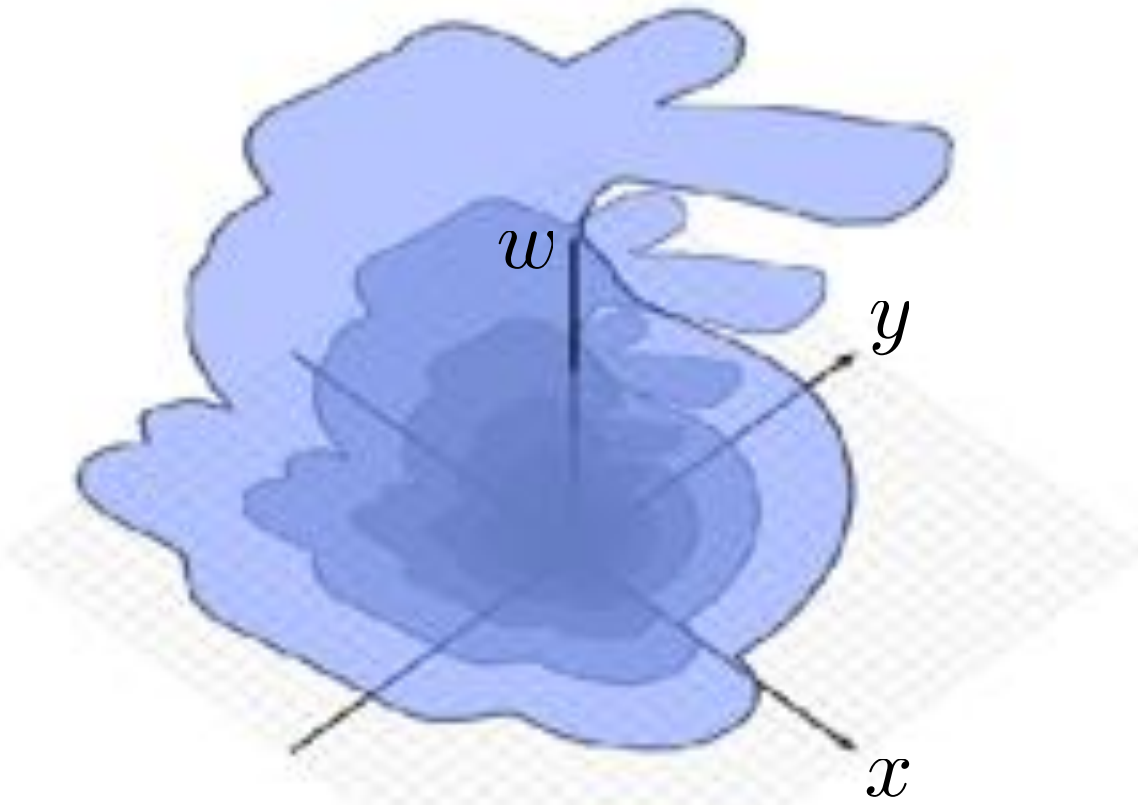
**2D-H points with $w=0$ represent 2D vectors
(think: directions are points at infinity)**

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

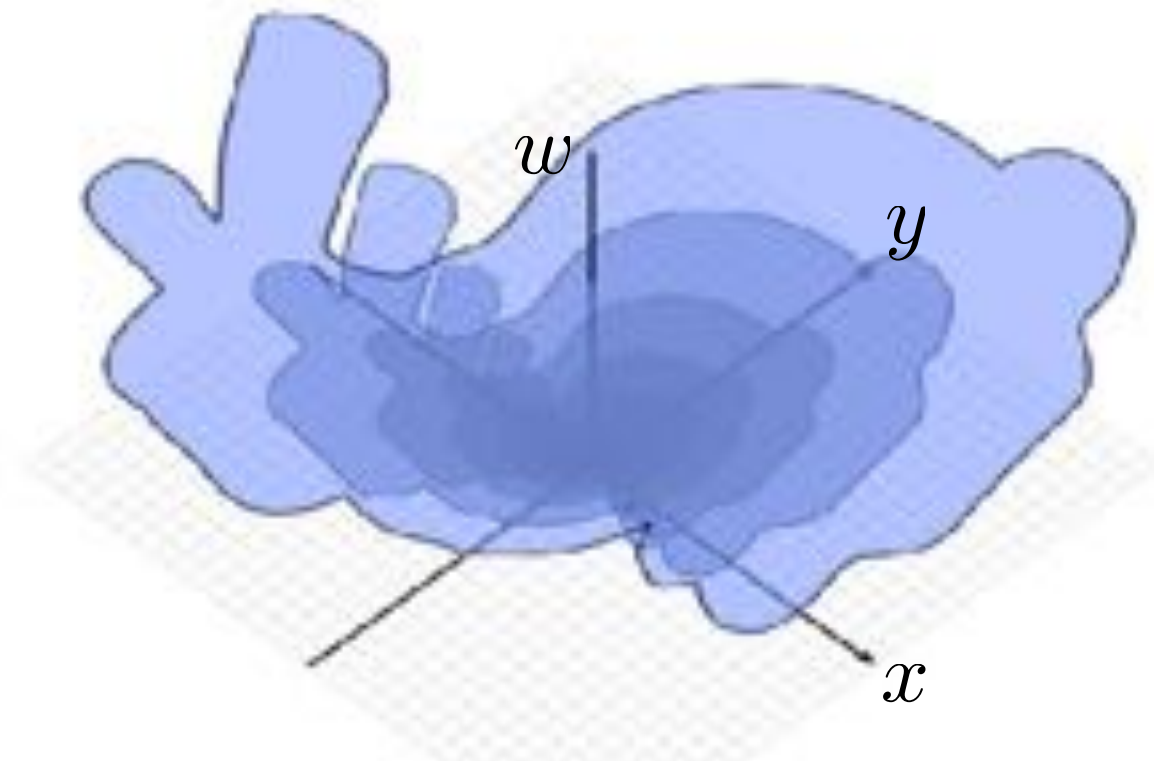
Note: translation does not modify directions:

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

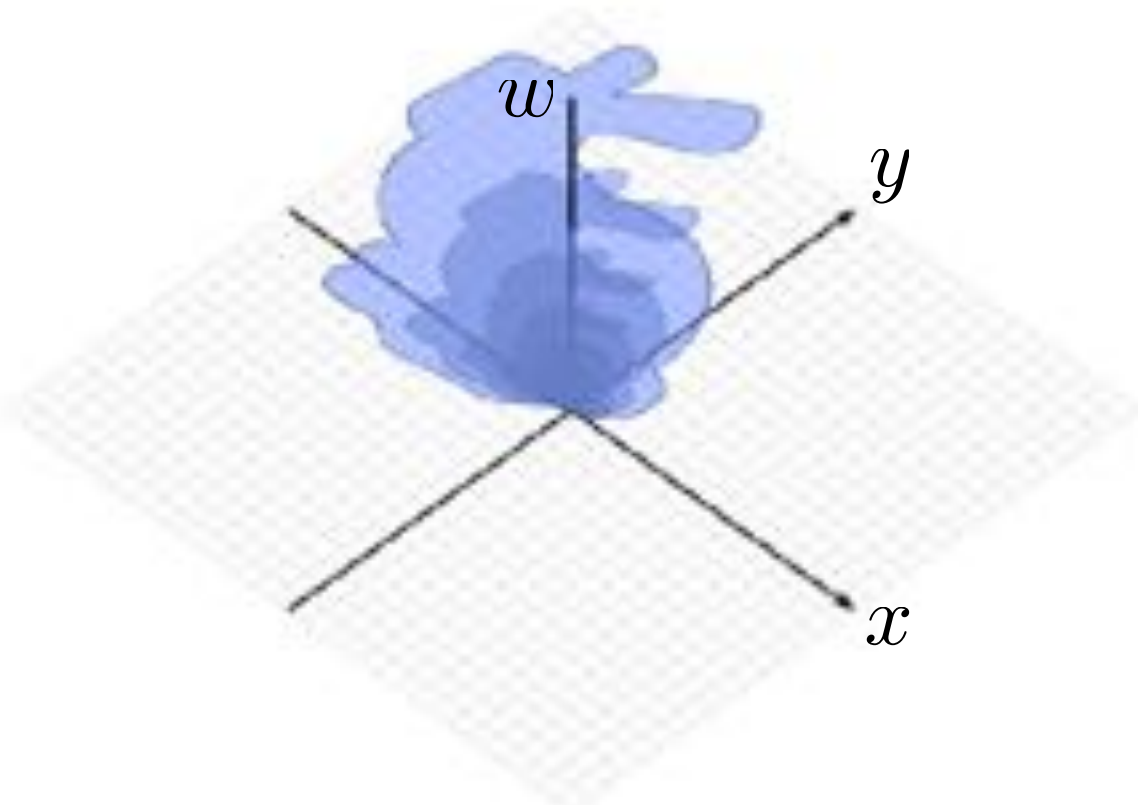
Visualizing 2D transformations in 2D-H



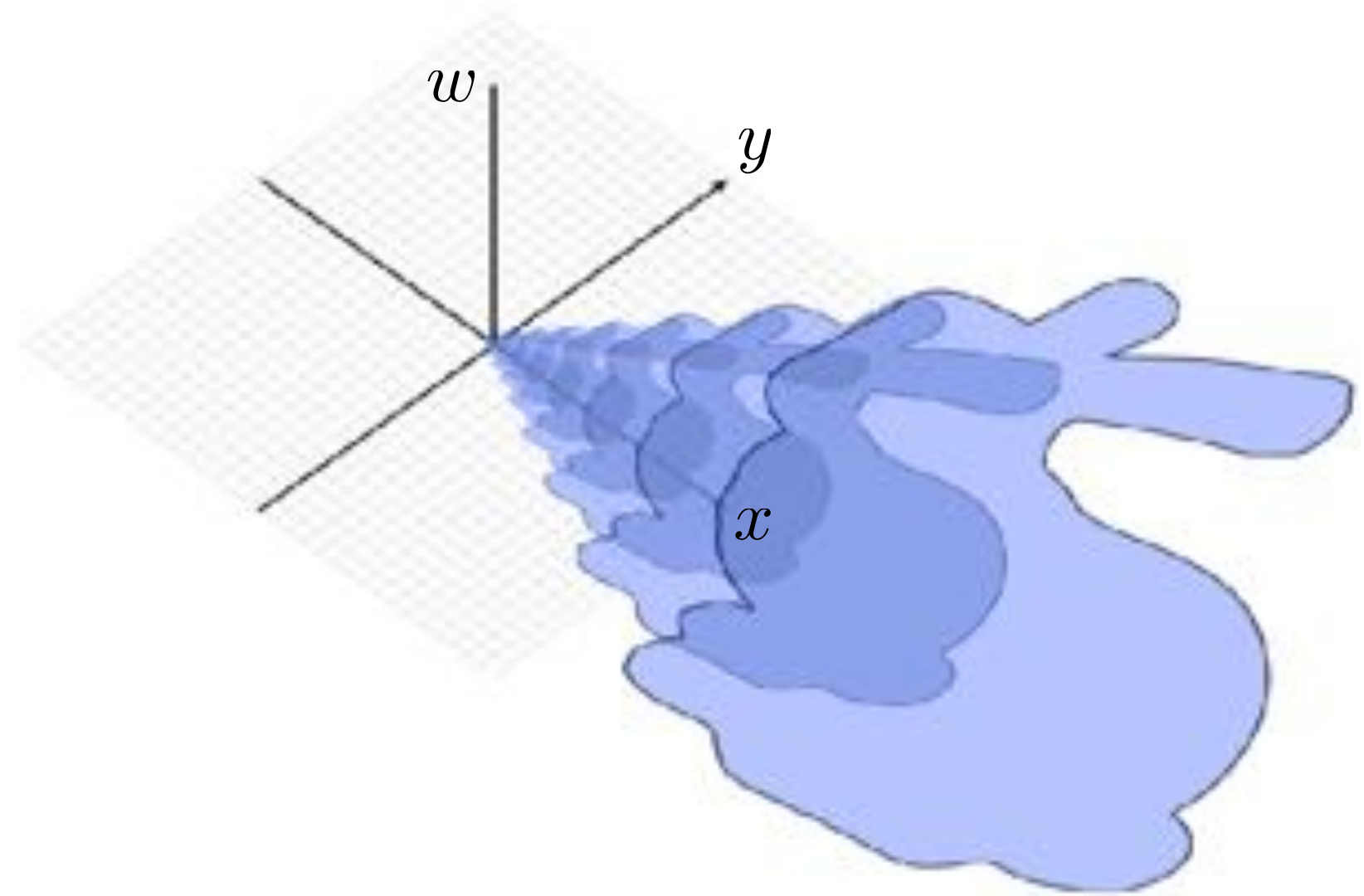
Original shape in 2D can be viewed as many copies, uniformly scaled by w .



2D rotation \leftrightarrow rotate around w



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



2D translate \leftrightarrow shear in 2D-H
(LINEAR!)

Moving to 3D (and 3D-H)

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

Scale:

$$\begin{array}{c} \mathbf{S}_s = \\ \text{3D} \end{array} \begin{bmatrix} \mathbf{S}_x & 0 & 0 \\ 0 & \mathbf{S}_y & 0 \\ 0 & 0 & \mathbf{S}_z \end{bmatrix} \quad \begin{array}{c} \mathbf{S}_s = \\ \text{3D-H} \end{array} \begin{bmatrix} \mathbf{S}_x & 0 & 0 & 0 \\ 0 & \mathbf{S}_y & 0 & 0 \\ 0 & 0 & \mathbf{S}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

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

Translate:

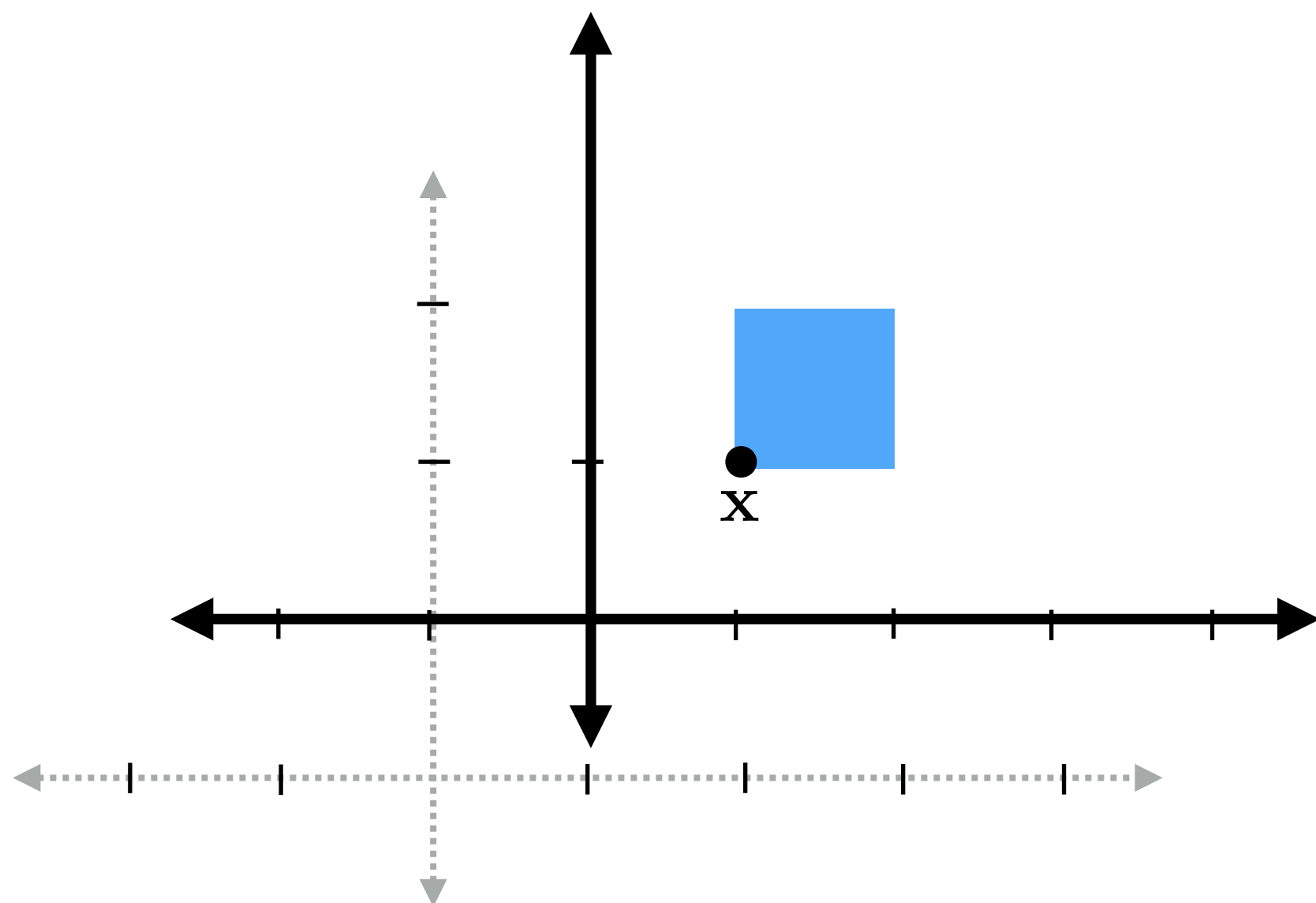
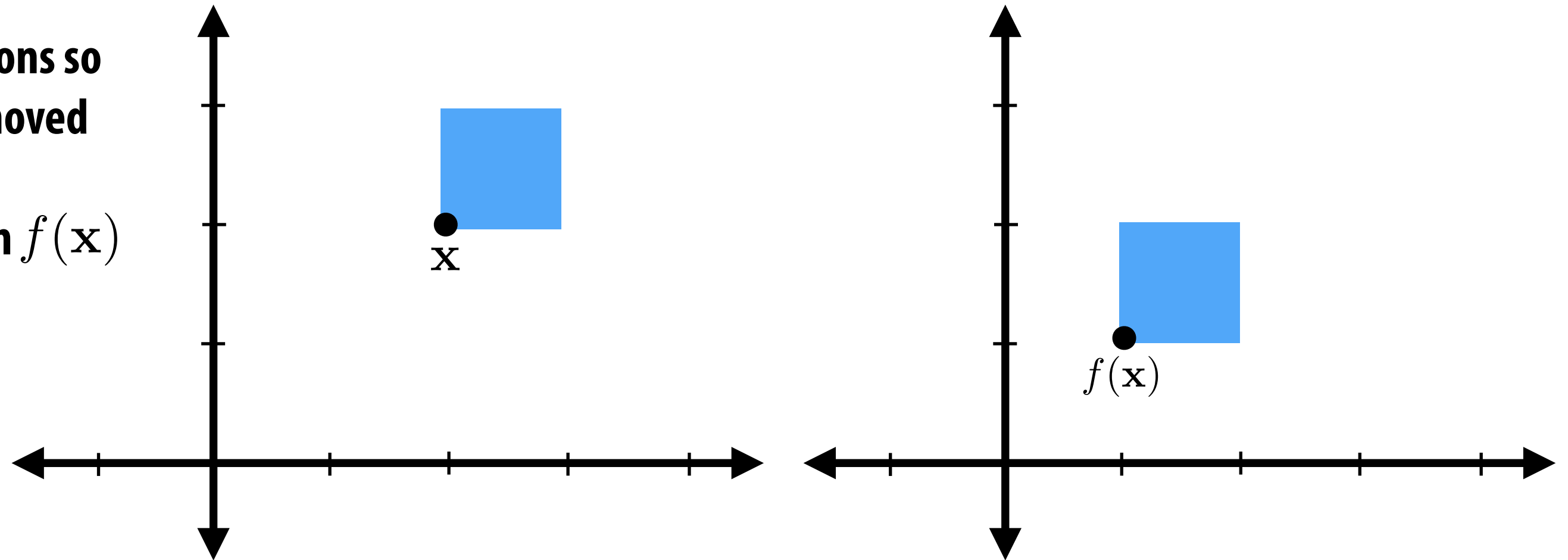
$$\mathbf{T}_b = \begin{array}{c} \text{3D-H} \\ \begin{bmatrix} 1 & 0 & 0 & \mathbf{b}_x \\ 0 & 1 & 0 & \mathbf{b}_y \\ 0 & 0 & 1 & \mathbf{b}_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Much more about rotations in next lecture!

Another way to think about transformations: change of coordinates

Interpretation of transformations so far in this lecture: points get moved

Point \mathbf{x} moved to new position $f(\mathbf{x})$



Alternative interpretation:

**Transformations induce a change of coordinates:
Representation of \mathbf{x} changes since point is now expressed in new coordinates**

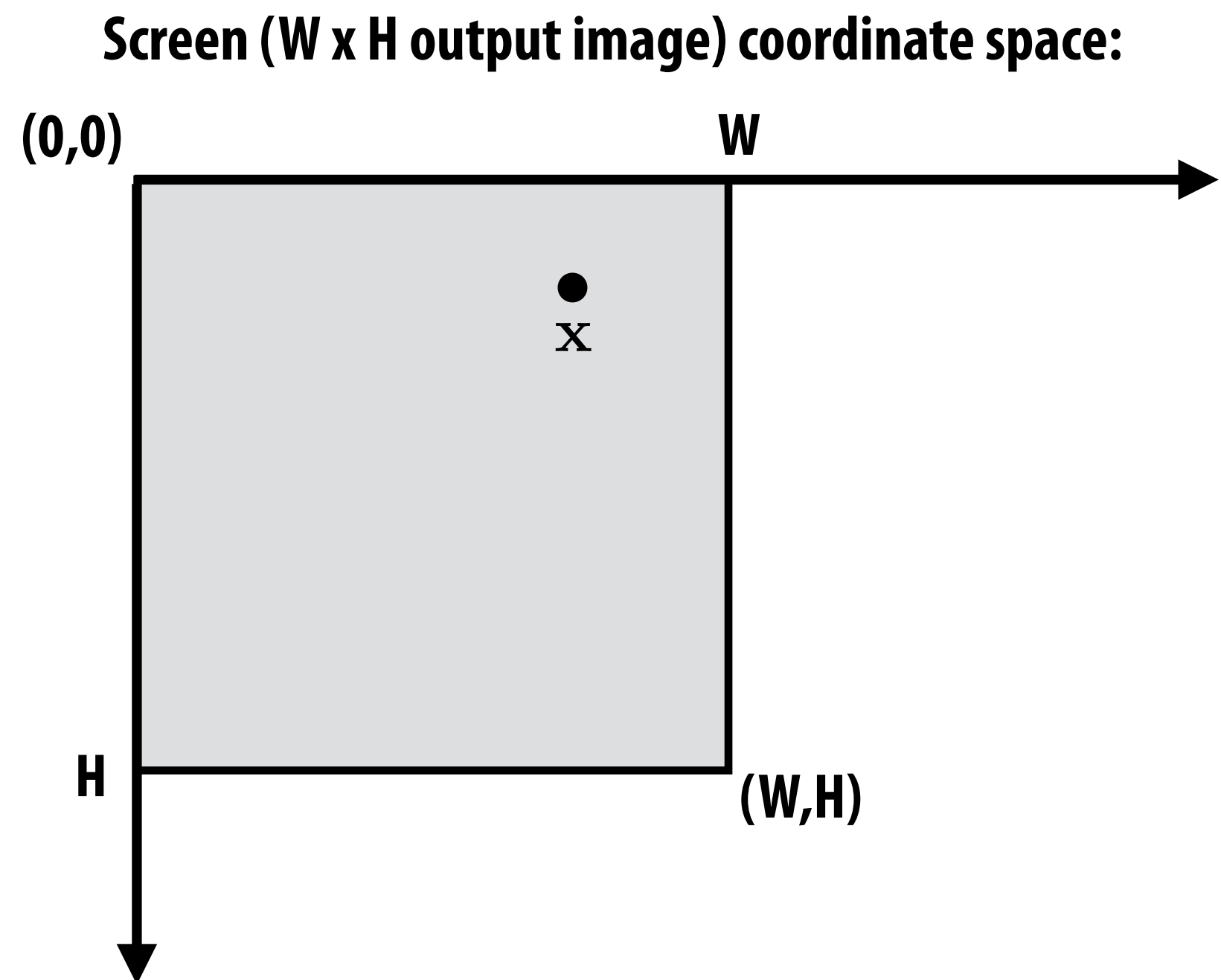
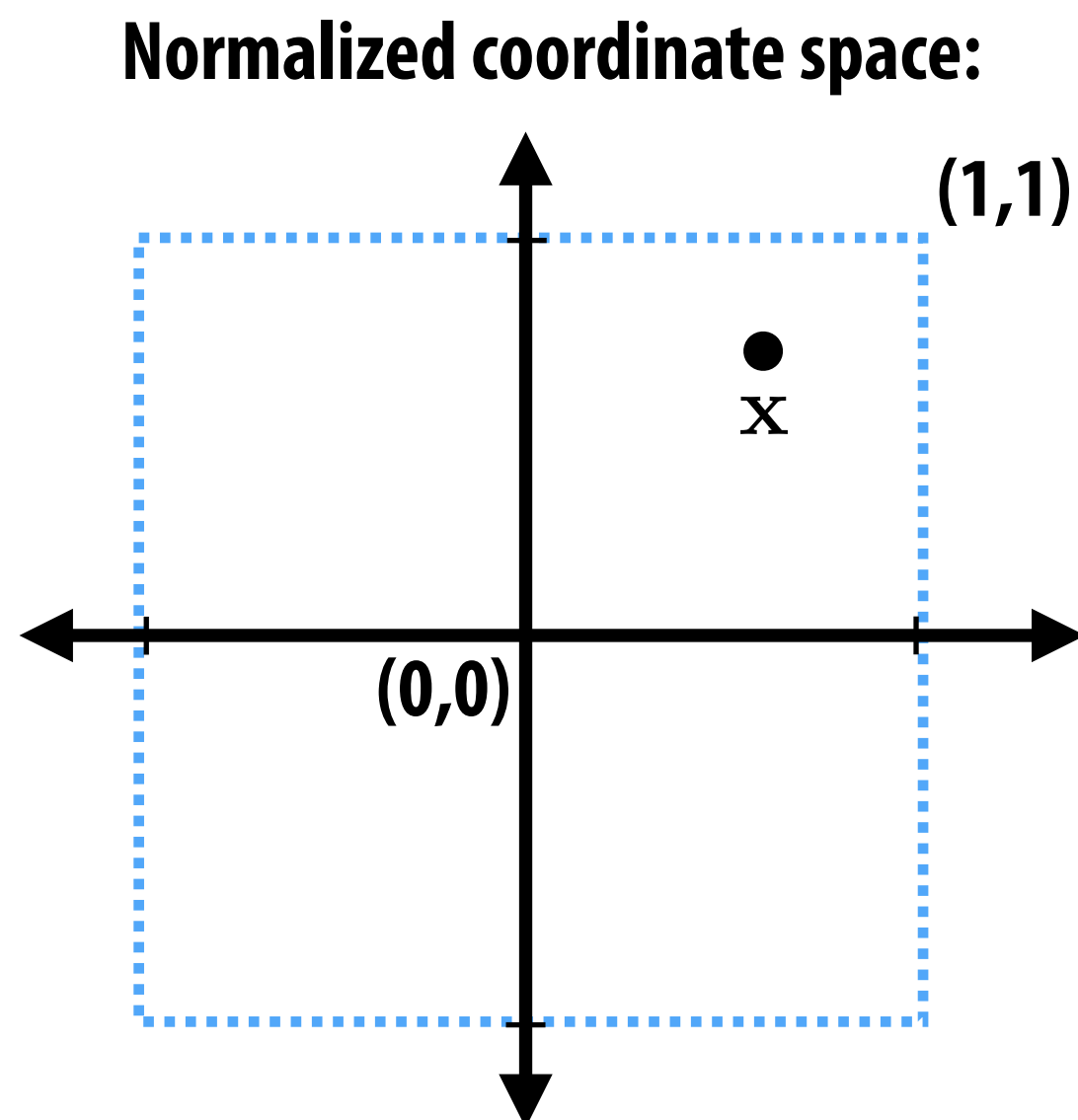
Screen transformation *

Convert points in normalized coordinate space to screen pixel coordinates

Example:

All points within $(-1,1)$ to $(1,1)$ region are on screen

$(1,1)$ in normalized space maps to $(W,0)$ in screen



Step 1: reflect about x

Step 2: translate by $(1,1)$

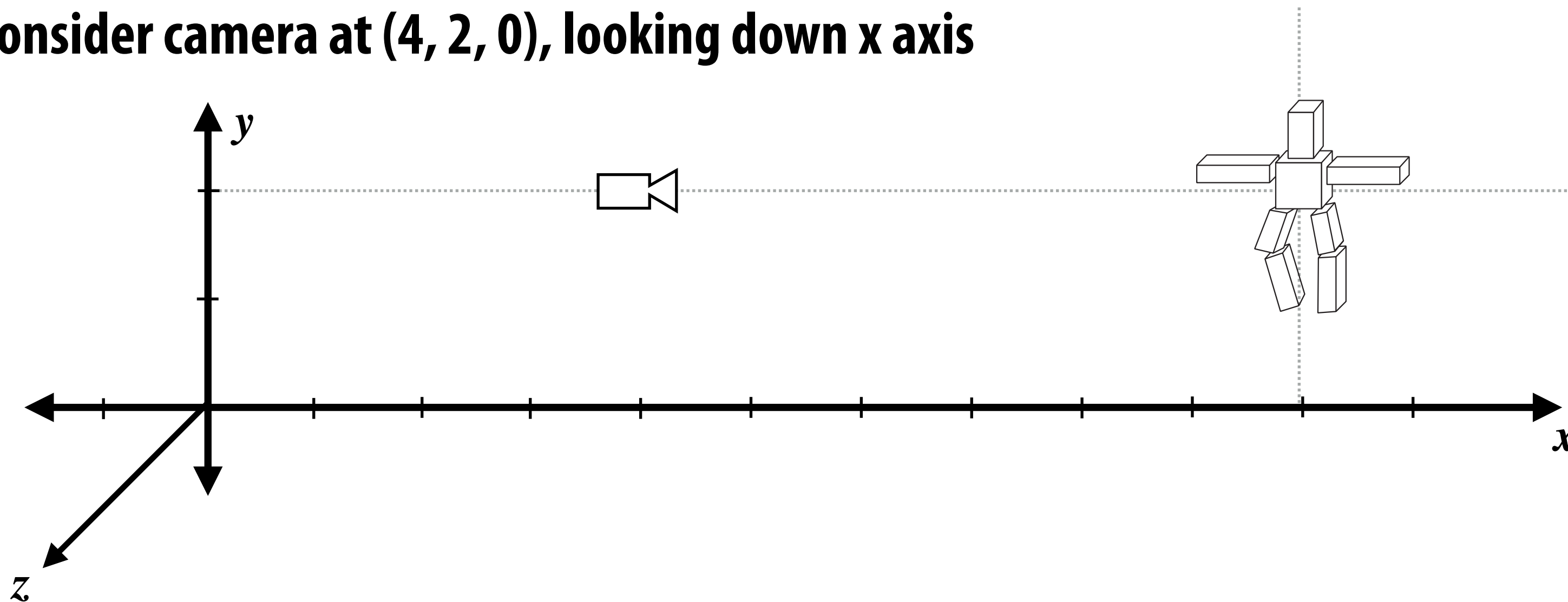
Step 3: scale by $(W/2, H/2)$

* Adopting convention that top-left of screen is $(0,0)$ to match SVG convention in Assignment 1.

Many 3D graphics systems like OpenGL place $(0,0)$ in bottom-left. In this case what would the transform be?

Example: simple camera transform

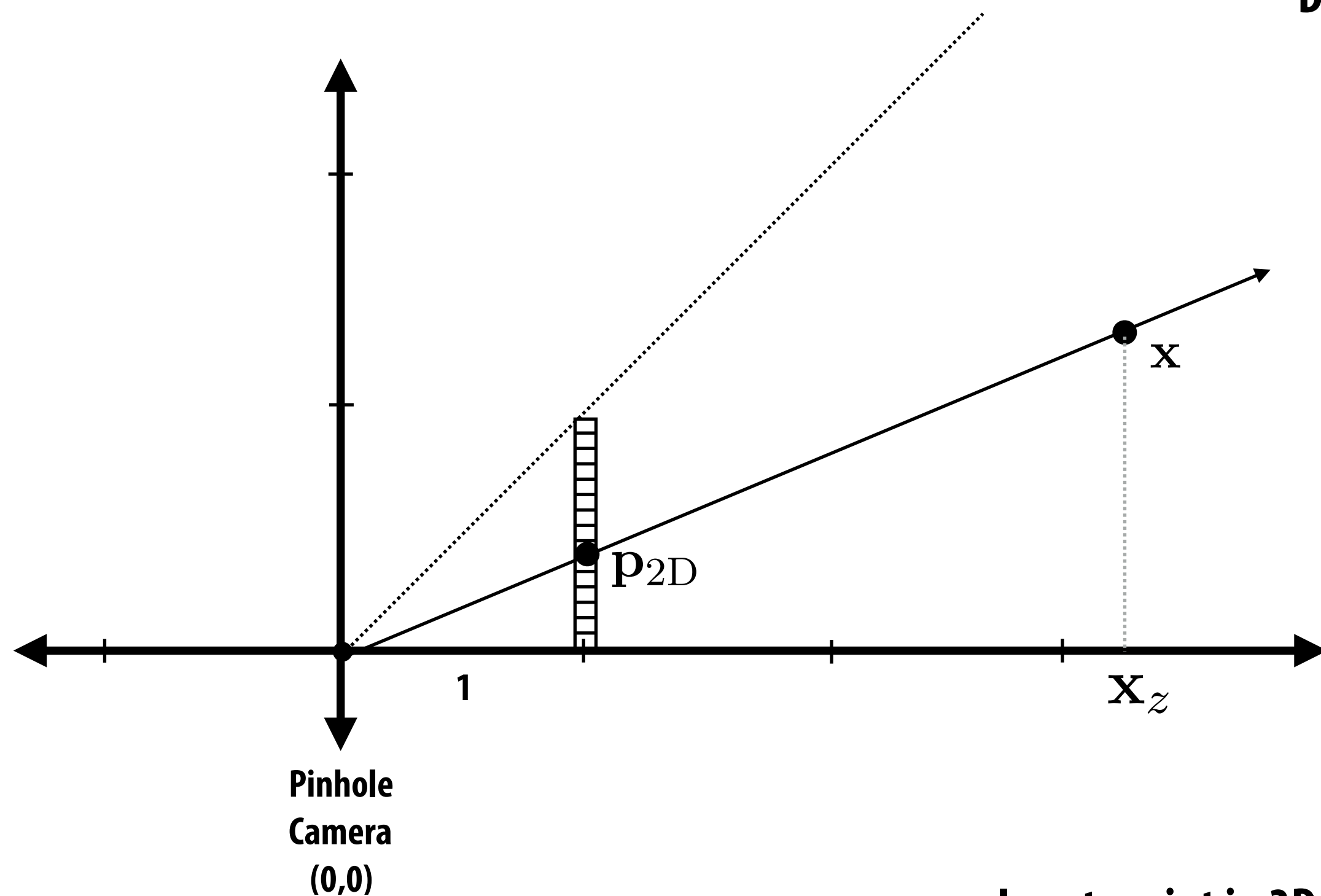
- Consider object in world at $(10, 2, 0)$
- Consider camera at $(4, 2, 0)$, looking down x axis



- Translating object vertex positions by $(-4, -2, 0)$ yields position relative to camera.
- Rotation about y by $-\pi/2$ gives position of object in coordinate system where camera's view direction is aligned with the z axis *

* The convenience of such a coordinate system will become clear on the next slide!

Basic perspective projection



Desired perspective projected result (2D point):

$$\mathbf{p}_{2D} = \left[\mathbf{x}_x / \mathbf{x}_z \quad \mathbf{x}_y / \mathbf{x}_z \right]^T$$

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Input: point in 3D-H

$$\mathbf{x} = \left[\mathbf{x}_x \quad \mathbf{x}_y \quad \mathbf{x}_z \quad 1 \right]$$

After applying \mathbf{P} : point in 3D-H

$$\mathbf{P}\mathbf{x} = \left[\mathbf{x}_x \quad \mathbf{x}_y \quad \mathbf{x}_z \quad \mathbf{x}_z \right]^T$$

After homogeneous divide:

$$\left[\mathbf{x}_x / \mathbf{x}_z \quad \mathbf{x}_y / \mathbf{x}_z \quad 1 \right]^T$$

(throw out third component)

Assumption:
Pinhole camera at (0,0) looking down z

Much more about perspective in later lecture!

Transformations 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

