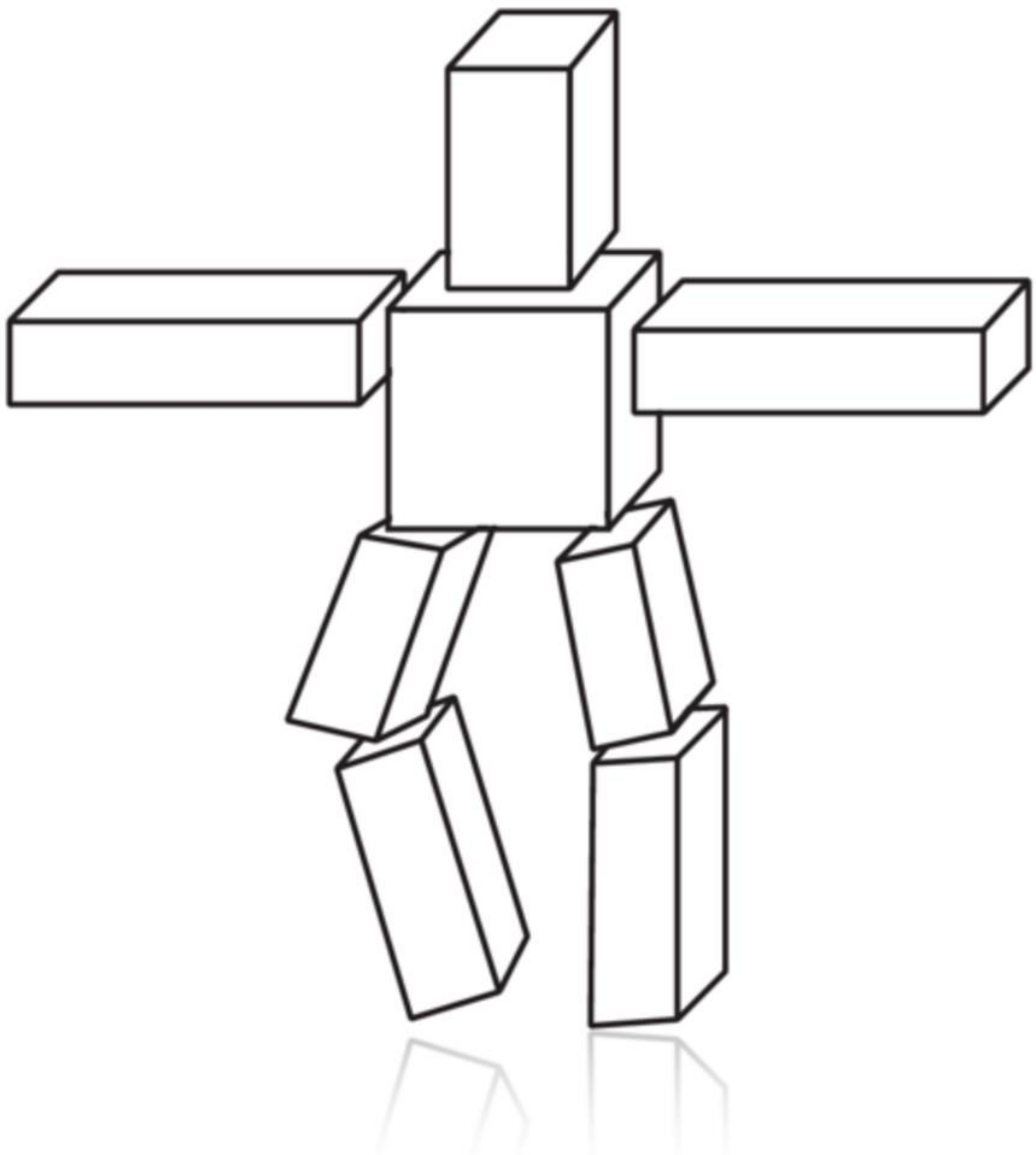
Lecture 3:

Transforms

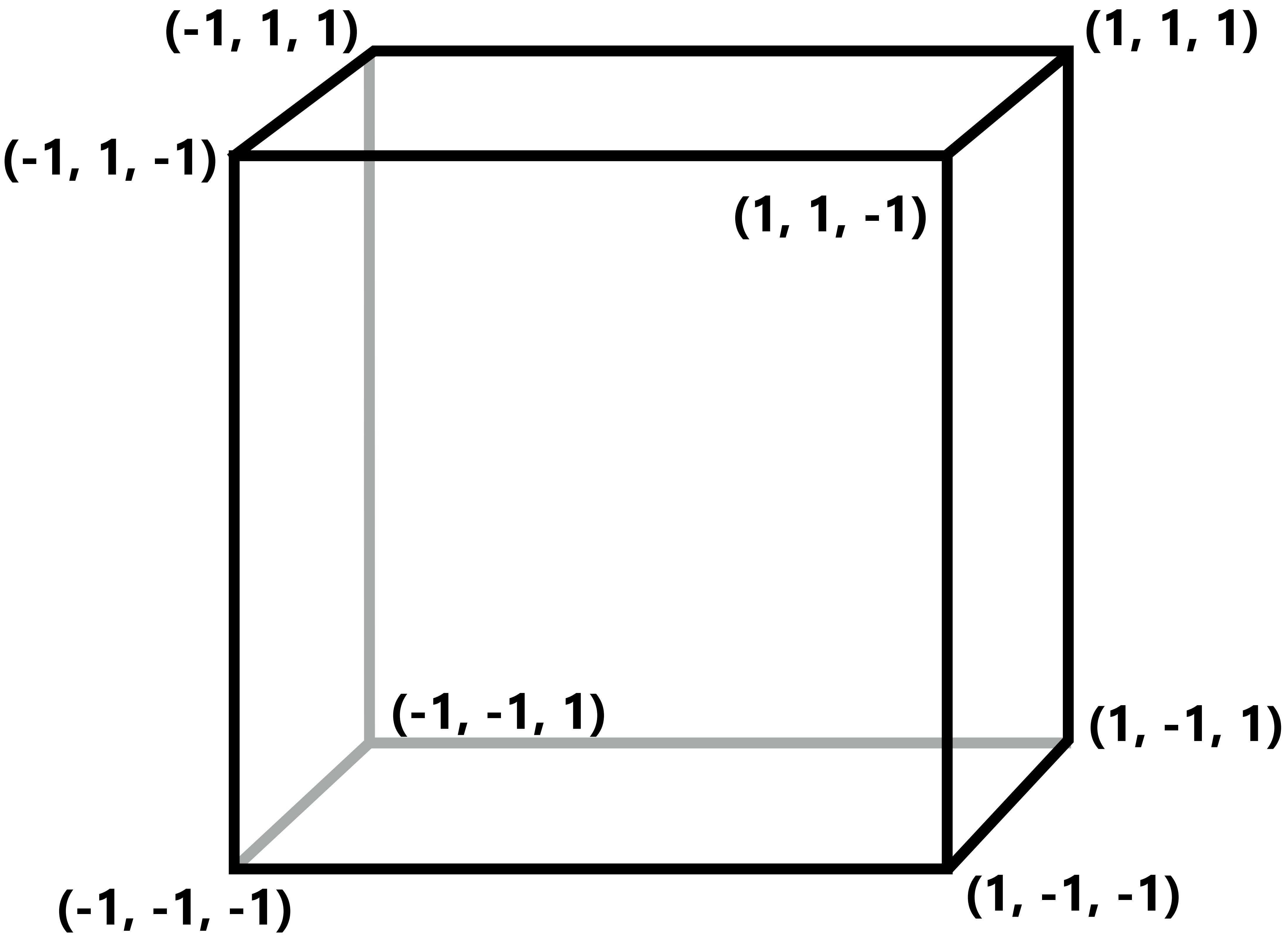
Brief recap…

- We now know how to…
  - represent/model a cube
  - rasterize edges/faces

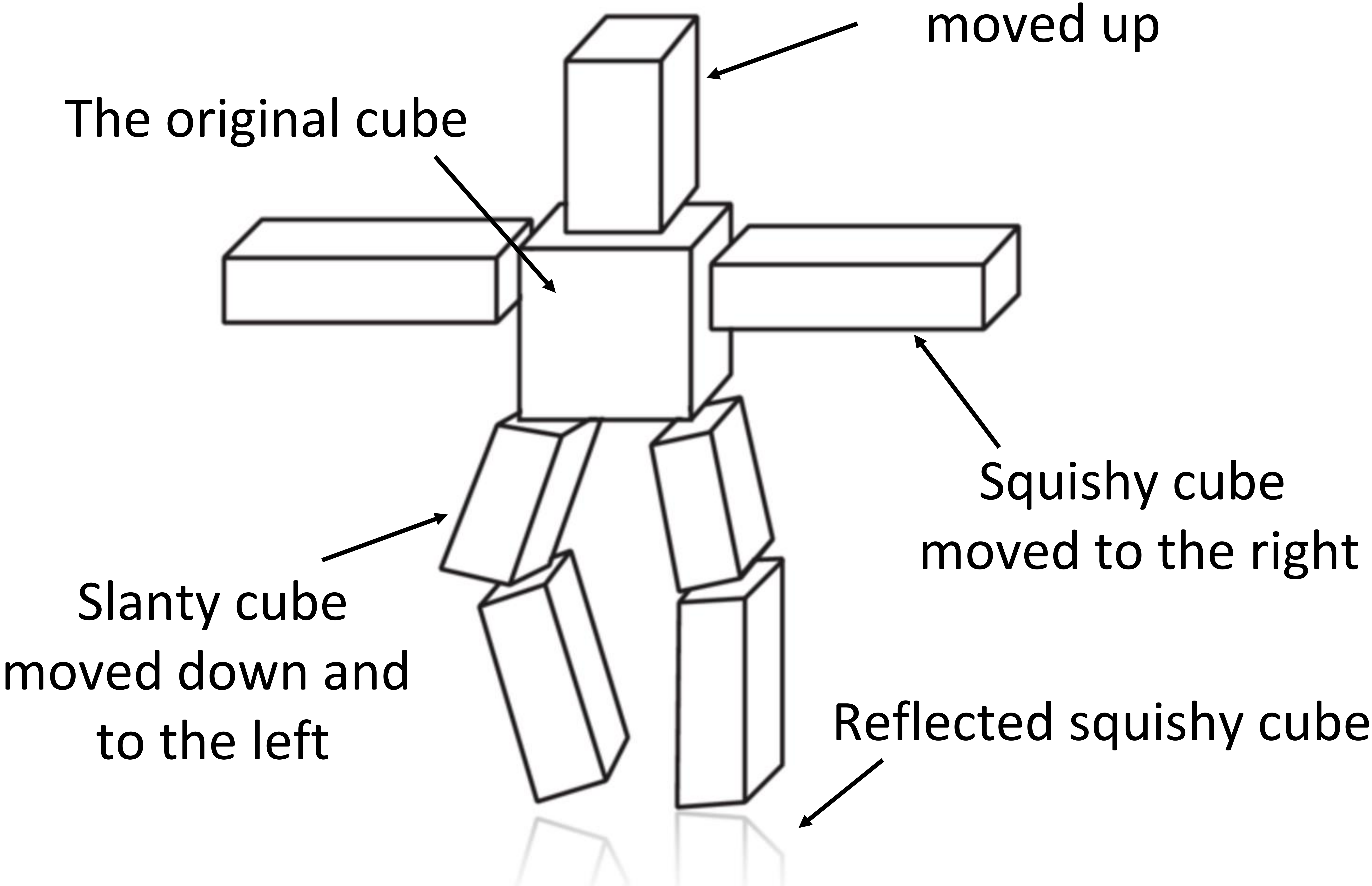
Now, what in the world is this?



Cube



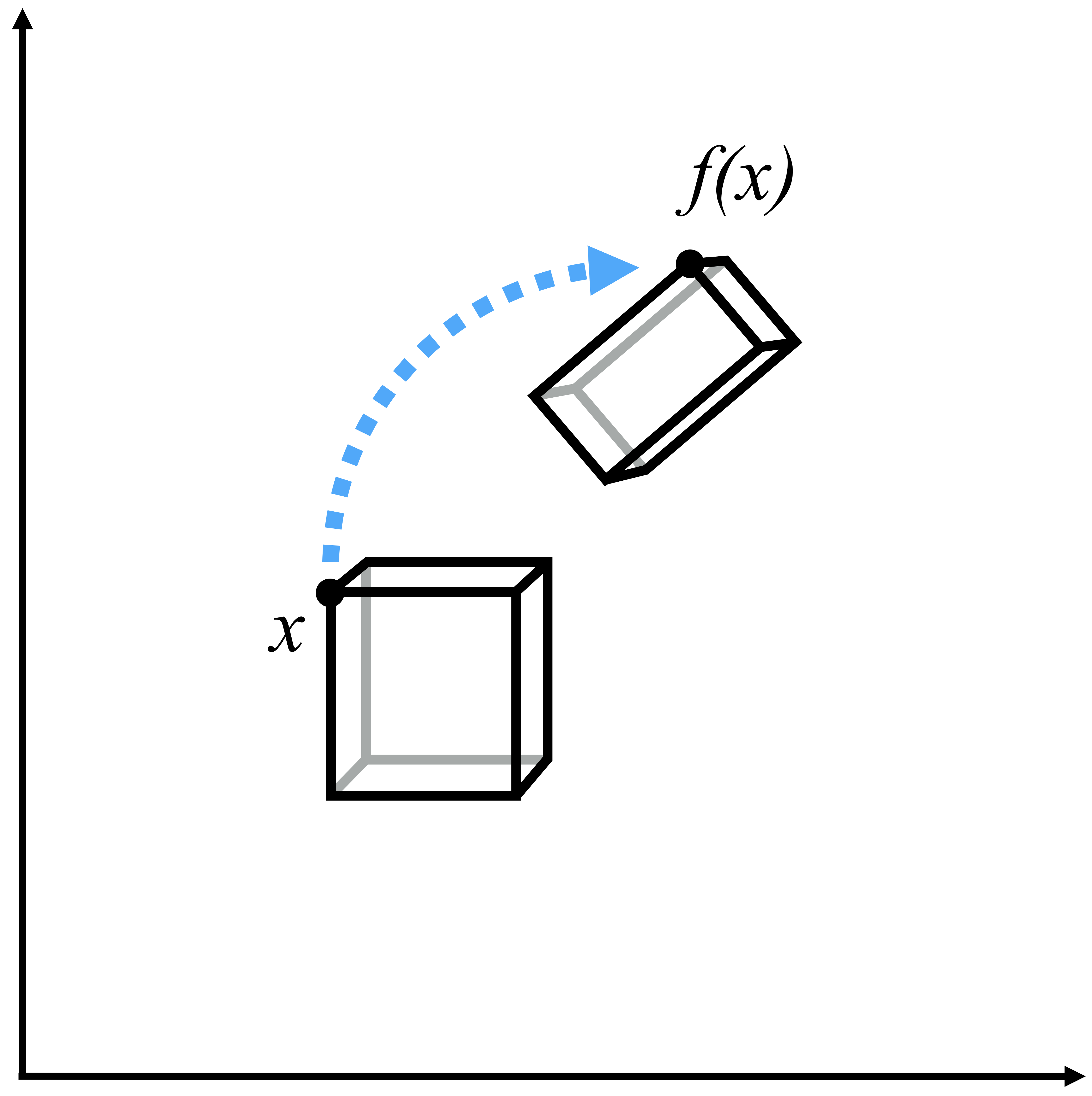
Cube man



It is just a collection of cubes that have been transformed!

Transformations are everywhere in CG…

f transforms x to $f(x)$

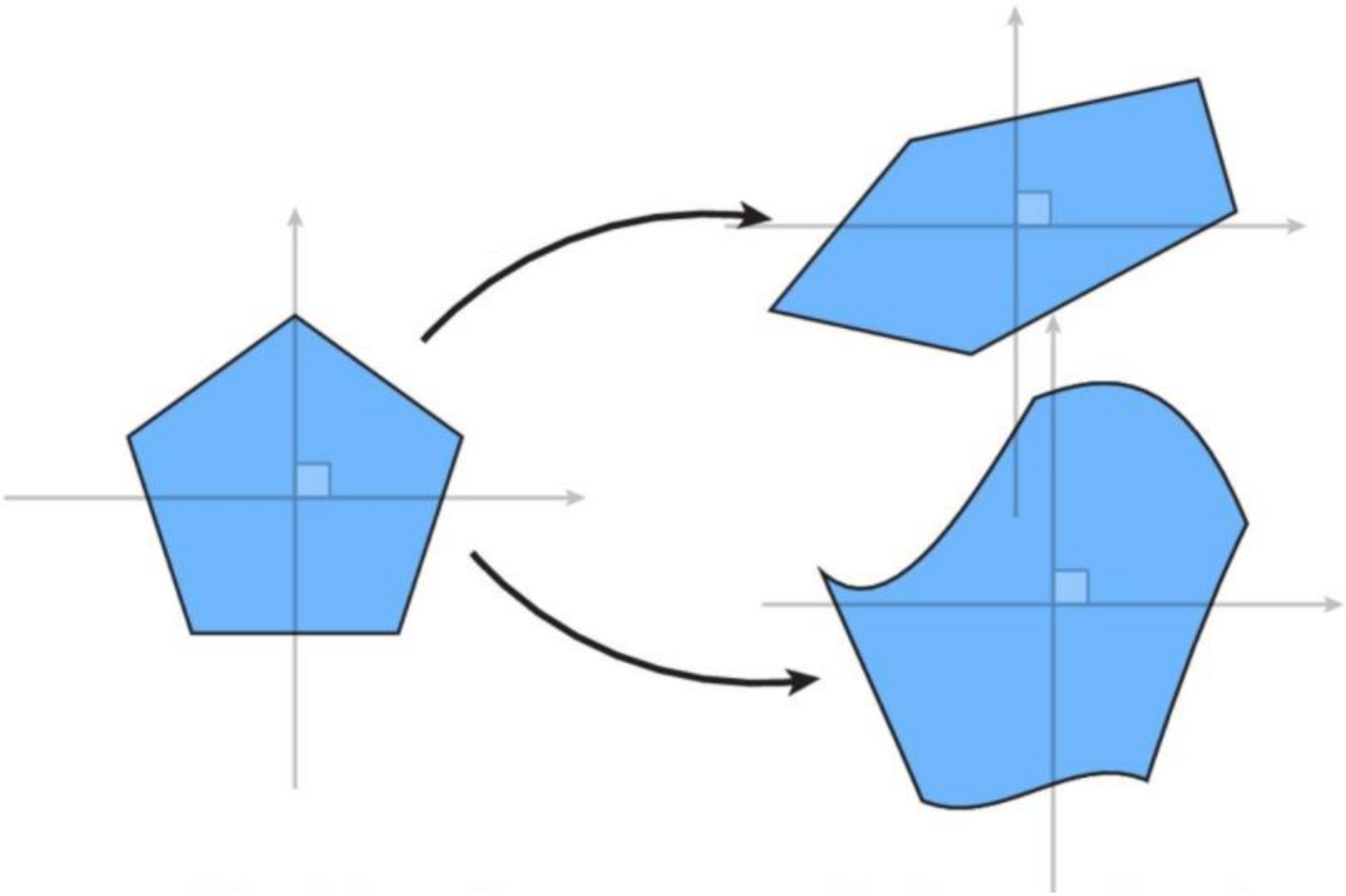


Linear transforms

- Linear algebra: the study of vector spaces and linear maps between them
- We'll get to what linear maps are in just one second
- But why (limit our scope to) linear maps?
  - Computationally speaking, easy to solve equations involving linear maps
  - Still very powerful!
  - Over a short distance, or a small amount of time, *all* maps can be approximated as linear maps (Taylor's theorem). This is used all over geometry, animation, rendering, image processing, etc…
  - Composition of linear transformations is linear, leading to uniform representation of transformations (e.g. in graphics card hardware and graphics API)

Linear maps

- **What is a linear map?**



Key idea: **linear maps take lines to lines...**

...while keeping the origin fixed.

Linear maps – algebraic definition

- **A map *f* is linear if it maps vectors to vectors, and if for all vectors *u*, *v* and scalars *a* we have:**

$$f(\boldsymbol{u} + \boldsymbol{v}) = f(\boldsymbol{u}) + f(\boldsymbol{v})$$

$$f(a\boldsymbol{u}) = af(\boldsymbol{u})$$

$$\begin{tikzcd}
\mathbf{x}, \mathbf{y} \arrow[r, "\text{add first}"] \arrow[d, "\text{apply f first}"] & \mathbf{x} + \mathbf{y} \arrow[d, "\text{then apply f}"] \\
f(\mathbf{x}), f(\mathbf{y}) \arrow[r, "\text{then add}"] & \begin{array}{c} f(\mathbf{x}) + f(\mathbf{y}) \\ = \\ f(\mathbf{x} + \mathbf{y}) \end{array}
\end{tikzcd}$$

Linear transforms

- **For maps between R^m and R^n (e.g. a map from 3D to 2D), we can give an even more explicit definition:**

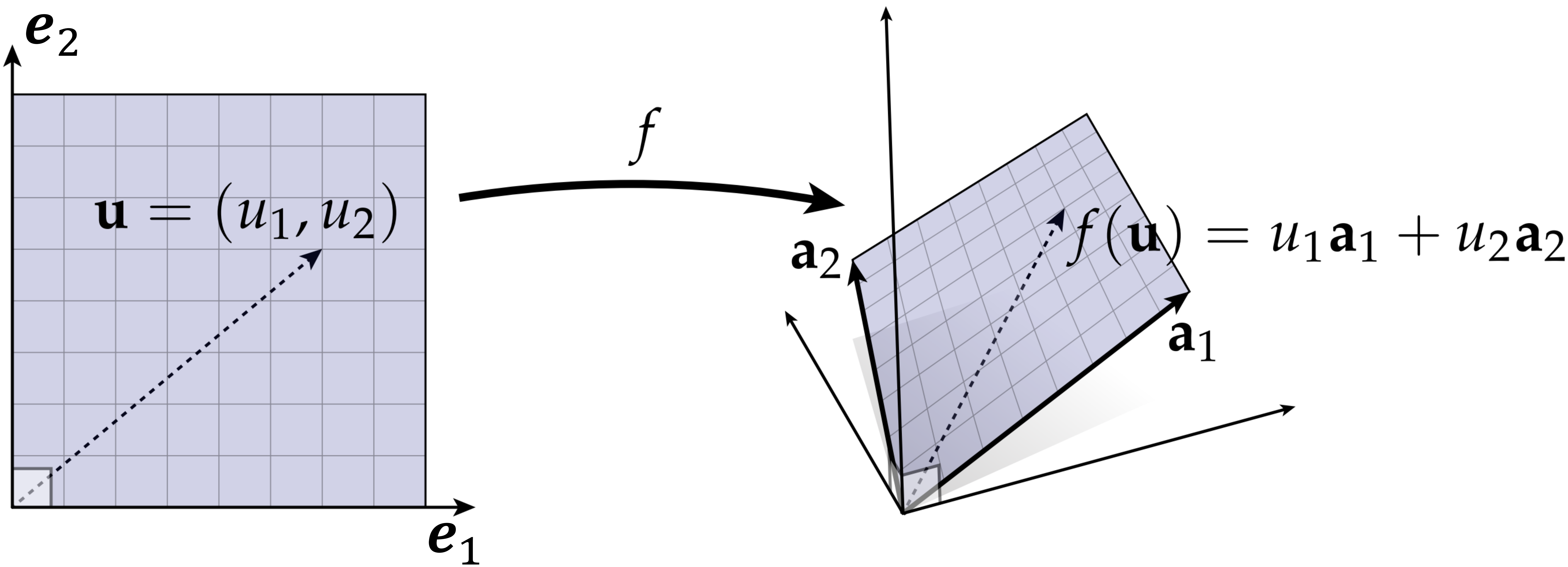
If a map can be expressed as

$$f(\boldsymbol{u}) = \sum_{i=1}^{m} u_i \boldsymbol{a}_i$$

with fixed vectors a_i, then it is linear

- **How do you show that this is a linear map?**
  - **It's called a linear combination**

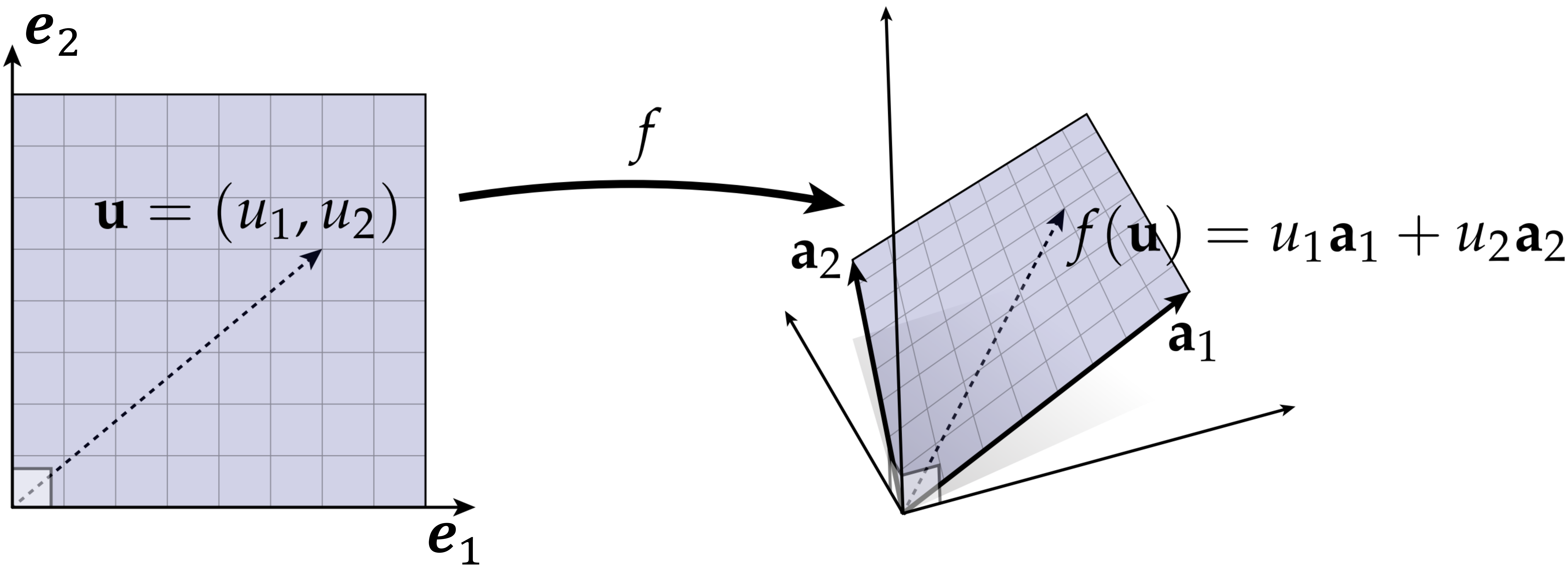
Linear transforms



Do you know…

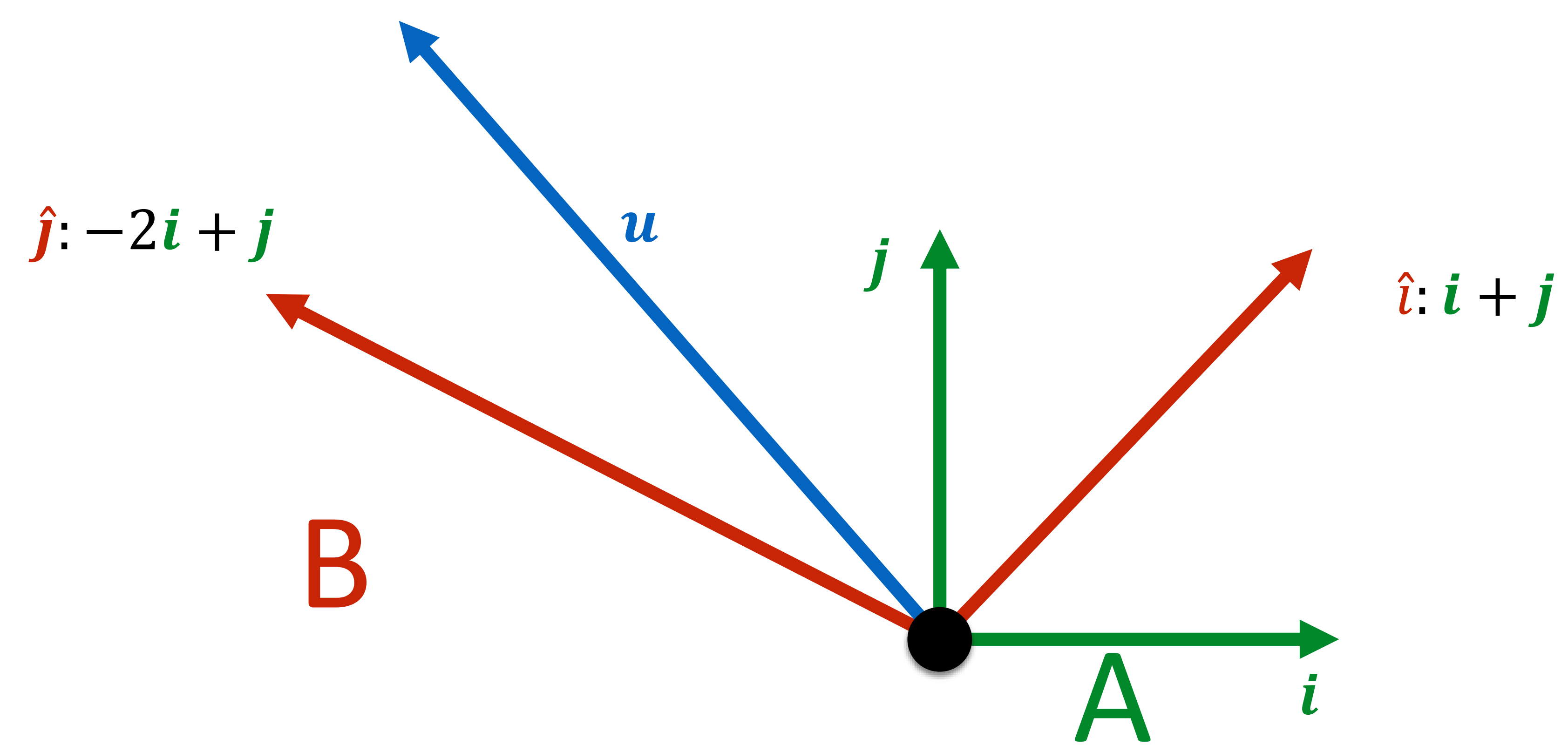
- **what u_1 and u_2 are?**
- **what a_1 and a_2 are?**

Linear transforms



- u is a linear combination of e_1 and e_2
- $f(u)$ is that same linear combination of a_1 and a_2
- a_1 and a_2 are $f(e_1)$ and f(e_2)
- by knowing what e_1 and e_2 map to, you know how to map the entire space!

Coordinate transformations



If vector **u** has coordinates (1,1) when expressed in frame B,

what are its coordinates in frame A?

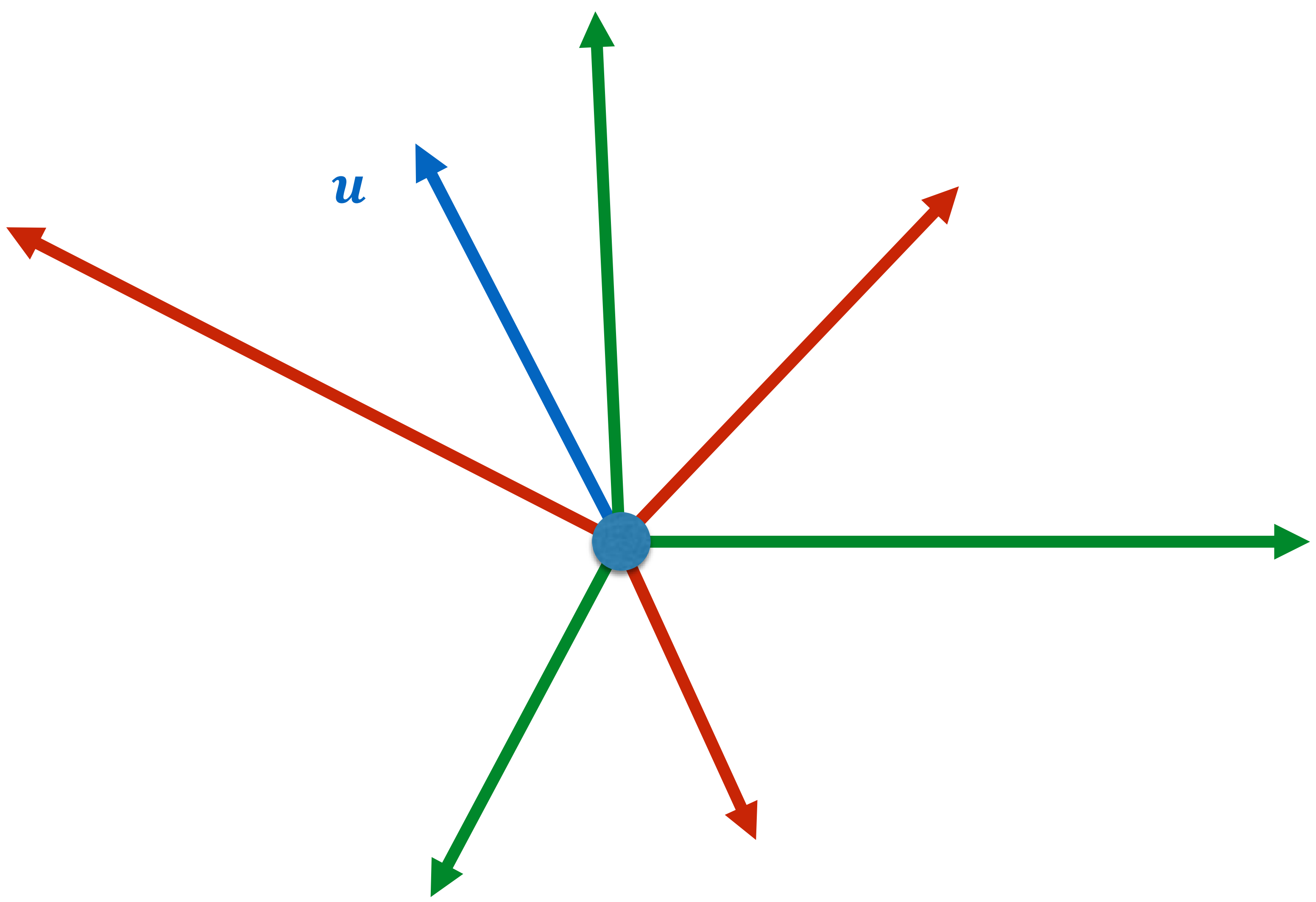
Vector **u** in expressed in coordinate frame B

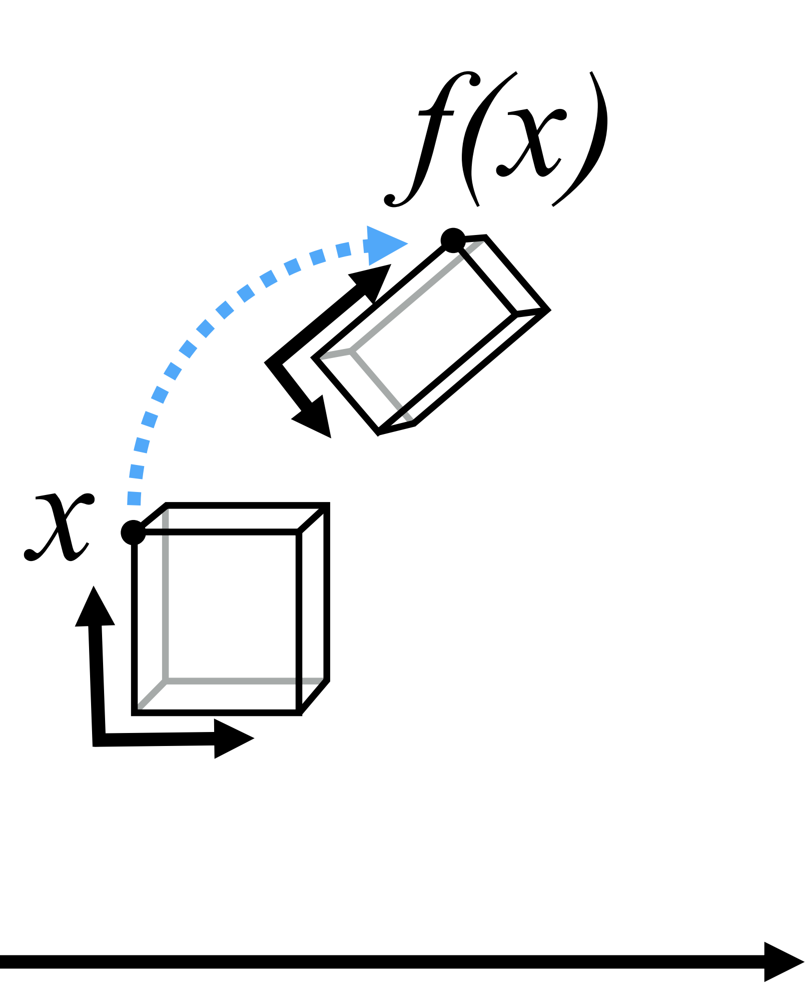
$$\underbrace{\boldsymbol{f}(\boldsymbol{u})} = \boldsymbol{f}(u_1\hat{\boldsymbol{\imath}} + u_2\hat{\boldsymbol{\jmath}}) = u_1\boldsymbol{f}(\hat{\boldsymbol{\imath}}) + u_2\boldsymbol{f}(\hat{\boldsymbol{\jmath}}) = u_1\begin{bmatrix}1\\1\end{bmatrix} + u_2\begin{bmatrix}-2\\1\end{bmatrix}$$

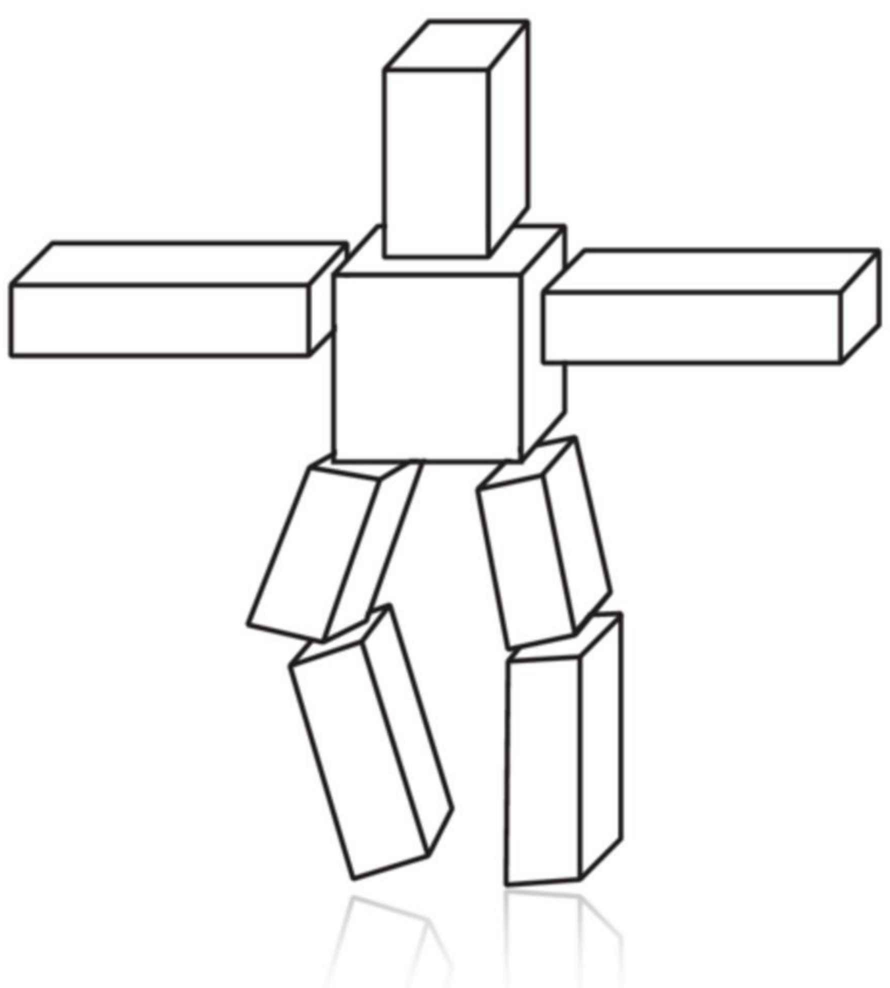
Same vector in coordinate frame A

Linear maps

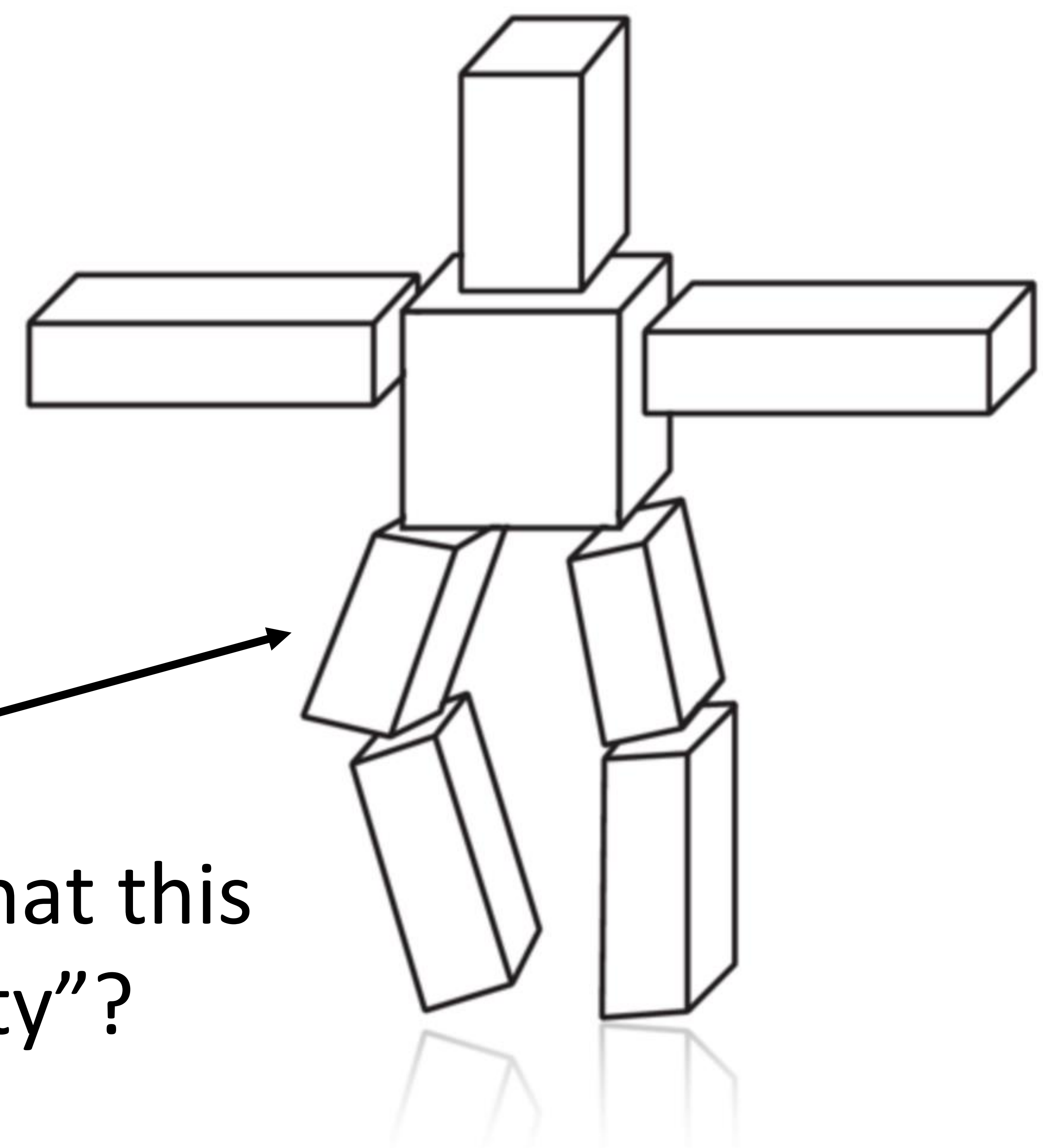
- **In graphics we often talk about changing coordinate frames (go from local to world to camera to screen coordinates)**
- **Equally useful to think about maps transforming a space (and everything in it!)**





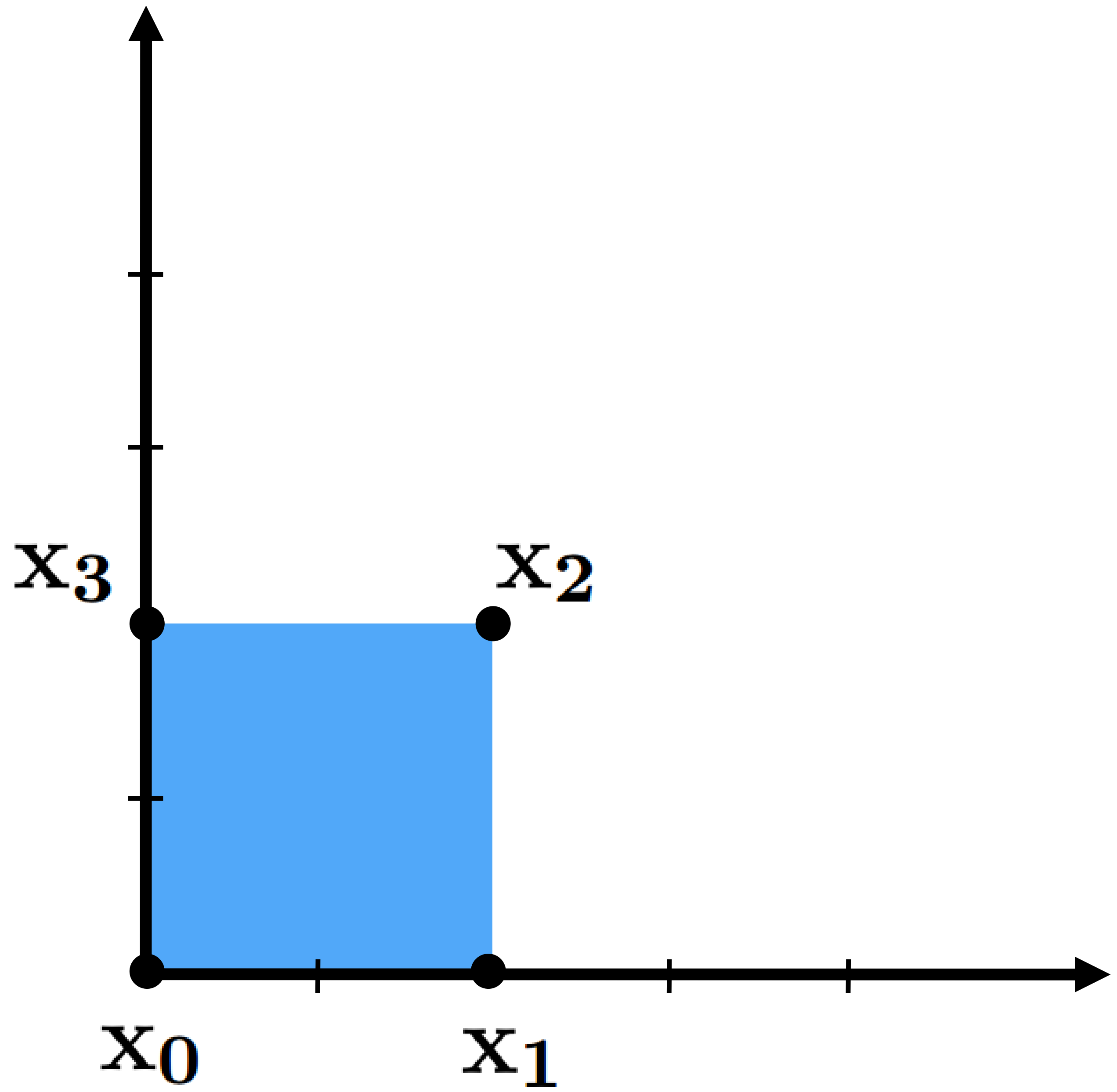


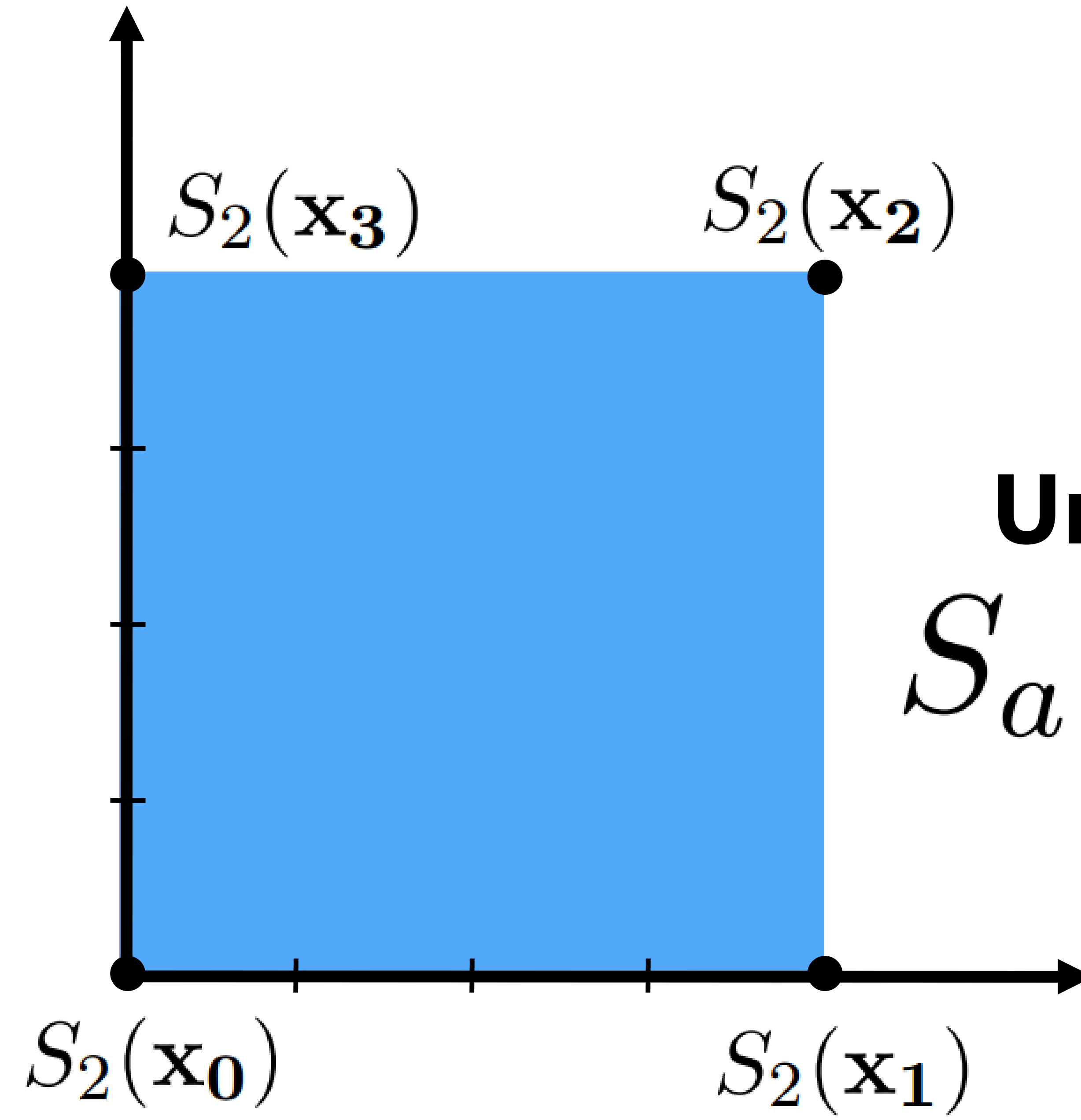
Let’s look at some transforms that are important in graphics…



How do you formally tell a computer that this cube should be “squished and slanty”?

Scale



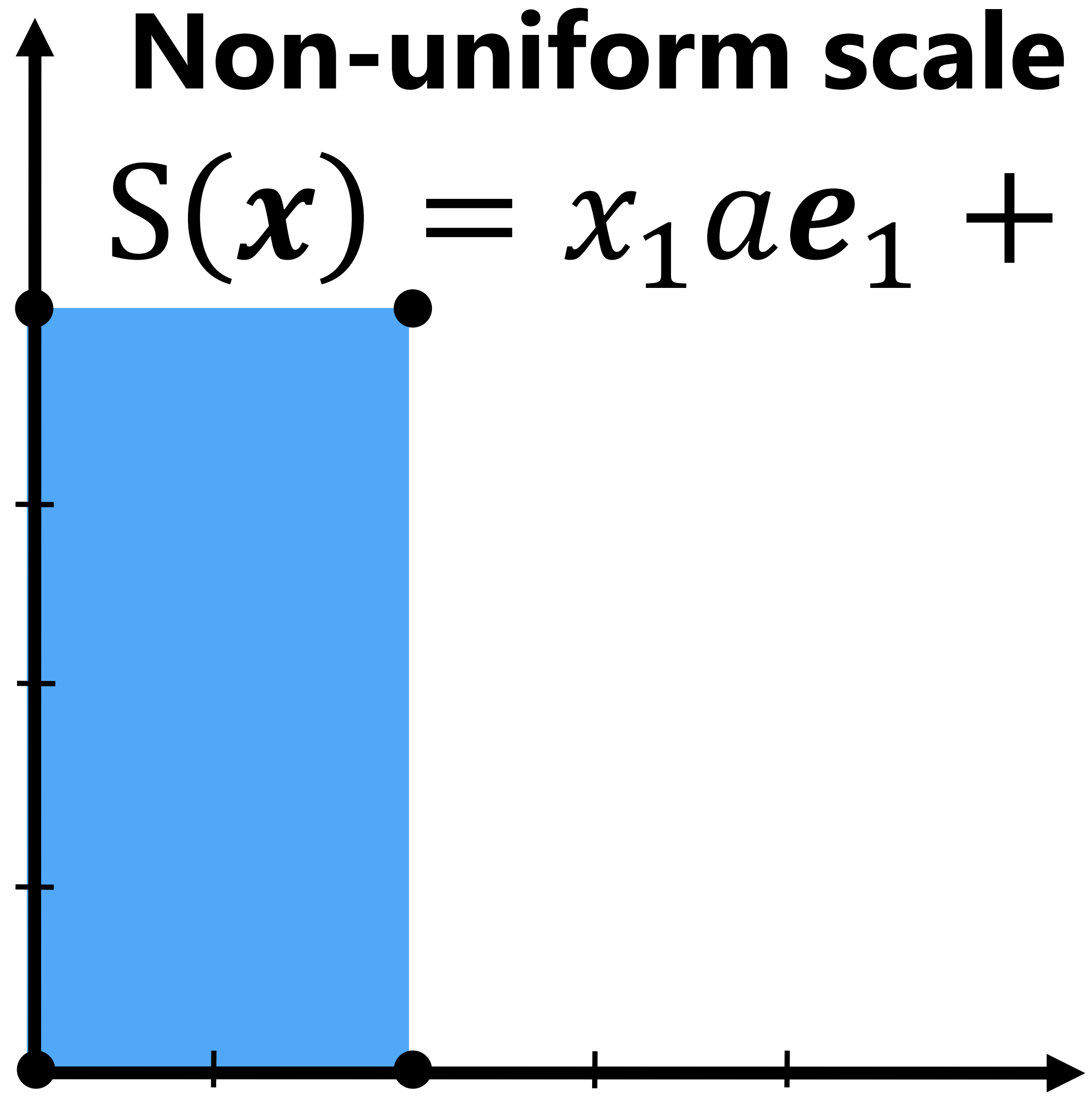


Uniform scale:

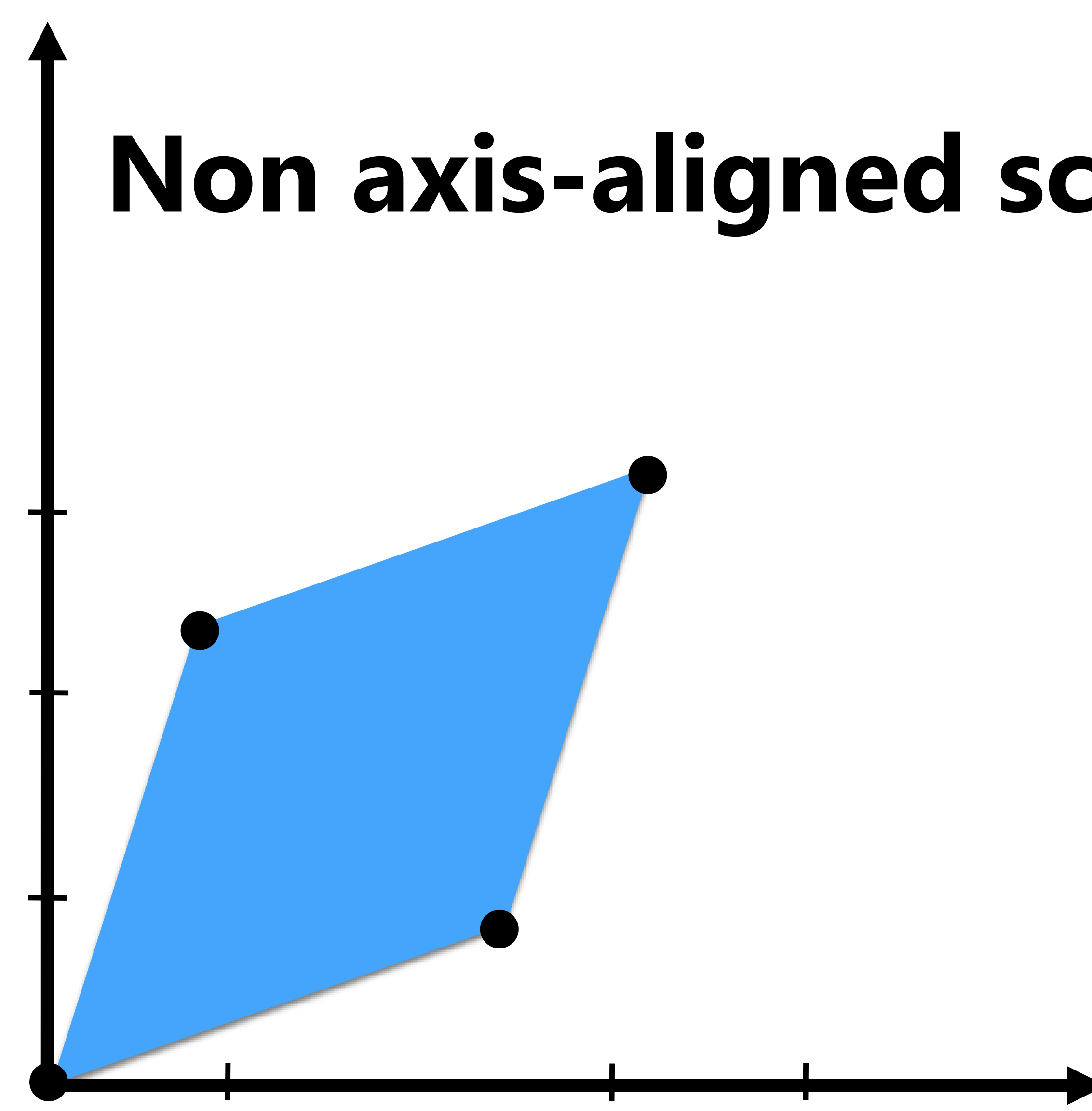
$$S_a(\mathbf{x}) = a\mathbf{x}$$

Non-uniform scale

$$S(\boldsymbol{x}) = x_1 a \boldsymbol{e}_1 + x_2 b \boldsymbol{e}_2$$

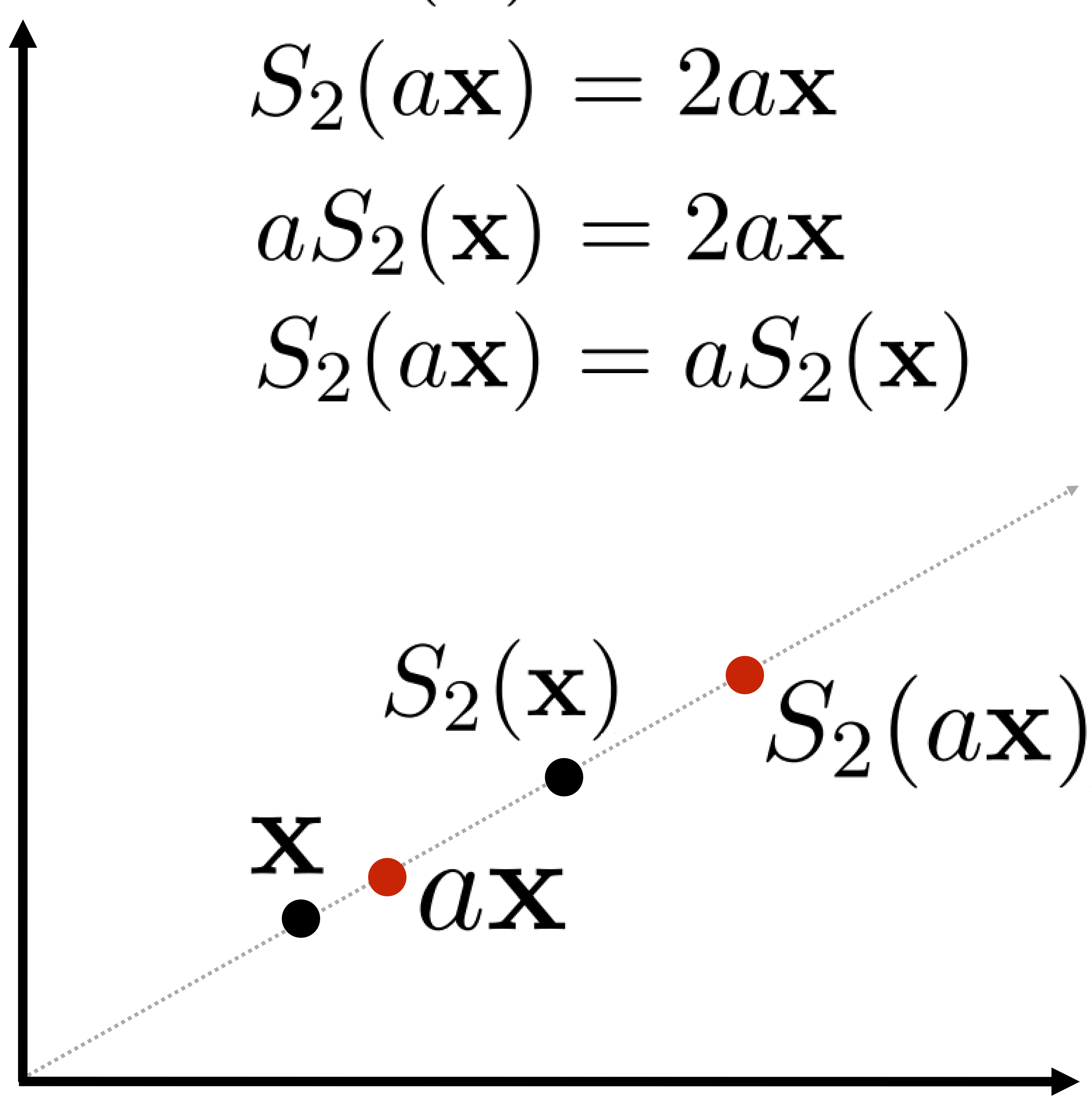


Non axis-aligned scaling??

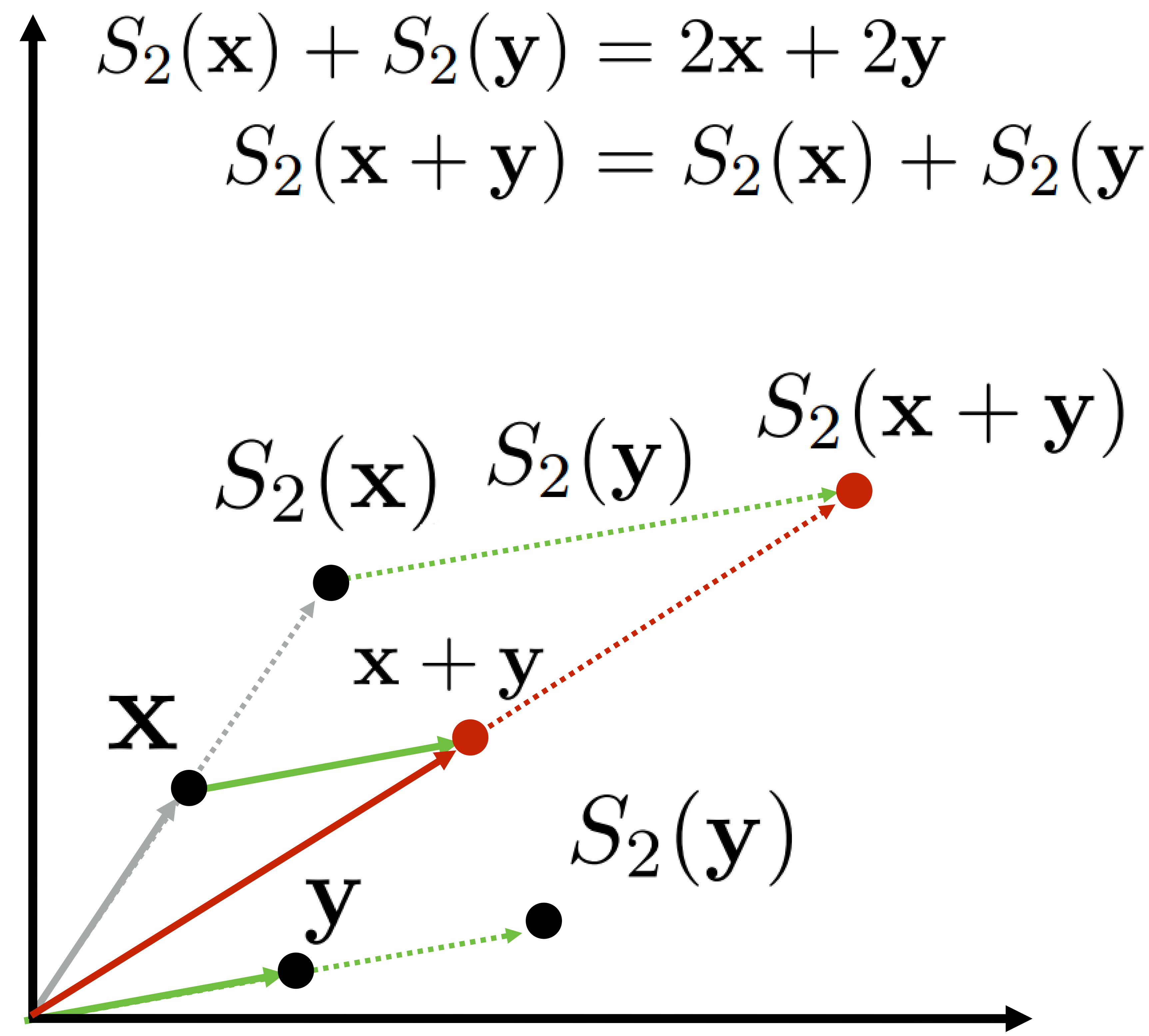


Is uniform scale a linear transform?

$$S_2(\mathbf{x}) = 2\mathbf{x}$$
$$S_2(a\mathbf{x}) = 2a\mathbf{x}$$
$$aS_2(\mathbf{x}) = 2a\mathbf{x}$$
$$S_2(a\mathbf{x}) = aS_2(\mathbf{x})$$

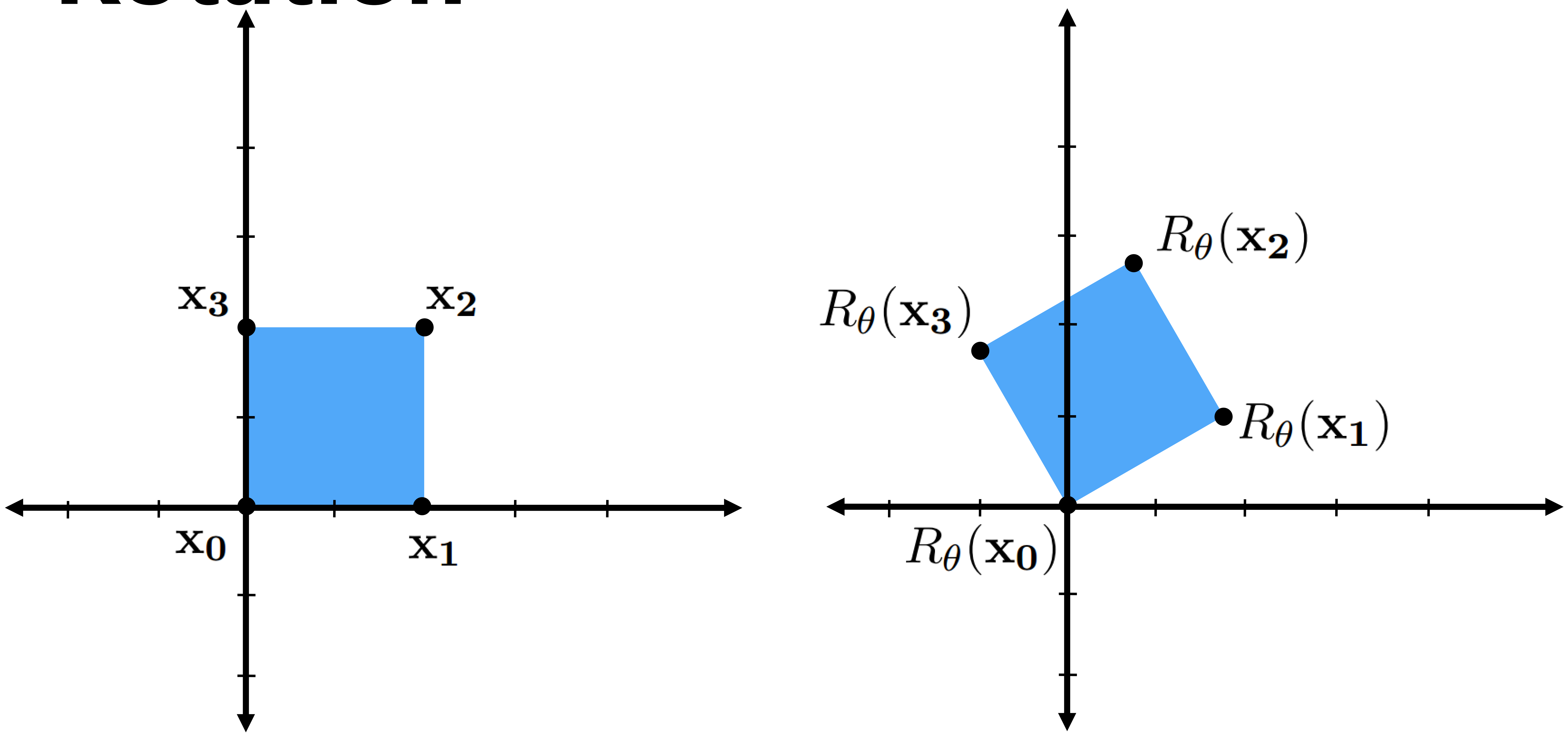


$$S_2(\mathbf{x}+\mathbf{y}) = 2(\mathbf{x}+\mathbf{y})$$
$$S_2(\mathbf{x}) + S_2(\mathbf{y}) = 2\mathbf{x} + 2\mathbf{y}$$
$$S_2(\mathbf{x}+\mathbf{y}) = S_2(\mathbf{x}) + S_2(\mathbf{y})$$



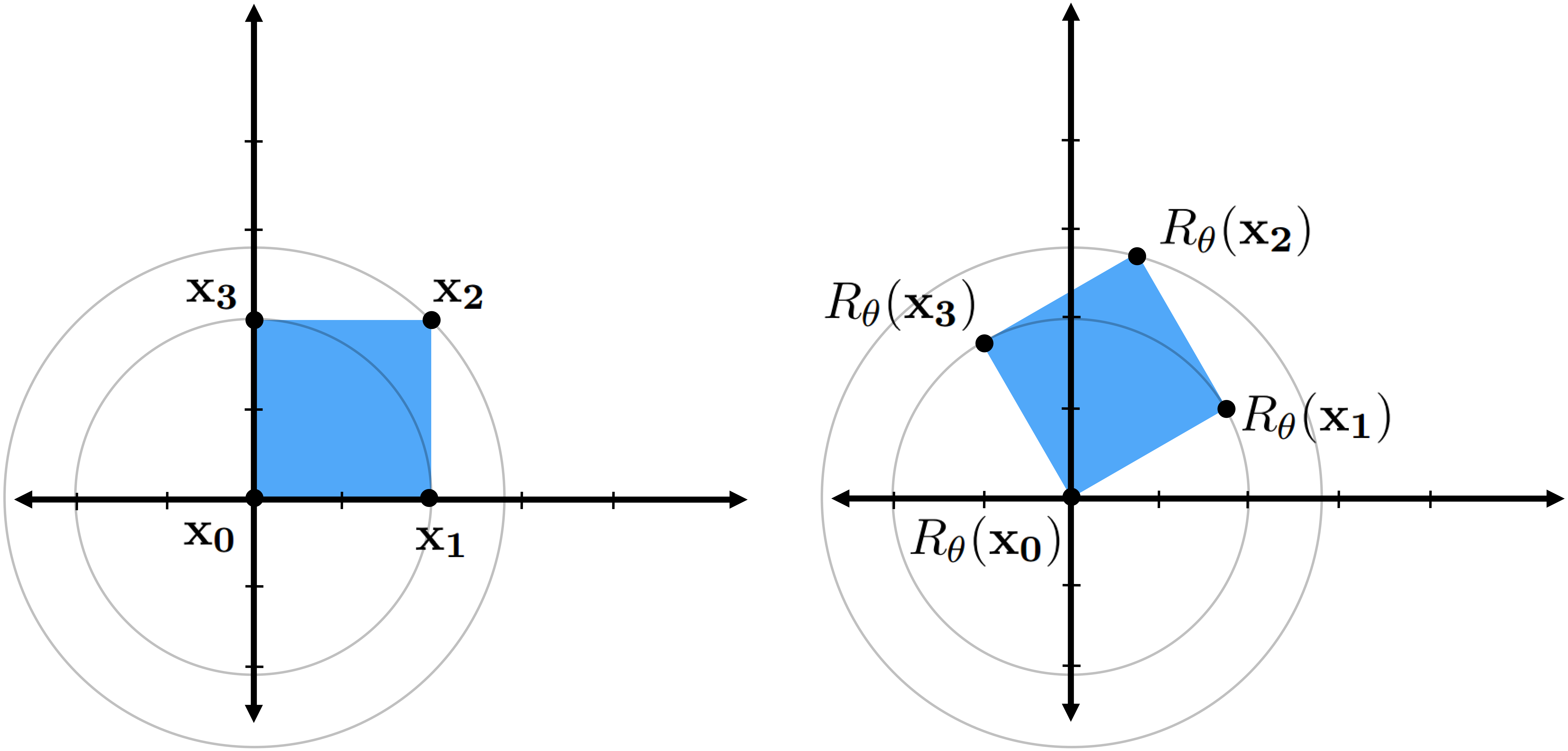
Yes!

Rotation



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

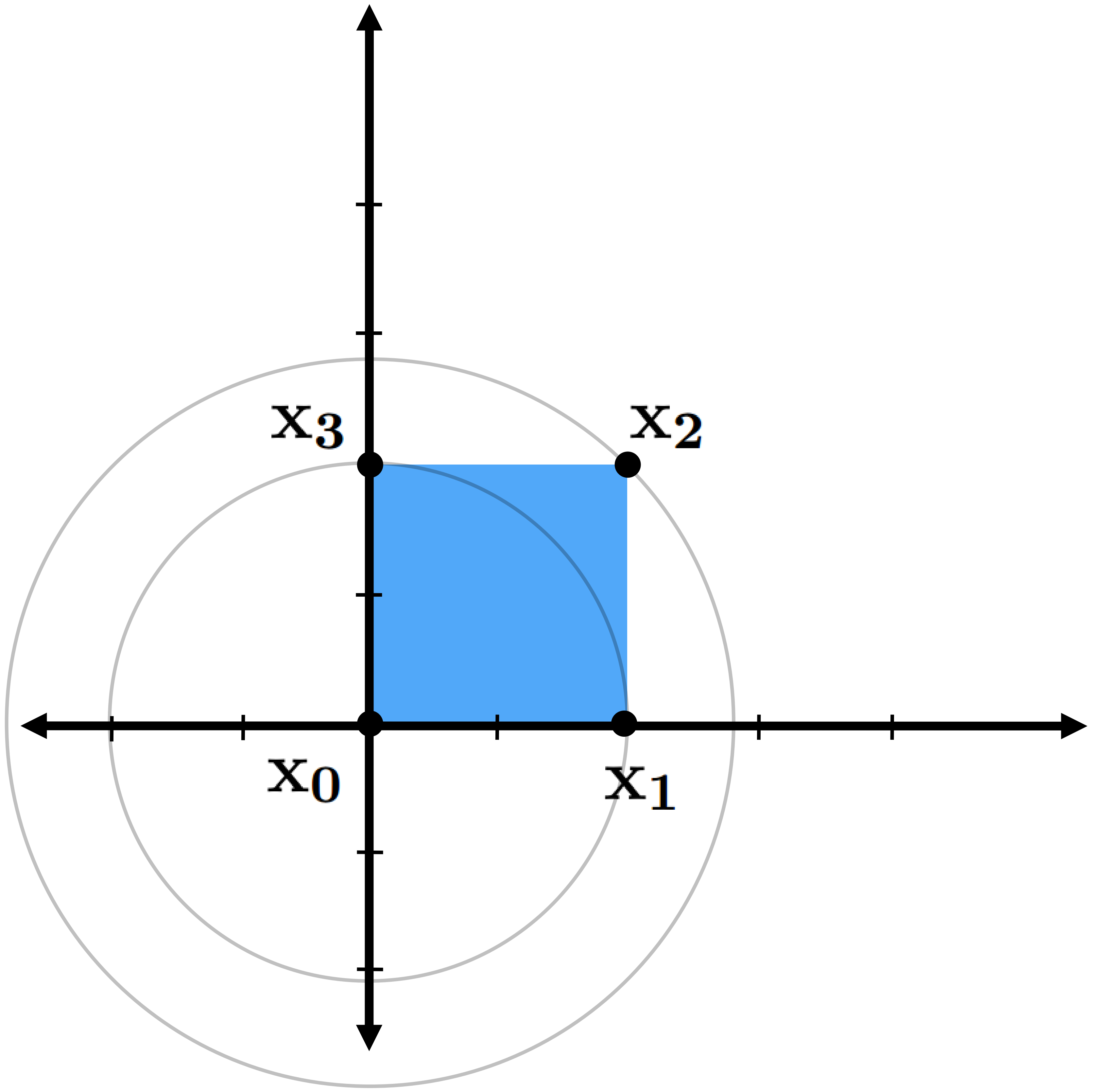
Rotation

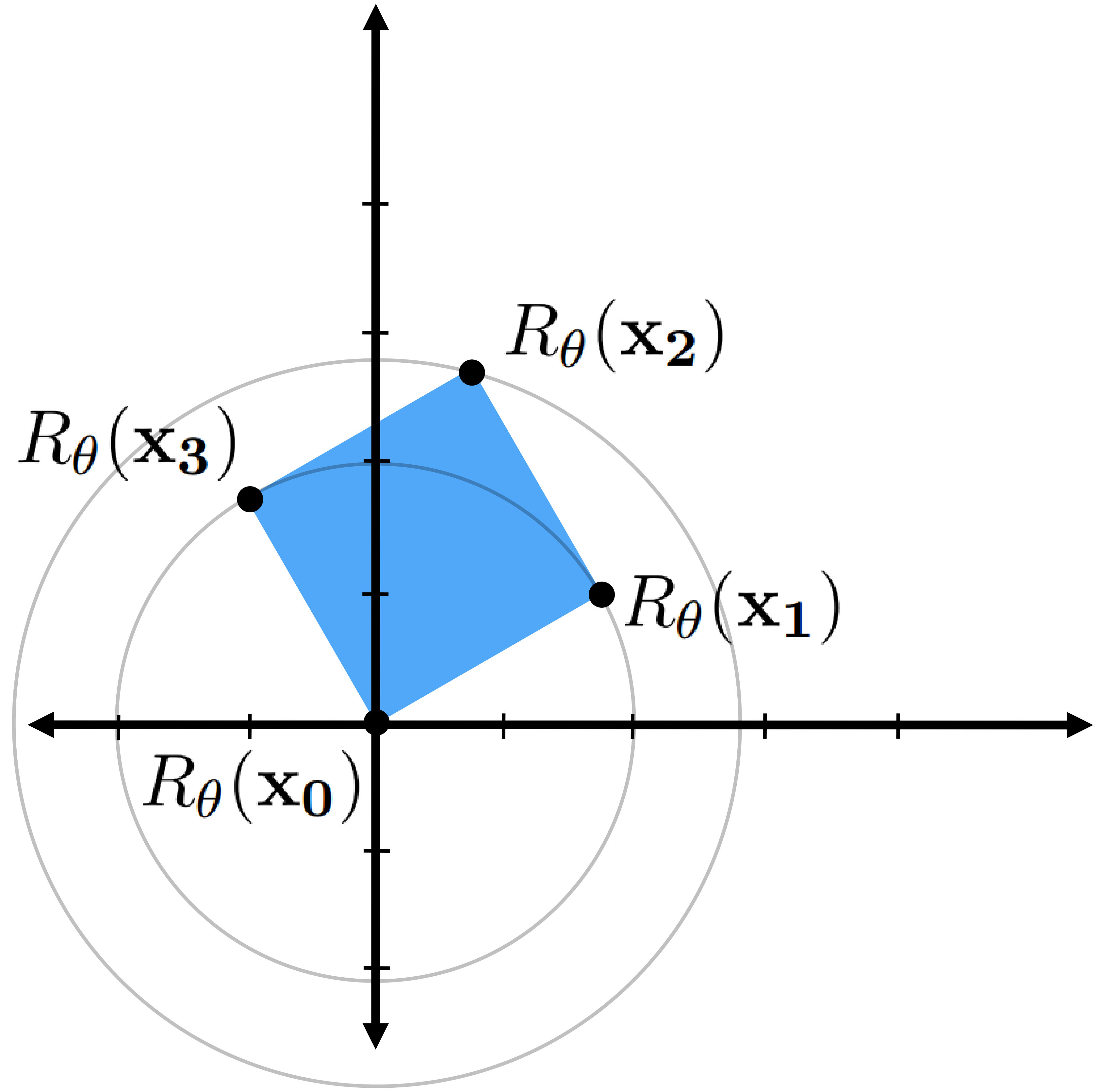


R_θ **= rotate counter-clockwise by** θ

As angle changes, points move along *circular* trajectories.

Rotation





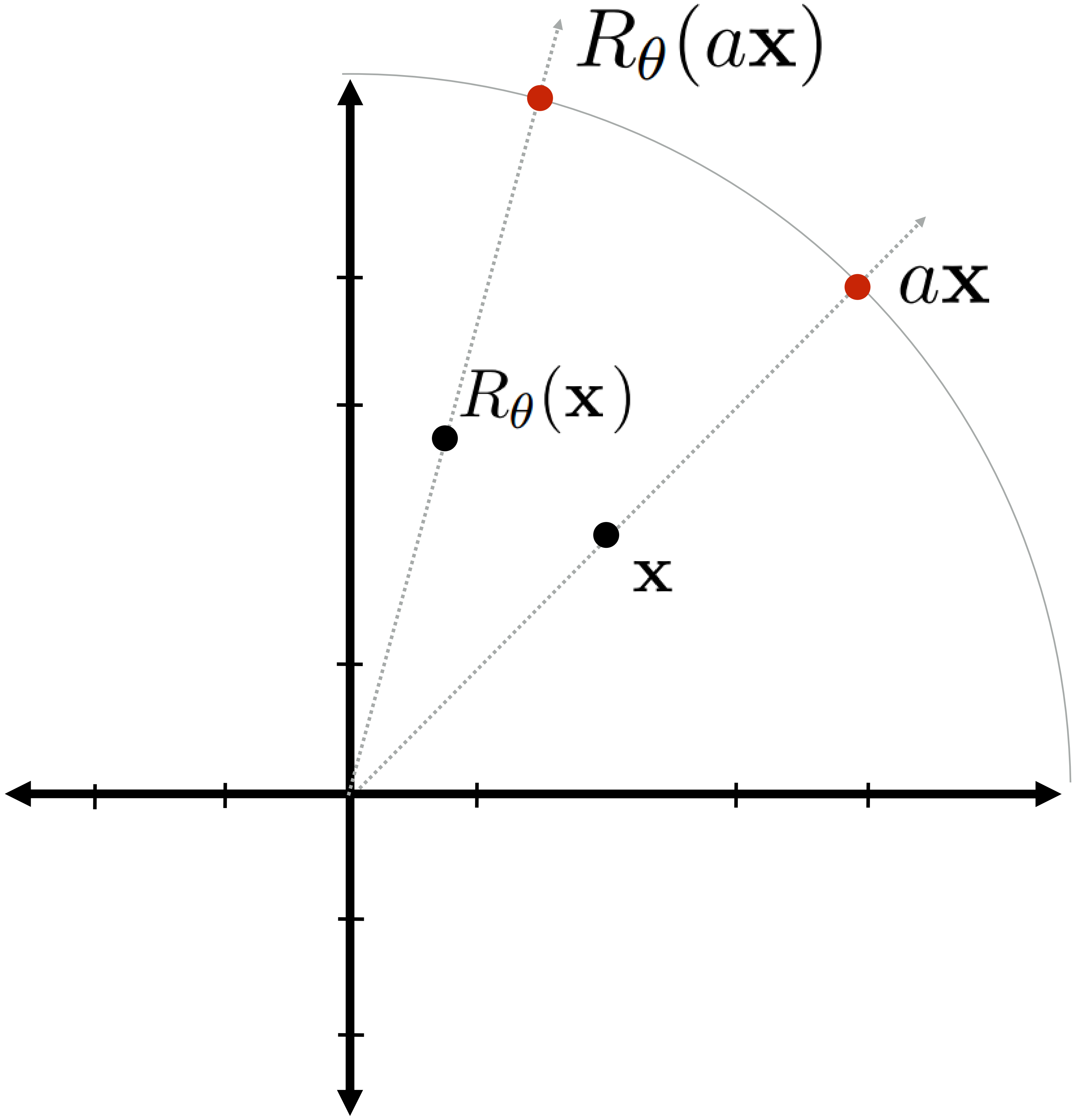
R_θ **= rotate counter-clockwise by** θ

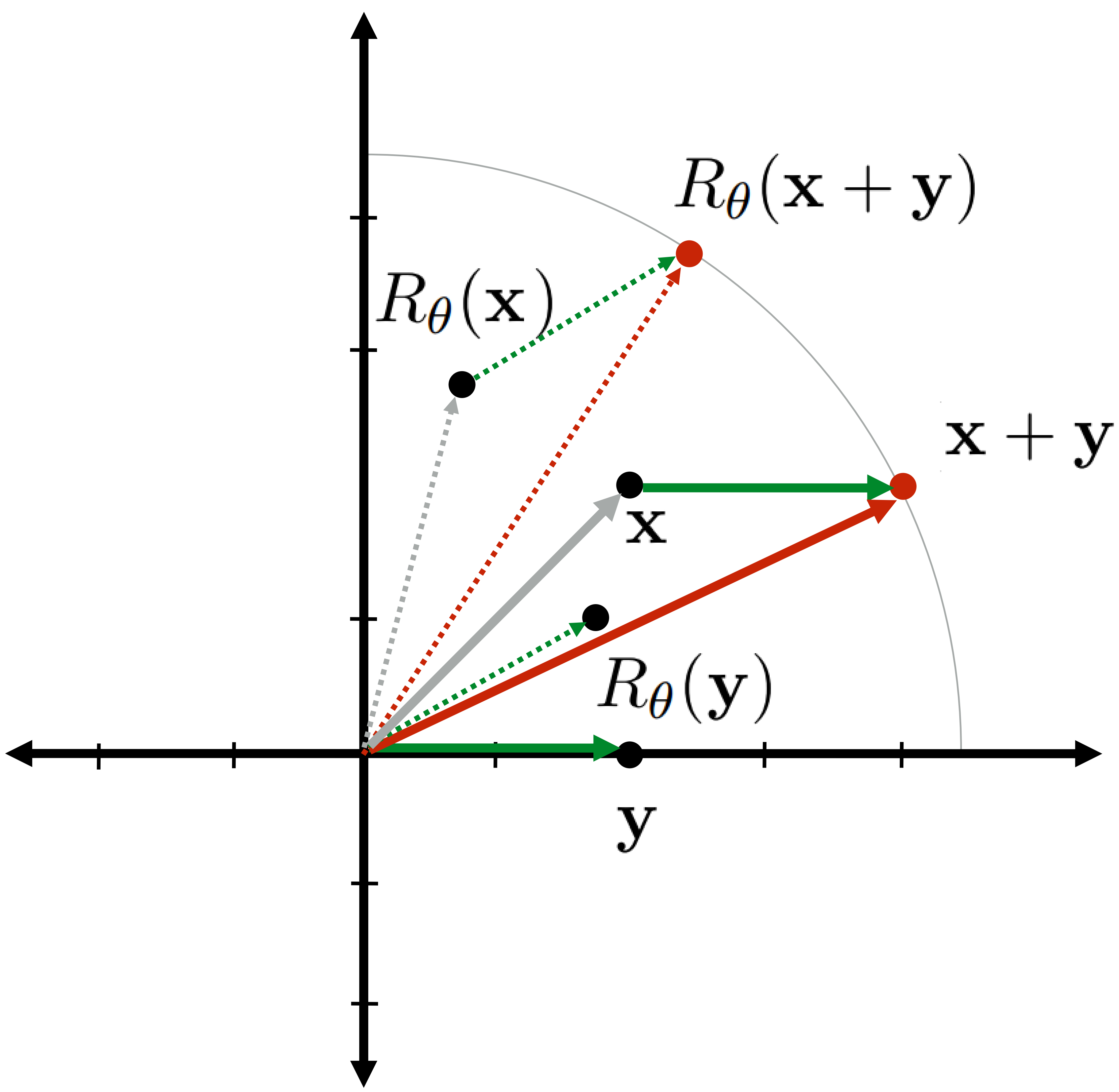
As angle changes, points move along *circular* trajectories.

Shape (distance between any two points) does not change!

(Rigid or isometric transformation)

Is rotation linear?

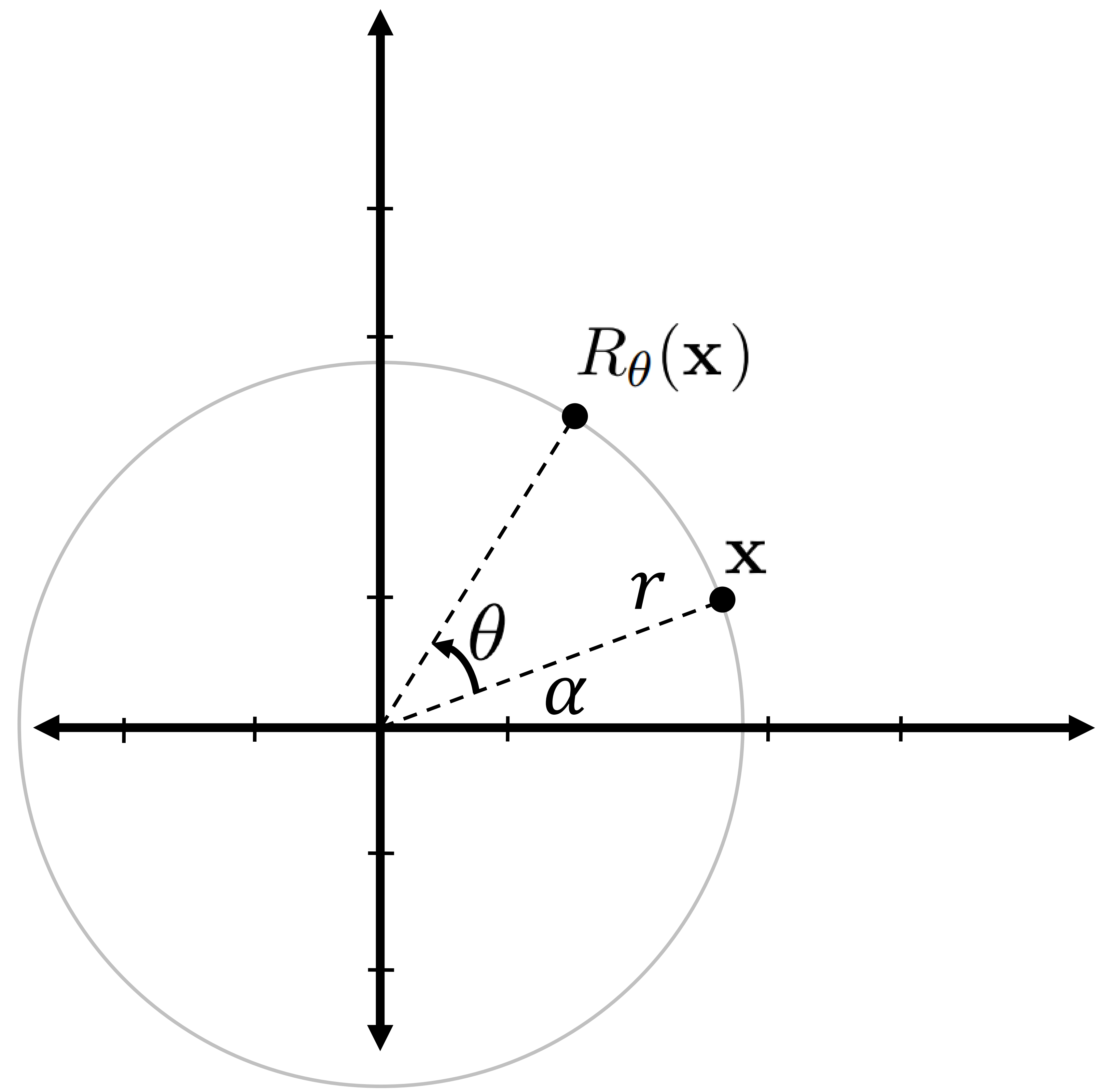




Yes!

Rotation

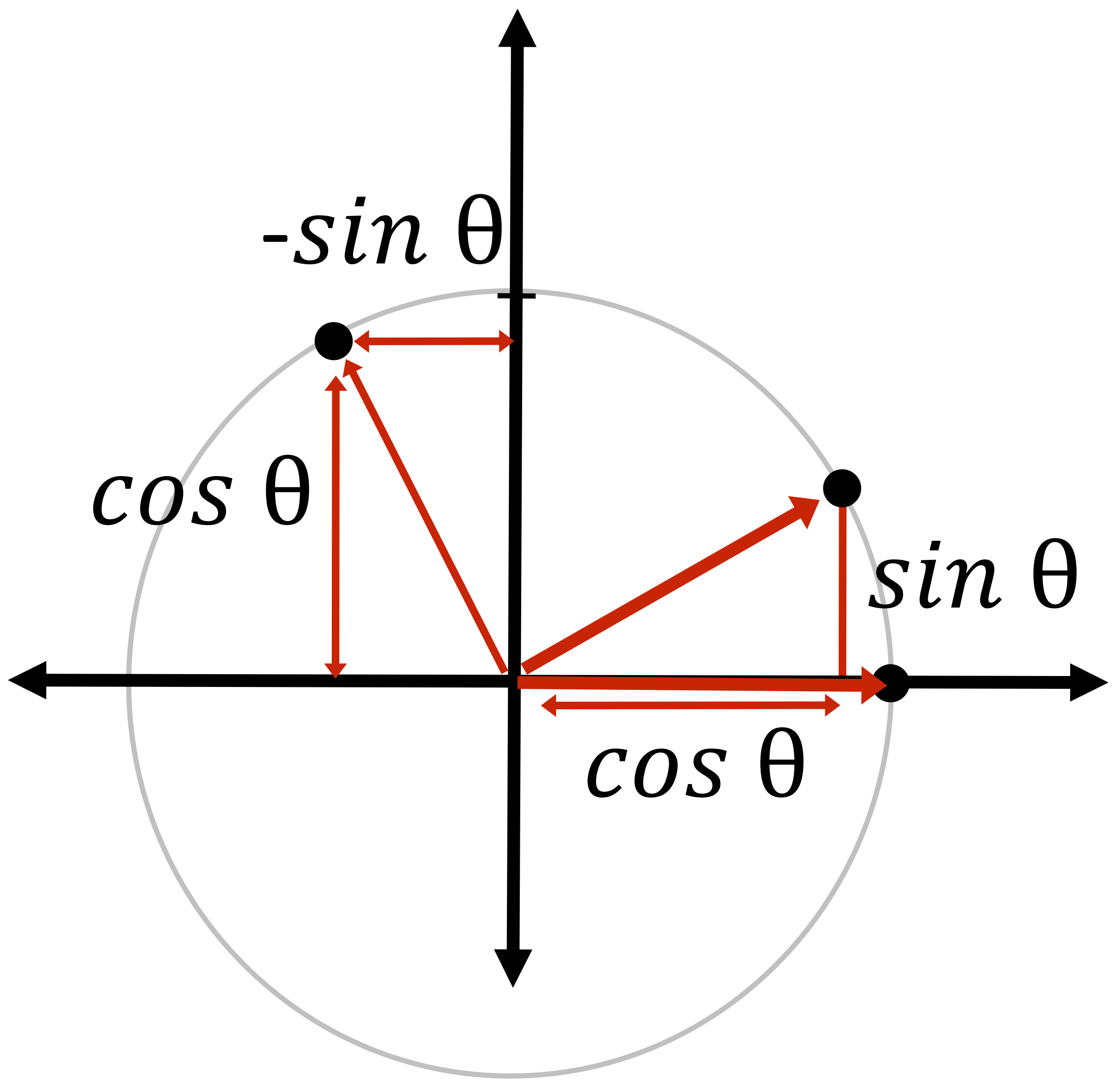
What does R_θ look like?



- **From x, compute α and r**
- **Write down $R_\theta(x)$ as a function of α, θ and r** (i.e. vector (r,0) rotated by $\alpha + \theta$)
- **Apply sum of angle formulae…**
- **Fine, but remember, we only need to know how $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ are transformed (if a linear map)!**

Rotation

So, what happens to vectors (1, 0) and (0, 1) after rotation by θ?



Answer:

$$R_\theta(\boldsymbol{e}_1) = (cos\ \theta, sin\ \theta) = \boldsymbol{a}_1$$

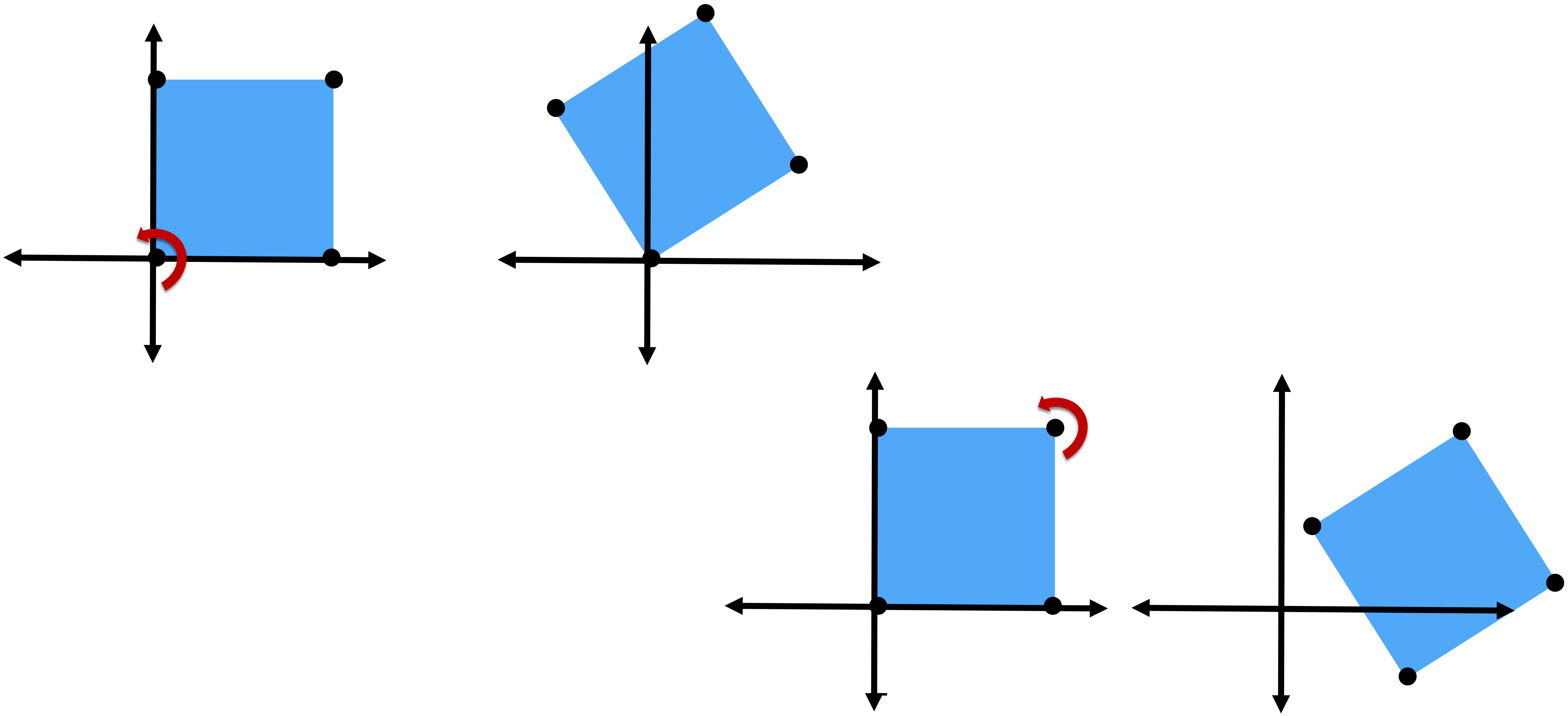
$$R_\theta(\boldsymbol{e}_2) = (-sin\ \theta, cos\ \theta) = \boldsymbol{a}_2$$

So:

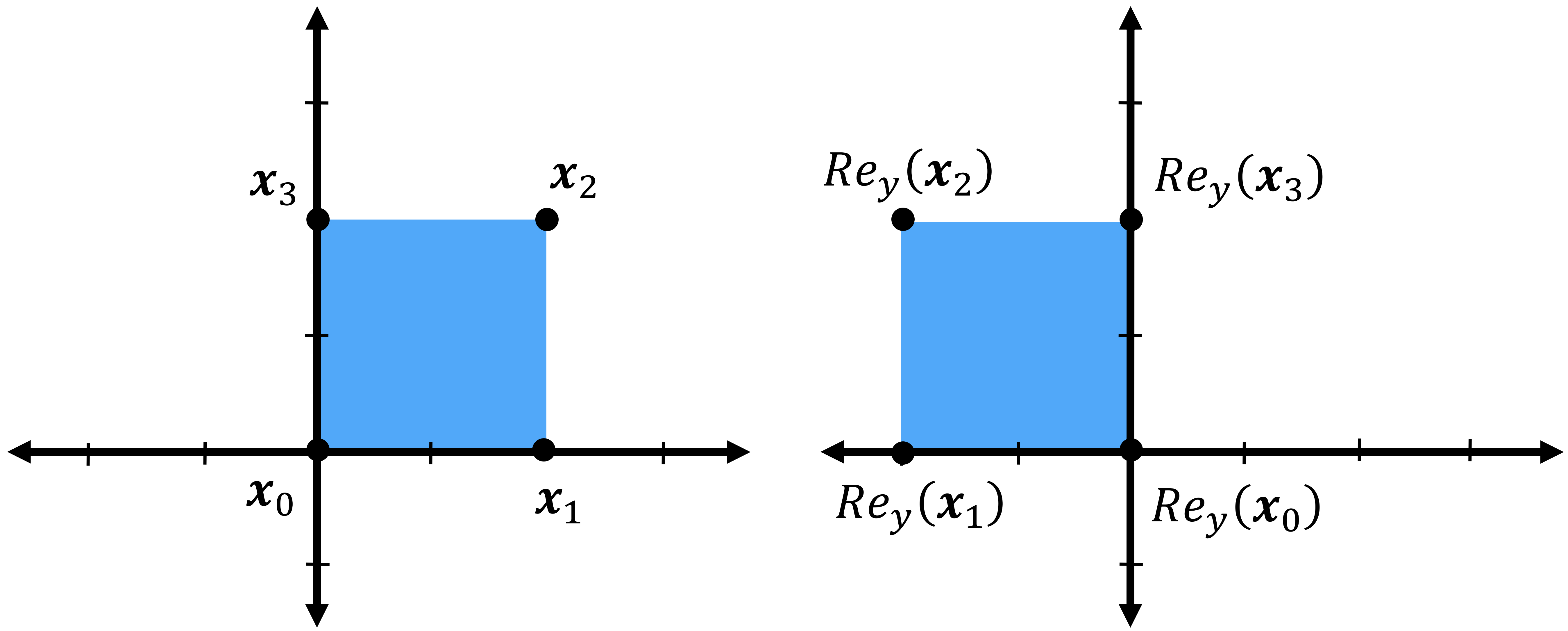
$$R_\theta(\boldsymbol{x}) = x_1\boldsymbol{a}_1 + x_2\boldsymbol{a}_2$$

Rotation

- **Note: all points are rotated about the origin**
- **What if we want to rotate about another point?**



Reflection



$Re_y(\boldsymbol{x})$ **: reflection about y-axis**

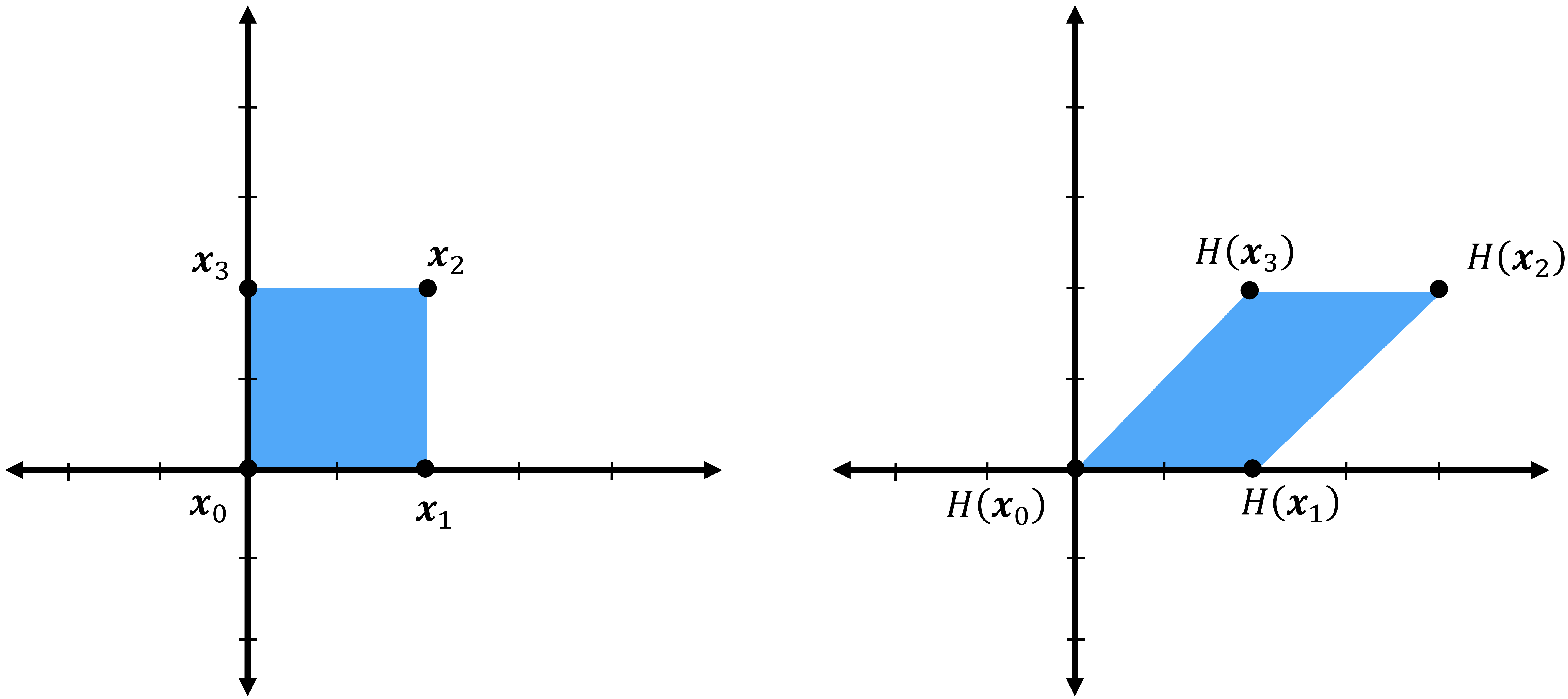
Reflections change “handedness”…

Do you know what $Re_y(\boldsymbol{x})$ **looks like?**

Is reflection a linear transform?

Do you know how to reflect about an arbitrary axis?

Shear (in x direction)

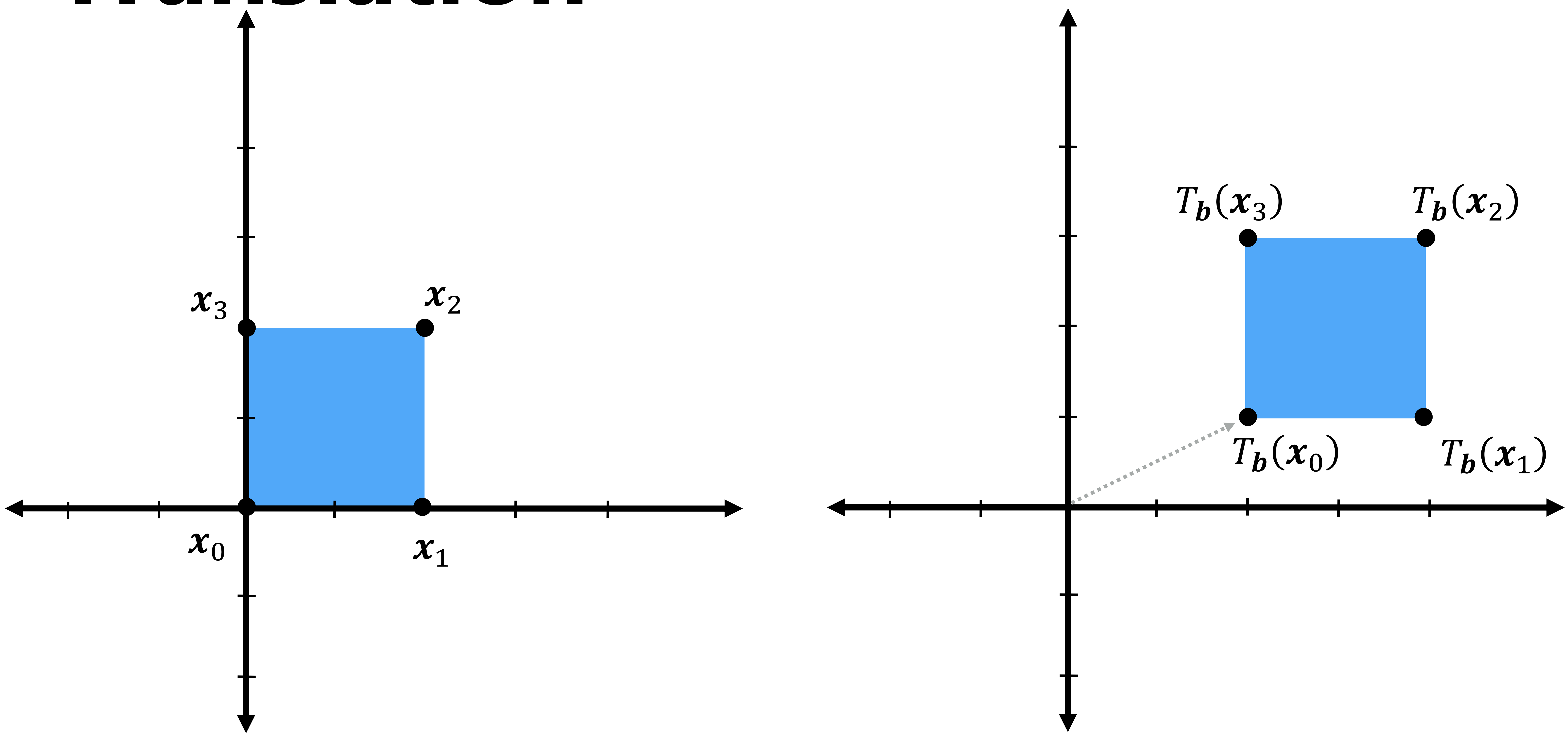


What does $H(x)$ look like?

$$\boldsymbol{H}_a(\boldsymbol{x}) = x_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + x_2 \begin{bmatrix} a \\ 1 \end{bmatrix}$$

Is shearing a linear transformation?

Translation

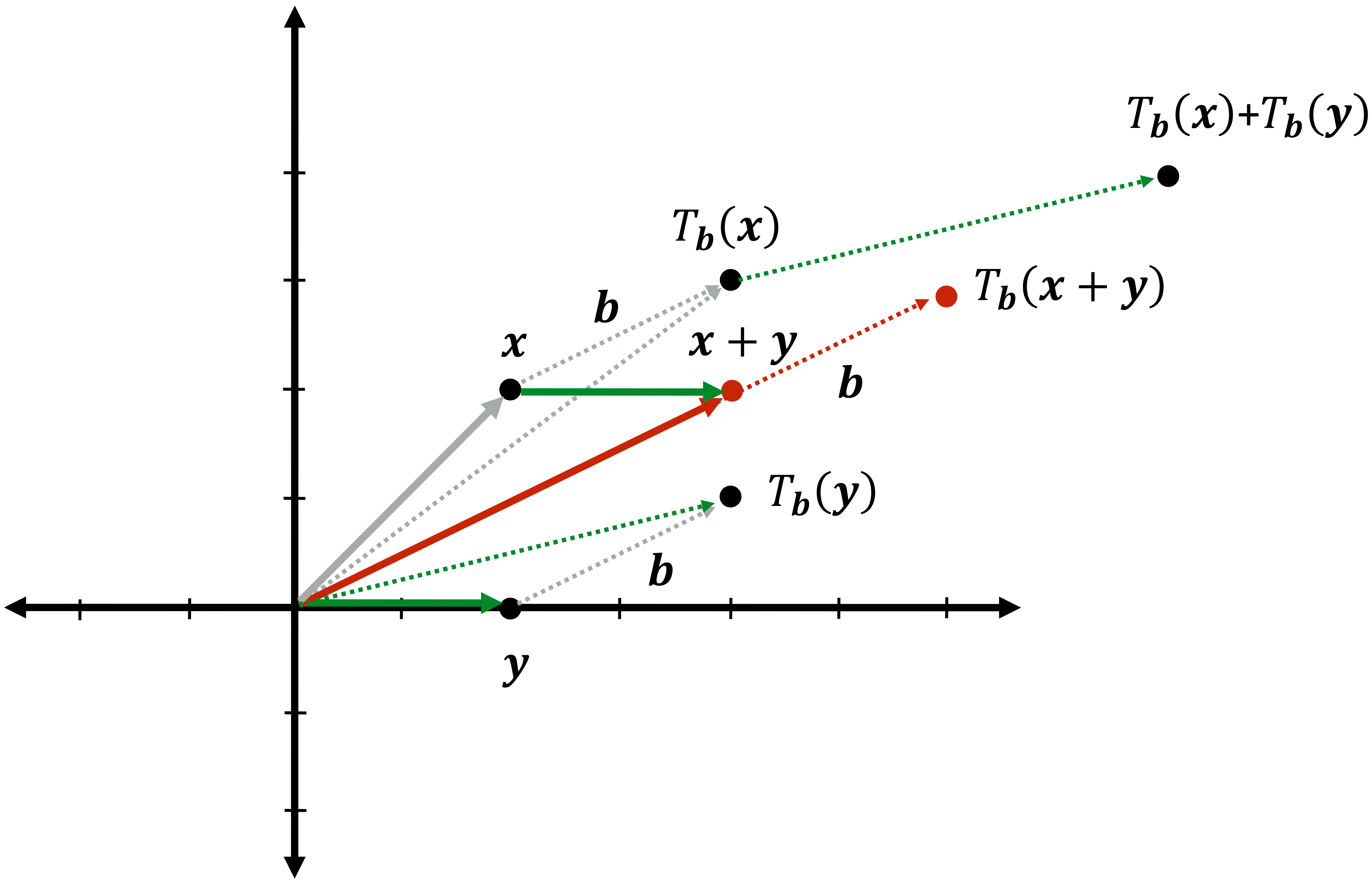


Let's write $T_b(\boldsymbol{x})$ in the form

$$T_{\boldsymbol{b}}(\boldsymbol{x}) = x_1 \begin{bmatrix} ? \\ ? \end{bmatrix} + x_2 \begin{bmatrix} ? \\ ? \end{bmatrix}$$

such that $T_{\boldsymbol{b}}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{b}$

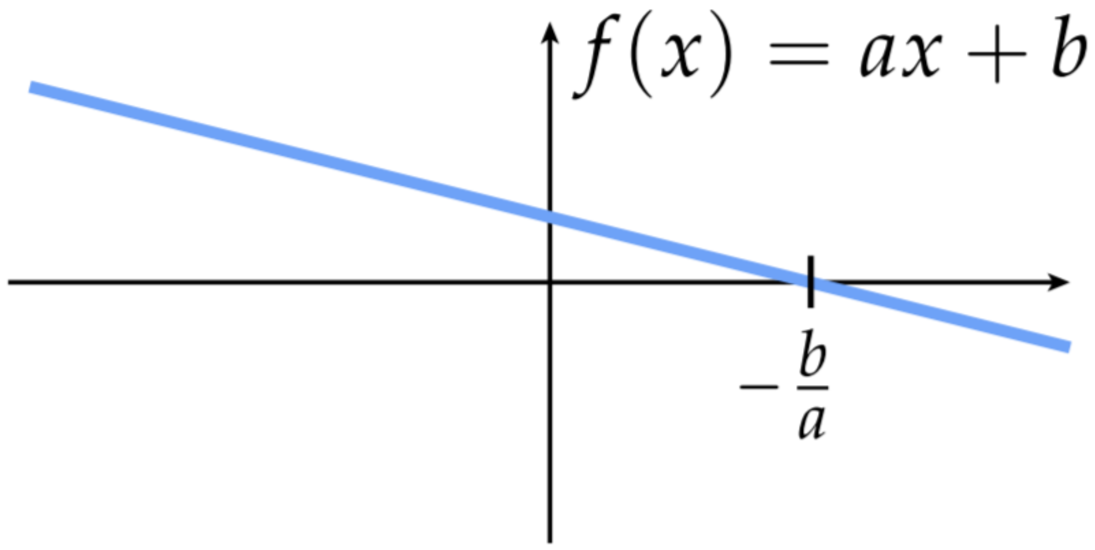
Is translation linear?



No. Translation is *affine*.

Linear vs Affine Maps

- **f(x) := ax + b is not a linear function!**
- **But easy to be fooled, since its graph is a line:**



- **However, it's not a line through the *origin* (f(0) != 0)**
- **Also, math corresponding to previous slide...**

$$f(x_1 + x_2) = a(x_1 + x_2) + b = ax_1 + ax_2 + b$$
$$f(x_1) + f(x_2) = (ax_1 + b) + (ax_2 + b) = ax_1 + ax_2 + 2b$$

When at first you don’t succeed…

- We’ll turn affine transformations into linear ones via

Homogeneous coordinates
(aka projective coordinates)

- But first, let’s use matrix notation to represent linear transforms

Linear transforms as matrix-vector products

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}}_{\boldsymbol{A}} * \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{\boldsymbol{x}} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 \\ a_{21}x_1 + a_{22}x_2 \end{bmatrix}$$

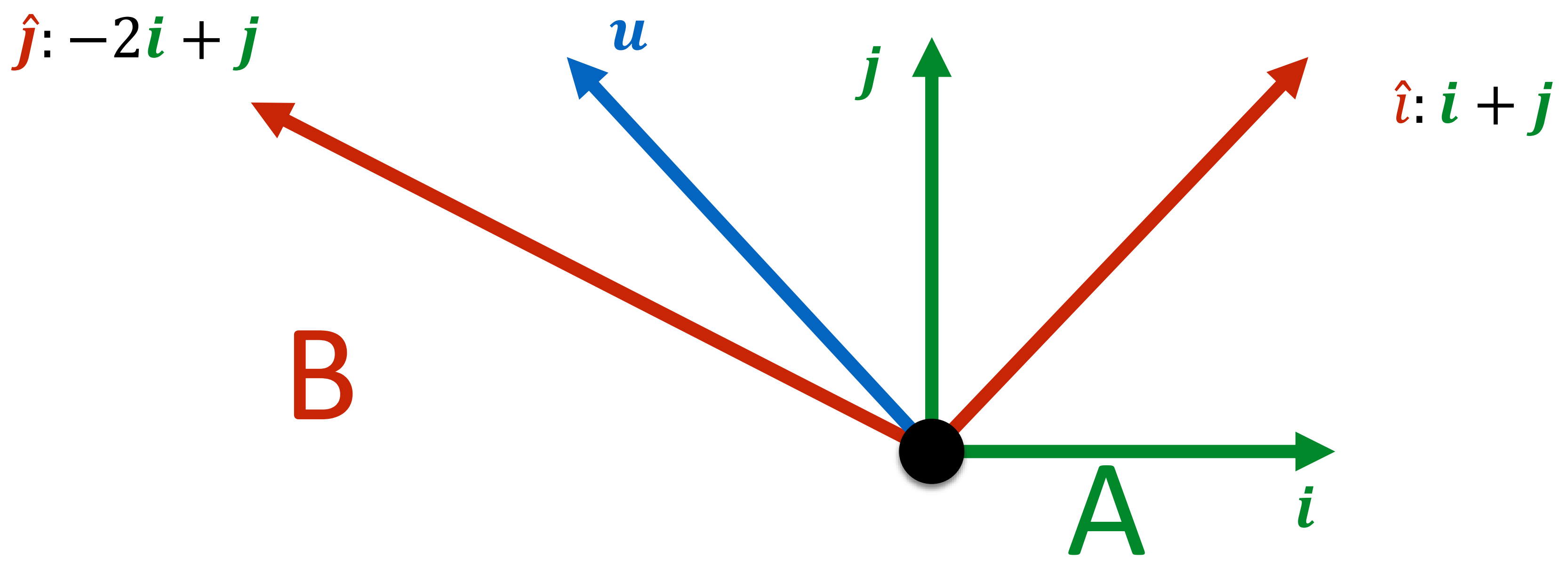
$$= x_1 \begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ a_{22} \end{bmatrix} = \underbrace{x_1 \boldsymbol{a_1} + x_2 \boldsymbol{a_2}}_{f(\boldsymbol{x}) = \sum_{i=1}^{m} x_i \boldsymbol{a}_i = \boldsymbol{Ax}}$$

Linear transforms as matrix-vector products

Change of coordinate systems

$$f(\boldsymbol{x}) = x_1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + x_2 \begin{bmatrix} -2 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & -2 \\ 1 & 1 \end{bmatrix} \boldsymbol{x}$$

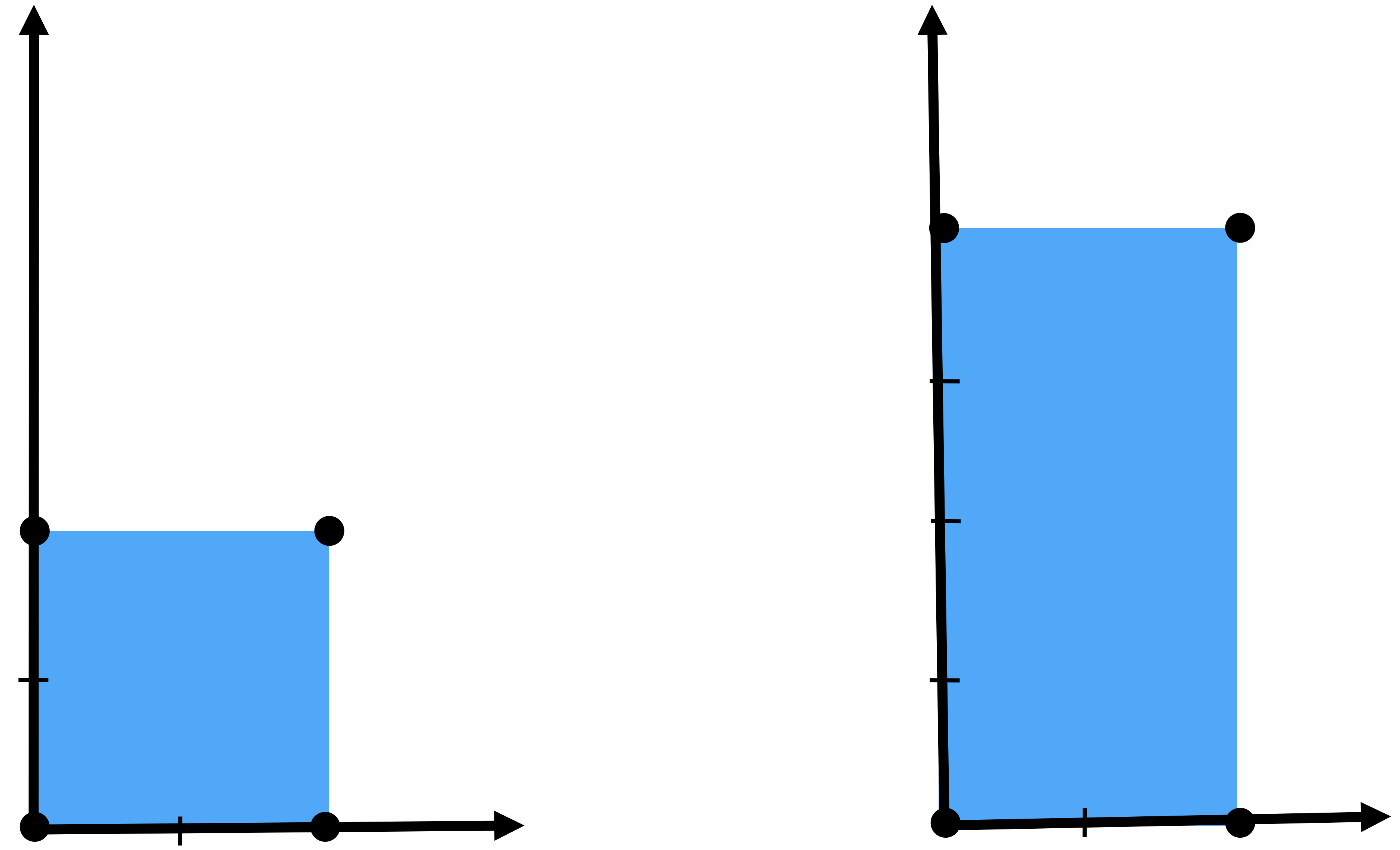


Linear transforms as matrix-vector products

Non-uniform scale

$$\mathrm{S}(\boldsymbol{x}) = x_1 a \boldsymbol{e}_1 + x_2 b \boldsymbol{e}_2$$

$$= \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \boldsymbol{x}$$



Linear transforms as matrix-vector products

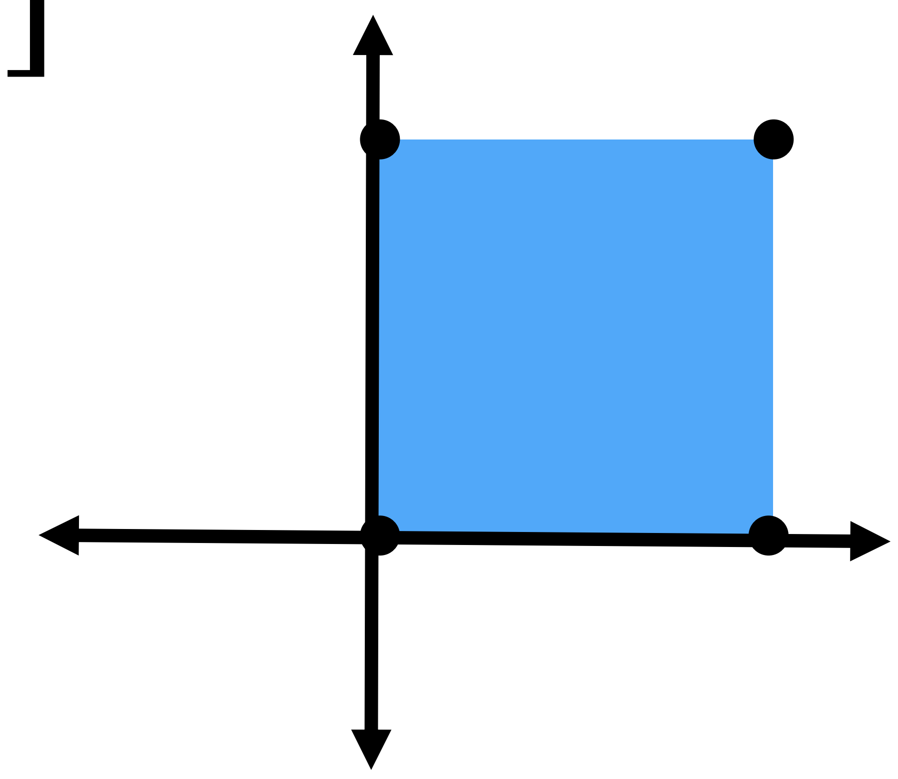
Rotation

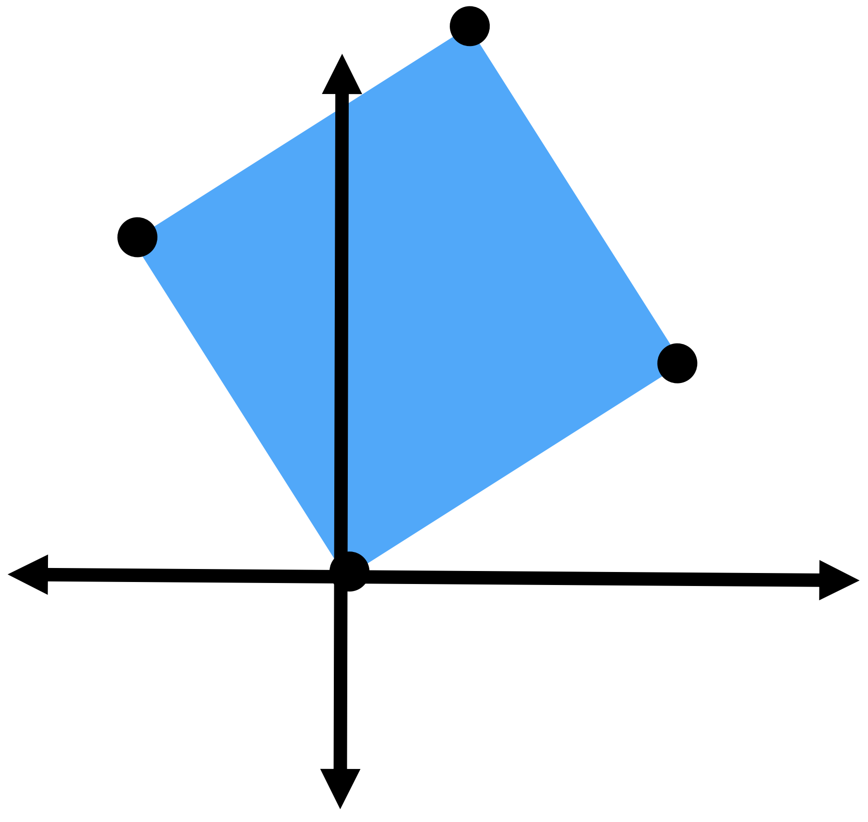
$$R_\theta(\boldsymbol{e}_1) = (cos\ \theta, sin\ \theta) = \boldsymbol{a}_1$$

$$R_\theta(\boldsymbol{e}_2) = (-sin\ \theta, cos\ \theta) = \boldsymbol{a}_2$$

$$R_\theta(\boldsymbol{x}) = x_1\boldsymbol{a}_1 + x_2\boldsymbol{a}_2$$

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



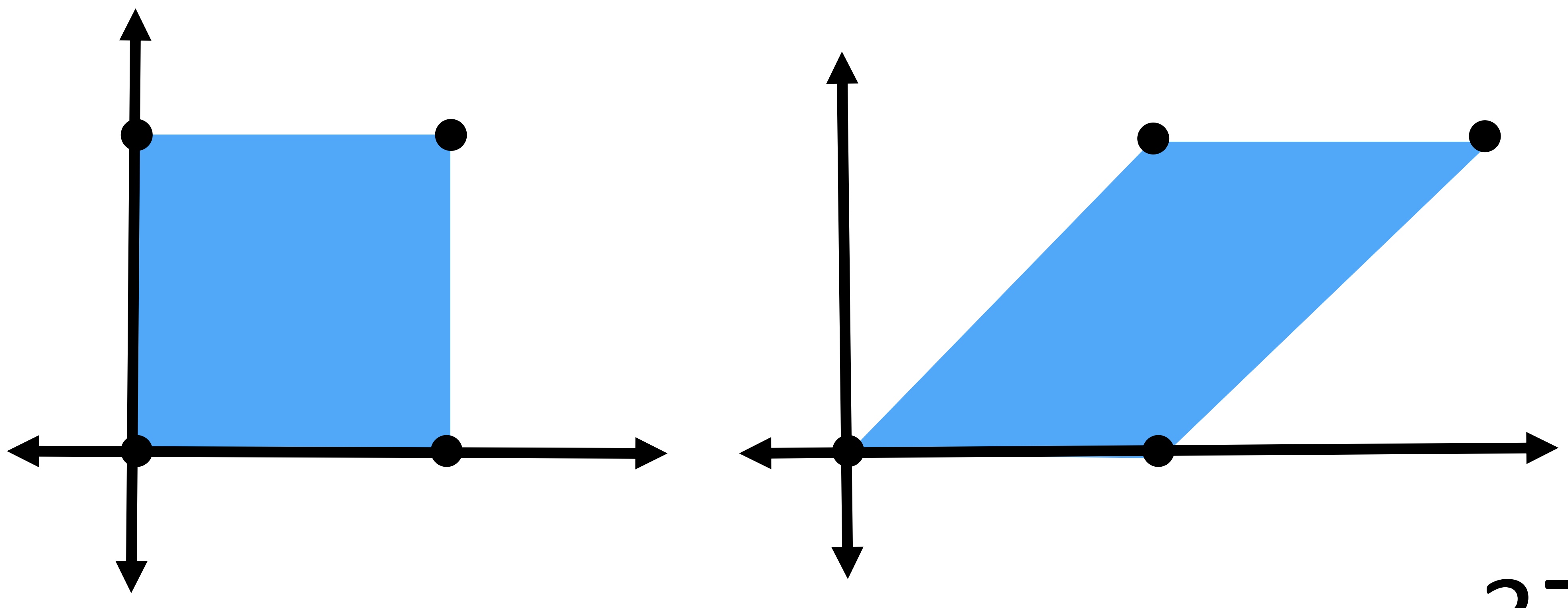


Linear transforms as matrix-vector products

Shear

$$H(\boldsymbol{x}) = x_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + x_2 \begin{bmatrix} a \\ 1 \end{bmatrix}$$

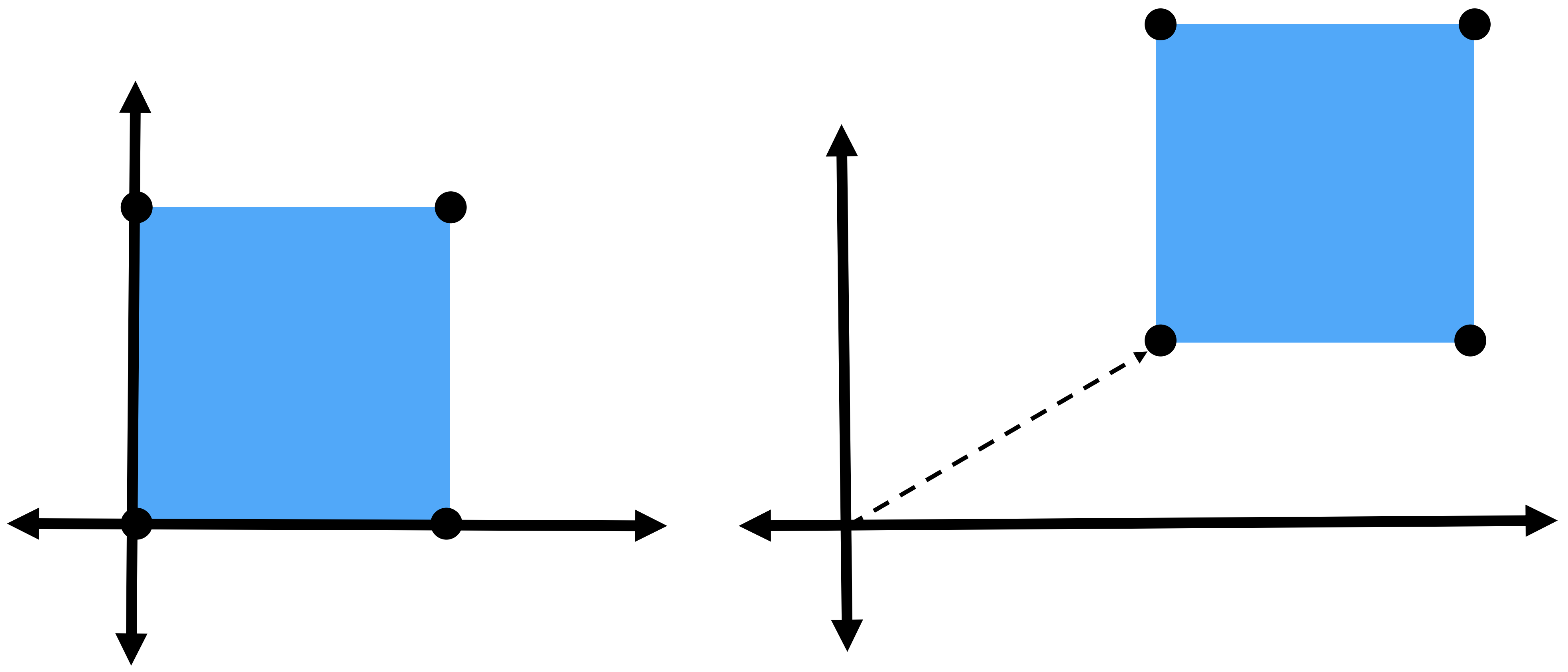
$$= \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \boldsymbol{x}$$



Linear transforms as matrix-vector products

Translation

Not a linear map*...



***when using Cartesian coordinates**

2D homogeneous coordinates (2D-H)

Key idea: "lift” 2D points to a 3D space

The 2D point (x_1, x_2) is represented as the 3-vector: $\begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$

And 2D transforms are represented by 3x3 matrices

For example: 2D rotation in homogeneous coordinates:

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$$

Q: how do the transforms we’ve seen so far affect the last coordinate?

Translation in 2D-H coords

Translation expressed as 3x3 matrix multiplication:

$$\boldsymbol{T}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{b} = \begin{bmatrix} 1 & 0 & b_1 \\ 0 & 1 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 + b_1 \\ x_2 + b_2 \\ 1 \end{bmatrix}$$

In homogeneous coordinates, translation is a linear transformation!

Translation in 2D-H coords

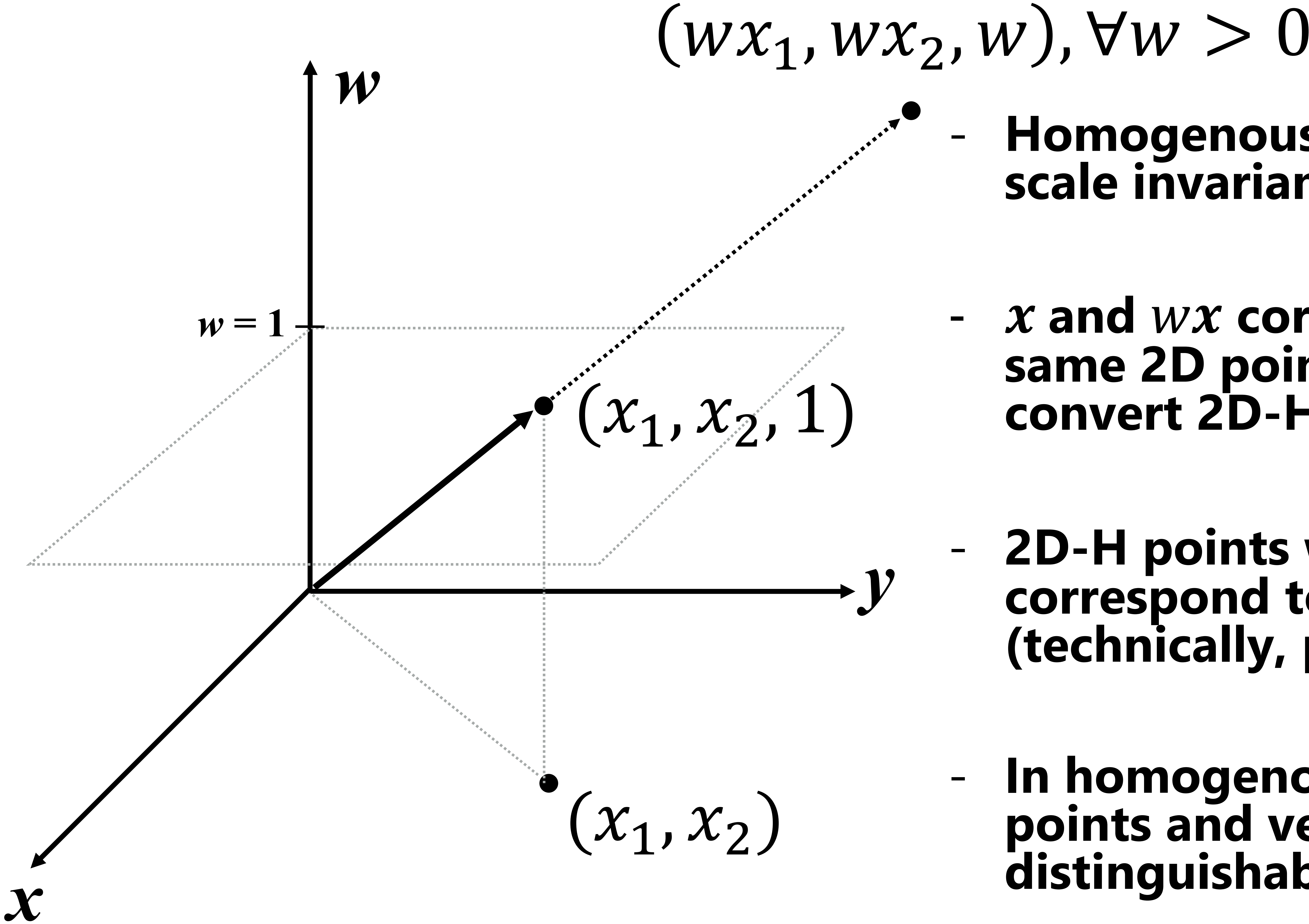
What is this magic?

$$\begin{bmatrix} 1 & 0 & b_1 \\ 0 & 1 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_1 + b_1 \\ x_2 + b_2 \\ x_3 \end{bmatrix}$$

Translation in 2D homogeneous coordinates is equivalent to shearing along x and y axes – a linear operation.

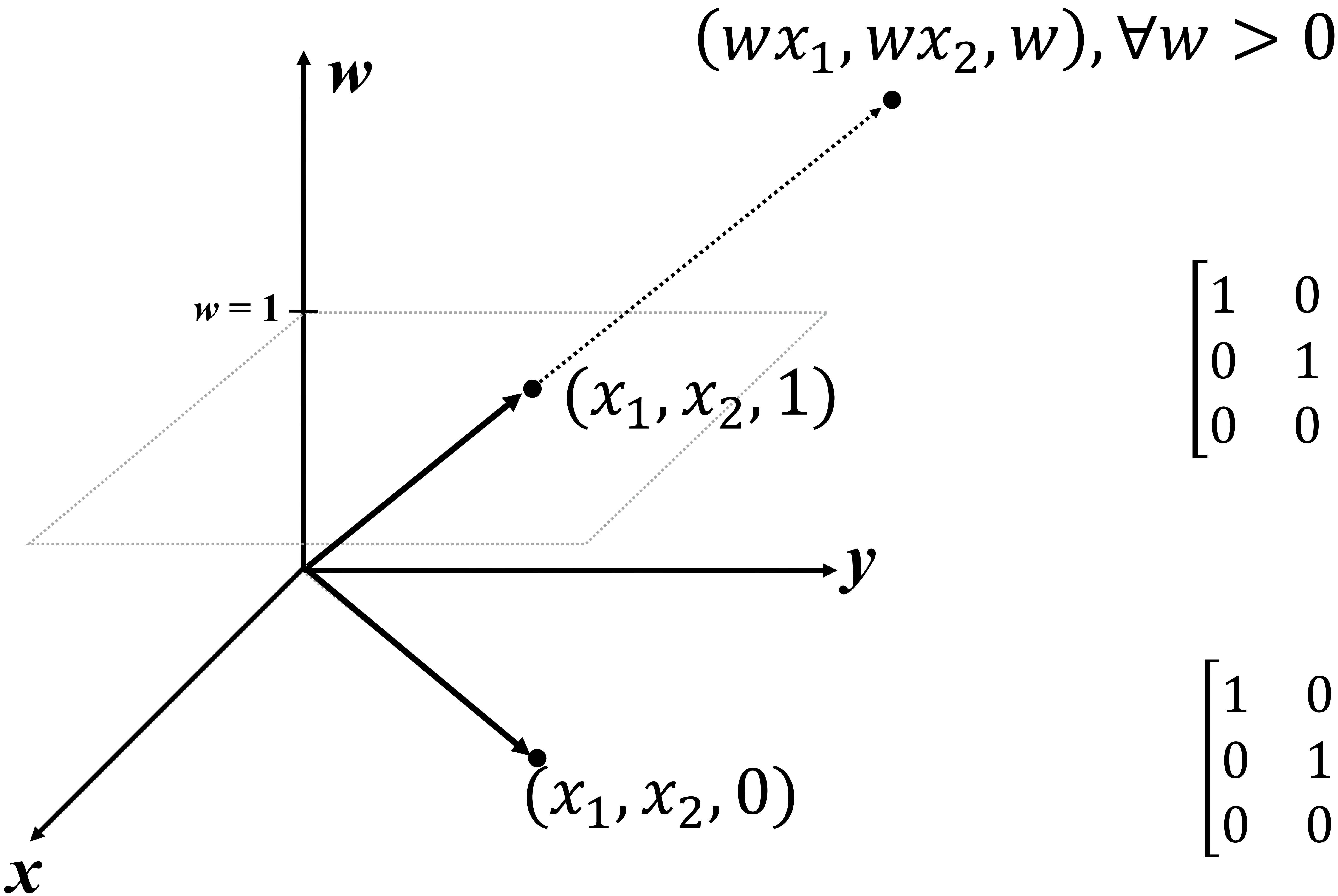
But why is x_3 set to 1? Could it not be 3.4182 instead?

Homogeneous coordinates



- **Homogenous coordinates are scale invariant**
- $\boldsymbol{x}$ **and** wx **correspond to the same 2D point (divide by** w **to convert 2D-H back to 2D)**
- **2D-H points with** $w = 0$ **correspond to 2D vectors (technically, points at infinity)**
- **In homogenous coordinates, points and vectors are distinguishable from each other!**

Homogeneous coordinates: points vs. vectors

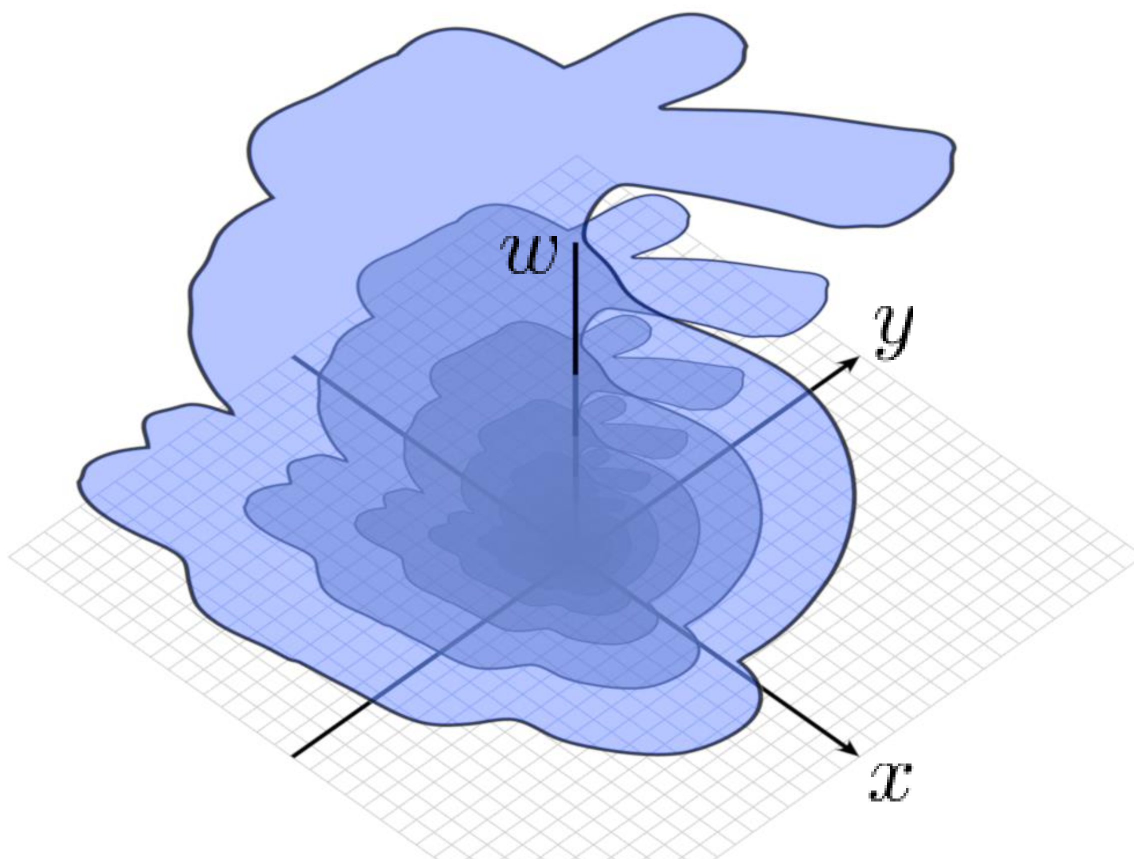


$$\begin{bmatrix} 1 & 0 & b_1 \\ 0 & 1 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$$

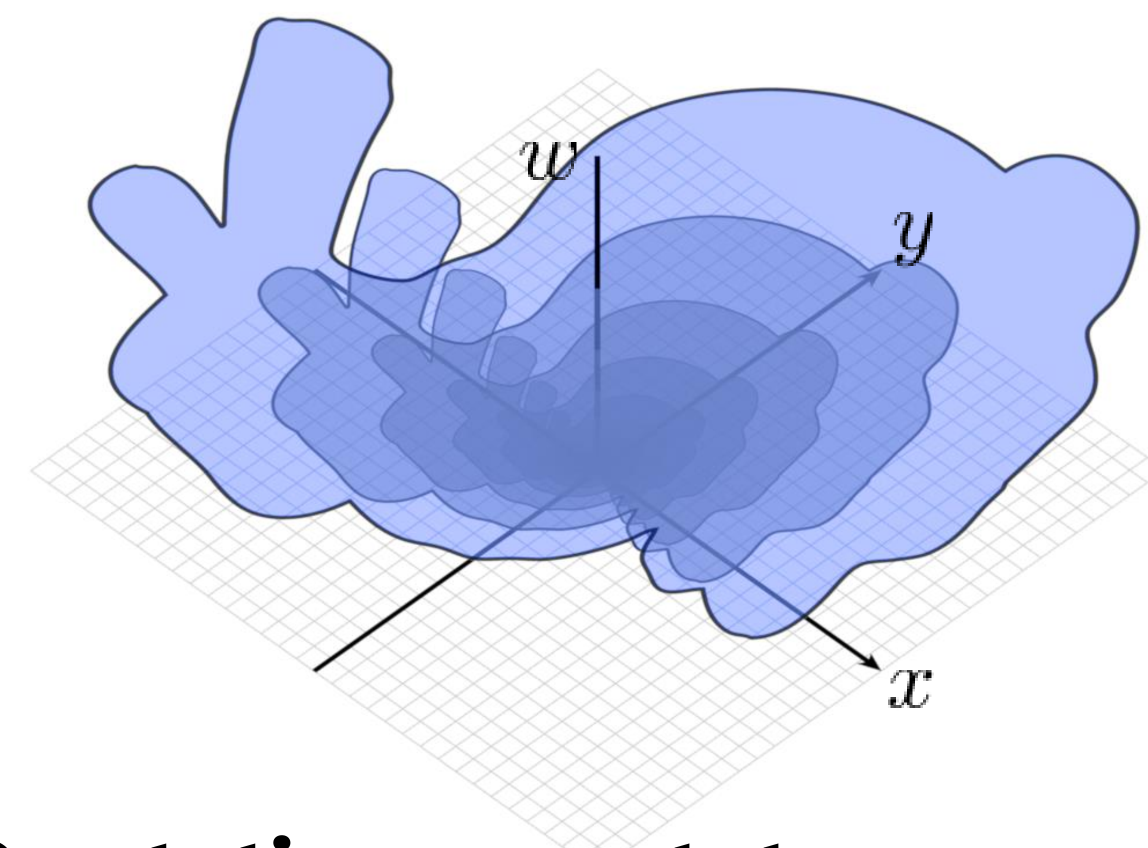
vs

$$\begin{bmatrix} 1 & 0 & b_1 \\ 0 & 1 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 0 \end{bmatrix}$$

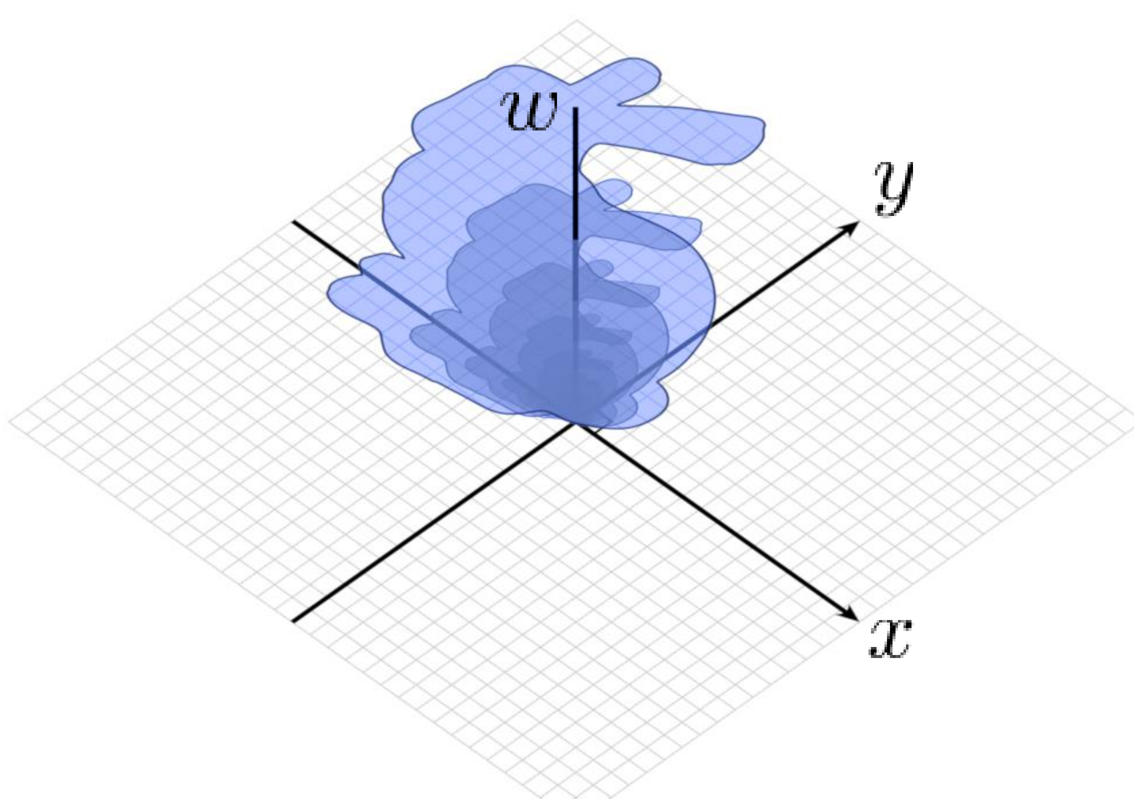
Visualizing 2D transformations in 2D-H



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

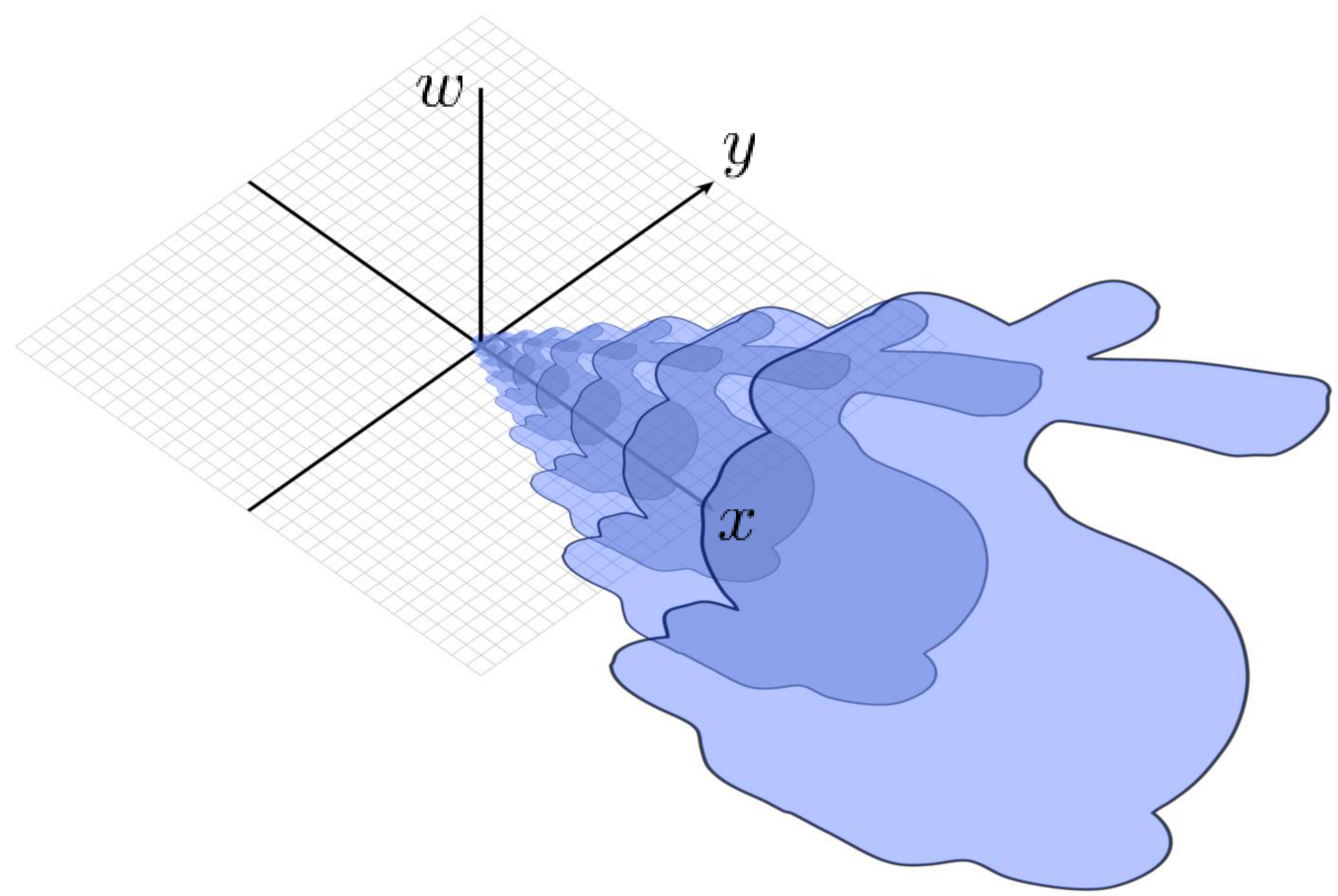


2D rotation ↔ rotate around w



2D scale ↔ scale x and y; preserve w

(Question: what happens to 2D shape if you scale x, y, and w uniformly?)



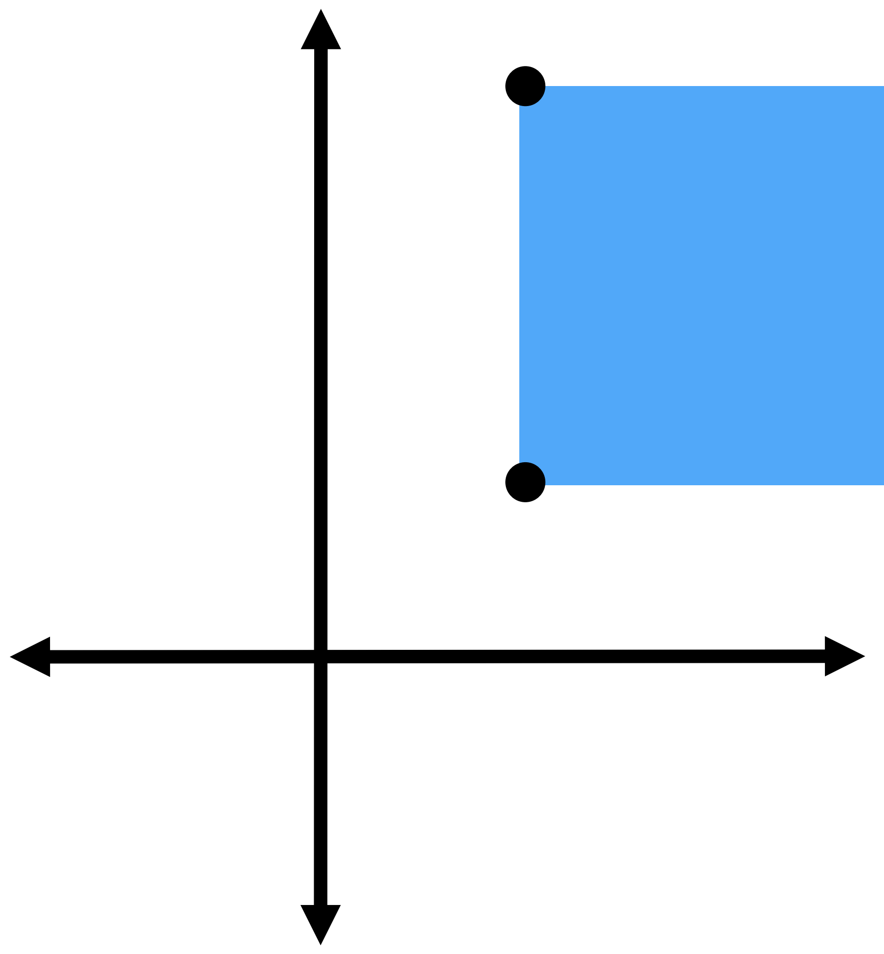
2D translate ↔ shear in xy

Summary so far…

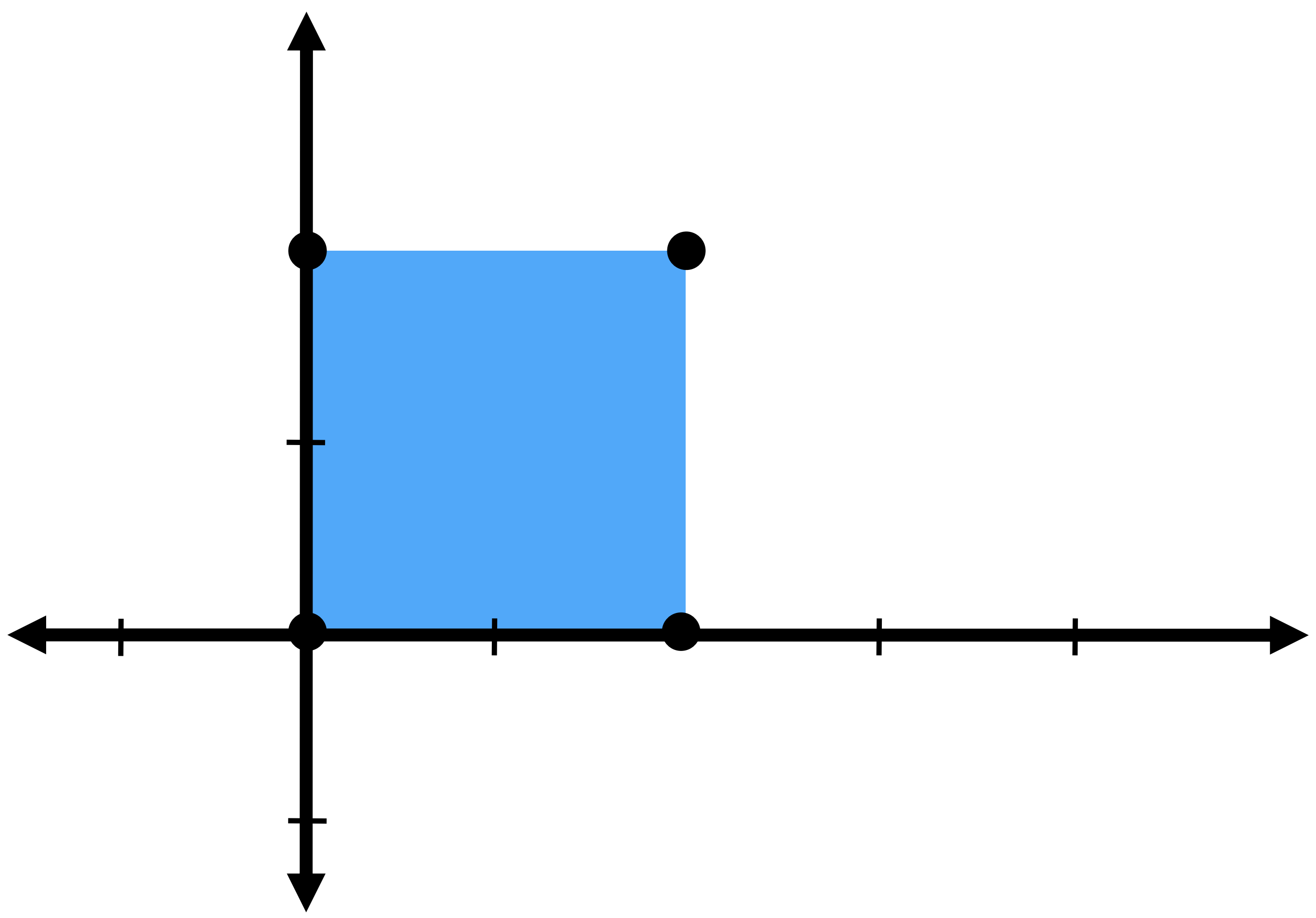
- We know how to transform (scale, rotate, reflect, shear, translate) 2D points and vectors
  - All these transforms are linear maps expressed as matrix-vector products when using (slightly) higher-dimensional homogenous coordinates
  - How about other types of transforms (e.g. rotate about an arbitrary point)?
  - How about 3D transforms?

Onto more complex transforms

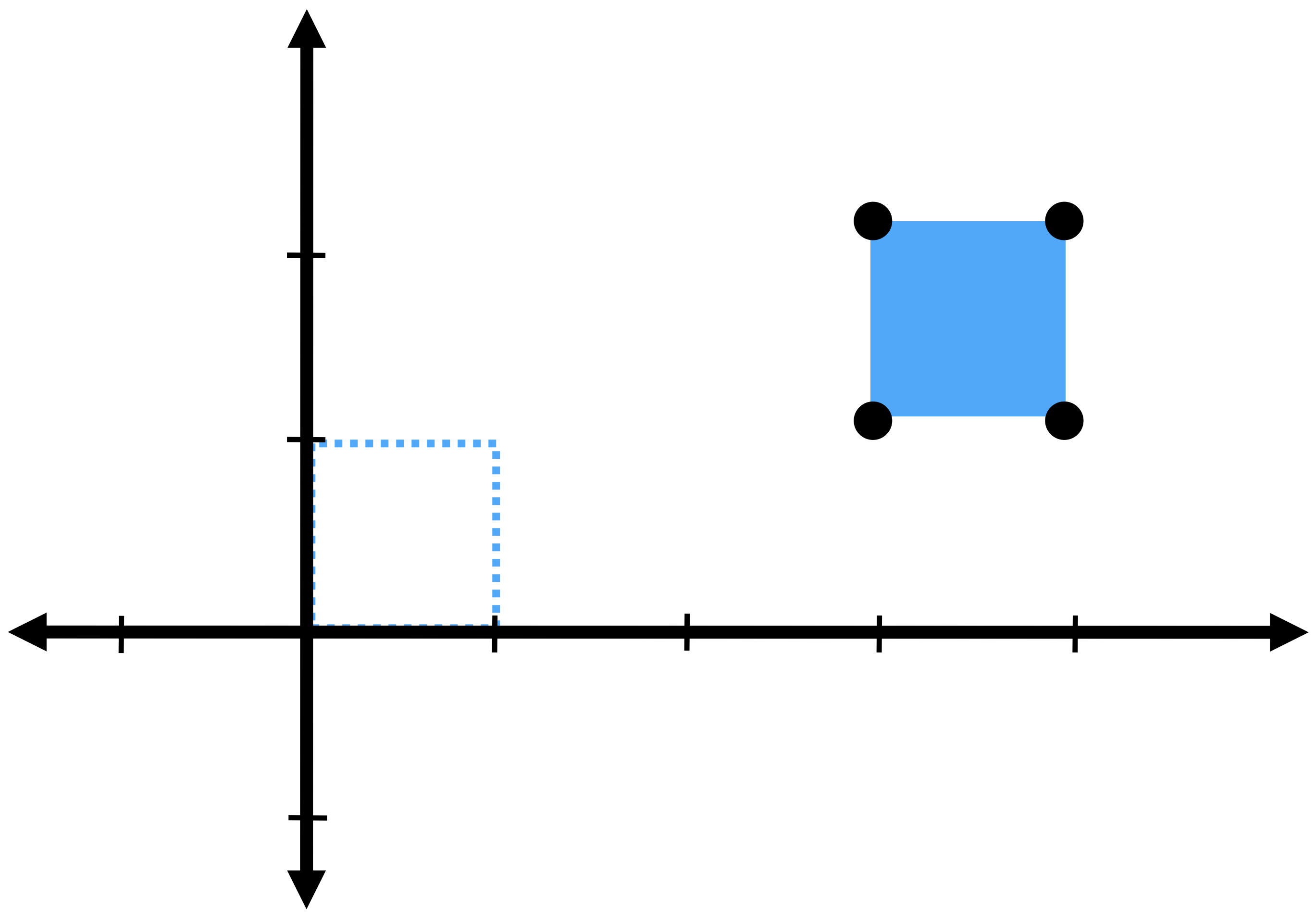
- **How would you transform this object such that it gets twice as large?**
  - **but "in place"…**



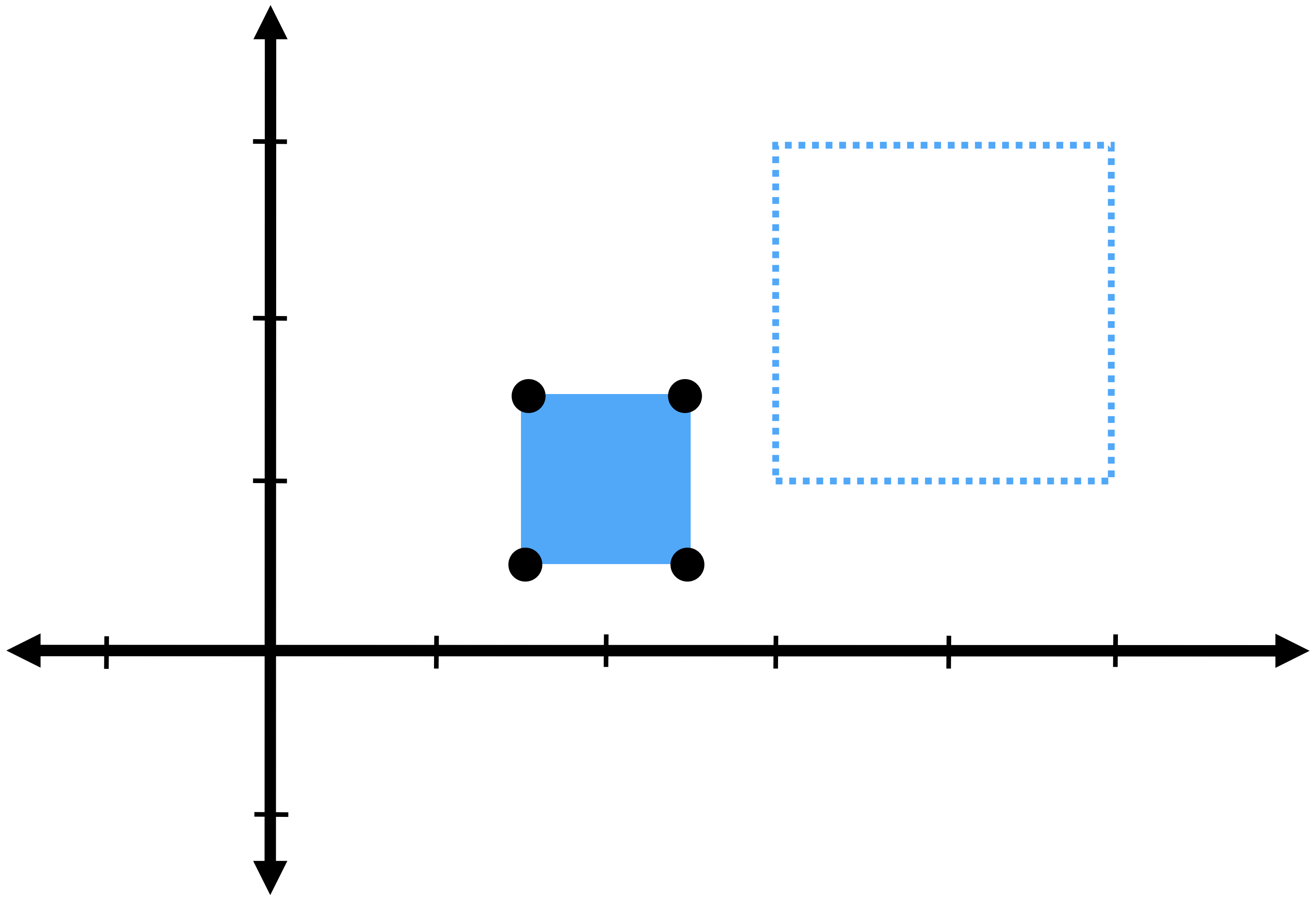
Composition of basic transforms



Scale by 0.5, then translate by (3,1)

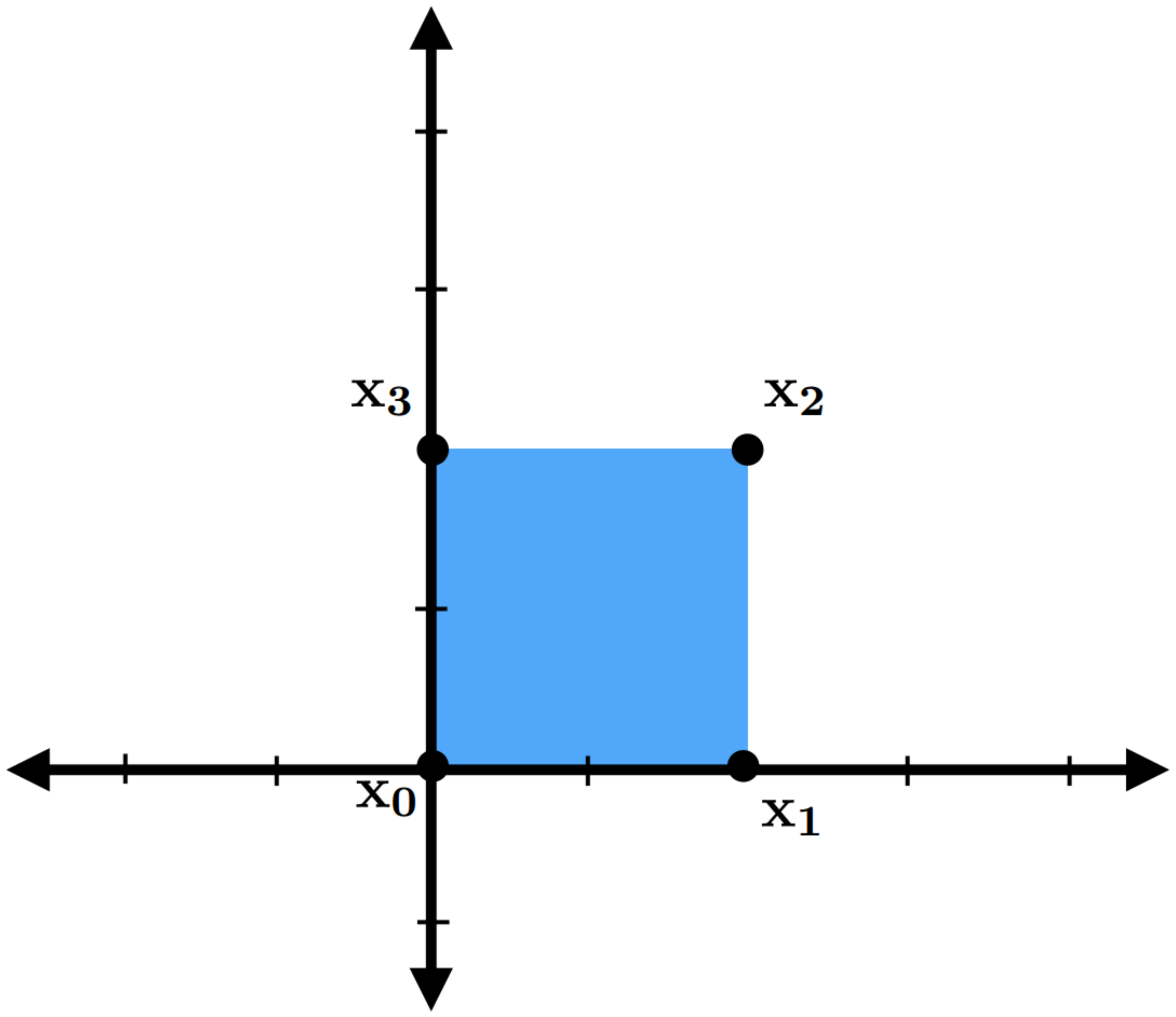


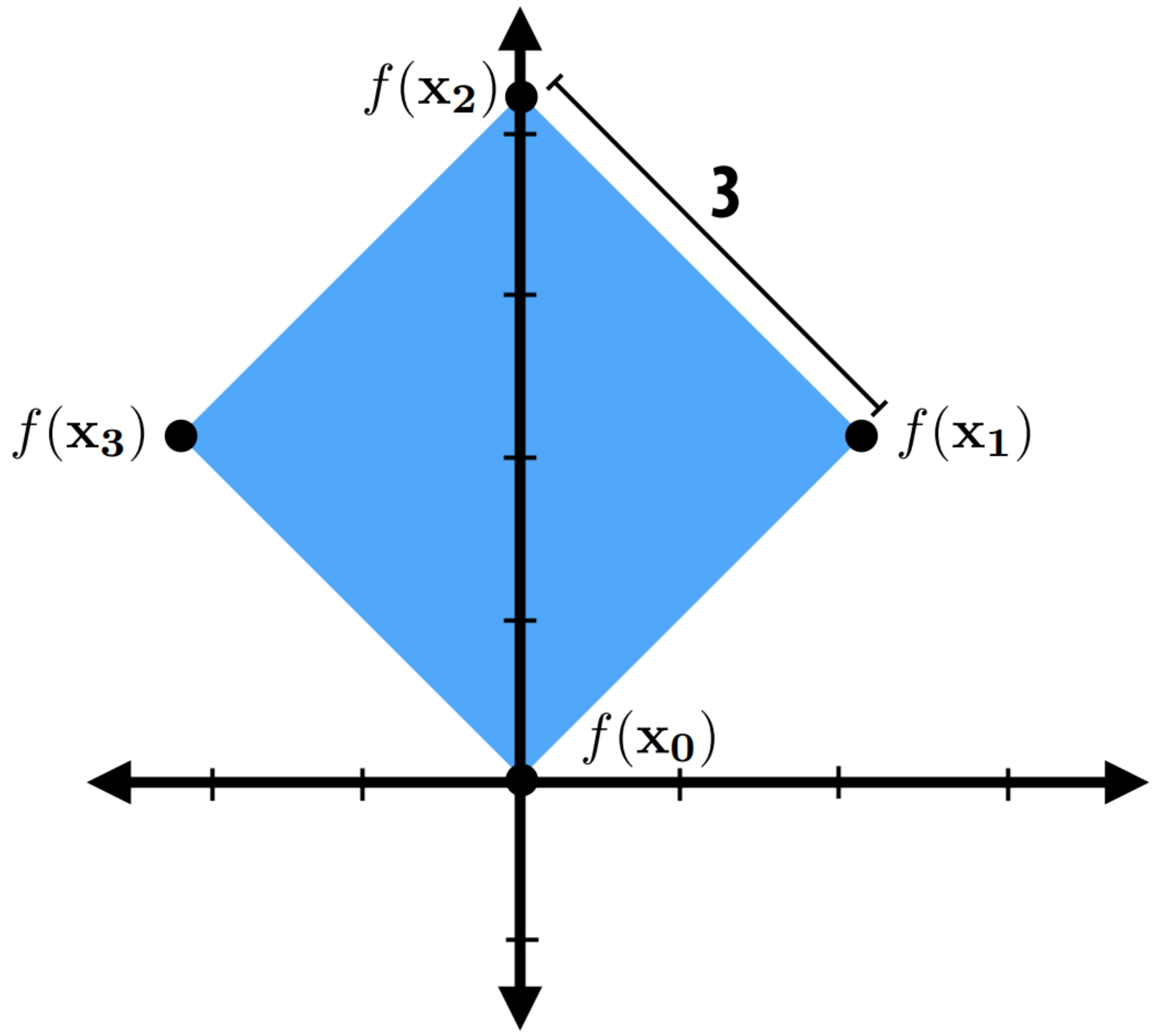
Translate by (3,1), then scale by 0.5



Note 1: order of composition matters!
Note 2: common source of bugs!

How do we compose linear transforms?



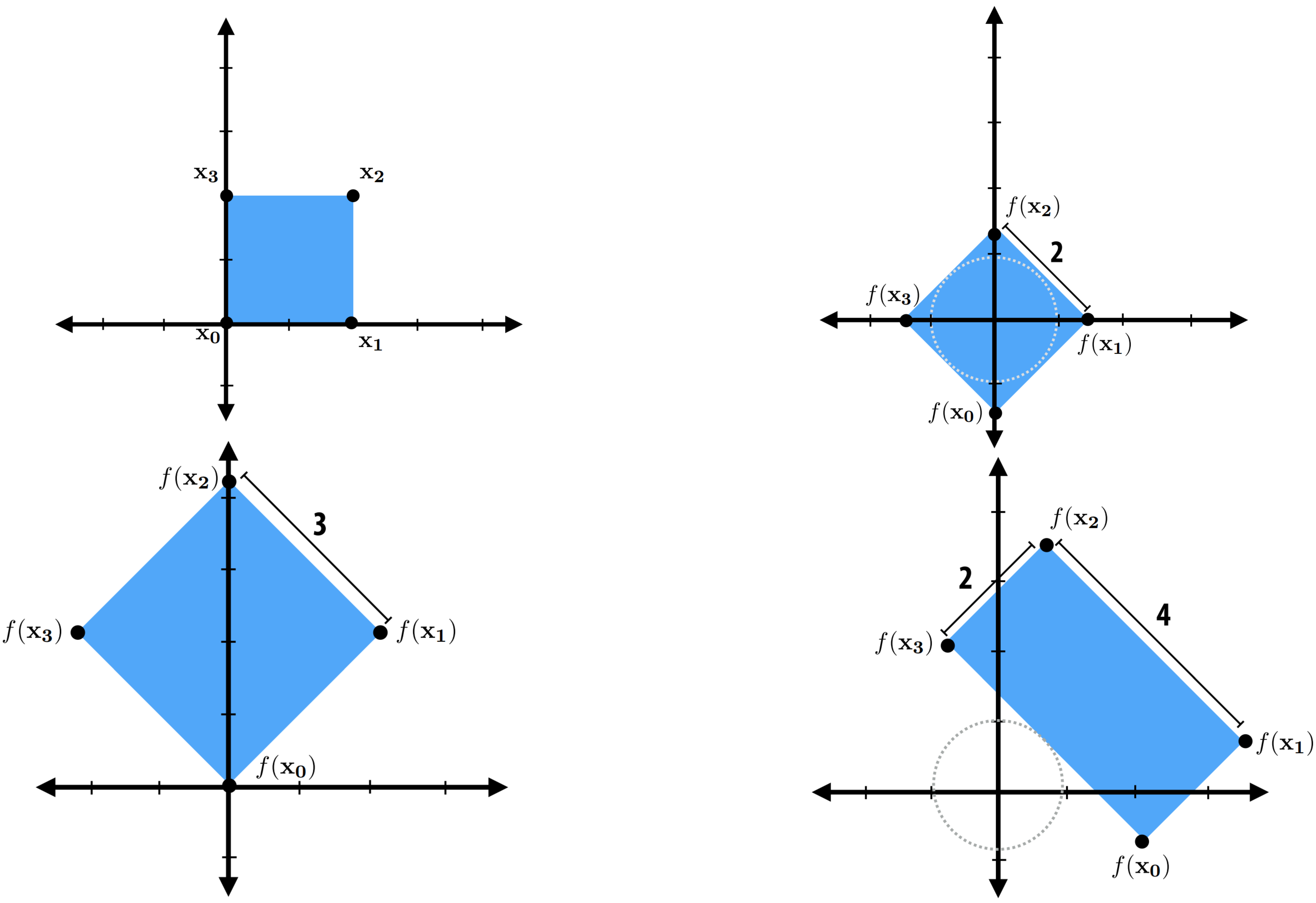


$$f(\mathbf{x}) = R_{\pi/4}\mathbf{S}_{[1.5,1.5]}\mathbf{x}$$

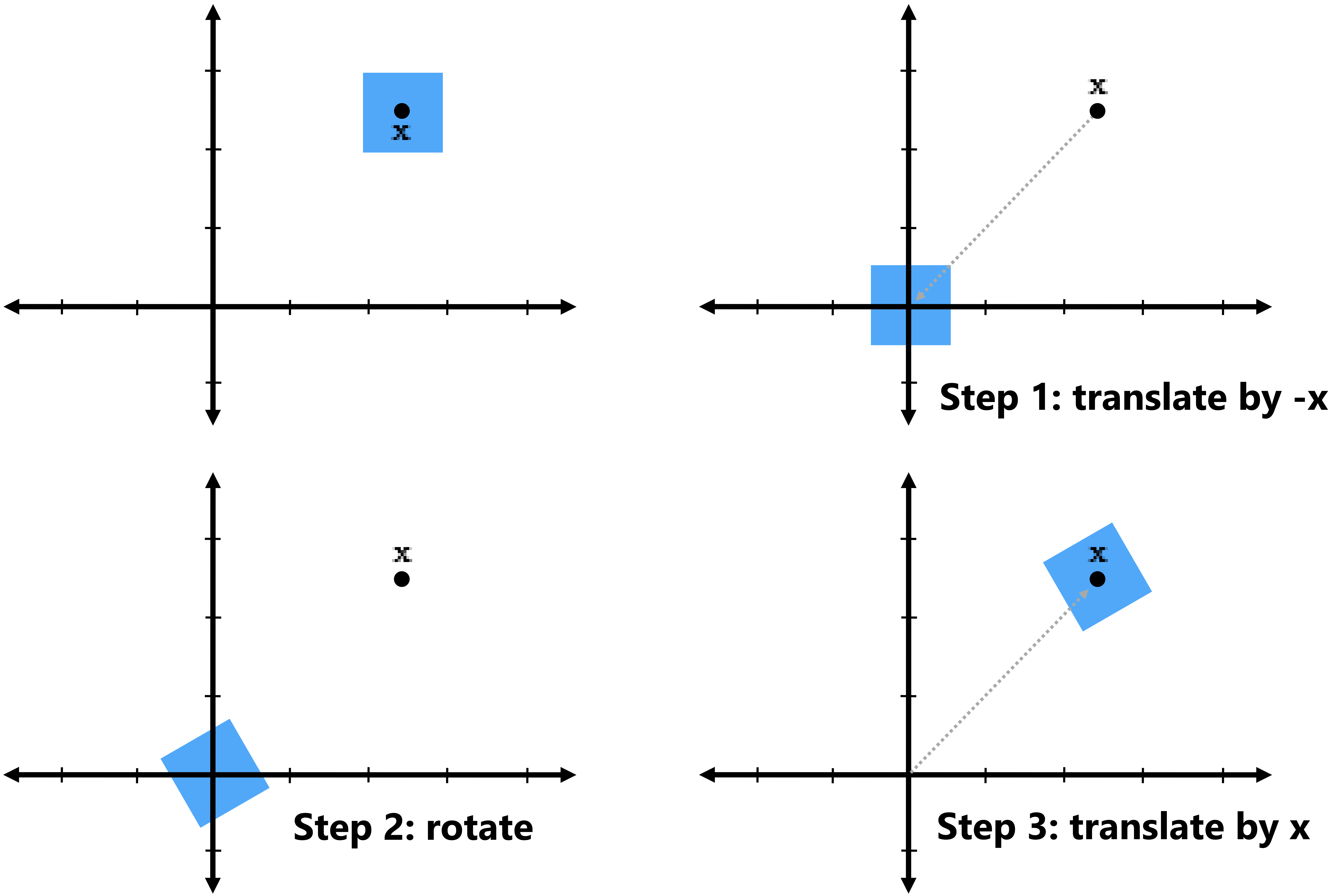
Compose linear transforms via matrix multiplication.

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

How would you perform these transformations?



Common pattern: rotation about point x



Q: In homogenous coordinates, what does the corresponding transformation matrix look like?

Transforms: 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

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

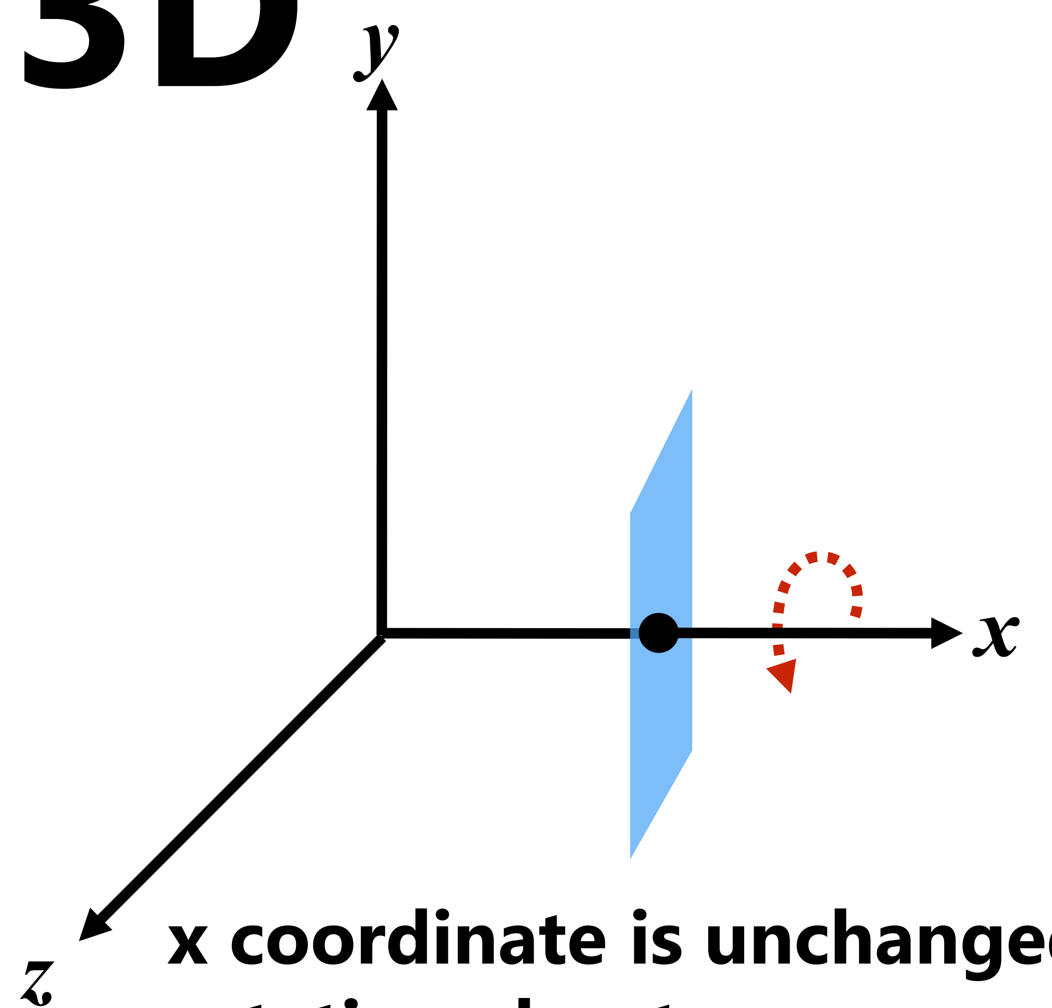
Shear (in x, based on y, z position):

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

Translate:

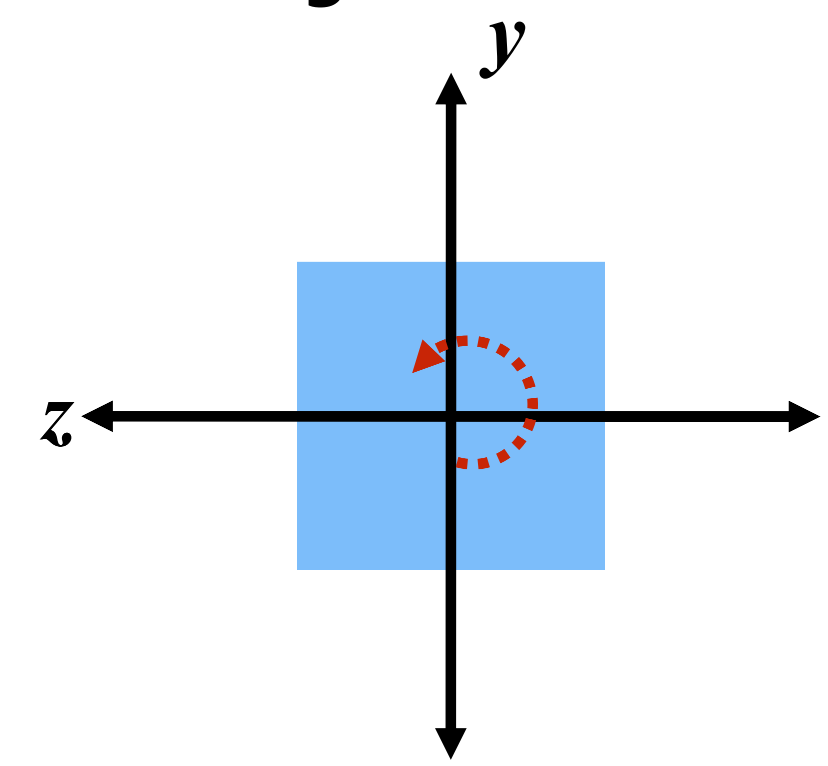
$$\mathbf{T_b} = \begin{bmatrix} 1 & 0 & 0 & \mathbf{b}_x \\ 0 & 1 & 0 & \mathbf{b}_y \\ 0 & 0 & 1 & \mathbf{b}_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotations in 3D

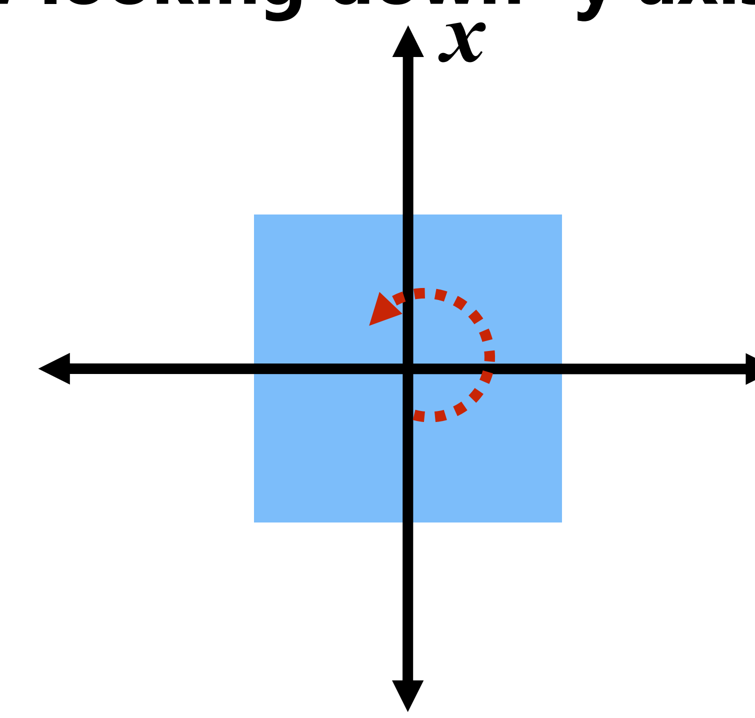


x coordinate is unchanged by rotation about x

View looking down -x axis:



View looking down -y axis:



Rotation about x axis:

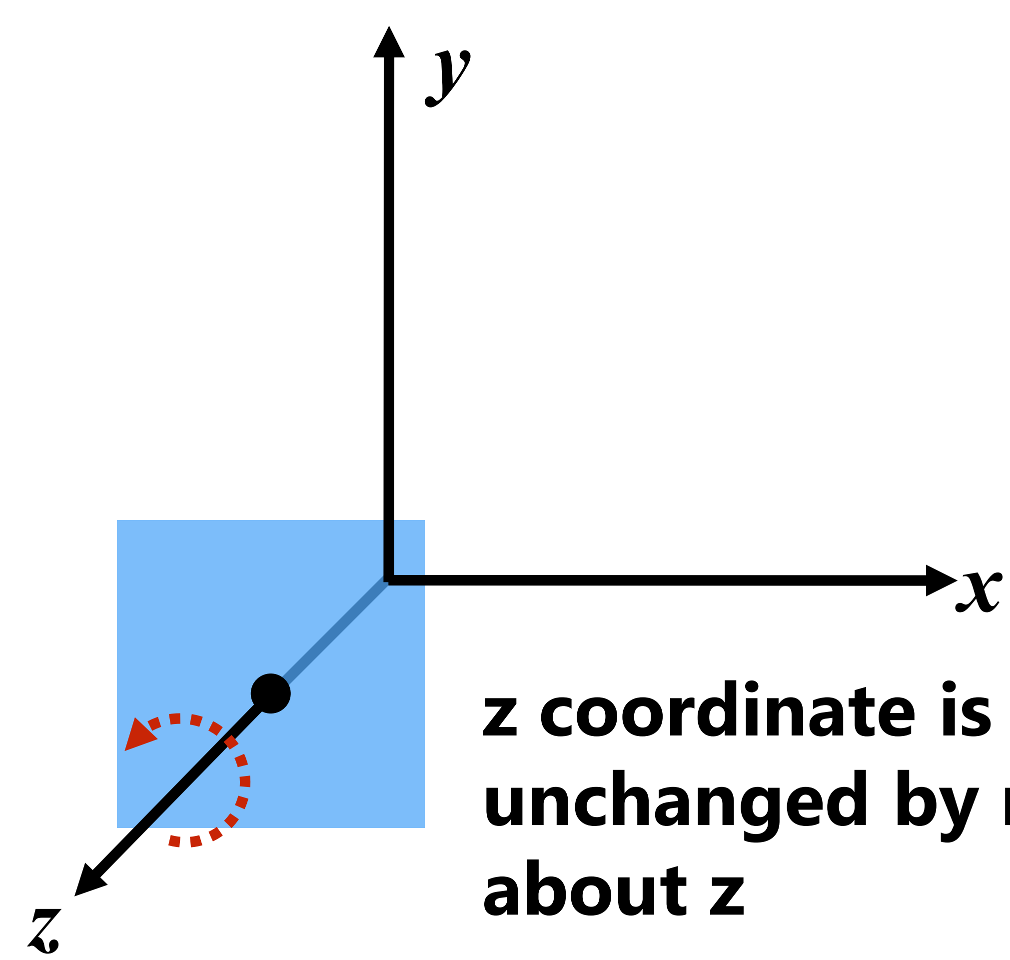
$$\mathbf{R}_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

Rotation about y axis:

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

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

$$\begin{bmatrix} \cos\theta + u_x^2\left(1-\cos\theta\right) & u_x u_y\left(1-\cos\theta\right) - u_z\sin\theta & u_x u_z\left(1-\cos\theta\right) + u_y\sin\theta \\ u_y u_x\left(1-\cos\theta\right) + u_z\sin\theta & \cos\theta + u_y^2\left(1-\cos\theta\right) & u_y u_z\left(1-\cos\theta\right) - u_x\sin\theta \\ u_z u_x\left(1-\cos\theta\right) - u_y\sin\theta & u_z u_y\left(1-\cos\theta\right) + u_x\sin\theta & \cos\theta + u_z^2\left(1-\cos\theta\right) \end{bmatrix}$$

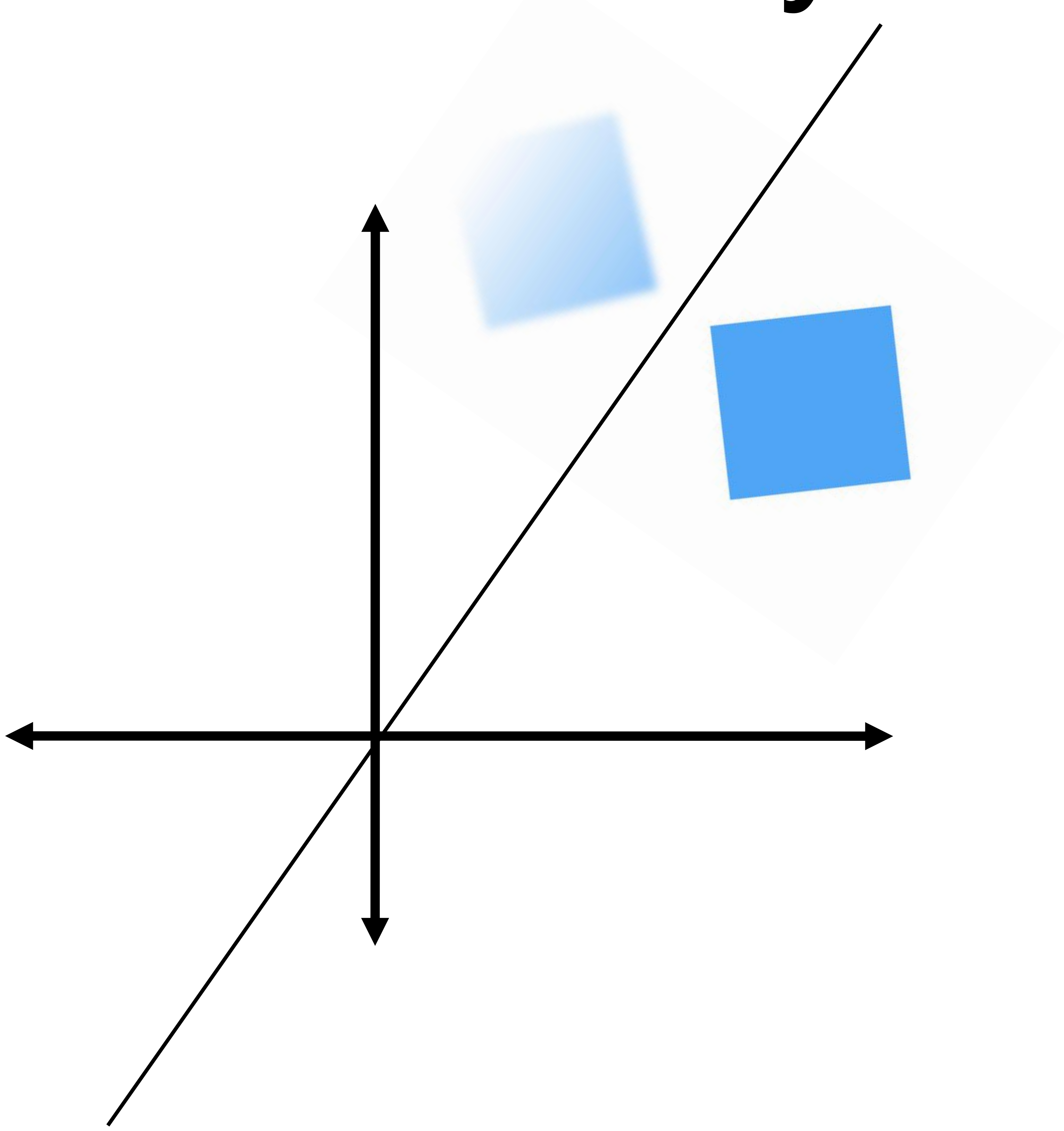
Just memorize this matrix! :-)

Q: Or, figure out how to derive it!

Hint: You already know how to rotate about the z-axis

Exercise

- **Reflection about an arbitrary line**



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

- Construct complex transformations as compositions of basic transforms

- Homogeneous coordinates allow non-linear transforms (e.g., affine, perspective projection) to be expressed as matrix-vector operations (linear transforms)
  - Matrix representation affords simple implementation and efficient composition