Lecture 3:

Math for Graphics and Transforms

Computer Graphics

CMU 15-462/15-662, Spring 2016

Notes

- **Use piazza to communicate with instructors and class**
  - **https://piazza.com/class/ijeillqemgr2l2**

- **Slides will have:**
  - **Things you should know**
  - **Practice questions**

- **Web page, office hours, and Assignment 1 coming soon**
  - **(hopefully later today — watch piazza for the links)**

Do you remember the secret to rasterizing a triangle?

Point-in-triangle test

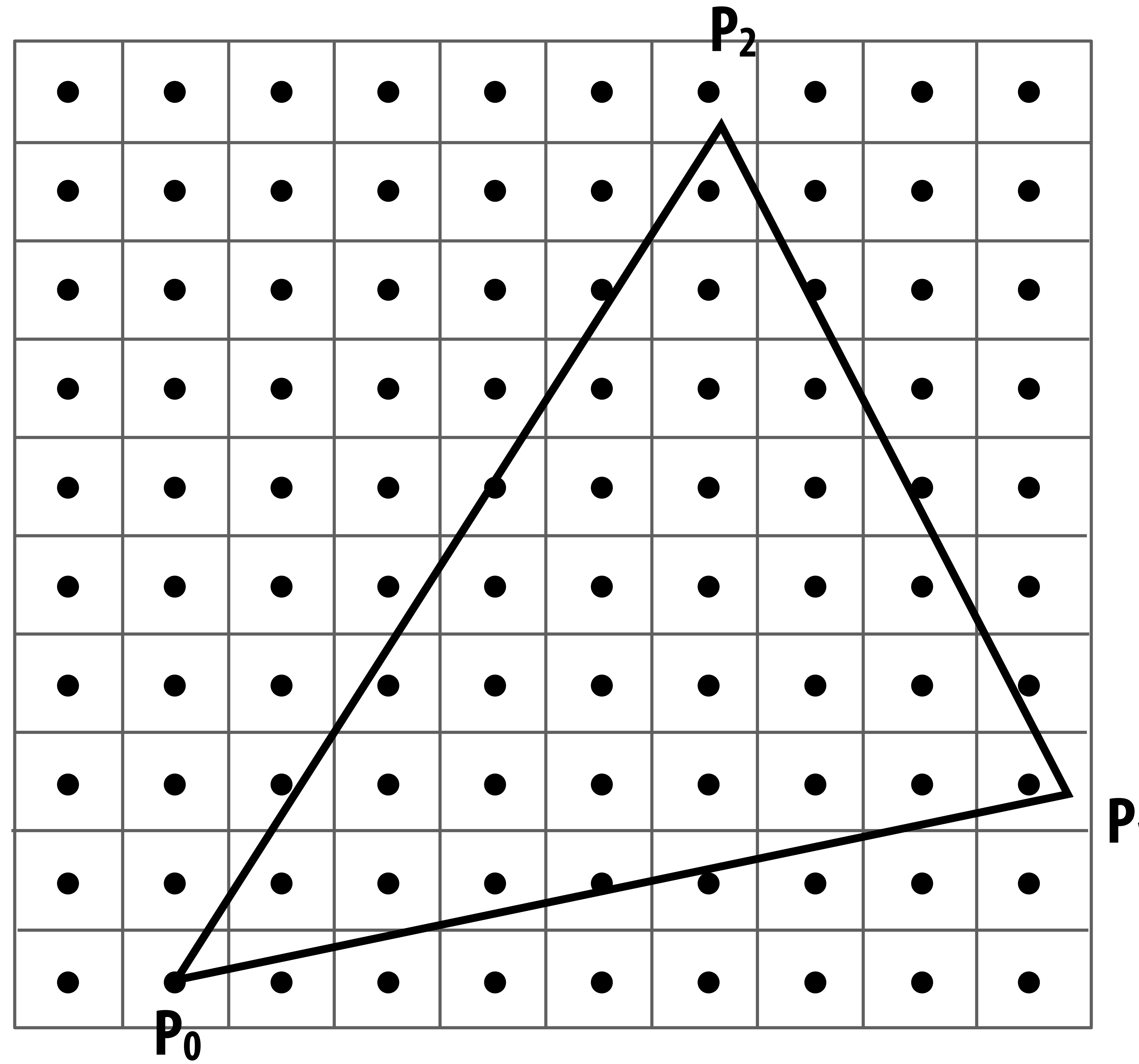
Compute triangle edge equations from projected positions of vertices

$P_i = (X_i, Y_i)$

$dX_i = X_{i+1} - X_i$
$dY_i = Y_{i+1} - Y_i$

$$E_i(x, y) = (x - X_i)\, dY_i - (y - Y_i)\, dX_i$$
$$= A_i x + B_i y + C_i$$

$E_i(x, y) = 0$: point on edge
> 0 : outside edge
< 0 : inside edge



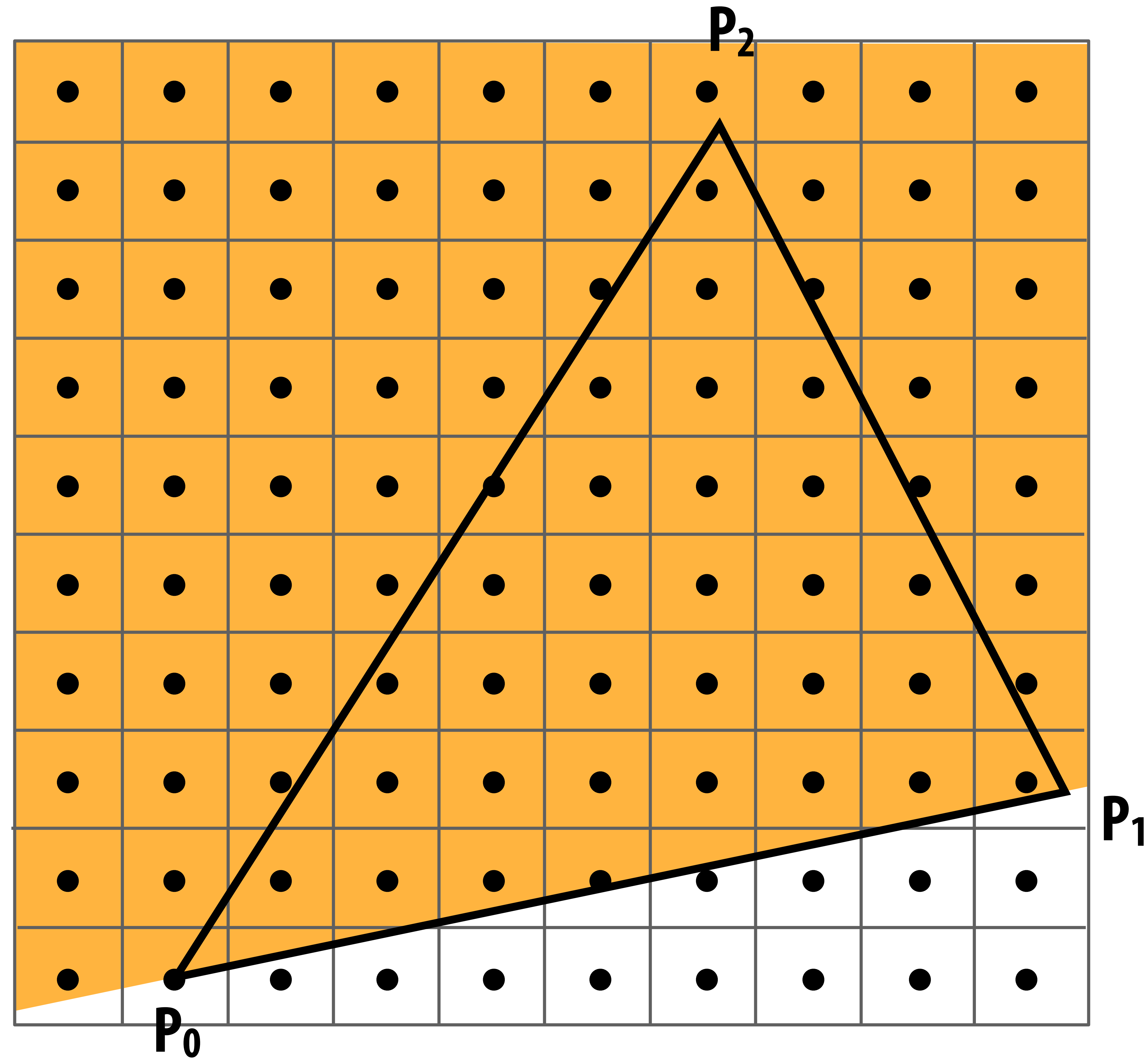
Point-in-triangle test

$P_i = (X_i, Y_i)$

$dX_i = X_{i+1} - X_i$
$dY_i = Y_{i+1} - Y_i$

$$E_i(x, y) = (x - X_i)\, dY_i - (y - Y_i)\, dX_i$$
$$= A_i\, x + B_i\, y + C_i$$

$E_i(x, y) = 0$: point on edge
> 0 : outside edge
< 0 : inside edge



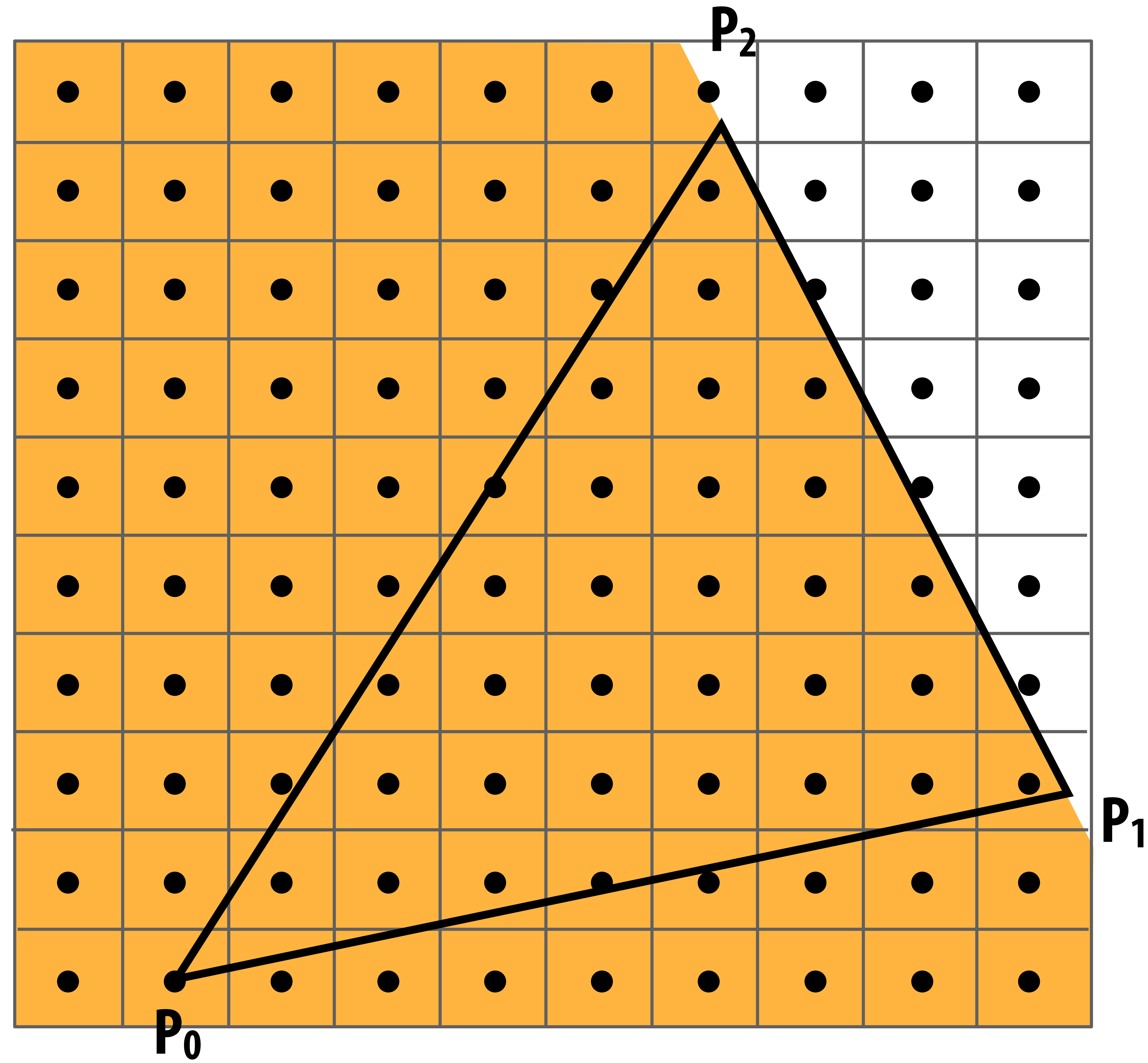
Point-in-triangle test

$P_i = (X_i, Y_i)$

$dX_i = X_{i+1} - X_i$
$dY_i = Y_{i+1} - Y_i$

$$E_i(x, y) = (x - X_i)\, dY_i - (y - Y_i)\, dX_i$$
$$= A_i x + B_i y + C_i$$

$E_i(x, y) = 0$: point on edge
> 0 : outside edge
< 0 : inside edge



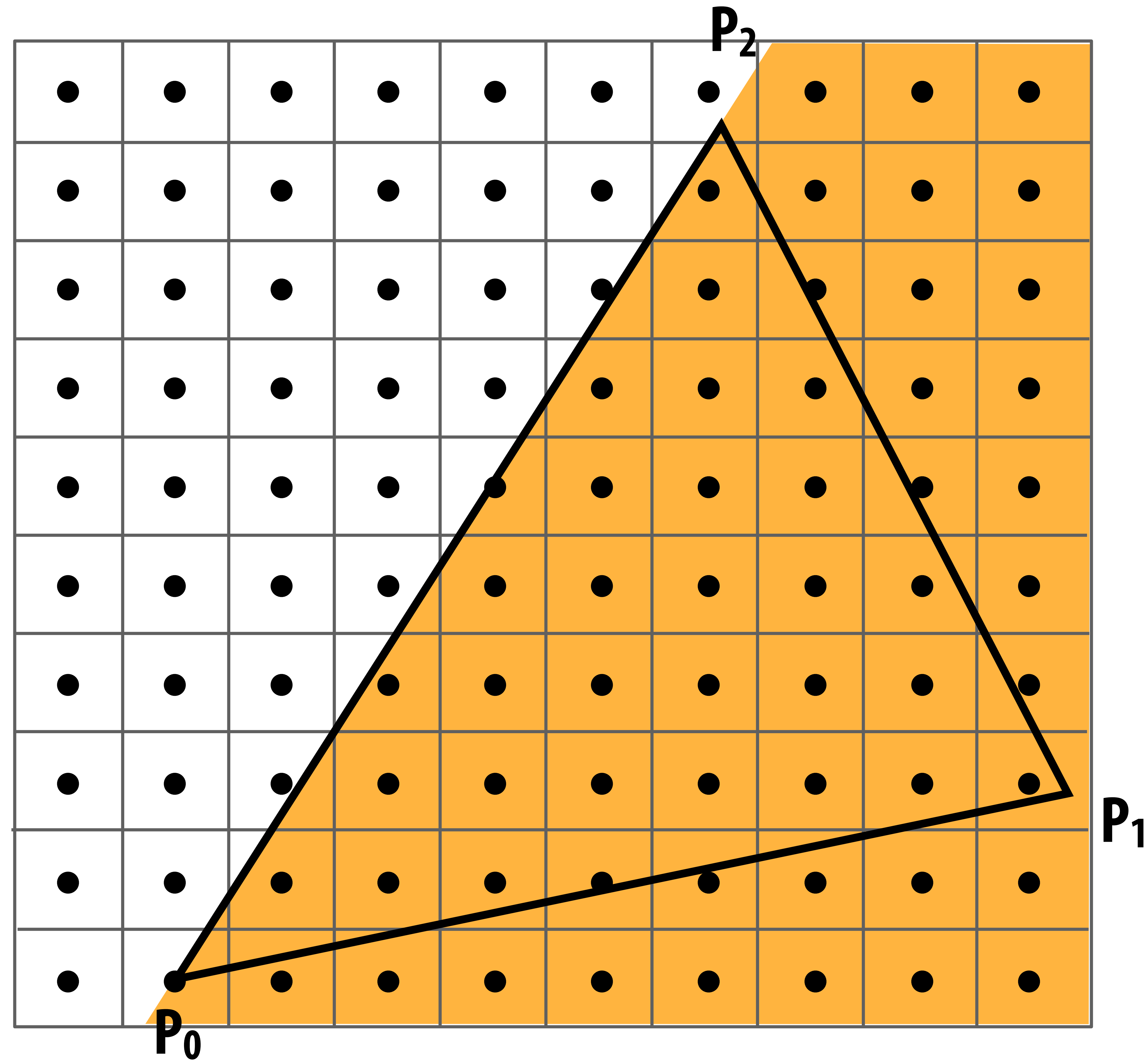
Point-in-triangle test

$P_i = (X_i, Y_i)$

$dX_i = X_{i+1} - X_i$
$dY_i = Y_{i+1} - Y_i$

$$\begin{aligned} E_i(x, y) &= (x - X_i)\, dY_i - (y - Y_i)\, dX_i \\ &= A_i\, x + B_i\, y + C_i \end{aligned}$$

$E_i(x, y) = 0$: point on edge
> 0 : outside edge
< 0 : inside edge

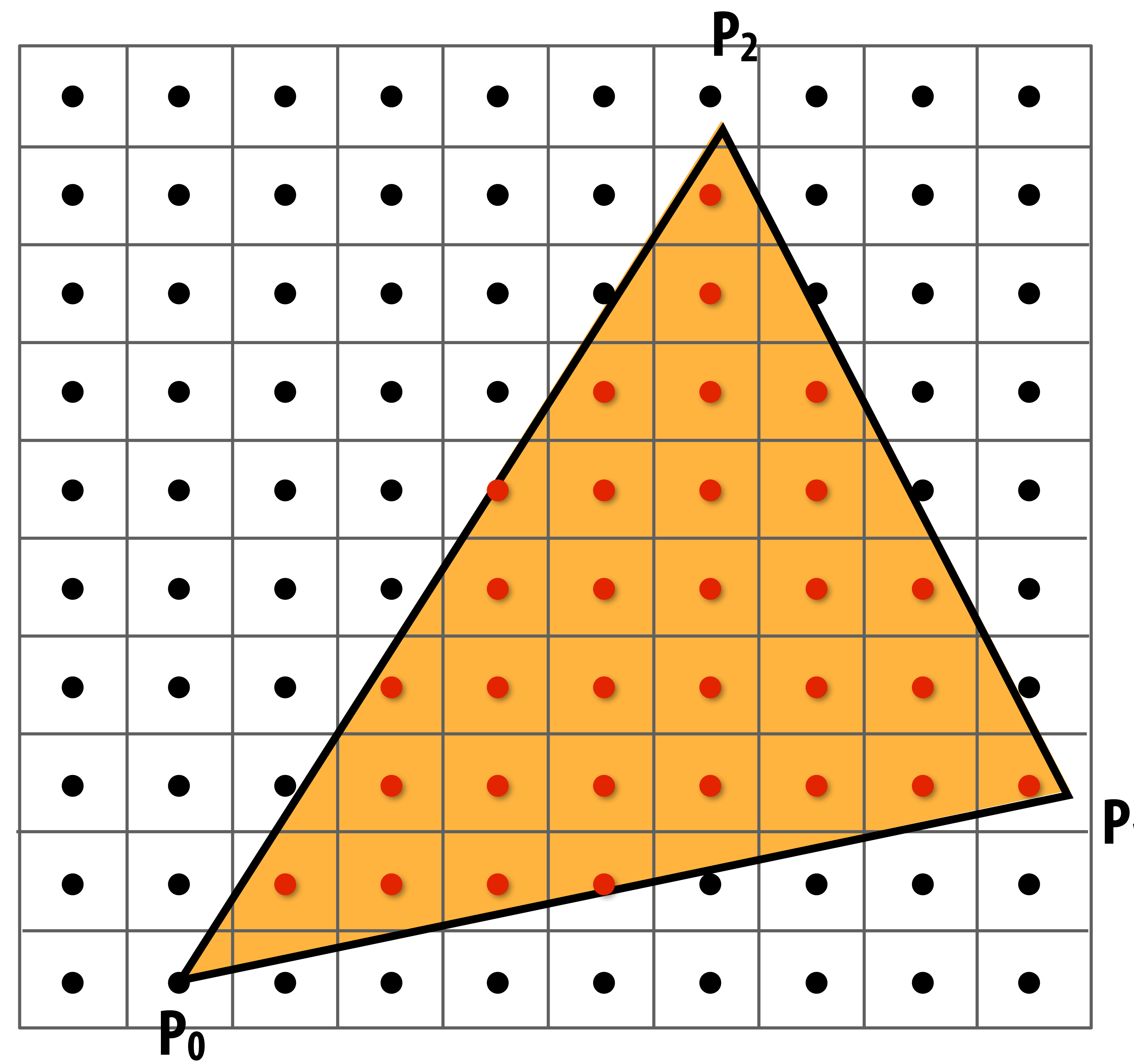


Point-in-triangle test

Sample point $s = (sx, sy)$ **is inside the triangle if it is inside all three edges.**

$$
\begin{aligned}
&inside(sx, sy) = \\
&\quad E_0(sx, sy) < 0 \;\&\& \\
&\quad E_1(sx, sy) < 0 \;\&\& \\
&\quad E_2(sx, sy) < 0;
\end{aligned}
$$

Note: actual implementation of $inside(sx,sy)$ **involves ≤ checks based on the triangle coverage edge rules (see beginning of lecture)**



Sample points inside triangle are highlighted red.

Incremental triangle traversal

$P_i = (X_i, Y_i)$

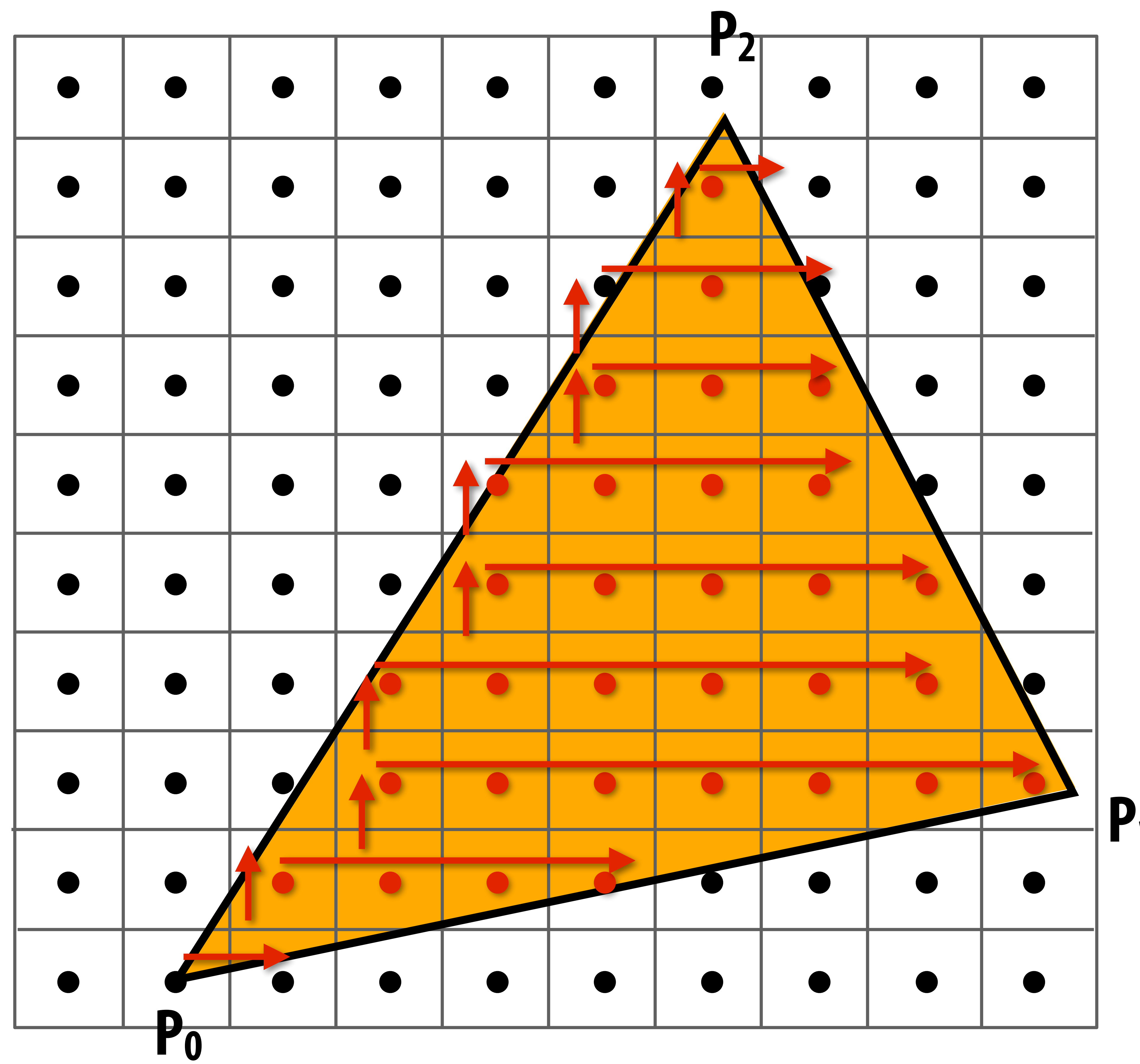
$dX_i = X_{i+1} - X_i$
$dY_i = Y_{i+1} - Y_i$

$$E_i(x, y) = (x - X_i)\, dY_i - (y - Y_i)\, dX_i$$
$$= A_i\, x + B_i\, y + C_i$$

$E_i(x, y) = 0$: point on edge
> 0 : outside edge
< 0 : inside edge

Efficient incremental update:

$$dE_i(x+1,y) = E_i(x,y) + dY_i = E_i(x,y) + A_i$$
$$dE_i(x,y+1) = E_i(x,y) + dX_i = E_i(x,y) + B_i$$



Incremental update saves computation:
Only one addition per edge, per sample test

Many traversal orders are possible: backtrack, zig-zag, Hilbert/Morton curves (locality maximizing)

Math for Computer Graphics

- Points vs. vectors
- Homogeneous coordinates
- Dot product
- Cross product
- Matrix multiplication
- Parametric line / surface
- Implicit line / surface

Points vs. Vectors

- **A point is a location in space. We might write it casually as**

$$P_i = (X_{i,} Y_i)$$

$$P_j = (X_{j,} Y_{j,} Z_j)$$

- **A vector has magnitude and direction. We might write it casually in a similar way**

$$V_i = (X_{i,} Y_i)$$

$$V_j = (X_{j,} Y_{j,} Z_j)$$

Homogeneous Coordinates

- **For graphics operations, we will often represent points and vectors in homogeneous coordinates. (We will soon see why this is useful.*)**

$$P_i = \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \qquad\qquad V_i = \begin{bmatrix} x_i \\ y_i \\ z_i \\ 0 \end{bmatrix}$$

3D point **3D vector**

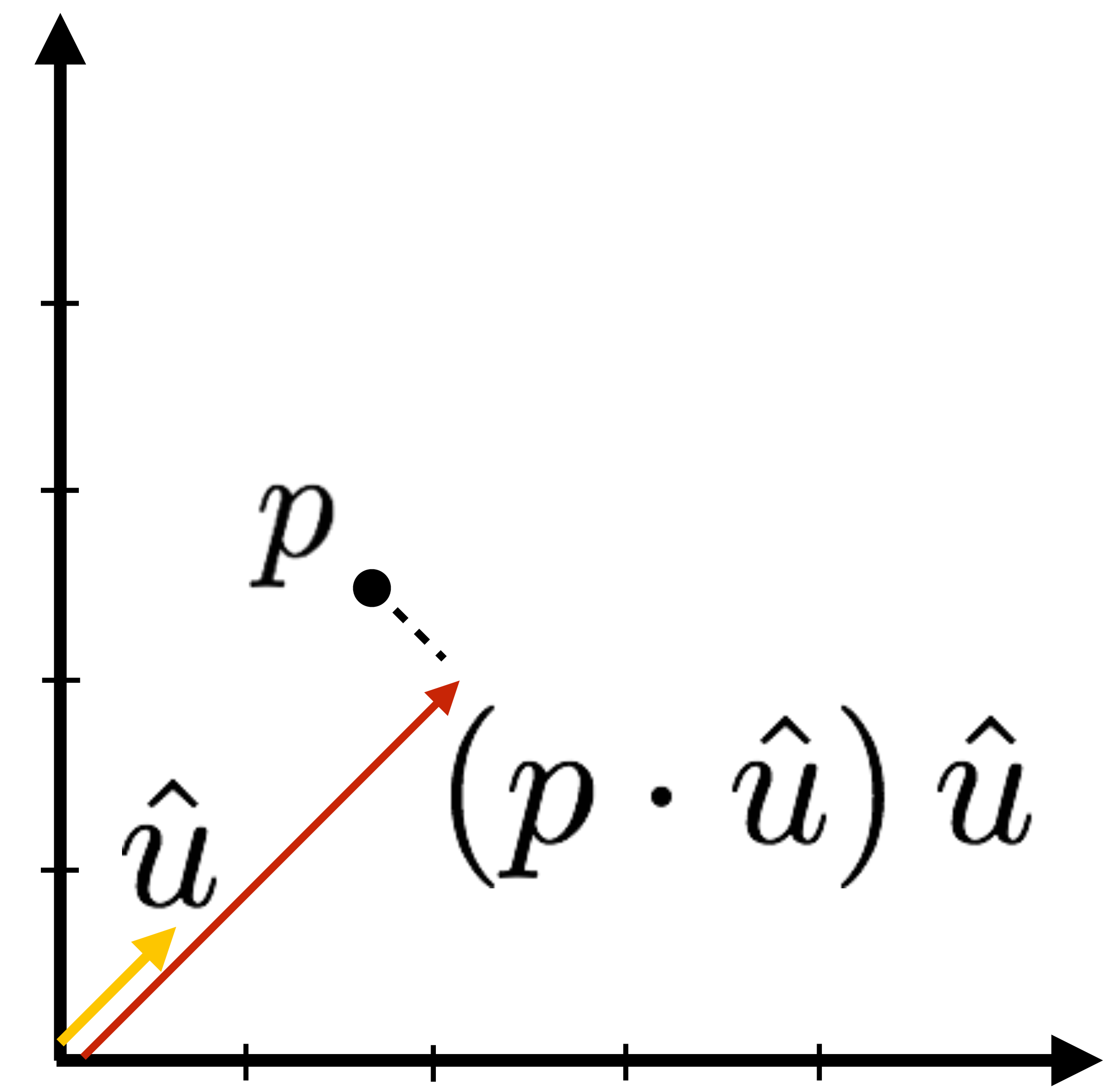
*Looking ahead — lifting points and vectors into homogeneous coordinate space allows us to perform affine operations as linear transforms.

Dot Product

- Dot products are required for many operations in graphics, including projection

$$p \cdot \hat{u} = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T \begin{bmatrix} \hat{u}_x \\ \hat{u}_y \\ \hat{u}_z \end{bmatrix} = p_x \hat{u}_x + p_y \hat{u}_y + p_z \hat{u}_z$$

- If $\hat{u}$ is a unit vector, $p \cdot \hat{u}$ gives the magnitude of the projection of p onto $\hat{u}$



Cross Product

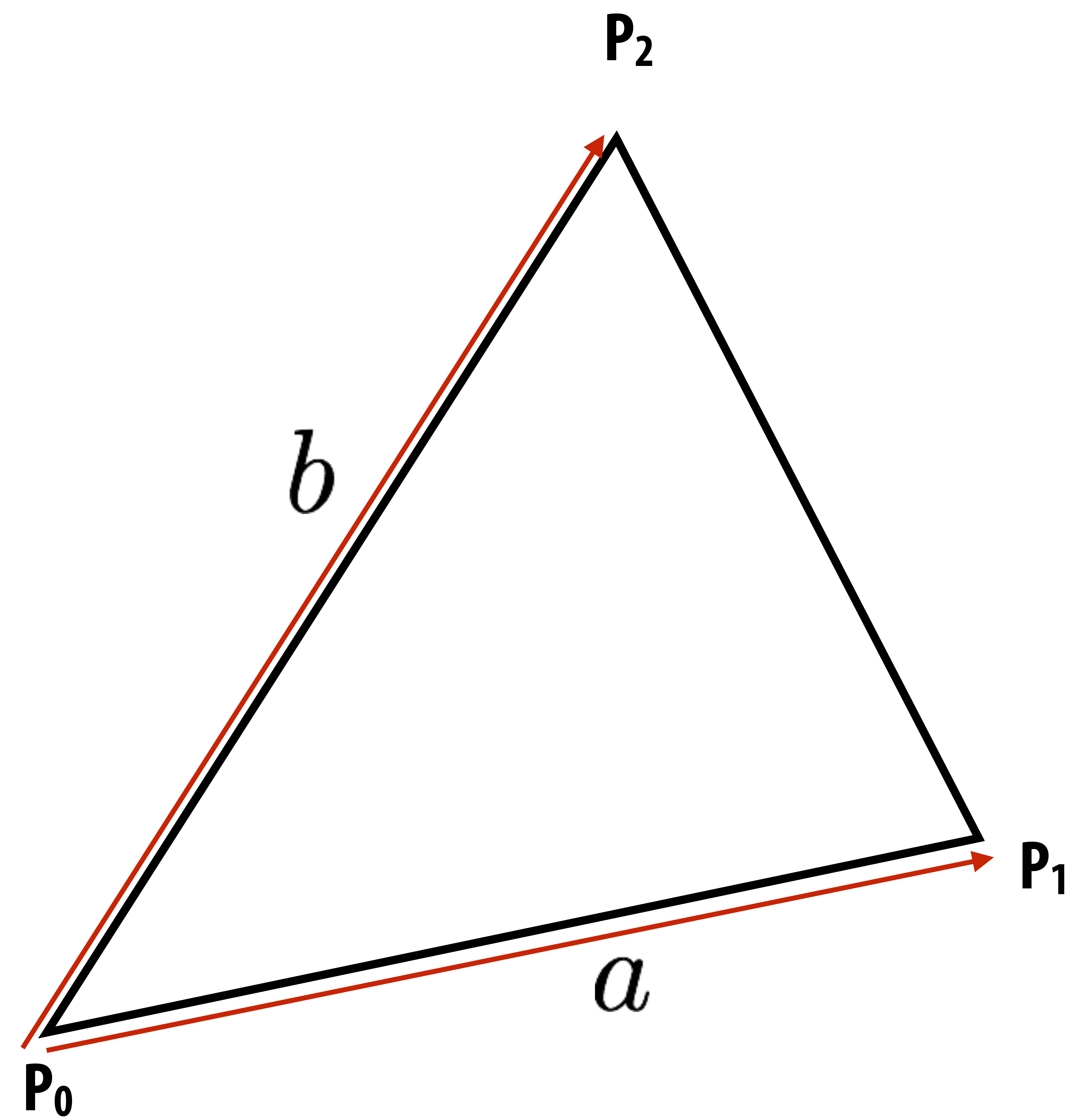
- **The cross product is also important for graphics operations. Examples are to compute a surface normal or to form an orthonormal coordinate system**

$$a = (P_1 - P_0)$$
$$b = (P_2 - P_0)$$

$$n = a \times b = \begin{bmatrix} a_y b_z - a_z b_y \\ a_z b_x - b_x a_z \\ a_x b_y - a_y b_x \end{bmatrix}$$

$$\|n\| = \sqrt{n_x^2 + n_y^2 + n_z^2}$$

$$\hat{n} = \frac{n}{\|n\|}$$



Matrix Multiplication

- **We will use matrix multiplication to compose transforms together, to transform points and vectors, and to do perspective projection**

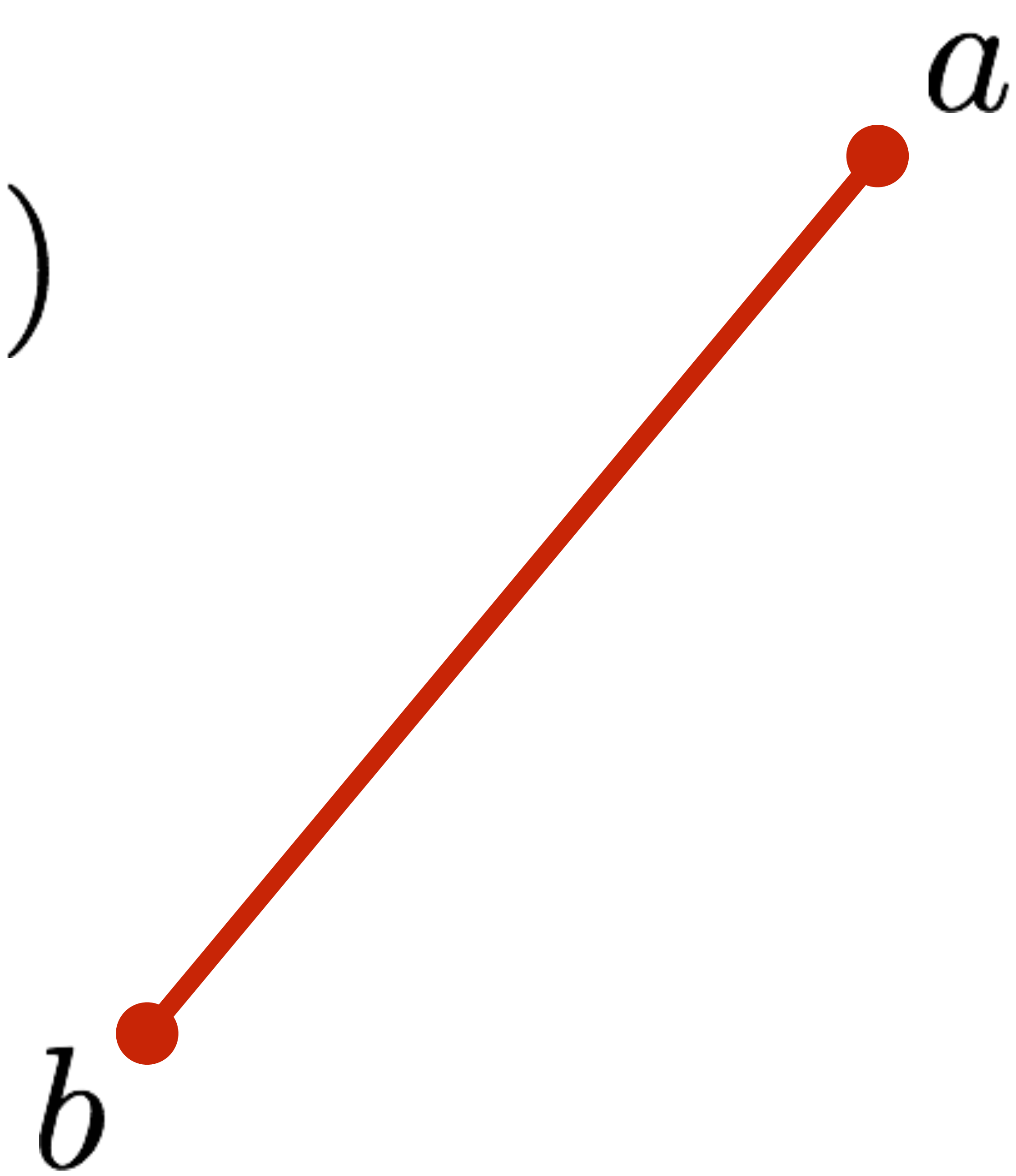
- **What do you think this matrix operation accomplishes?**

$$\begin{bmatrix} u_x & u_y & u_z & t_x \\ v_x & v_y & v_z & t_y \\ w_x & w_y & w_z & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} u_x P_x + u_y P_y + u_z P_z + t_x \\ v_x P_x + v_y P_y + v_z P_z + t_y \\ w_x P_x + w_y P_y + w_z P_z + t_z \\ 1 \end{bmatrix} = \begin{bmatrix} u \cdot P + t_x \\ v \cdot P + t_y \\ w \cdot P + t_z \\ 1 \end{bmatrix}$$

Parametric representation of a line

- **A line can be expressed as a function of a single parameter**

$$P(t) = a + t(b - a)$$



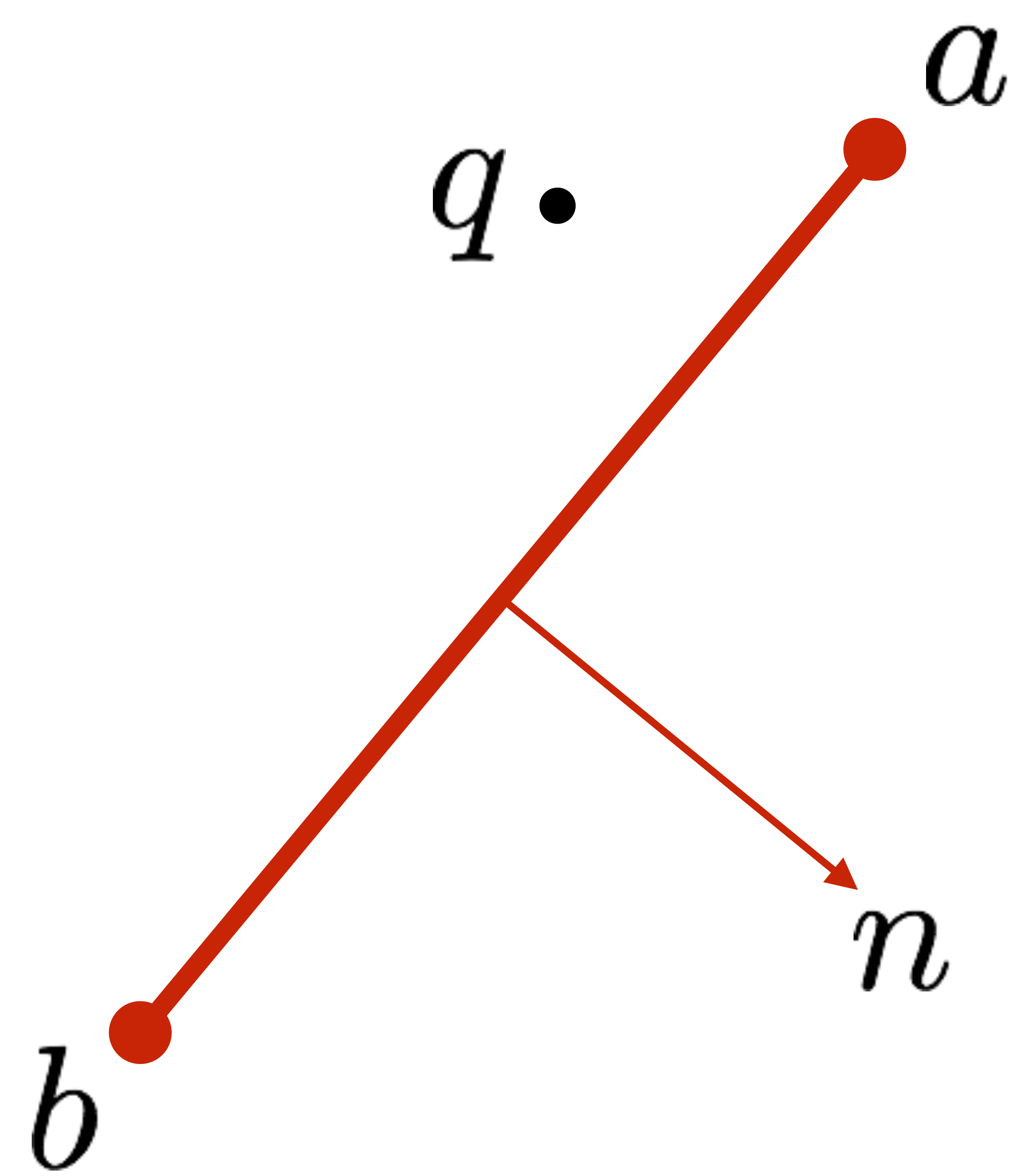
Implicit representation of a line

- **A line can also be expressed implicitly, with an expression that evaluates to zero only for points that are exactly on the line**

$$v = (b - a)$$

$$n = (v \times \hat{z}) = \begin{bmatrix} v_y \\ -v_x \end{bmatrix}$$

$$E(q) = (q - a) \cdot n$$



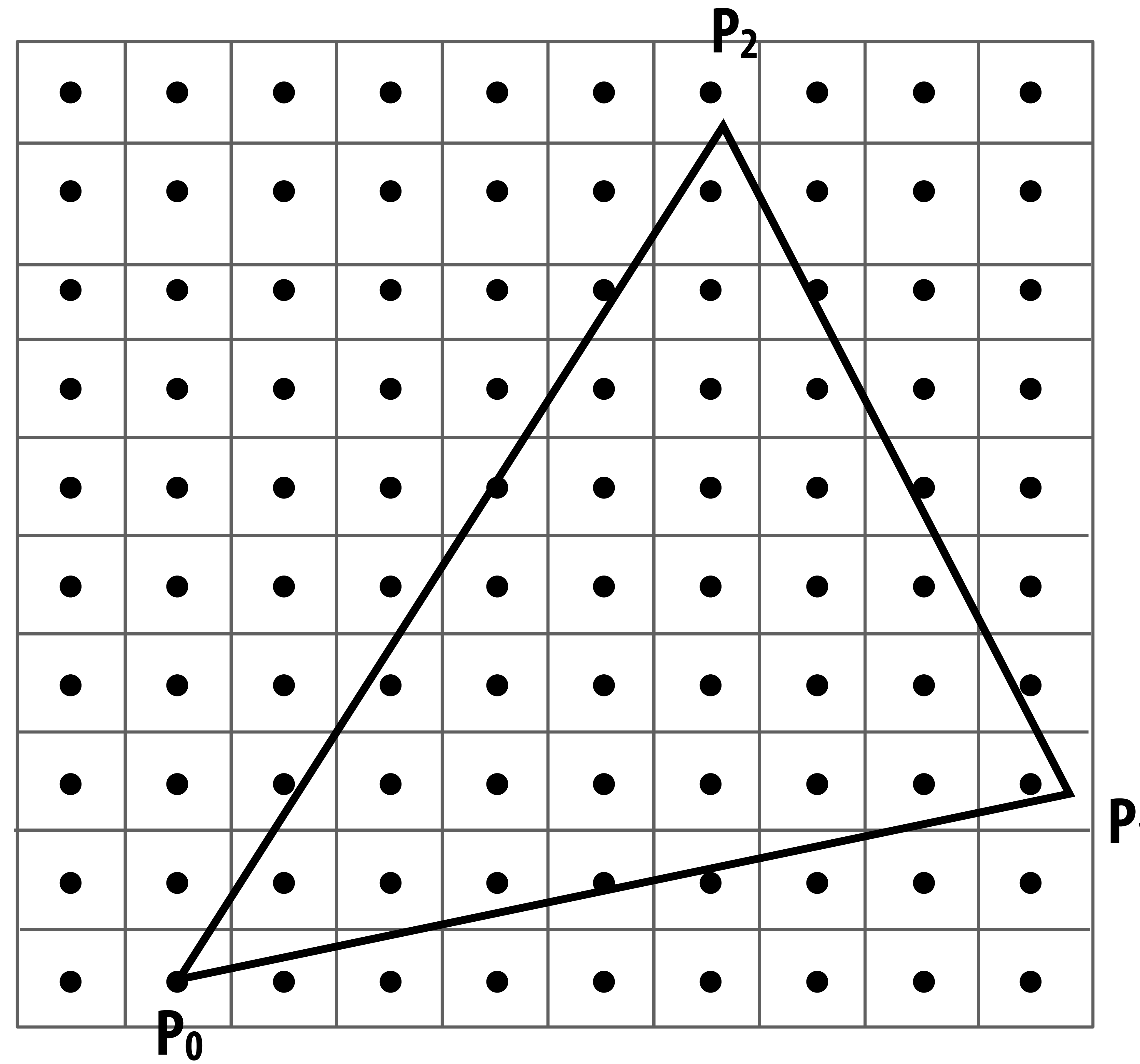
Now, it should be easy to derive and understand this edge equation

$P_i = (X_i, Y_i)$

$dX_i = X_{i+1} - X_i$
$dY_i = Y_{i+1} - Y_i$

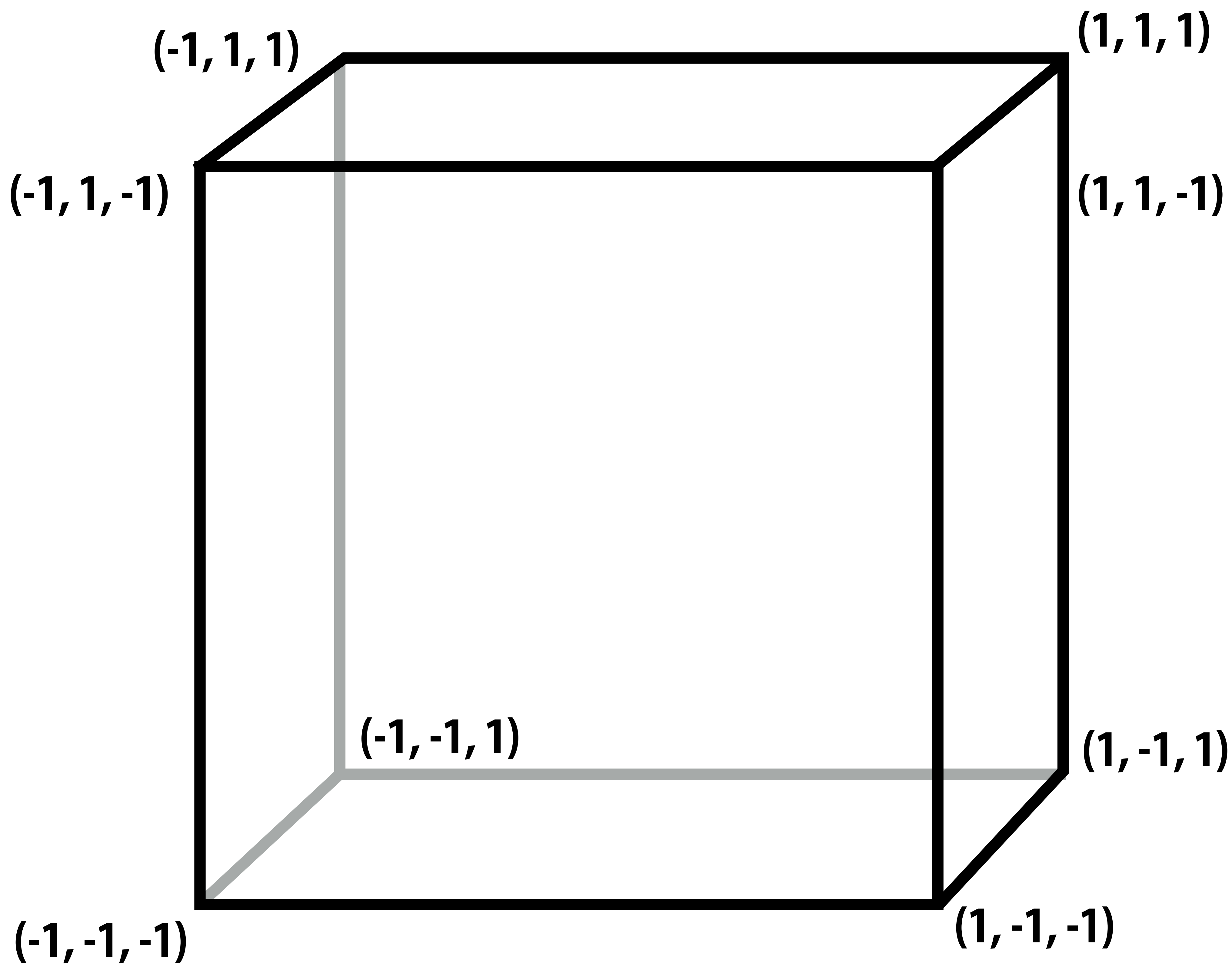
$$E_i(x, y) = (x - X_i)\, dY_i - (y - Y_i)\, dX_i$$
$$= A_i\, x + B_i\, y + C_i$$

$E_i(x, y) = 0$: point on edge
> 0 : outside edge
< 0 : inside edge

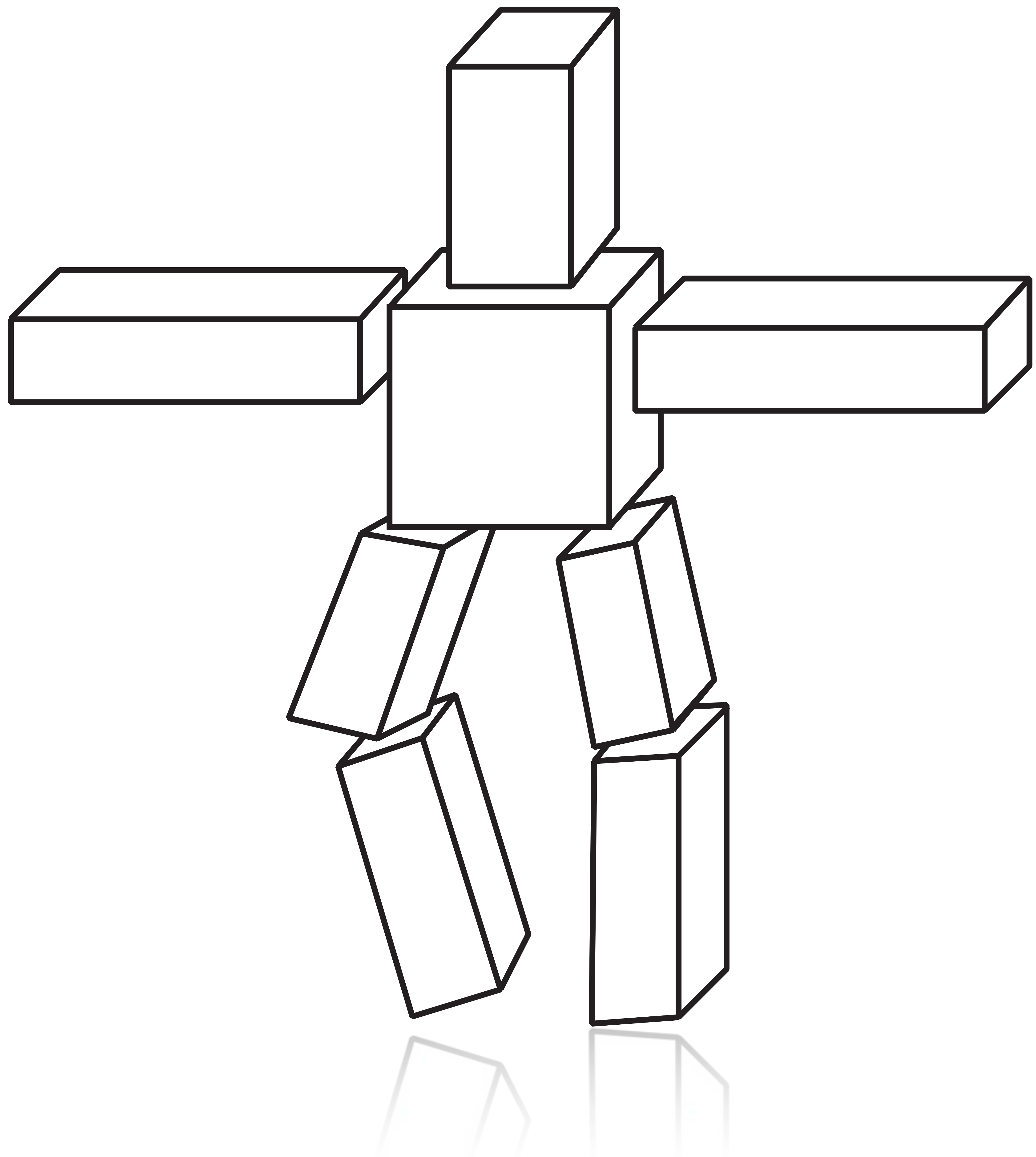


Next topic: Transforms

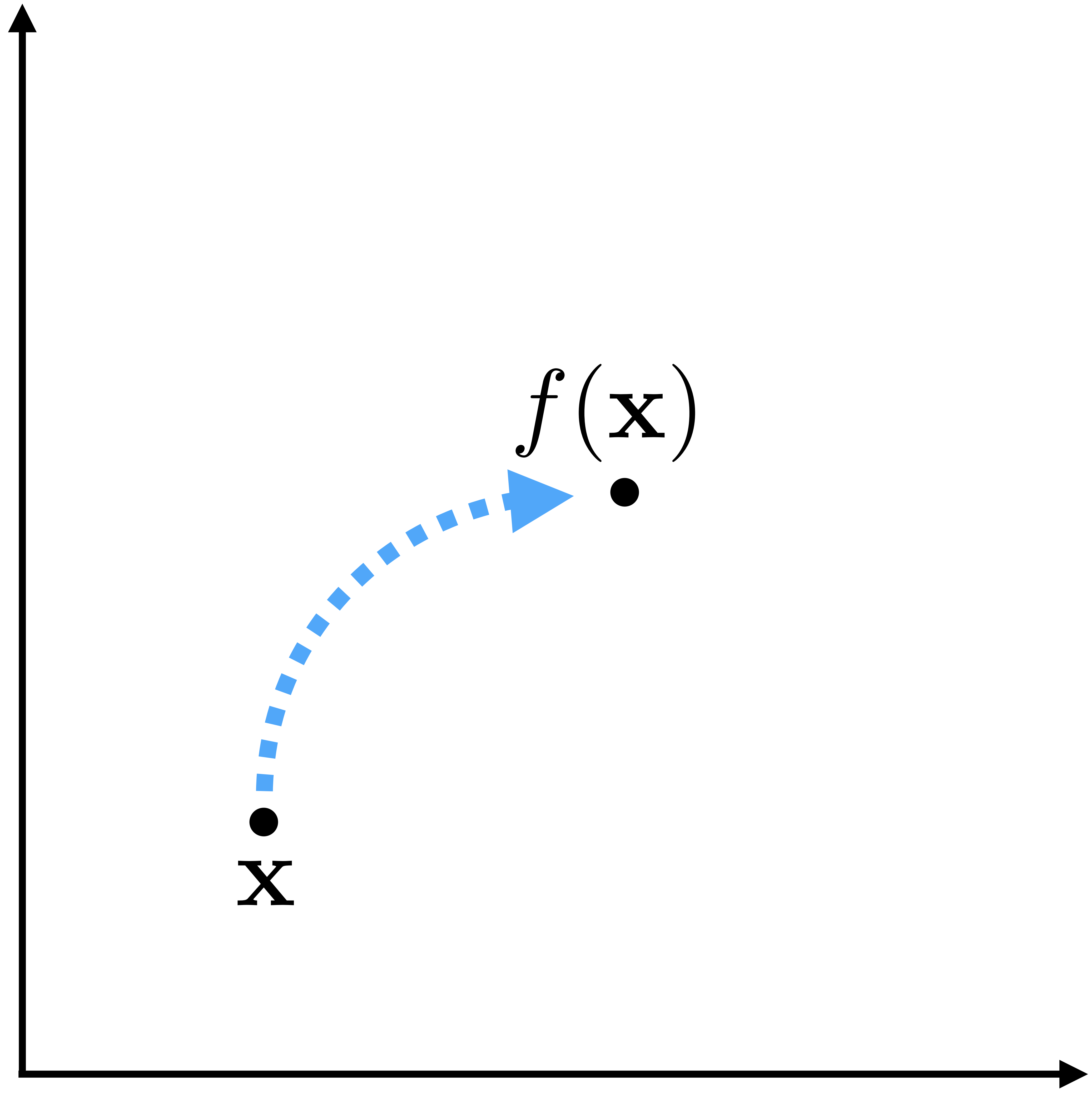
Cube



Cube man



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



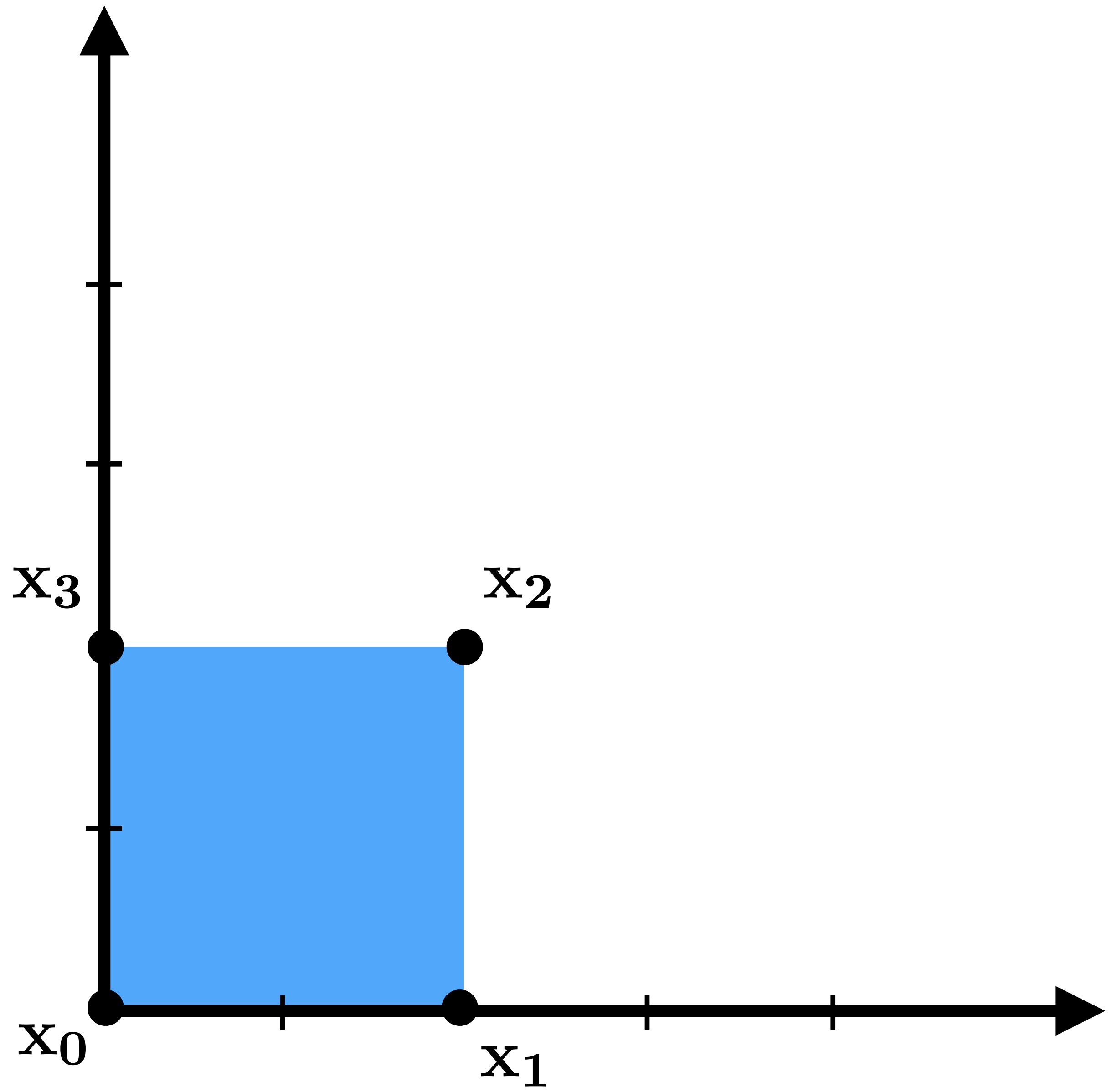
Linear transforms

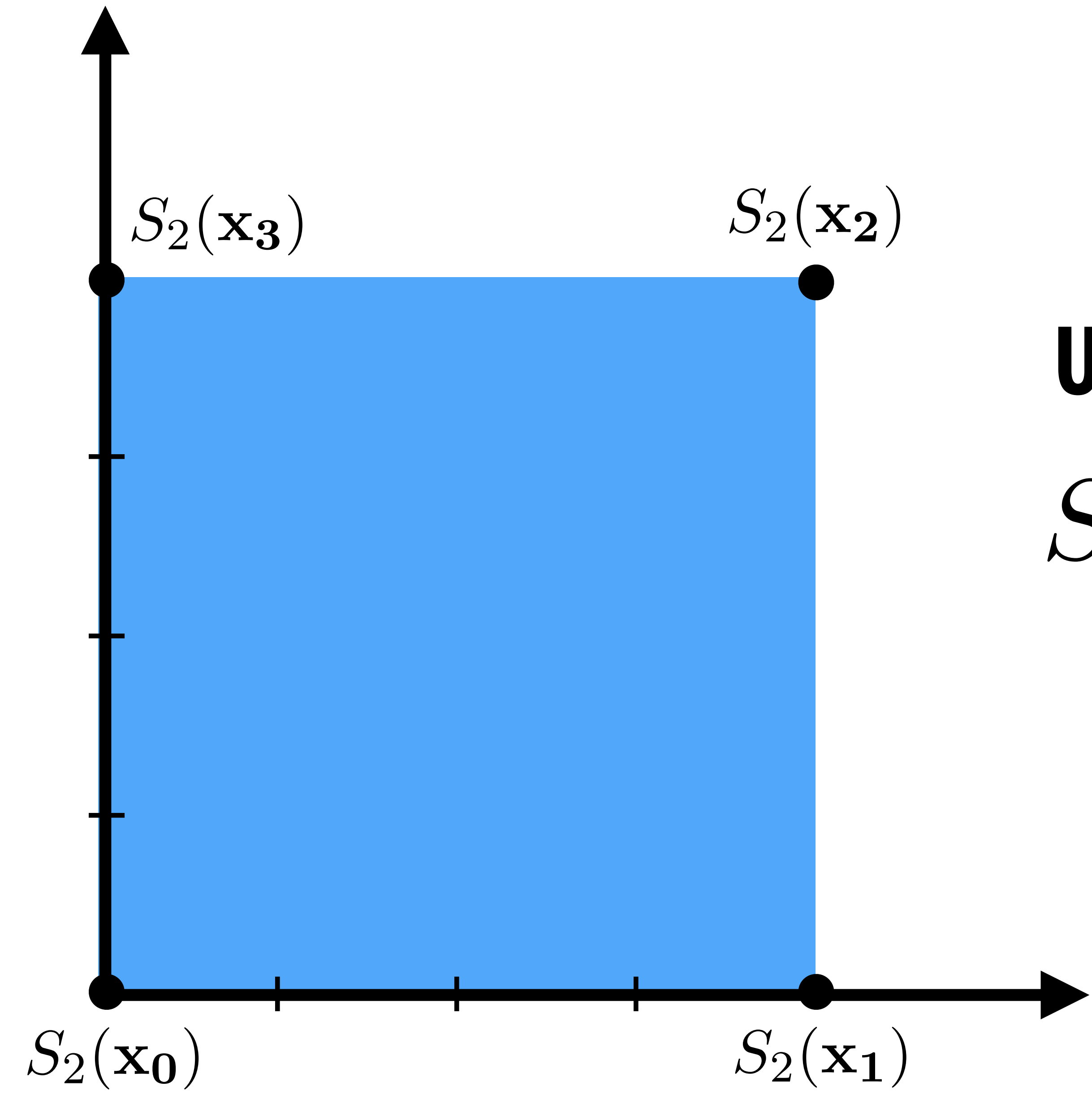
Transform f is linear if and only if:

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

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

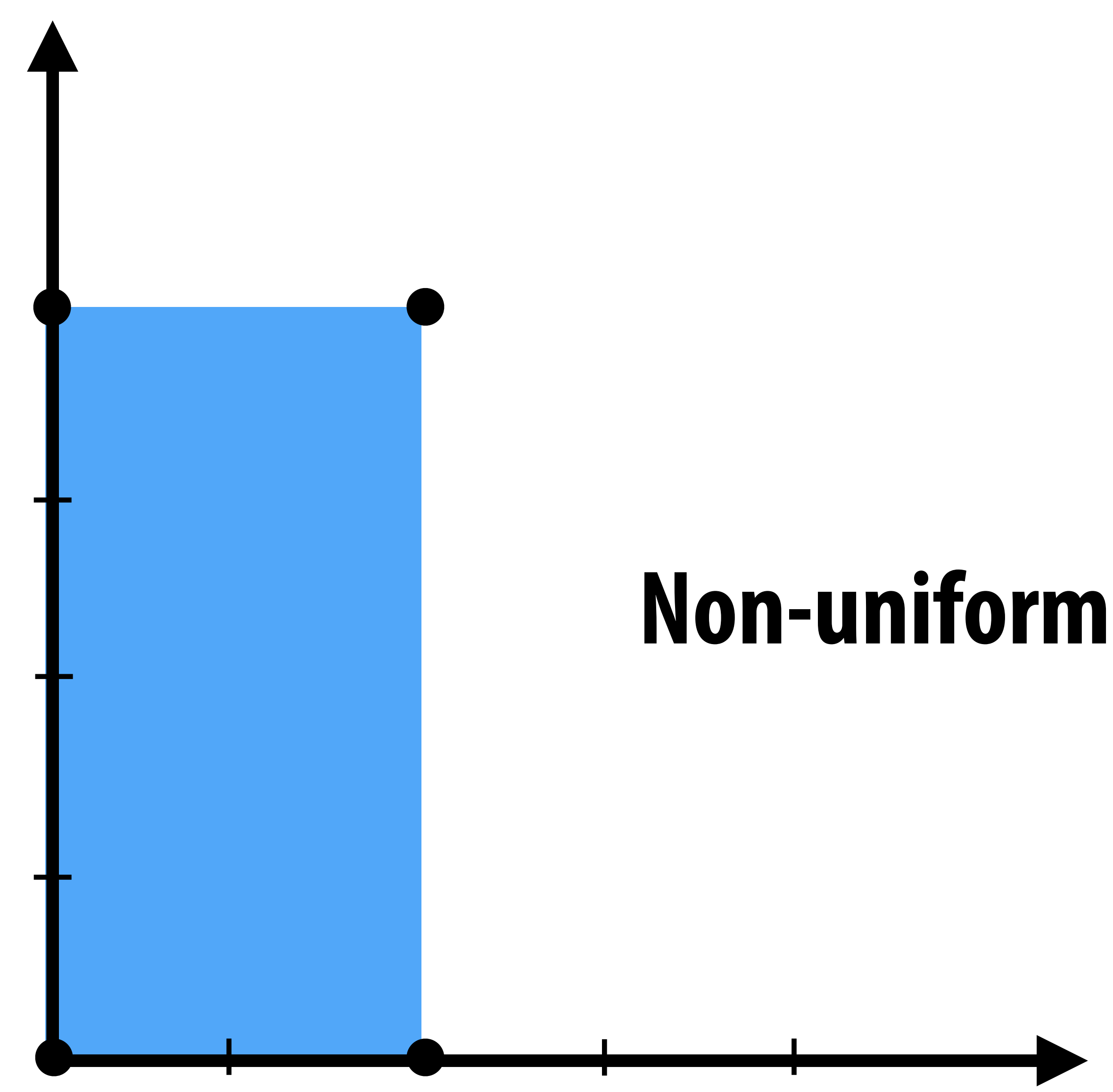
Scale





Uniform scale:

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



Non-uniform scale??

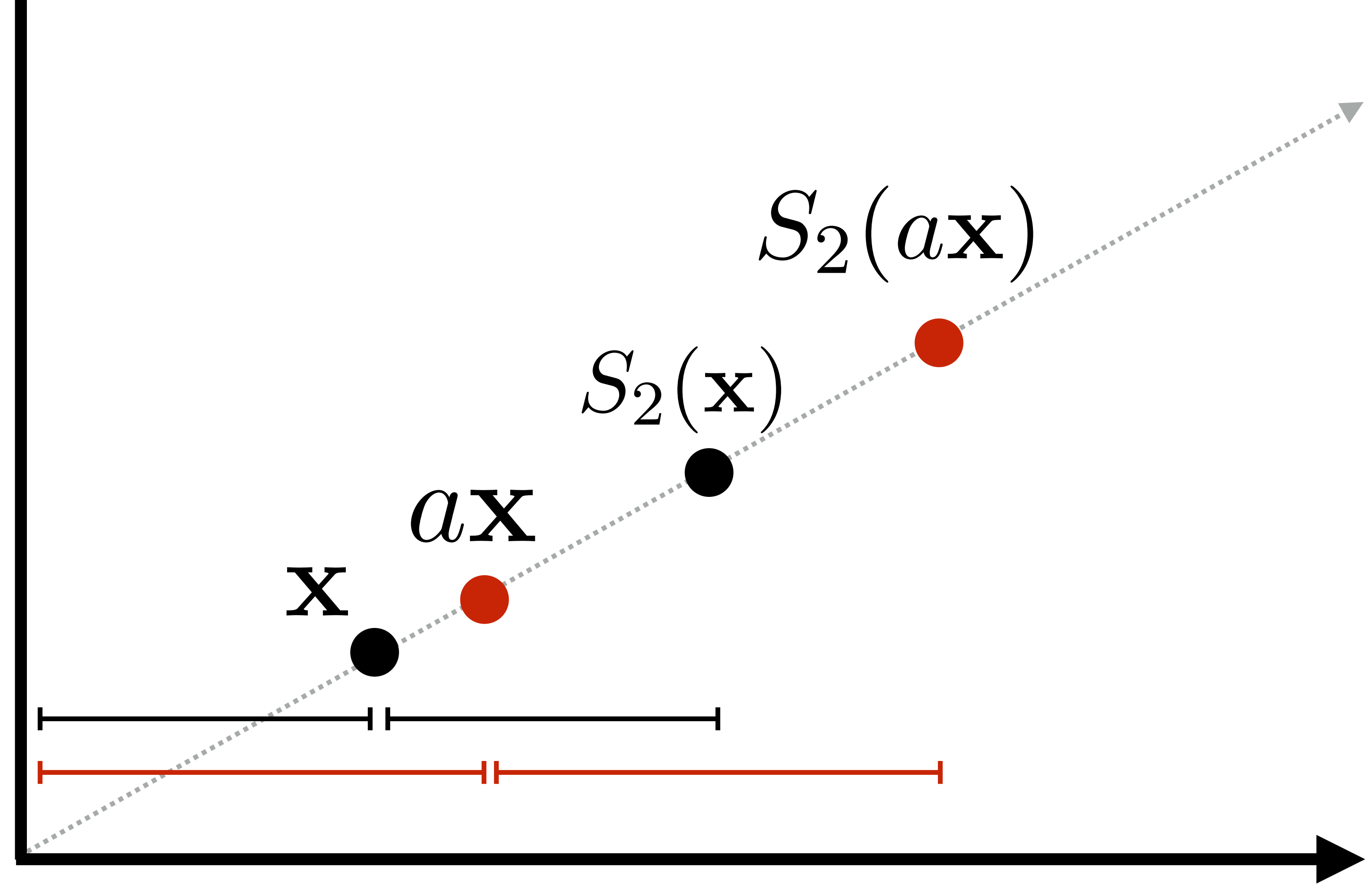
Is scale a linear transform?

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

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

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

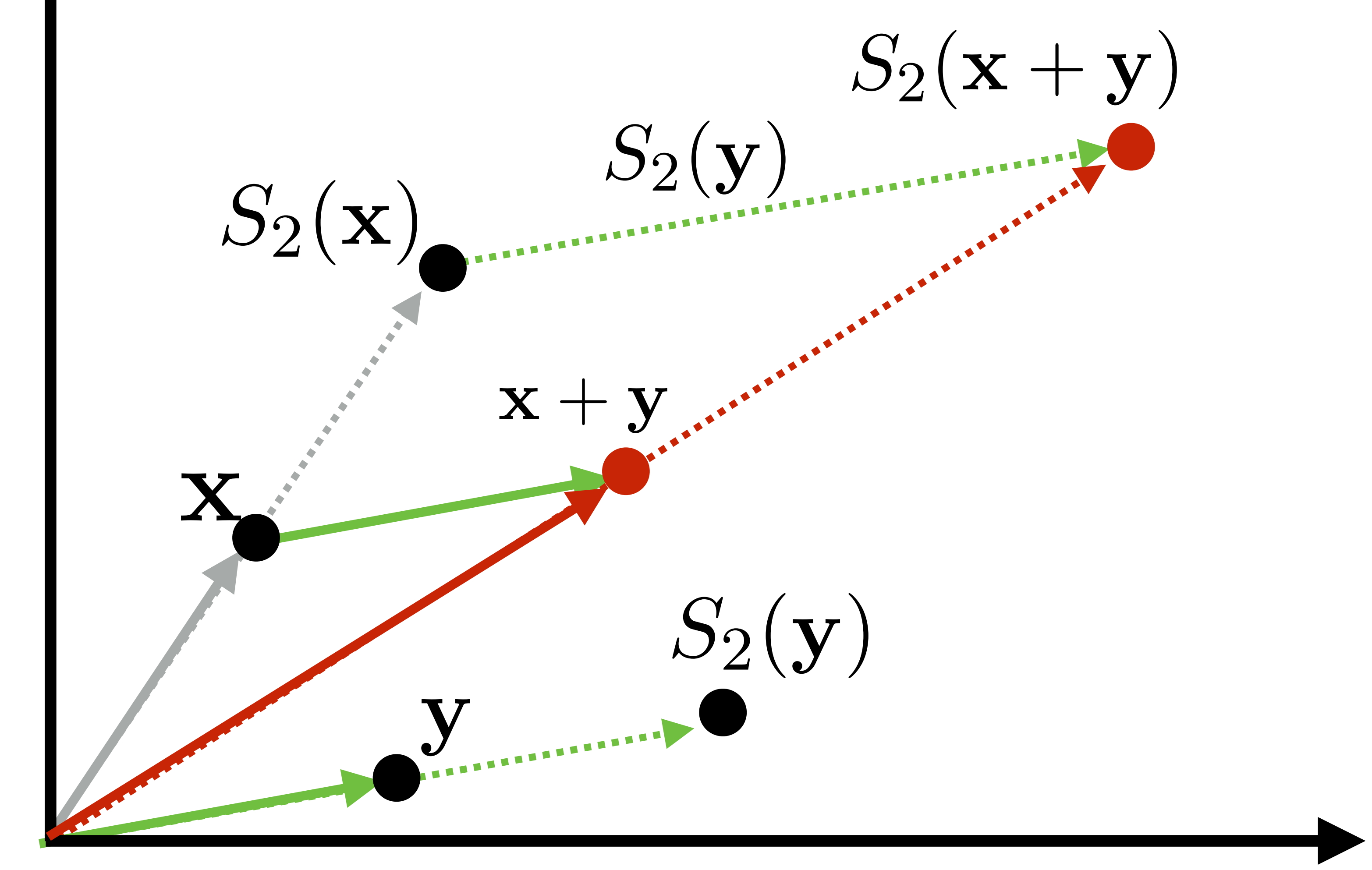
$$S_2(a\mathbf{x}) = aS_2(\mathbf{x})$$



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

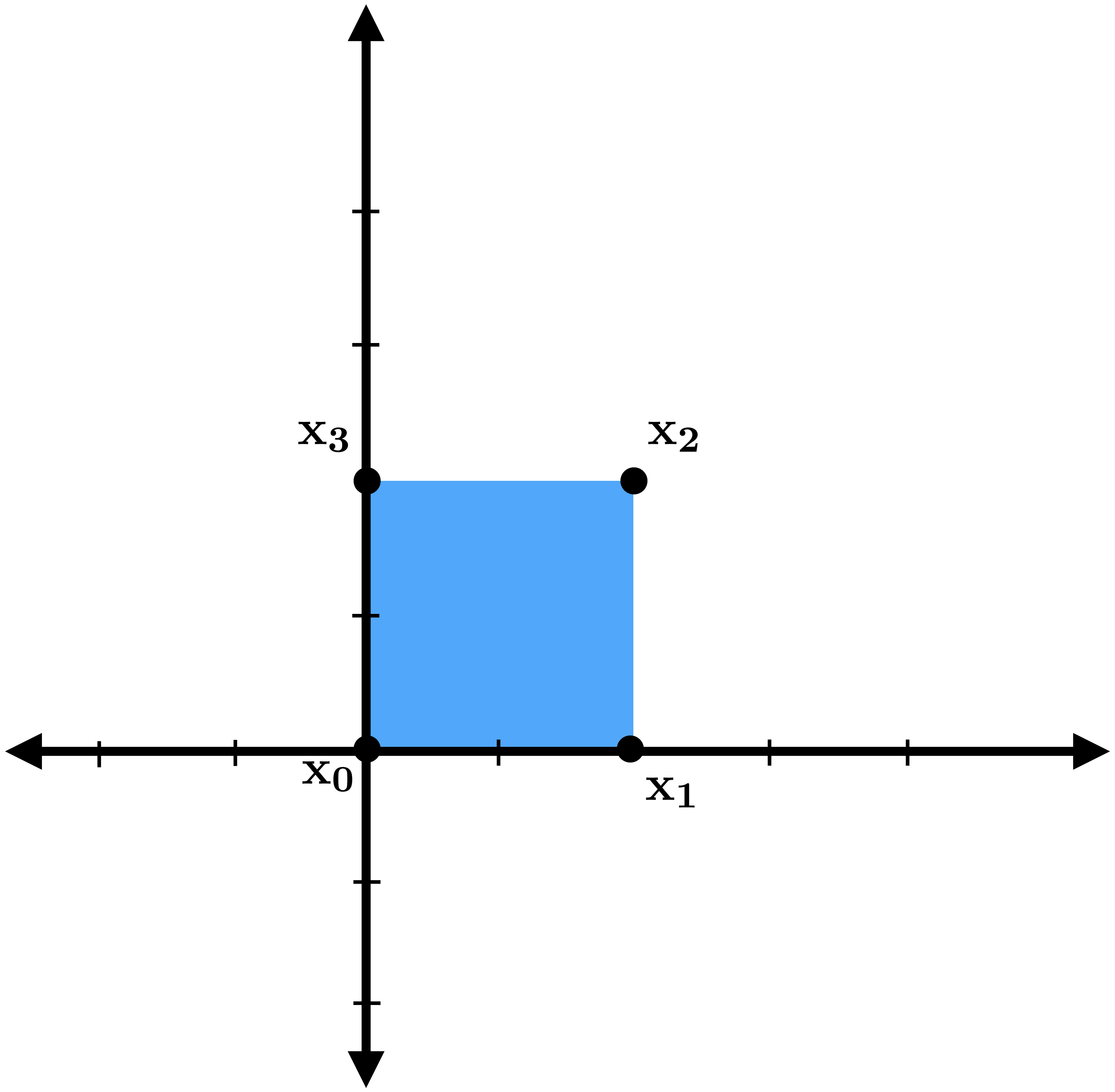
$$S_2(\mathbf{x}) + S_2(\mathbf{y}) = 2\mathbf{x} + 2\mathbf{y}$$

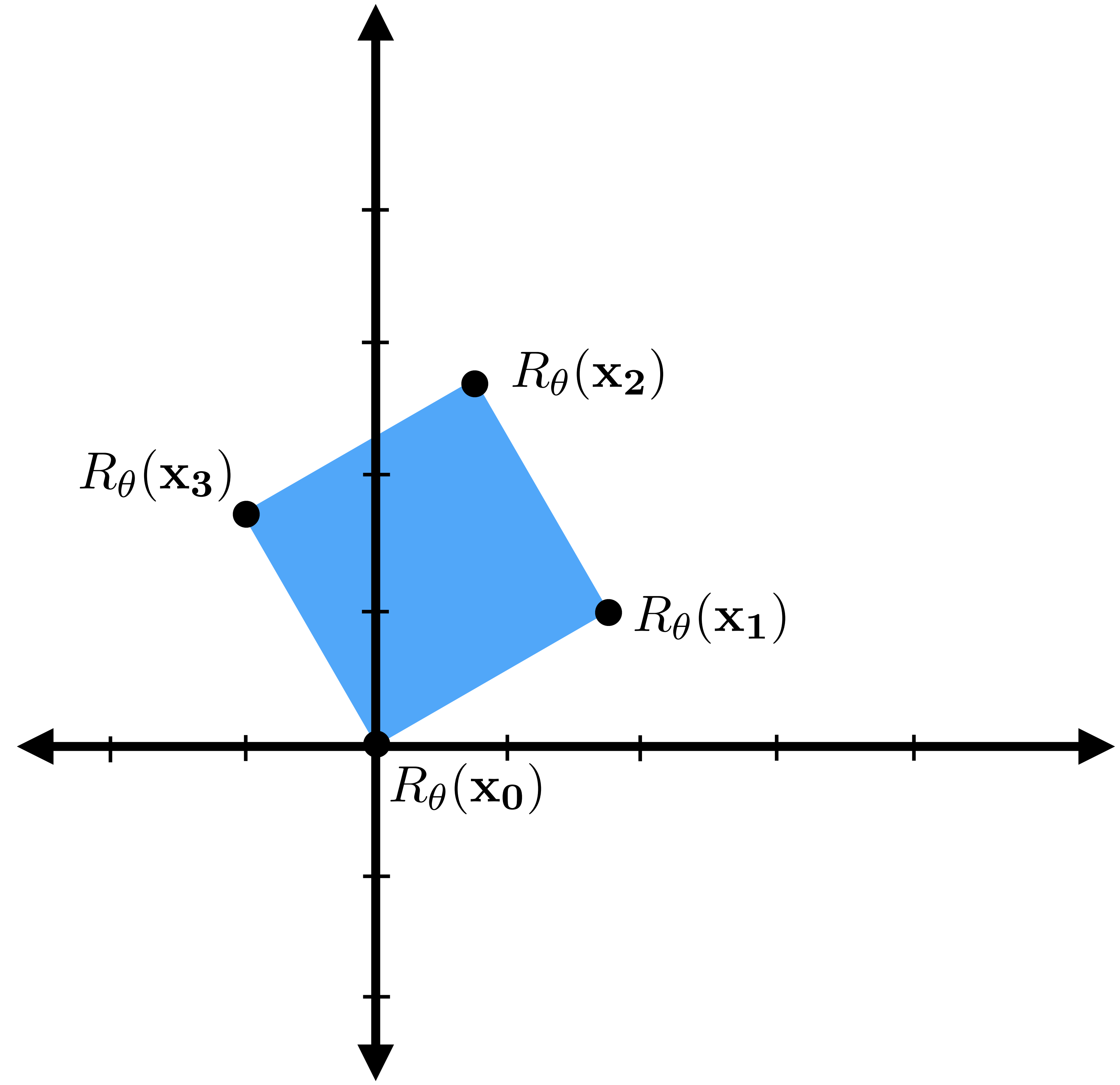
$$S_2(\mathbf{x}+\mathbf{y}) = S_2(\mathbf{x}) + S_2(\mathbf{y})$$



Yes!

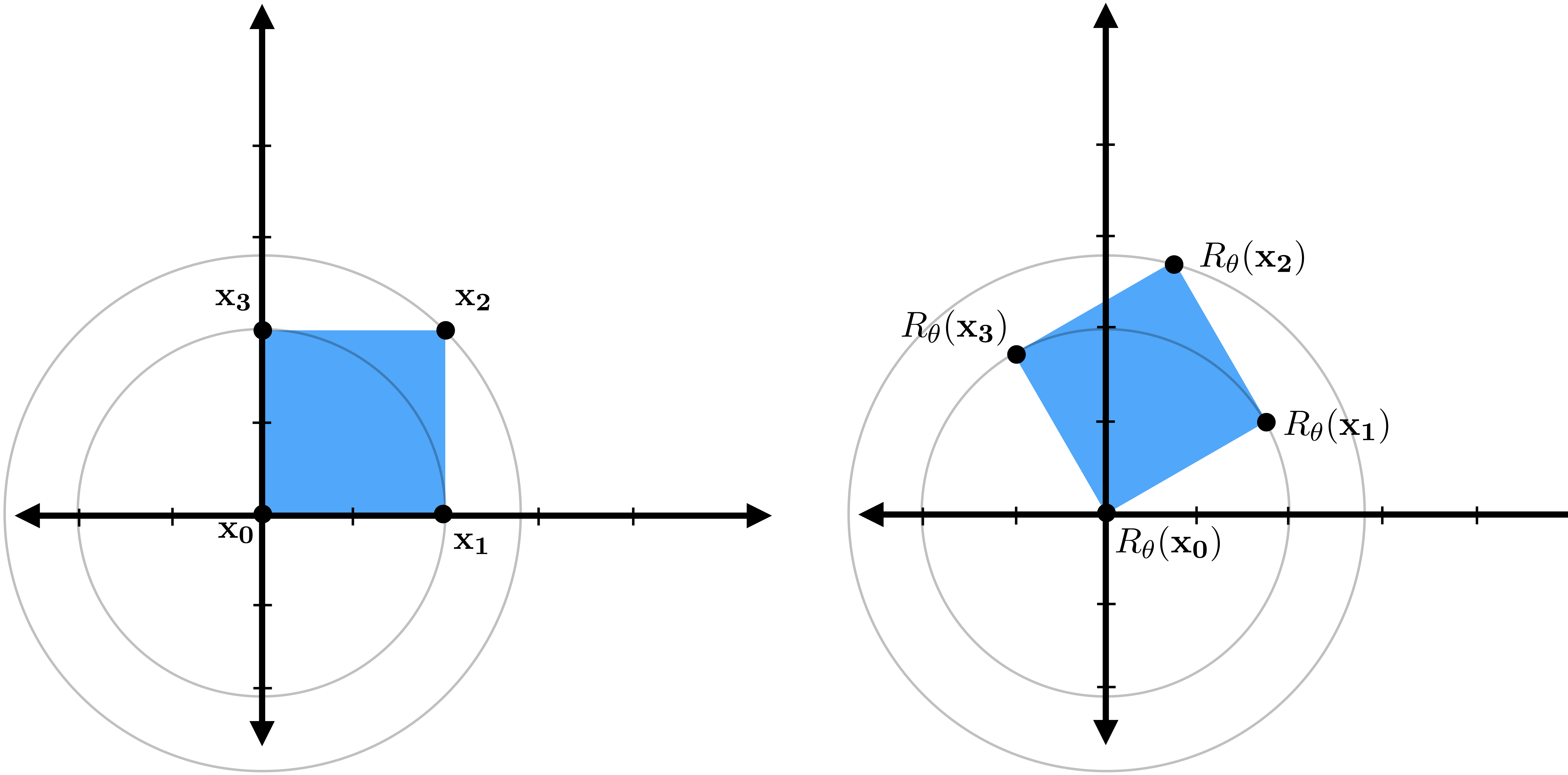
Rotation





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

Rotation as Circular Motion

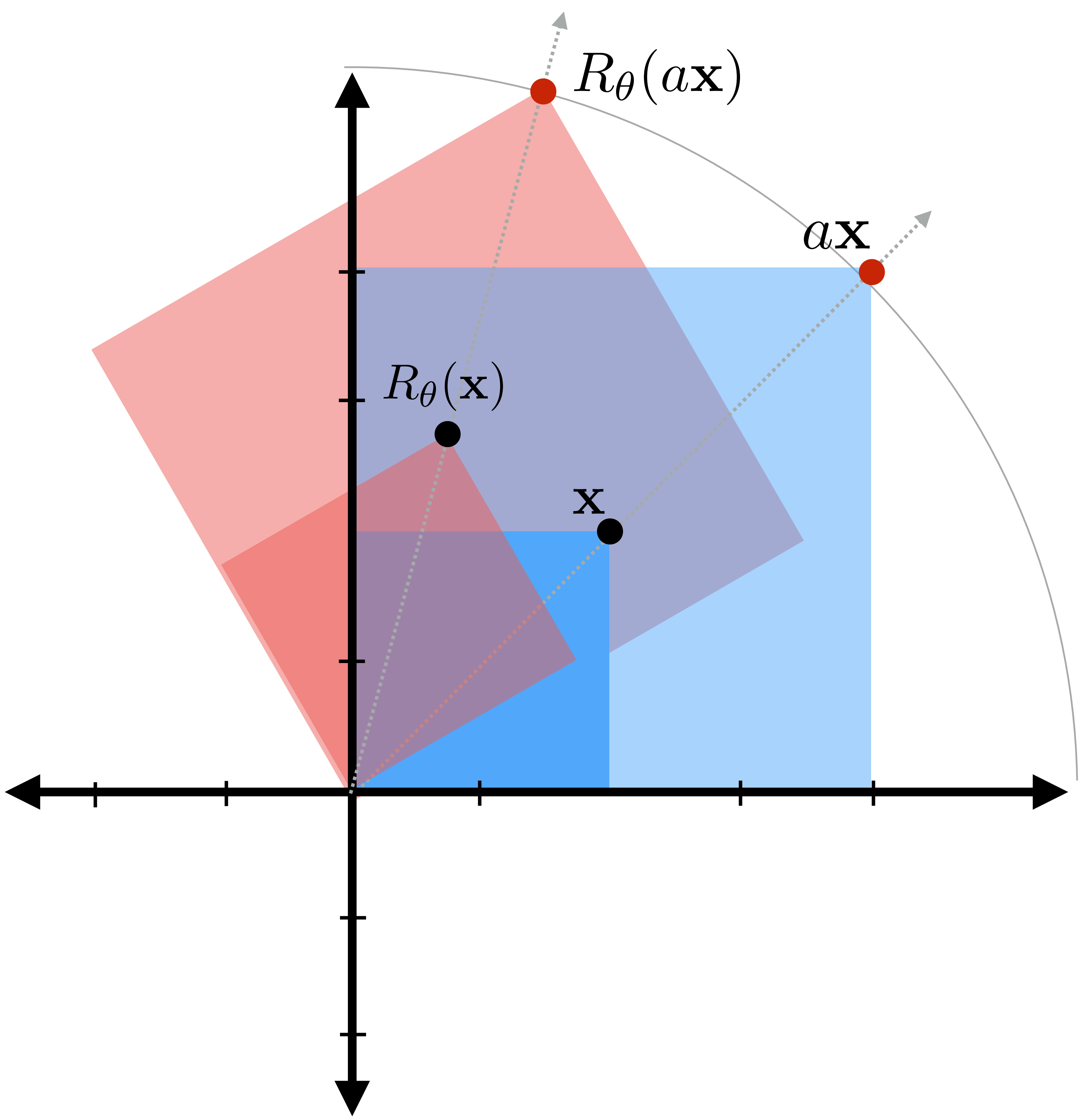


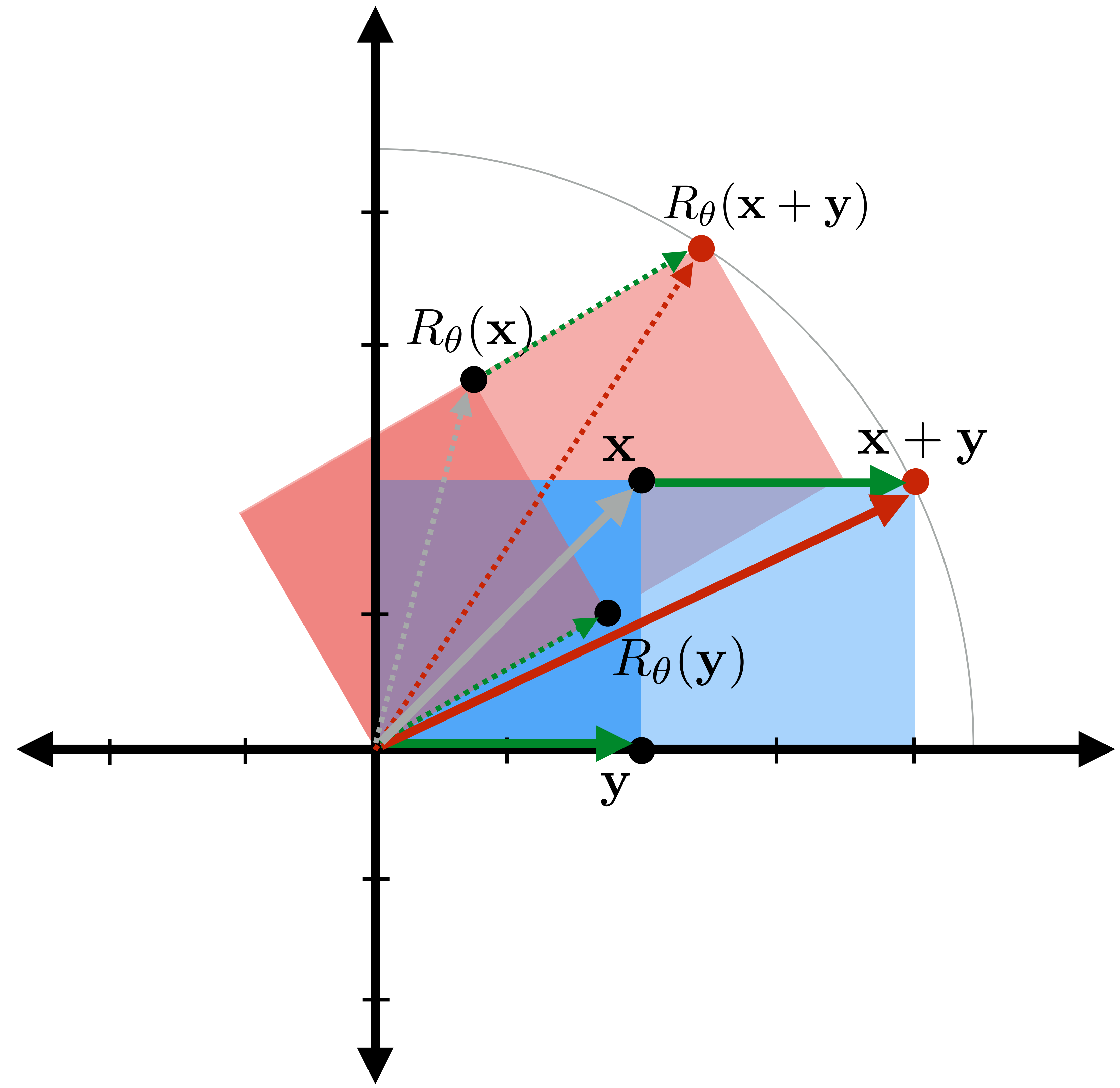
R_θ **= rotate counter-clockwise by** θ

As angle changes, points move along circular trajectories.

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

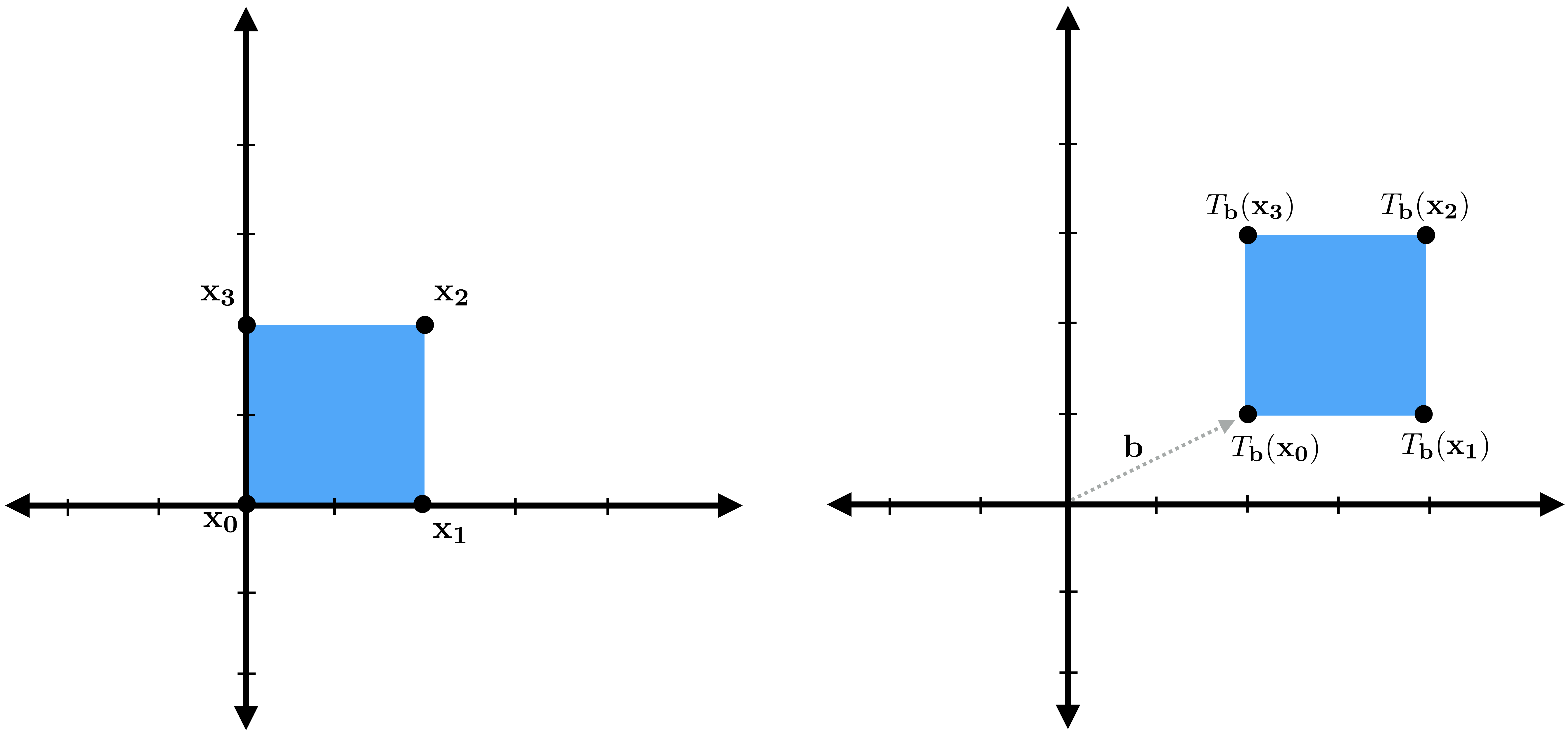
Is rotation linear?





Yes!

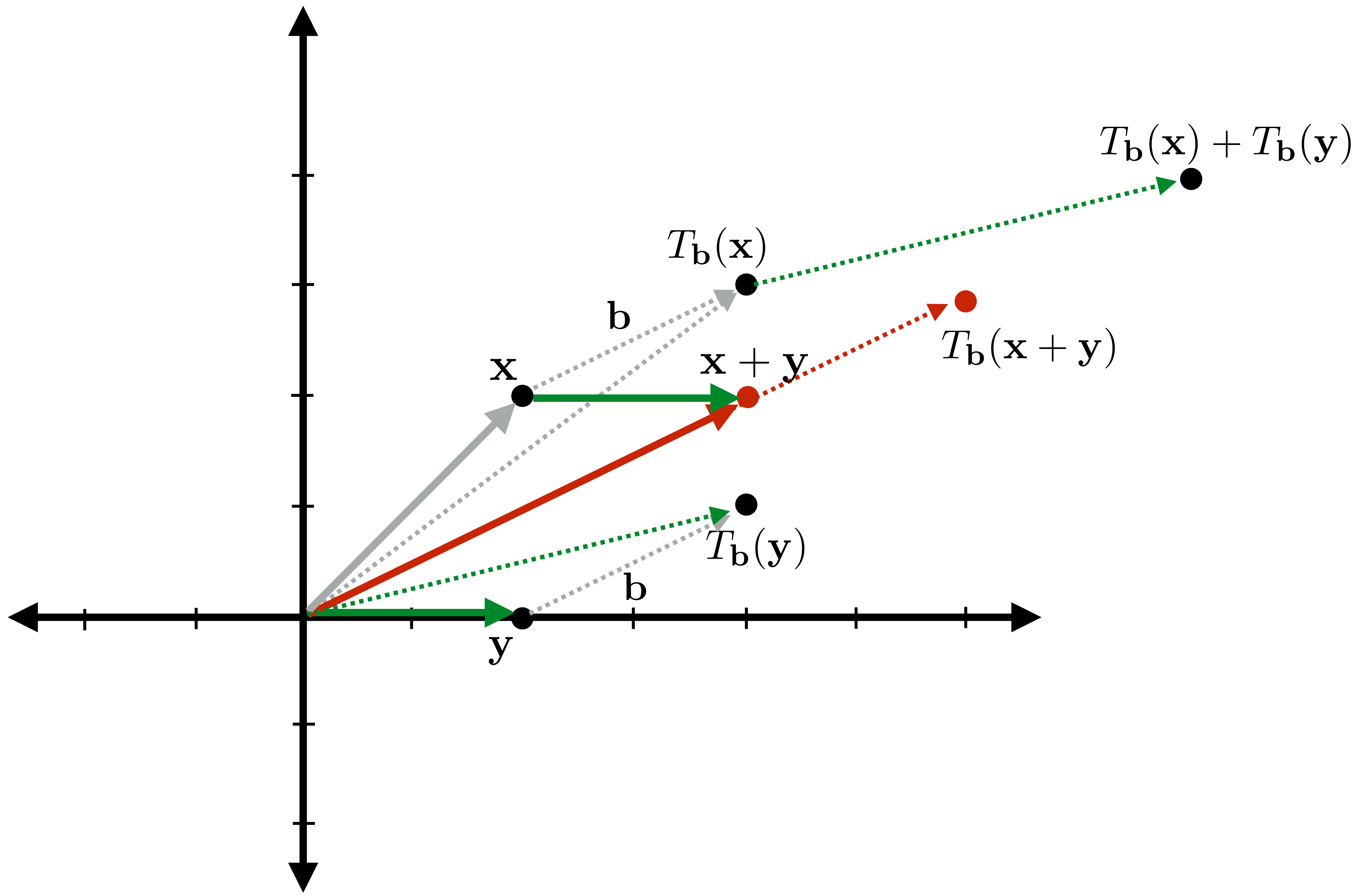
Translation



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

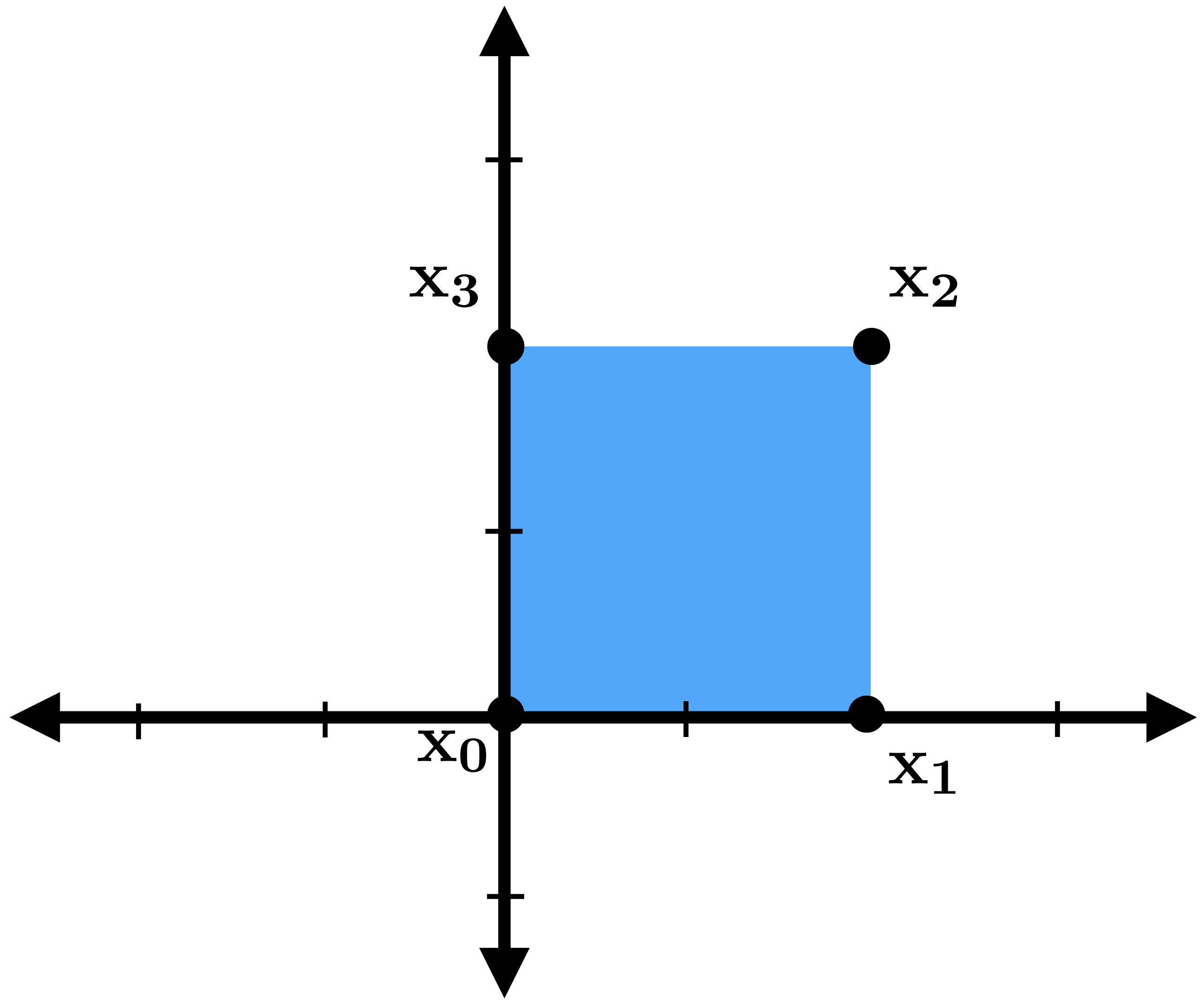
$$T_{\mathbf{b}}(\mathbf{x}) = \mathbf{x} + \mathbf{b}$$

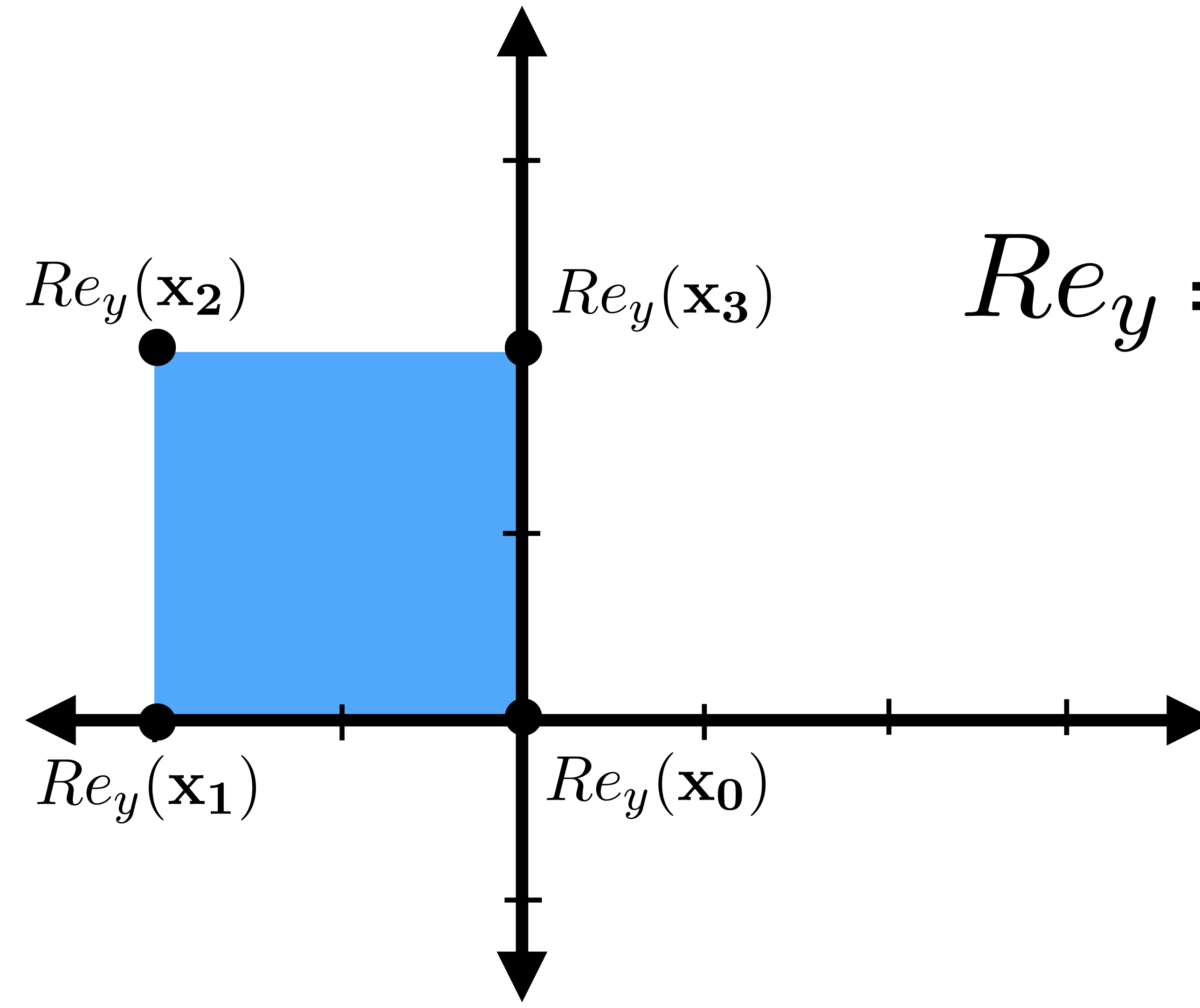
Is translation linear?



No. Translation is affine.

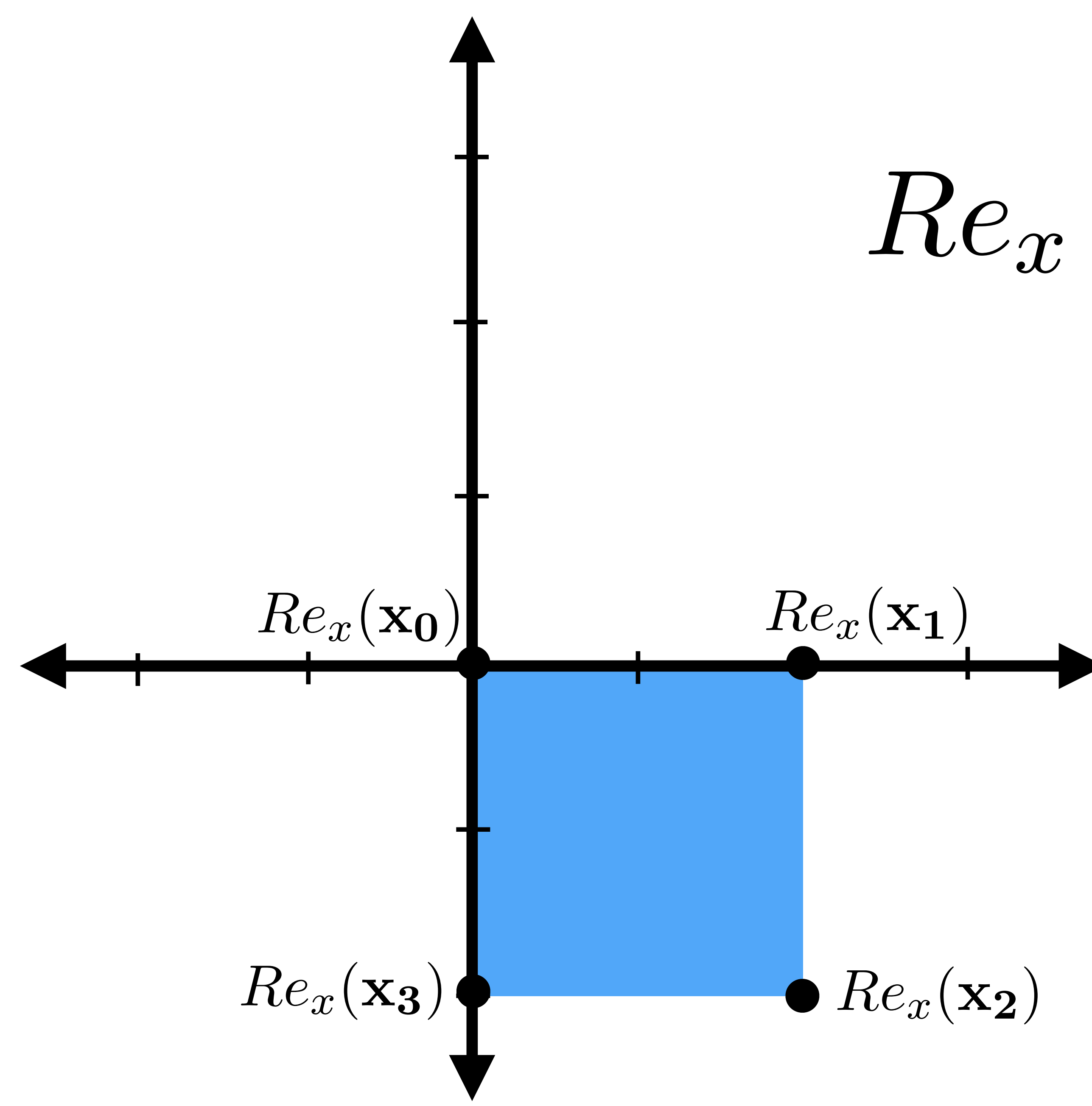
Reflection



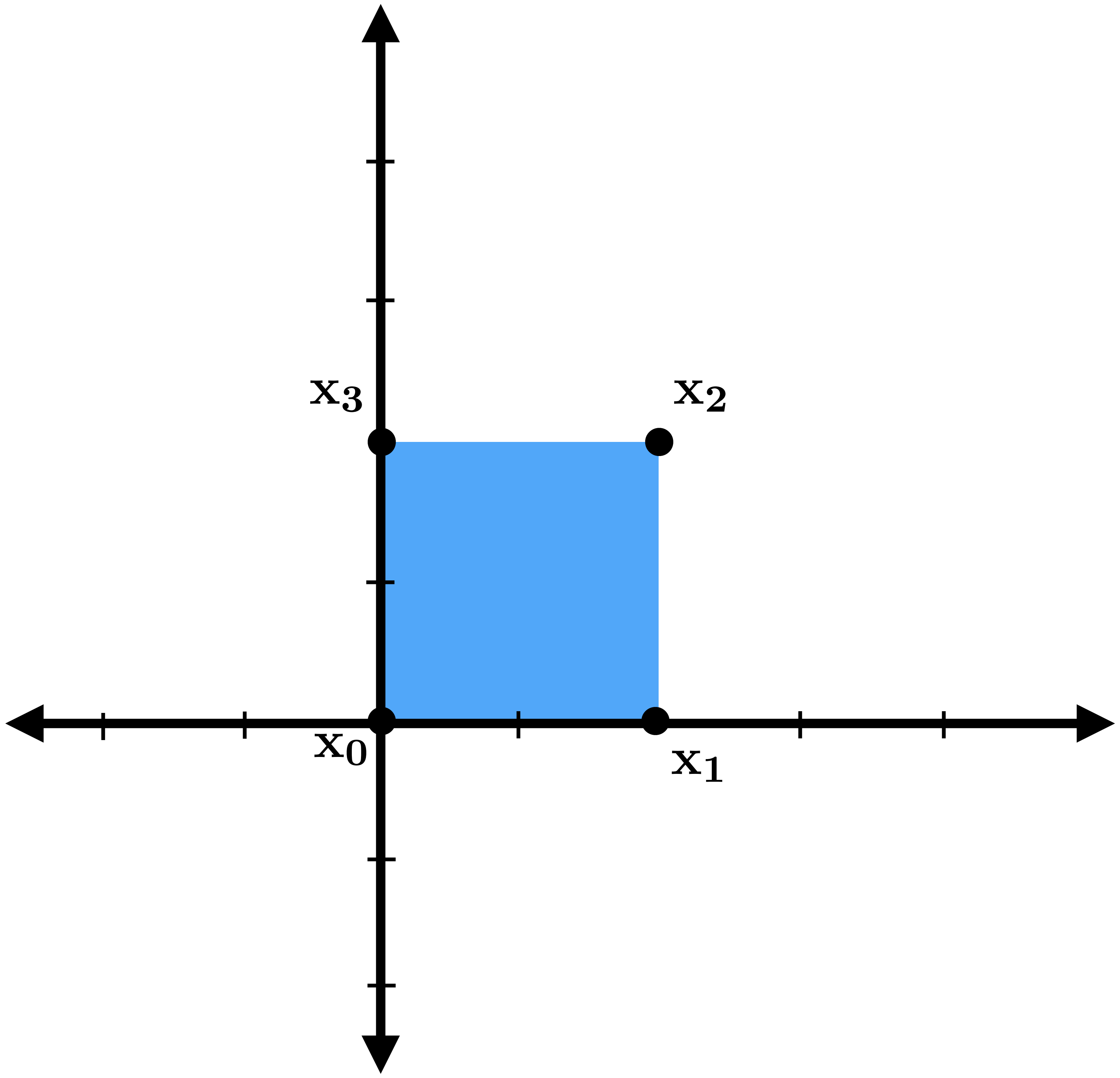


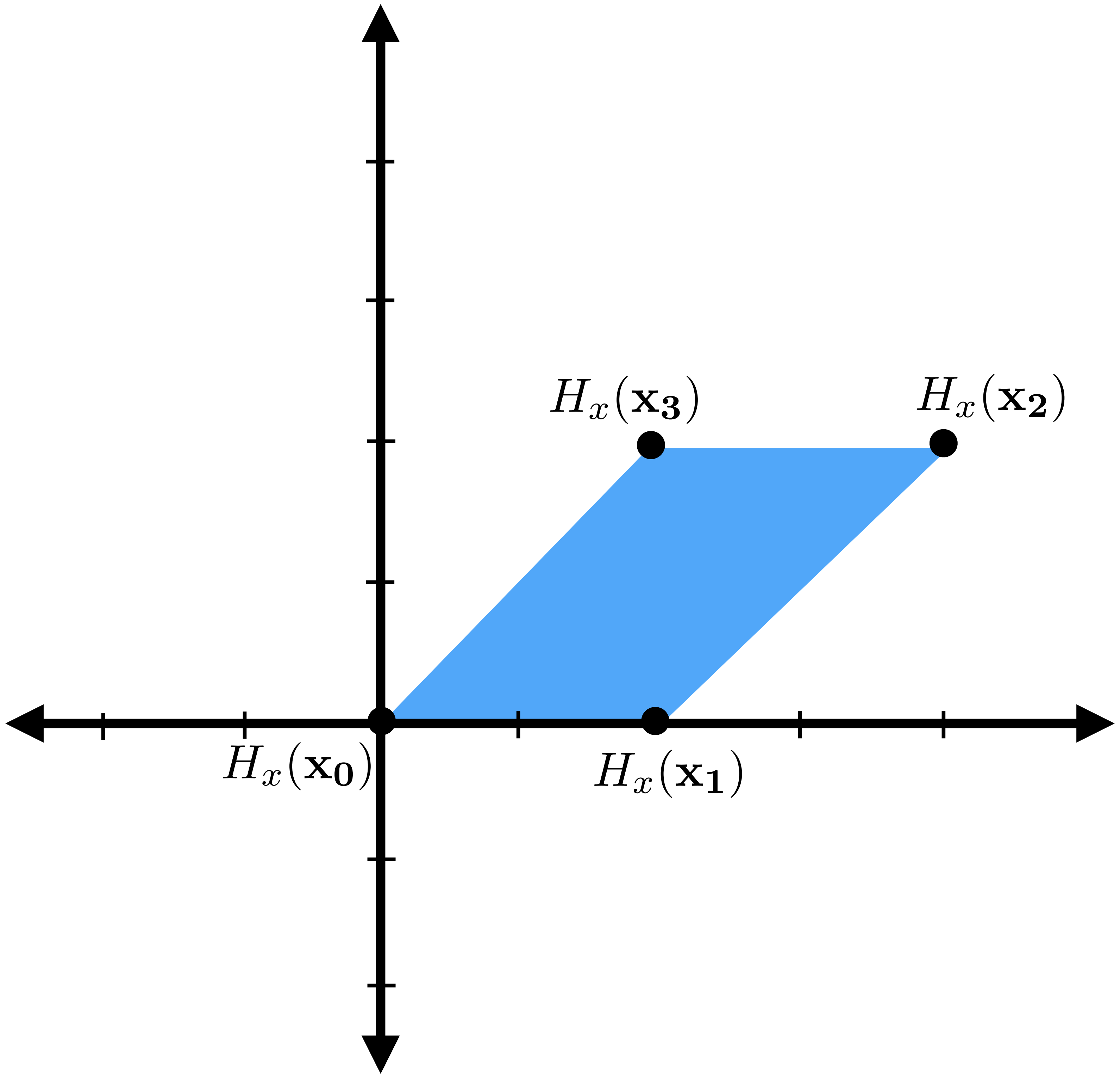
Re_y = **reflection about *y***

Re_x = **reflection about *x***

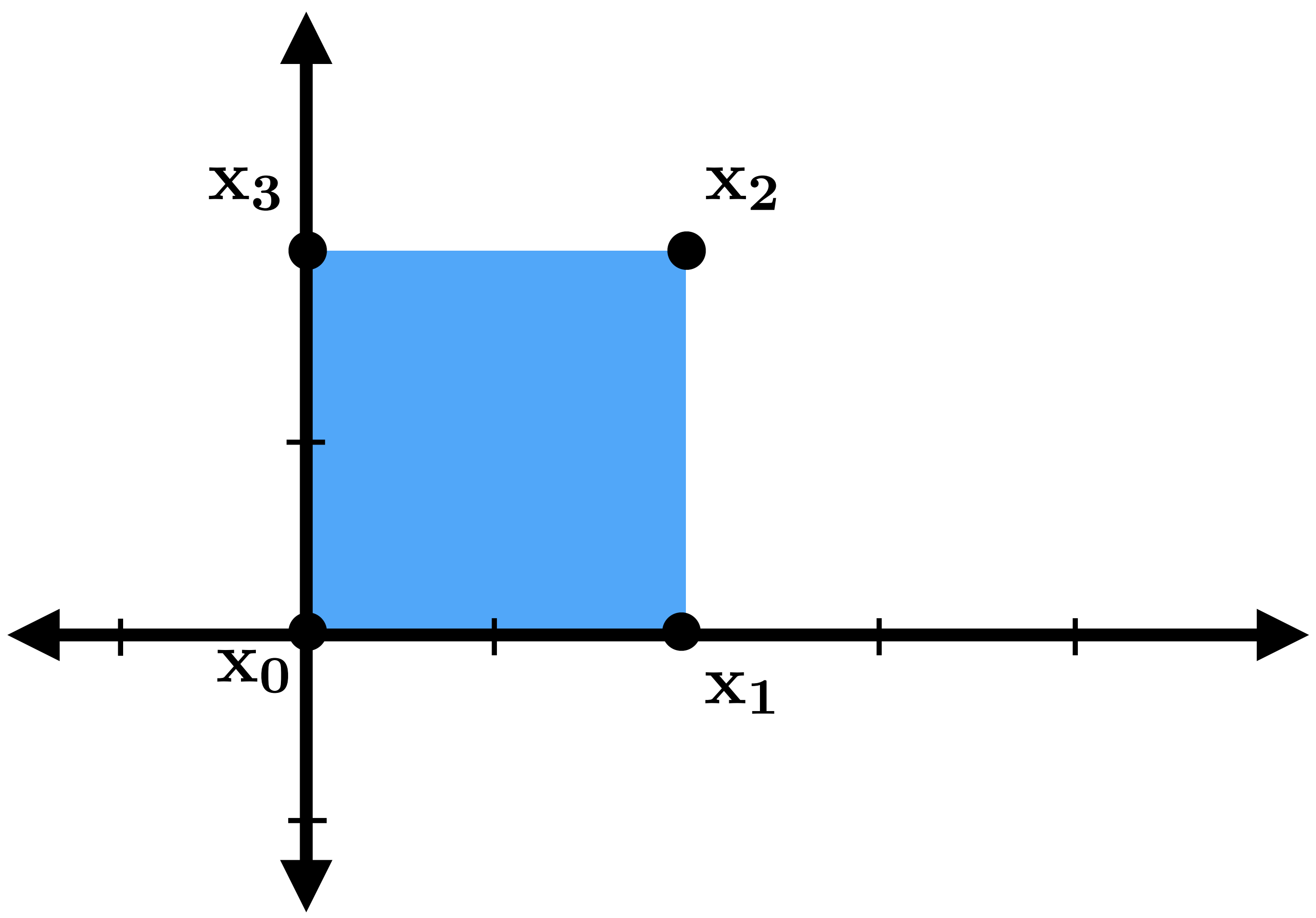


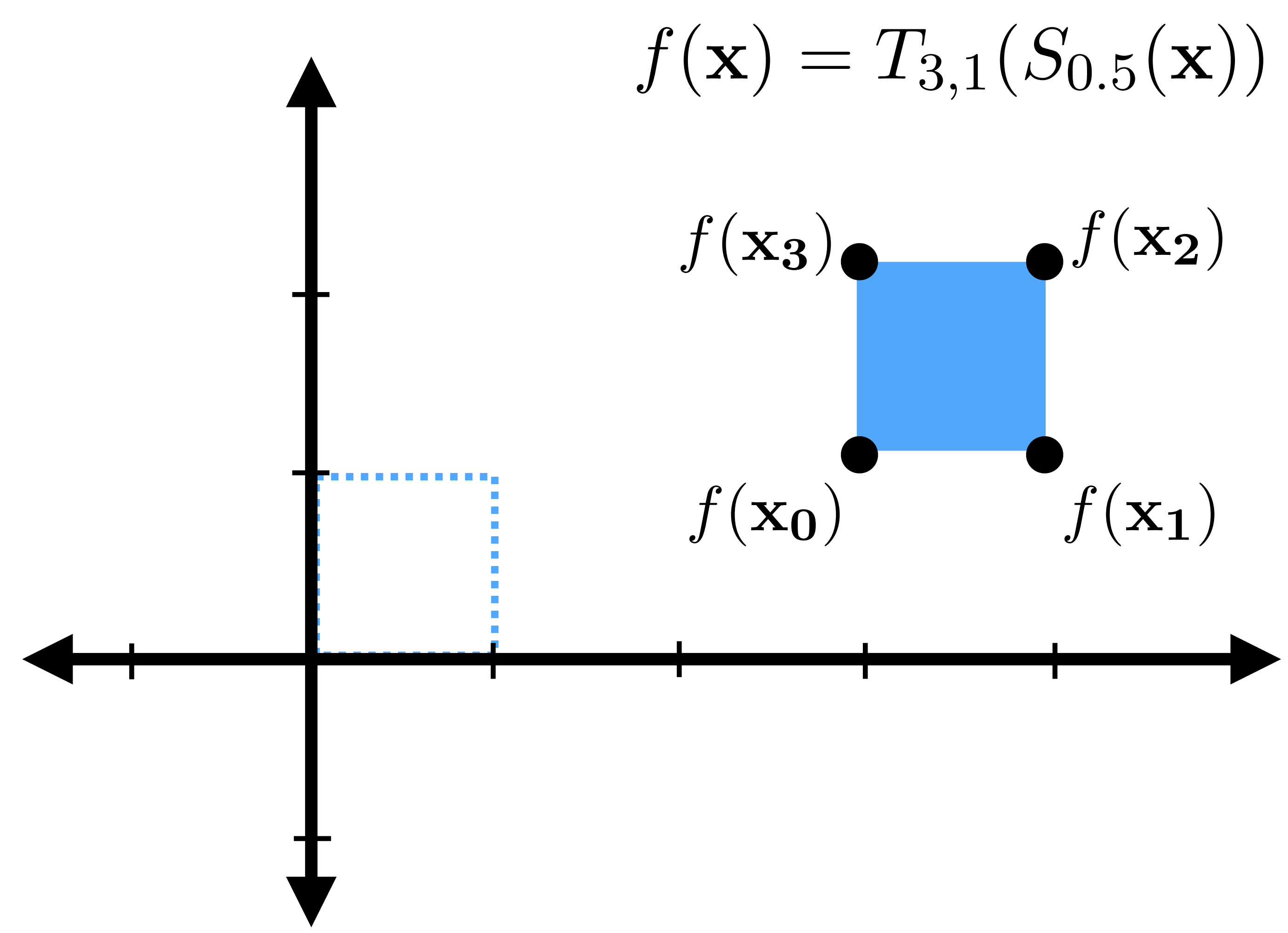
Shear (in x direction)

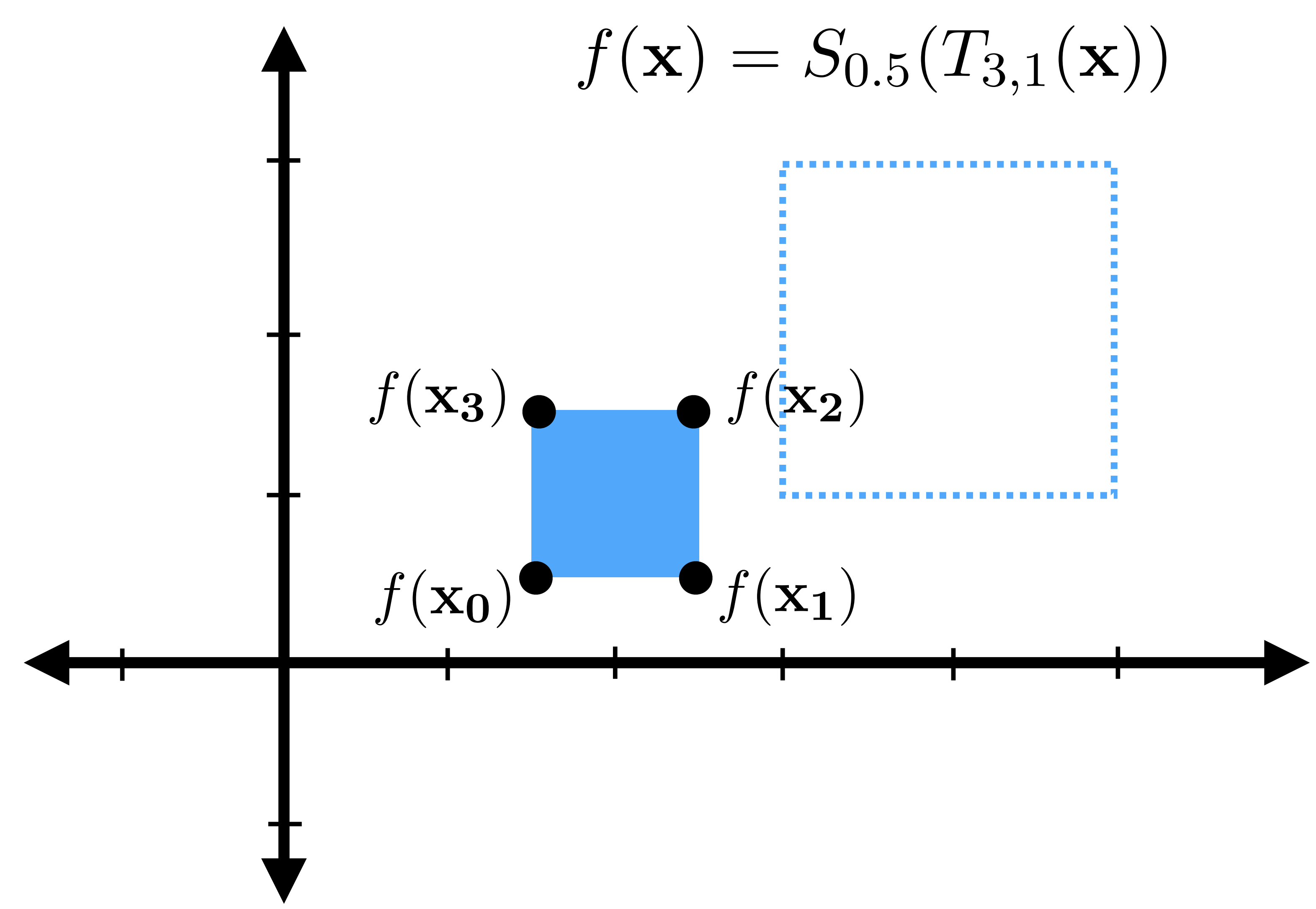




Compose basic transforms to construct more complex transforms





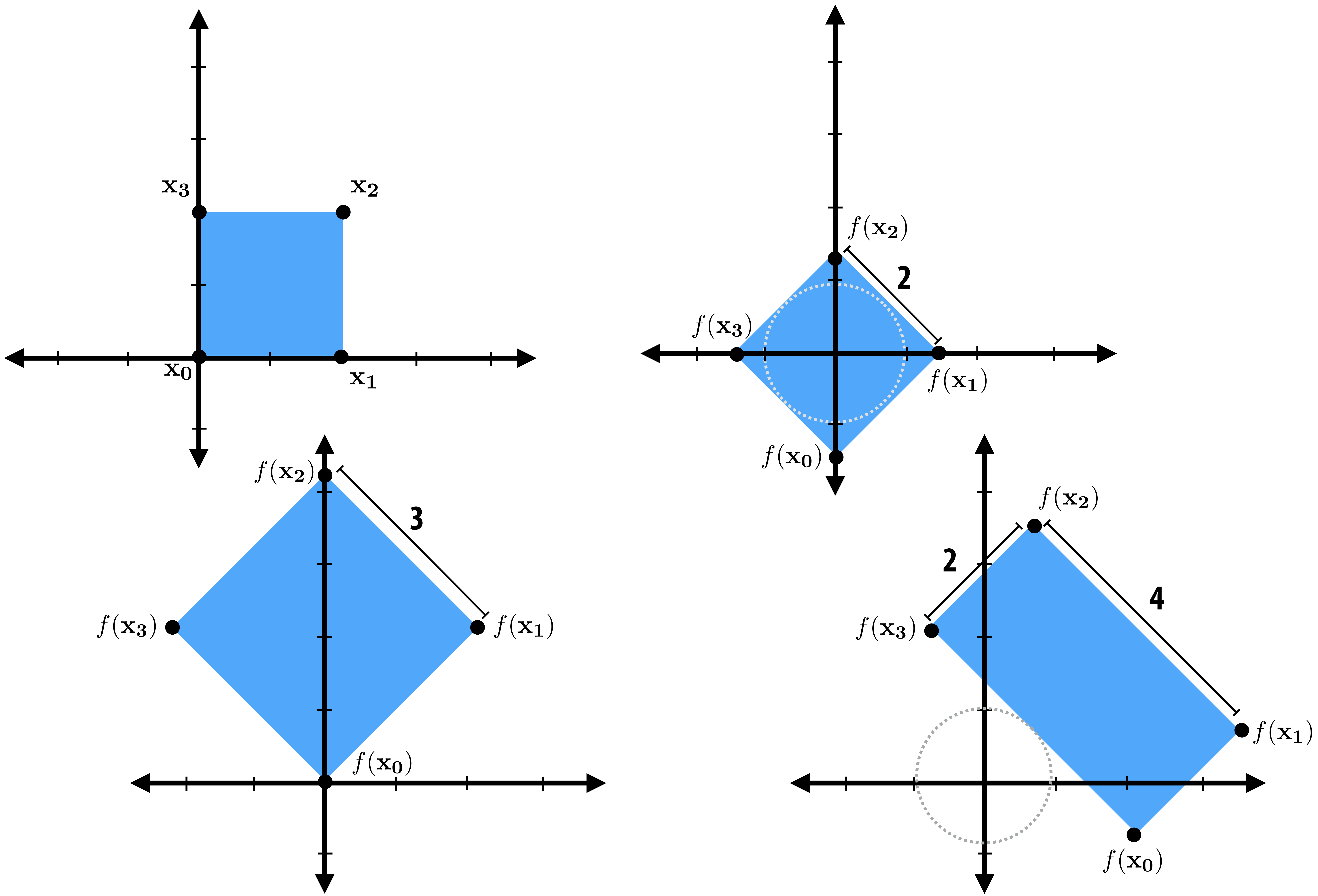


Note: order of composition matters

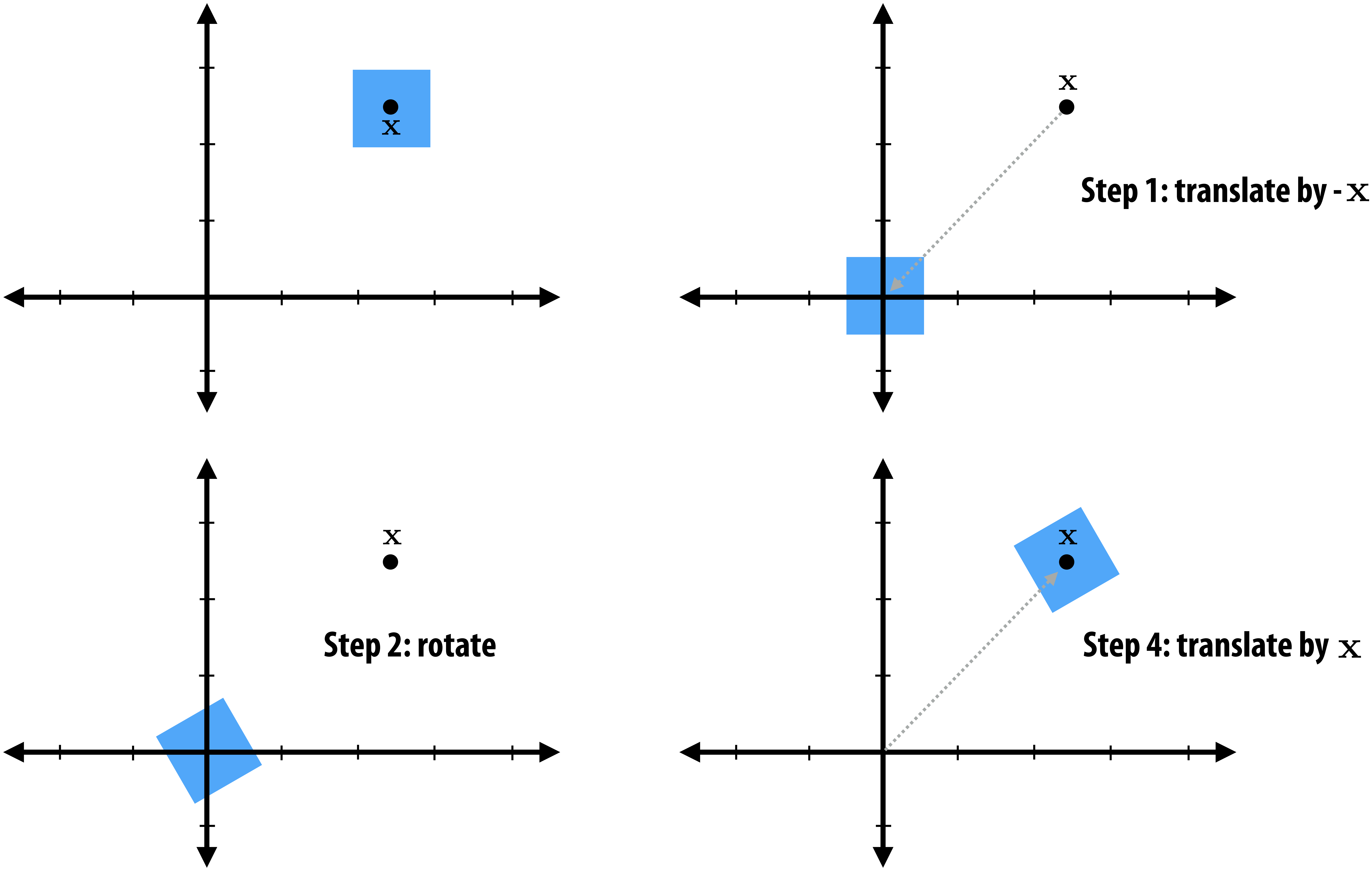
Top-right: scale, then translate

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)

What you should know

1. Express points and vectors using homogeneous coordinates.
2. Perform a dot product and demonstrate how to use the dot product for projection (e.g., projection of a point onto a coordinate axis).
3. Perform a cross product and demonstrate how to use it to compute a surface normal.
4. Perform matrix multiplication.
5. Derive a parametric expression for a line between two points.
6. Prove that your parametric expression is correct. Discuss whether it is unique.
7. Derive an implicit expression for an edge between two points.
8. Prove that your implicit expression is correct. Discuss whether it is unique.
9. Use an implicit edge expression to determine whether a (2D) point is inside a (projected) triangle.
10. Create an algorithm to rasterize a triangle from a set of "inside-triangle" tests.
11. Make that algorithm efficient so that not every pixel needs to be tested.