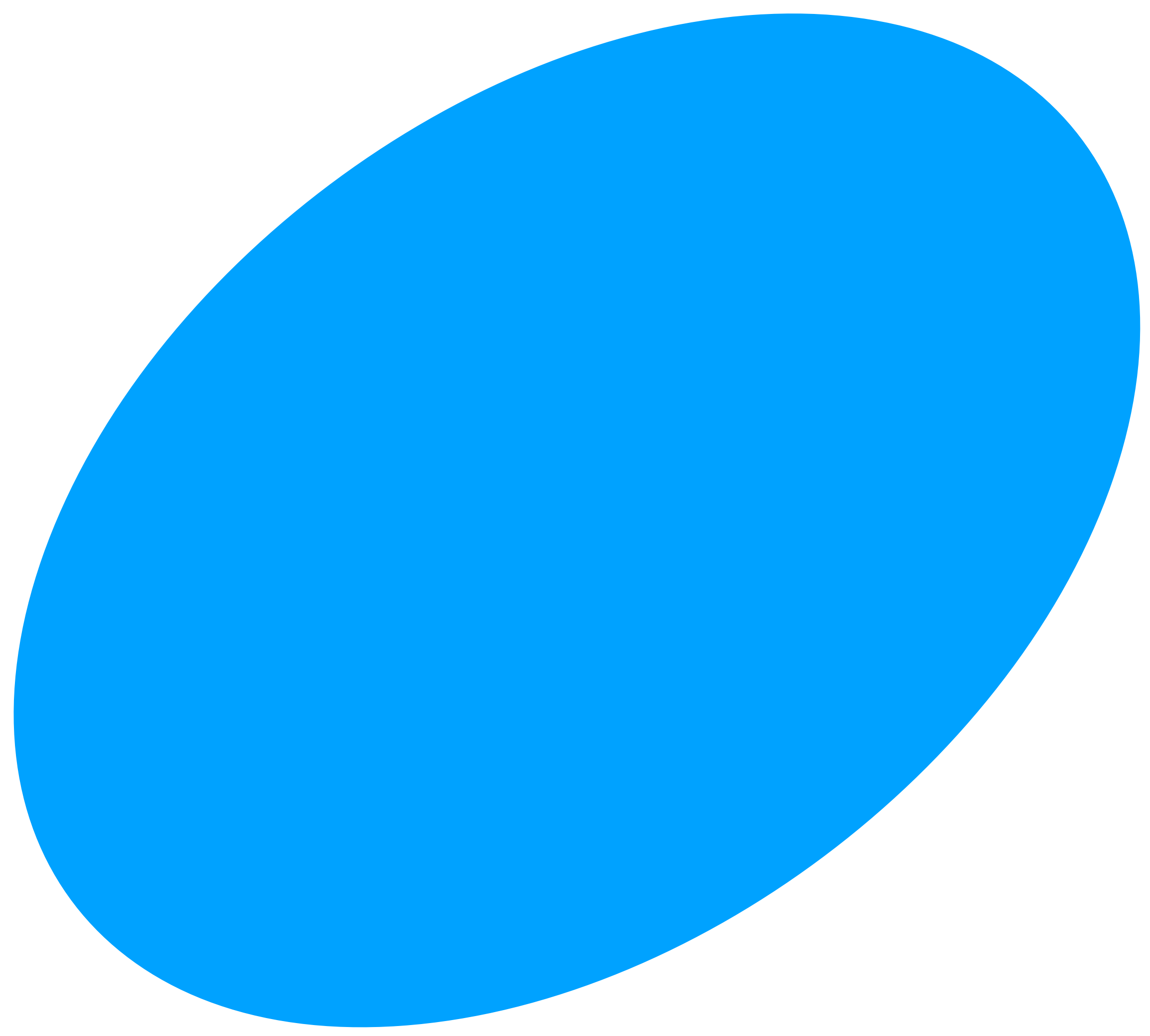
(2D) Linear transformations

UCSD CSE 167

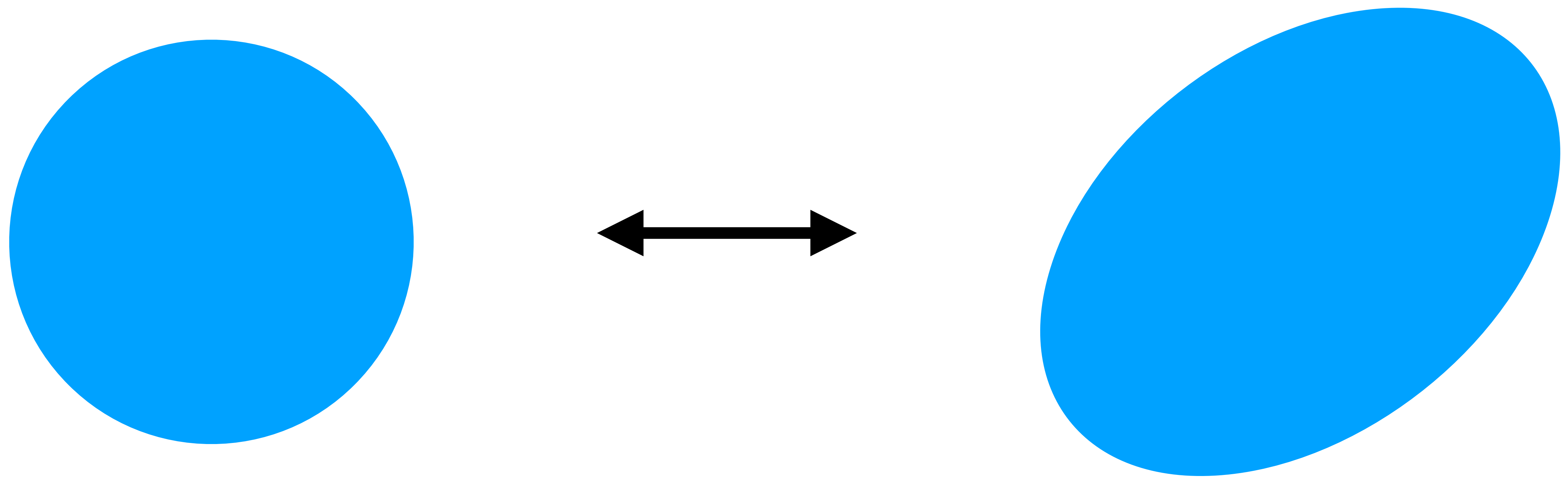
Tzu-Mao Li

Motivation: rendering an ellipse

testing whether a point is inside an ellipse is slightly more complex



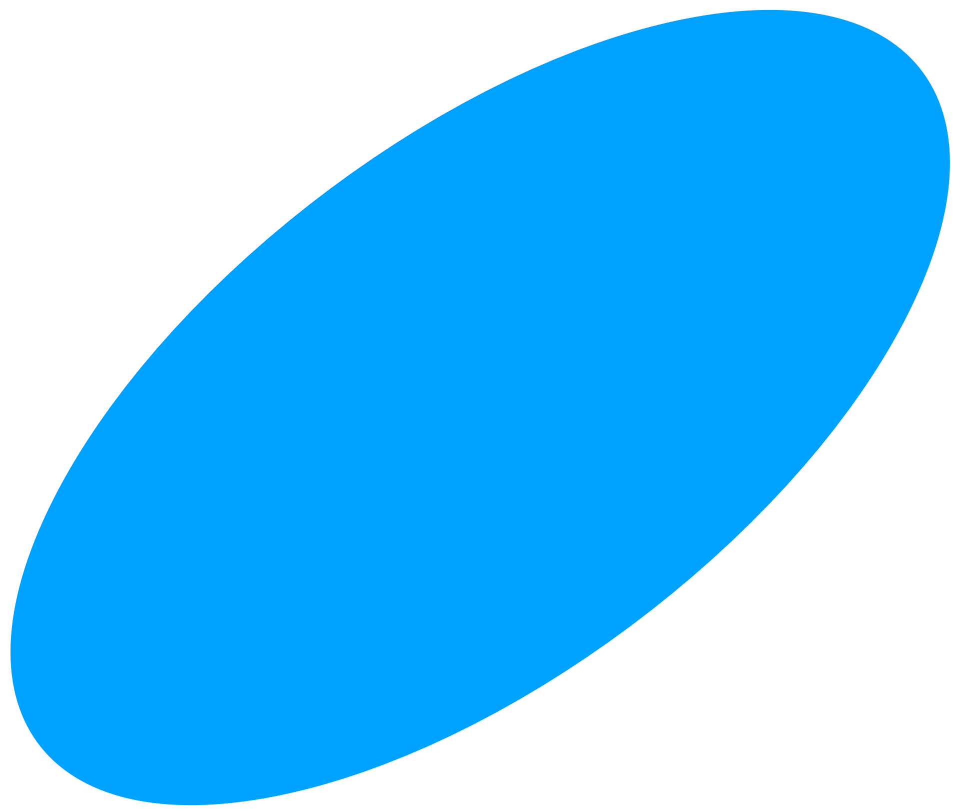
Idea: map the ellipse to a simple circle



Motivation 2: instancing

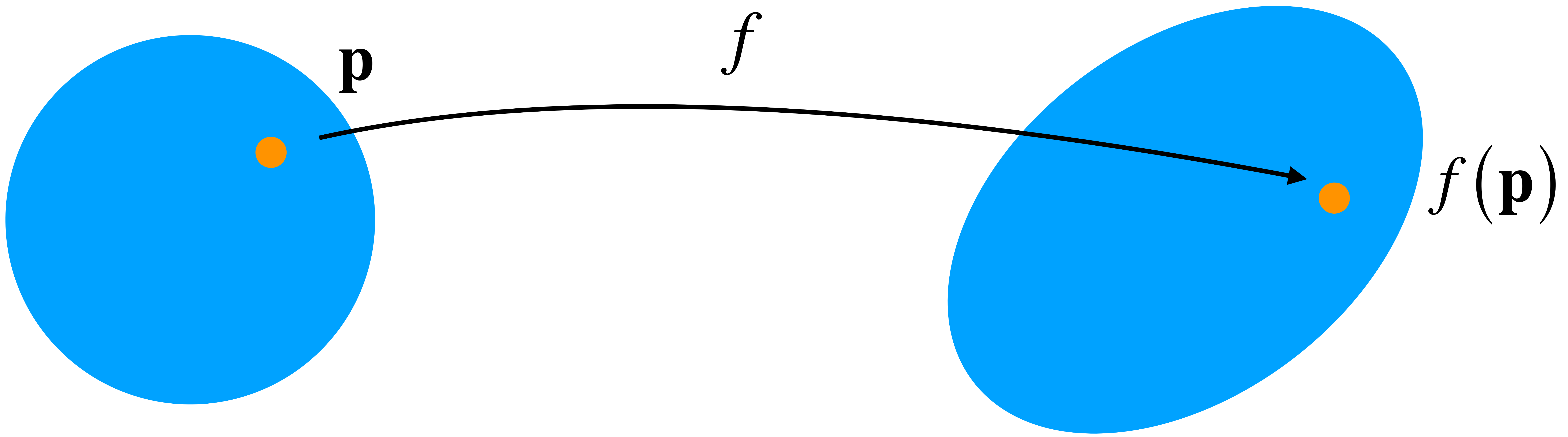
only want to store one star

Motivation 3: animation



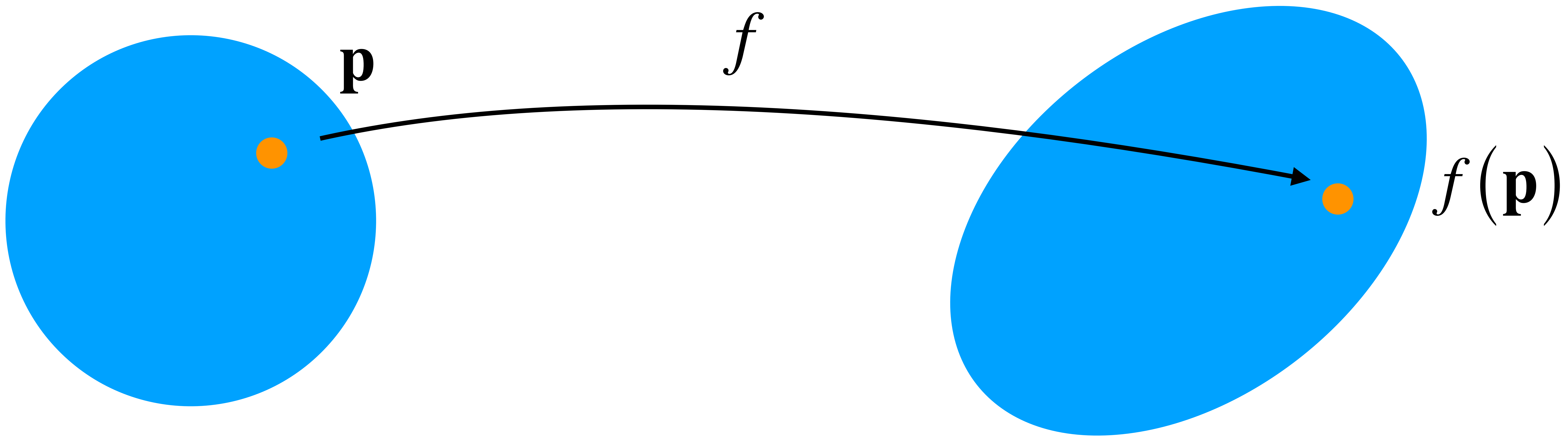
Transformation

each point $\mathbf{p}$ is mapped to a point $f(\mathbf{p})$



Linear Transformation

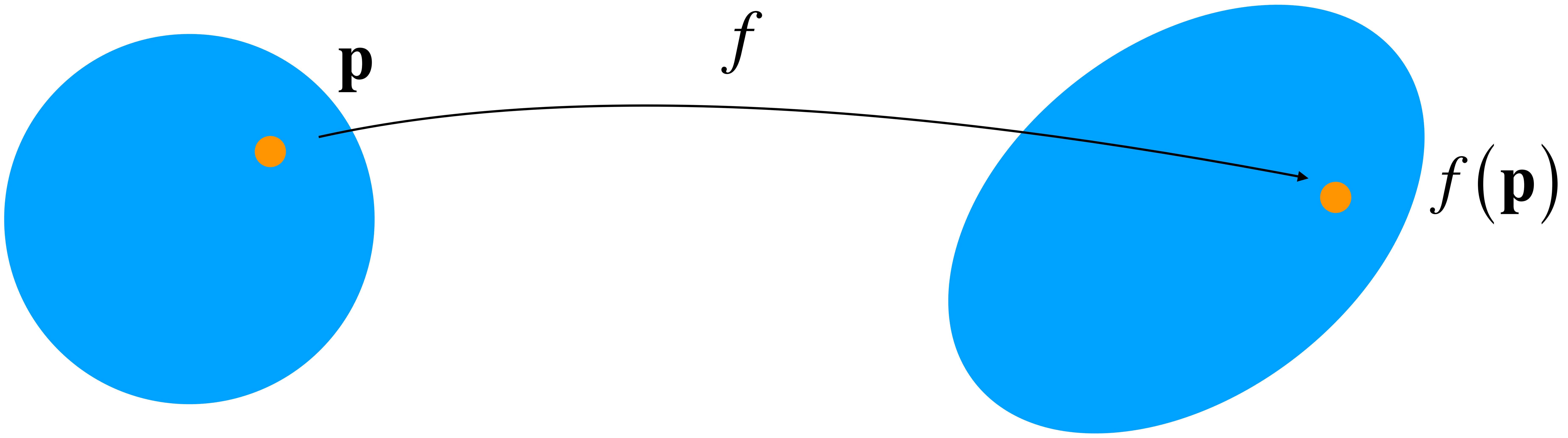
each point $\mathbf{p}$ is mapped to a point $f(\mathbf{p})$
by a **linear function** f



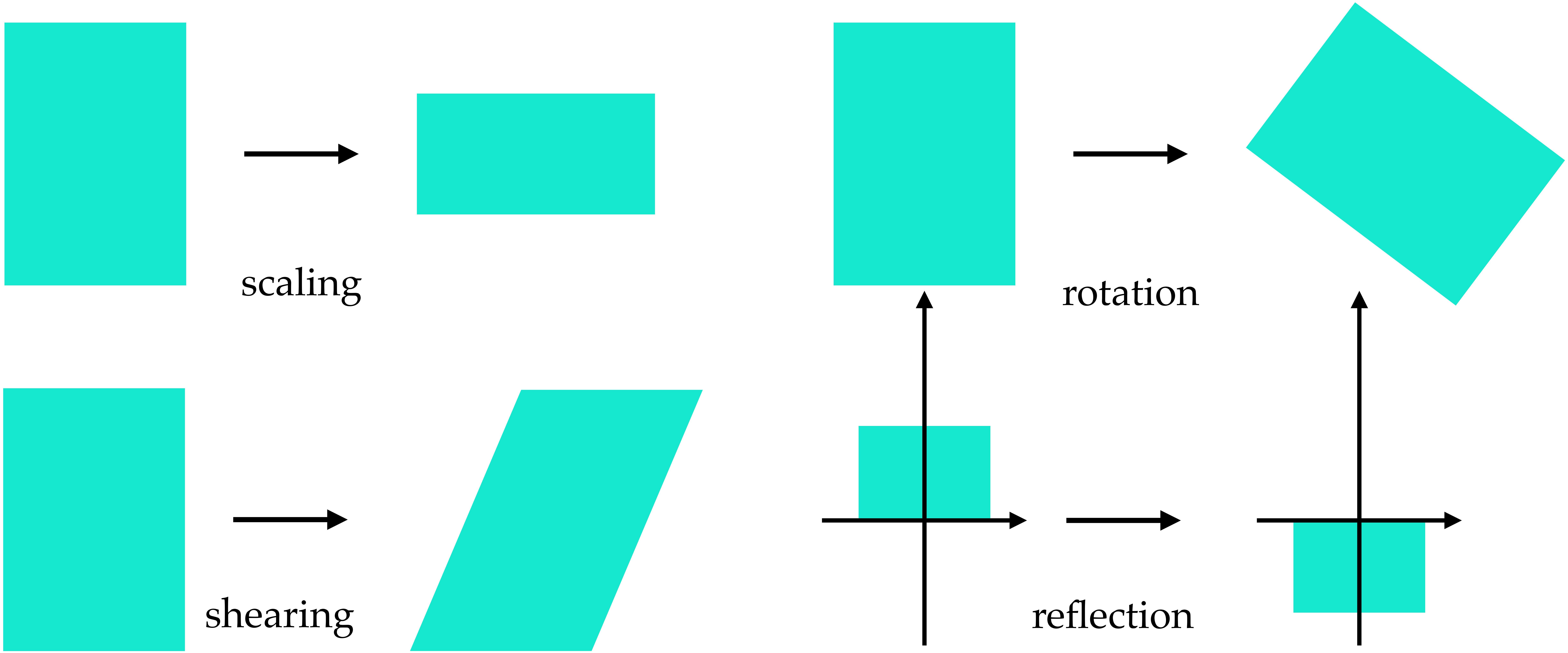
(we will define linearity later)

Why linear transformation?

- linear transformations are easier to analyze and well understood
  - e.g., there are well-established ways to compute their inverses
- linear transformations are expressive
- non-linear transformations can often be broken down to many small linear transformations



Things linear transformation can do

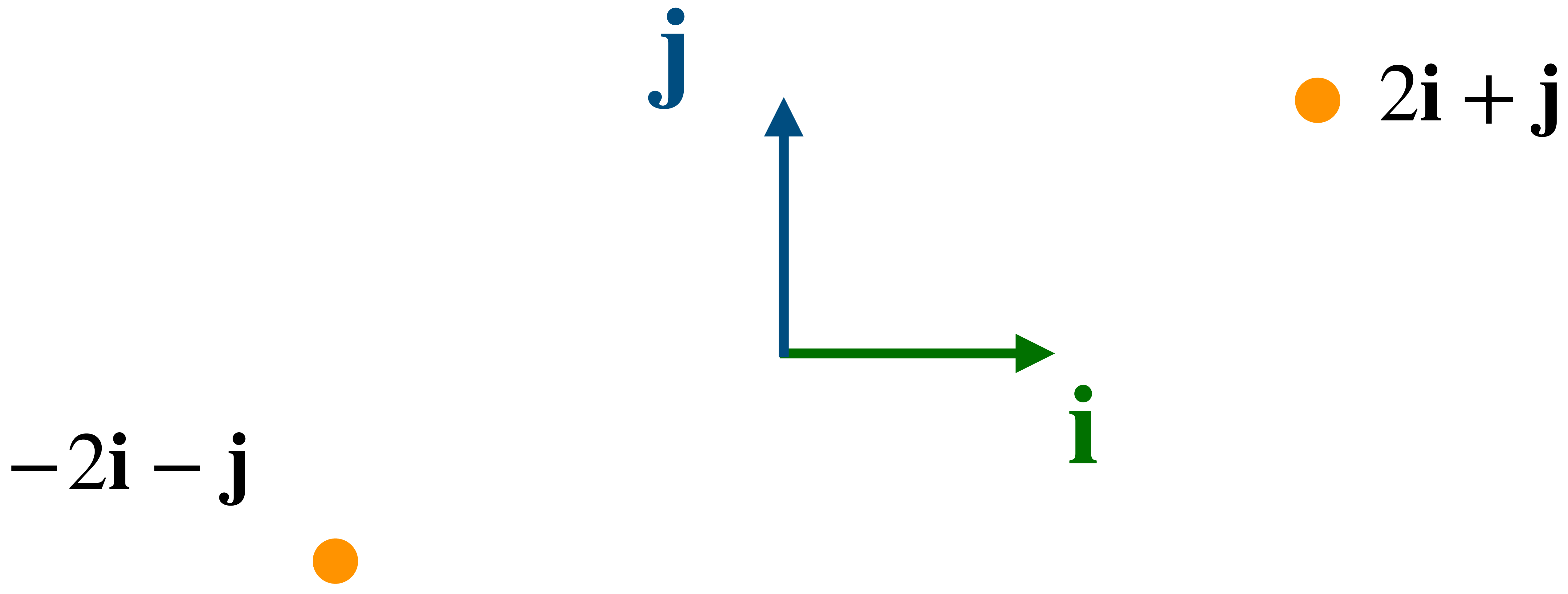


Linear algebra

- excellent resources:
  - Essence of Linear Algebra https://www.youtube.com/playlist?list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab
  - Immersive Linear Algebra http://immersivemath.com/ila/index.html

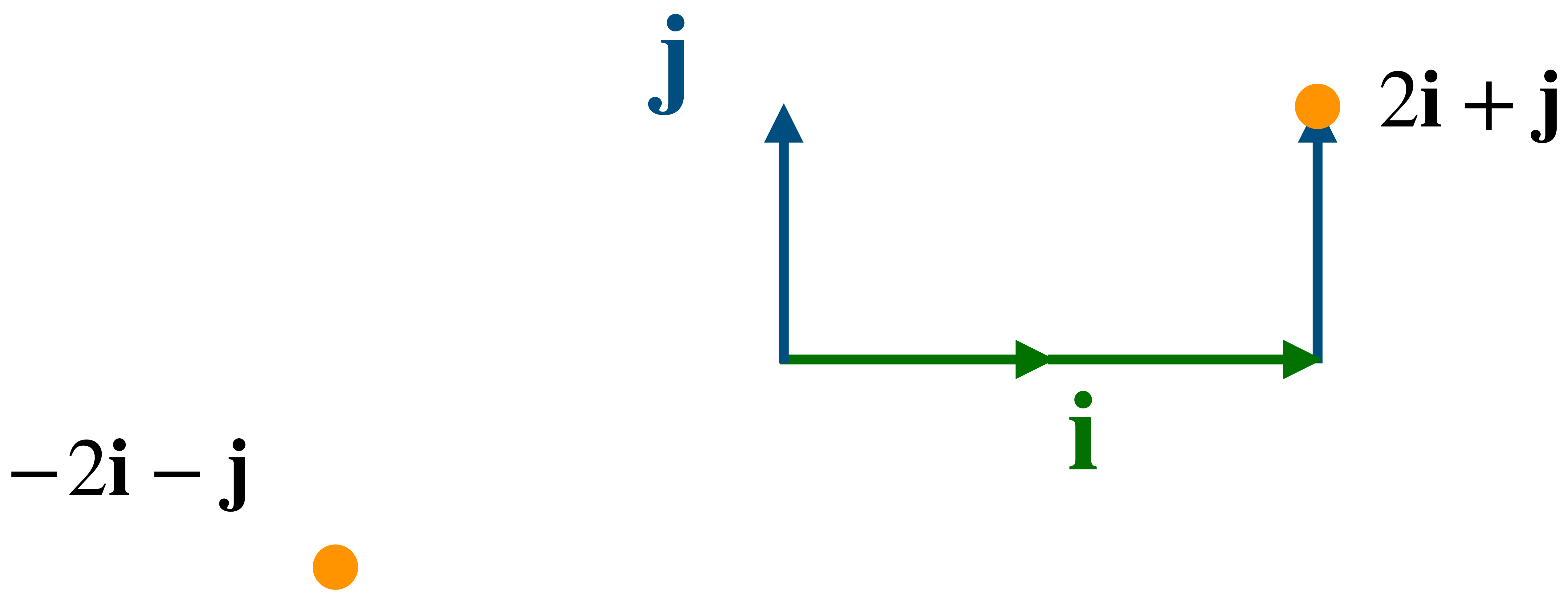
Linear basis

each point in the 2D space can be written as a linear combination of two basis vectors **i** and **j**



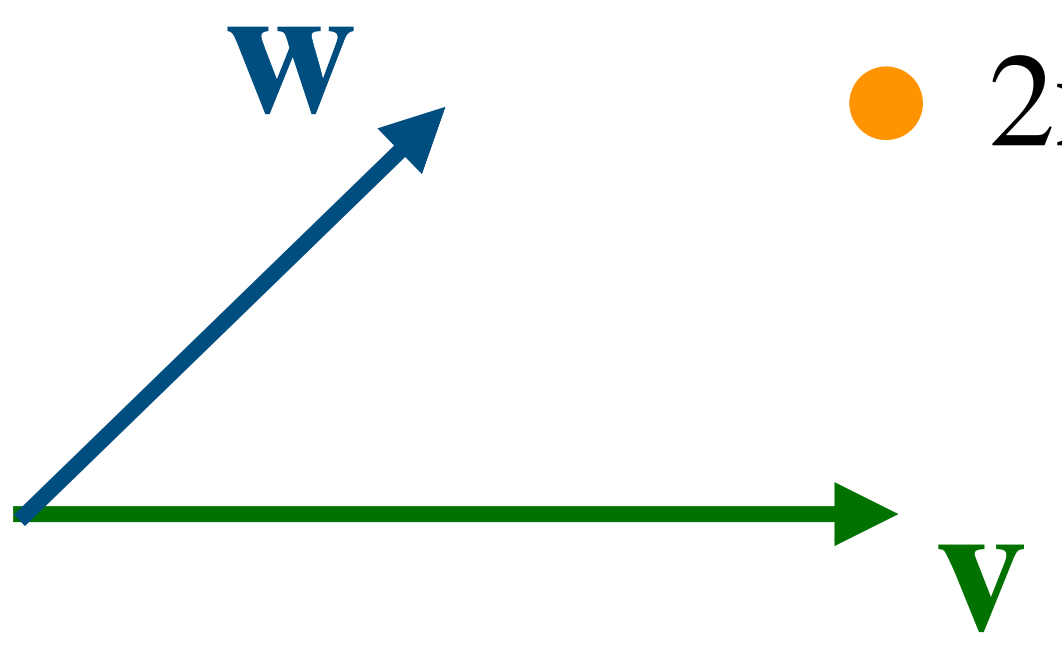
Linear basis

each point in the 2D space can be written as a linear combination of two basis vectors **i** and **j**



Linear basis

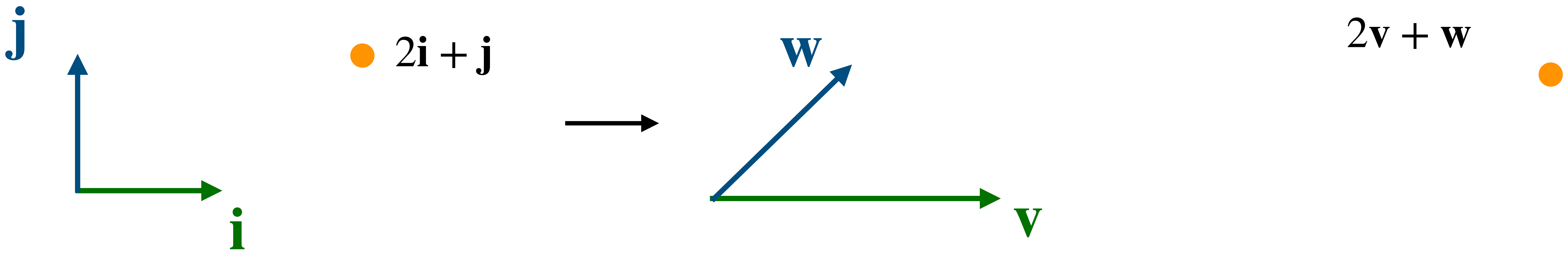
we can use a different linear basis to represent the same 2D space
(and they don't need to be perpendicular to each other!)



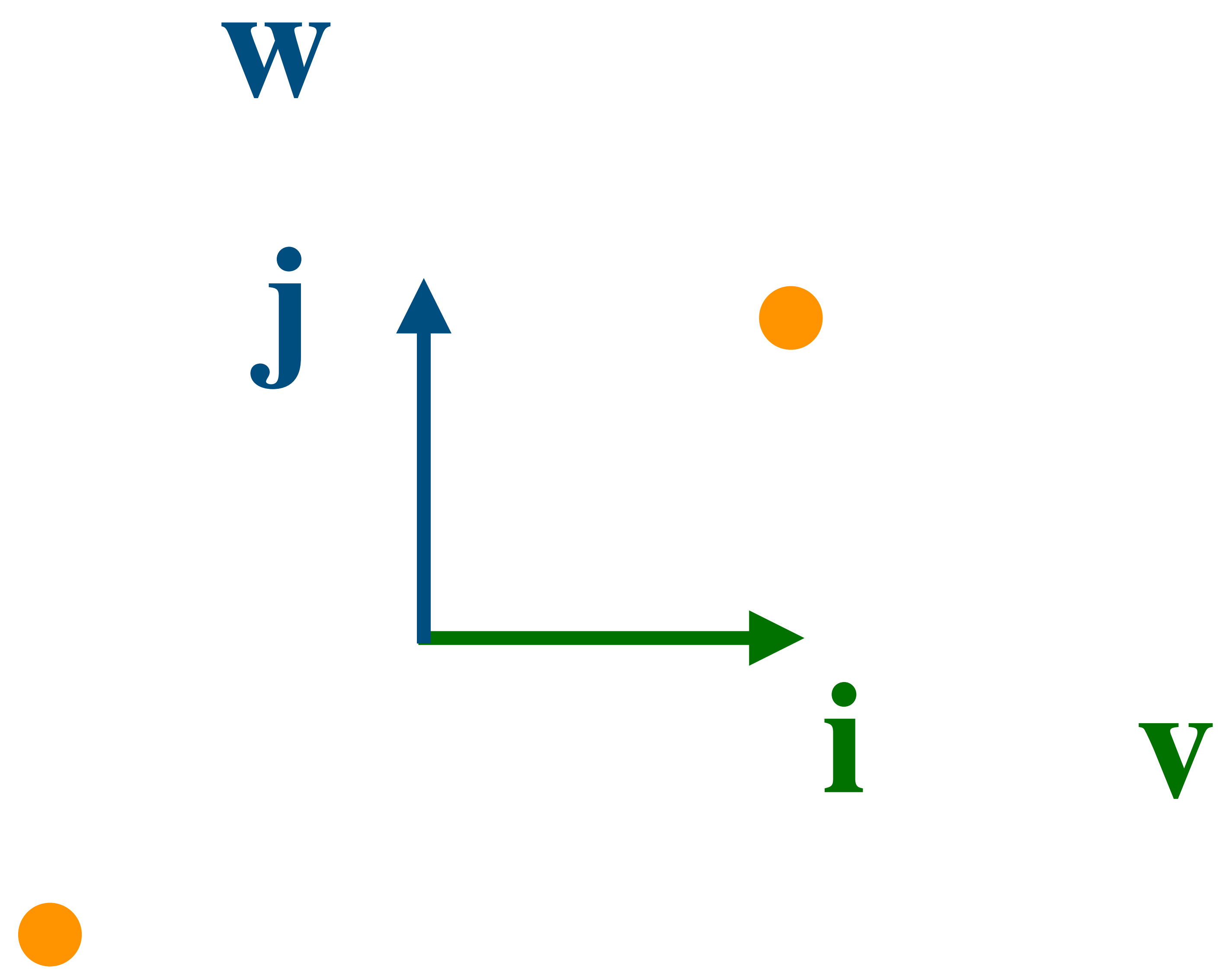
$$2\mathbf{i} + \mathbf{j} = \frac{1}{2}\mathbf{v} + \mathbf{w}$$

$$-2\mathbf{i} - \mathbf{j} = -\frac{1}{2}\mathbf{v} - \mathbf{w}$$

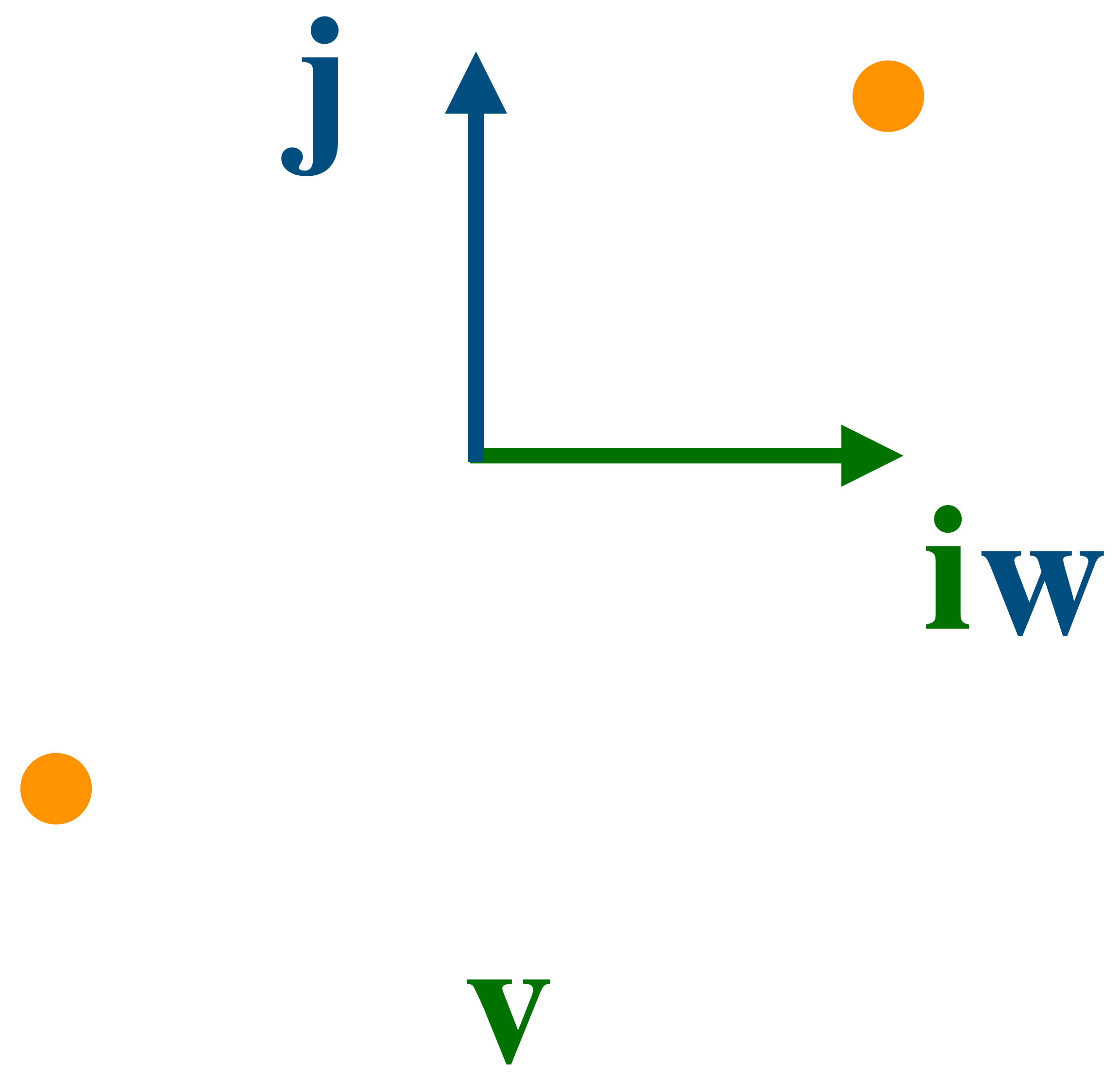
Linear transformation = preserves the coordinates, but changes the basis



Example: scaling



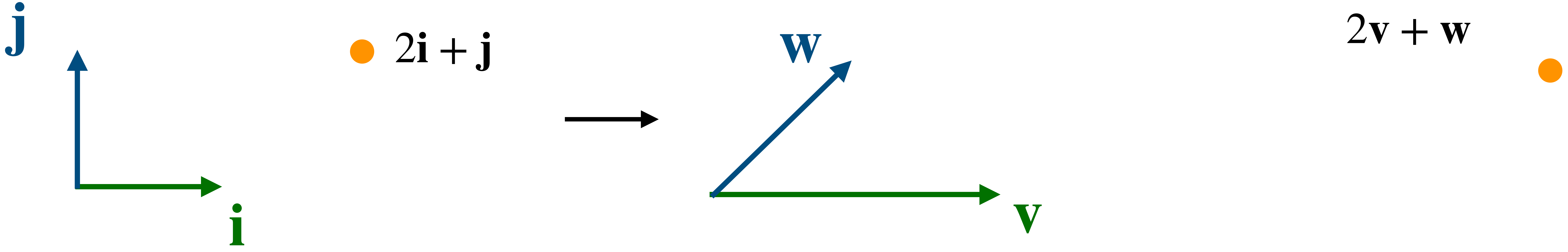
Example: rotation



Computing linear transformation

- step 1: write down the new basis in terms of the old basis

$$\mathbf{v} = 2\mathbf{i} \qquad \mathbf{w} = \mathbf{i} + \mathbf{j}$$

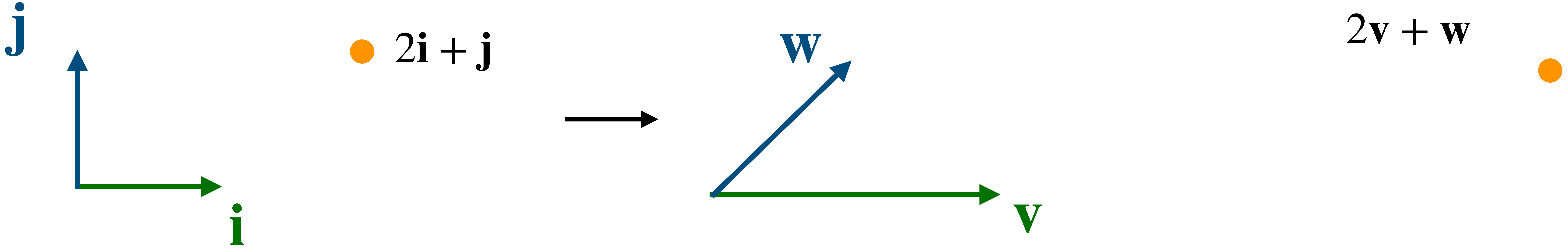


Computing linear transformation

- step 1: write down the new basis in terms of the old basis

$$\mathbf{v} = 2\mathbf{i} \qquad \mathbf{w} = \mathbf{i} + \mathbf{j}$$

- step 2: substitute $2\mathbf{v} + \mathbf{w} = 2\left(2\mathbf{i}\right) + \left(\mathbf{i} + \mathbf{j}\right) = 5\mathbf{i} + \mathbf{j}$

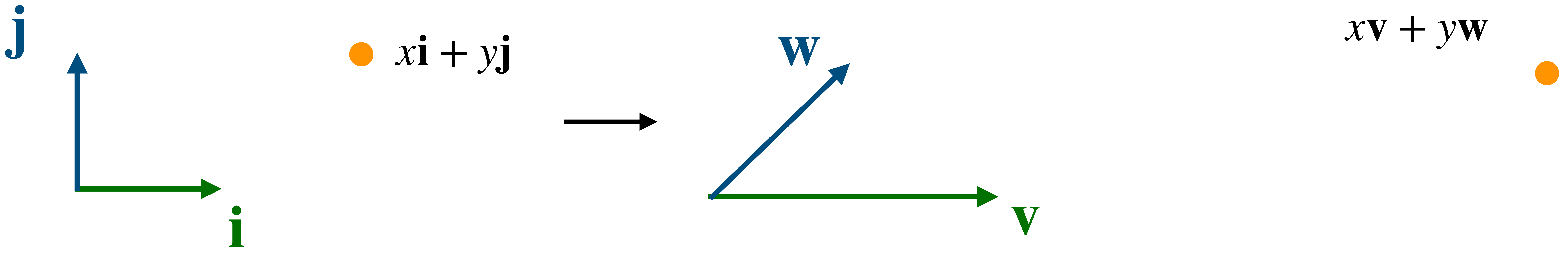


Computing linear transformation

- step 1: write down the new basis in terms of the old basis

$$\mathbf{v} = 2\mathbf{i} \qquad \mathbf{w} = \mathbf{i} + \mathbf{j}$$

- step 2: substitute $x\mathbf{v} + y\mathbf{w} = x\left(2\mathbf{i}\right) + y\left(\mathbf{i} + \mathbf{j}\right) = \left(2x + y\right)\mathbf{i} + y\mathbf{j}$

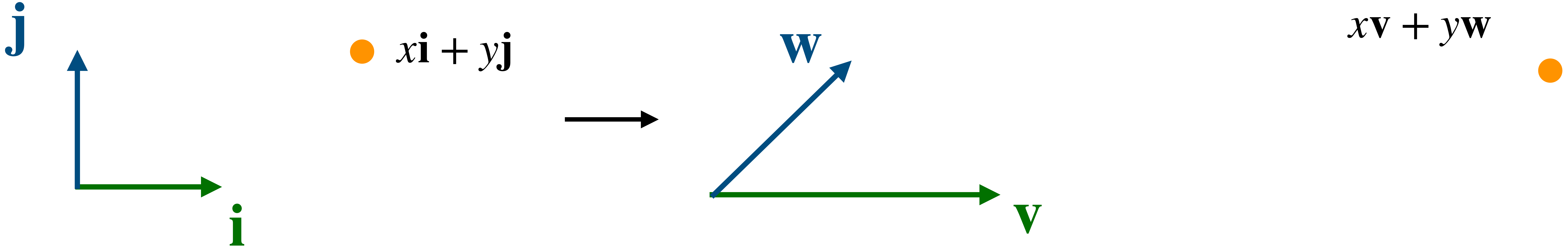


Representing linear transformation as a matrix

$\mathbf{v} = 2\mathbf{i}$

$\mathbf{w} = \mathbf{i} + \mathbf{j}$

$$\begin{array}{cc} \mathbf{v} & \mathbf{w} \\ \end{array}$$
$$\begin{bmatrix} \quad & \quad \\ \quad & \quad \end{bmatrix}$$



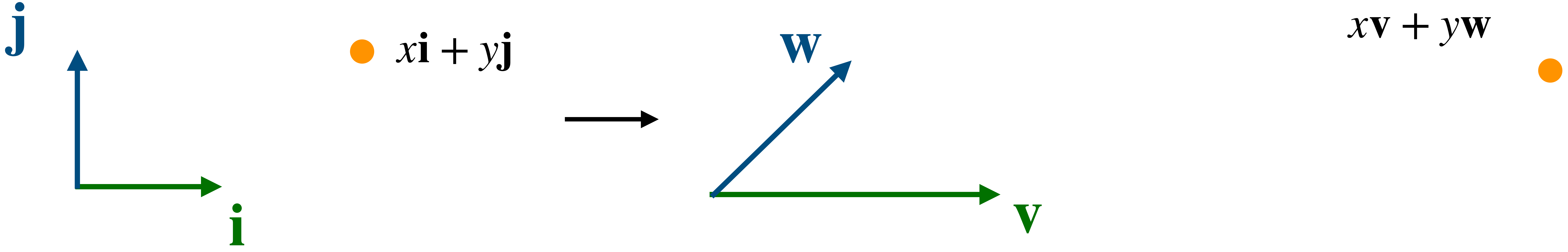
Representing linear transformation as a matrix

$\mathbf{v} = 2\mathbf{i}$

$\mathbf{w} = \mathbf{i} + \mathbf{j}$

$$\begin{array}{cc} \mathbf{v} & \mathbf{w} \\ \multicolumn{2}{c}{\begin{bmatrix} \boxed{\begin{matrix} 2 \\ 0 \end{matrix}} & \end{bmatrix}} \end{array}$$

where does **i** go?



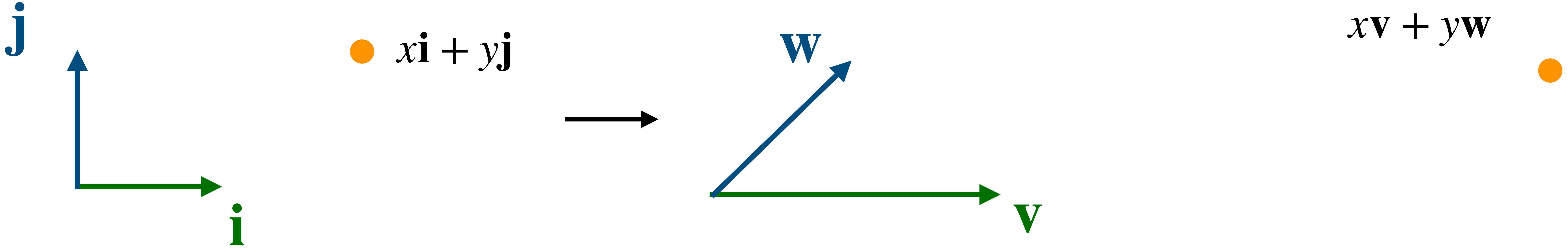
Representing linear transformation as a matrix

$\mathbf{v} = 2\mathbf{i}$

$\mathbf{w} = \mathbf{i} + \mathbf{j}$

$$\begin{array}{cc} \mathbf{v} & \mathbf{w} \\ \multicolumn{2}{c}{\begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}} \end{array}$$

where does $\mathbf{j}$ go?

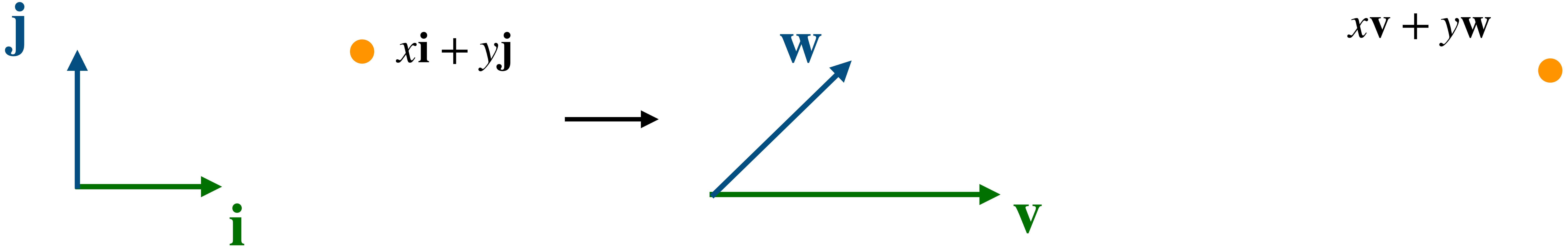


Applying linear transformation as matrix-vector product

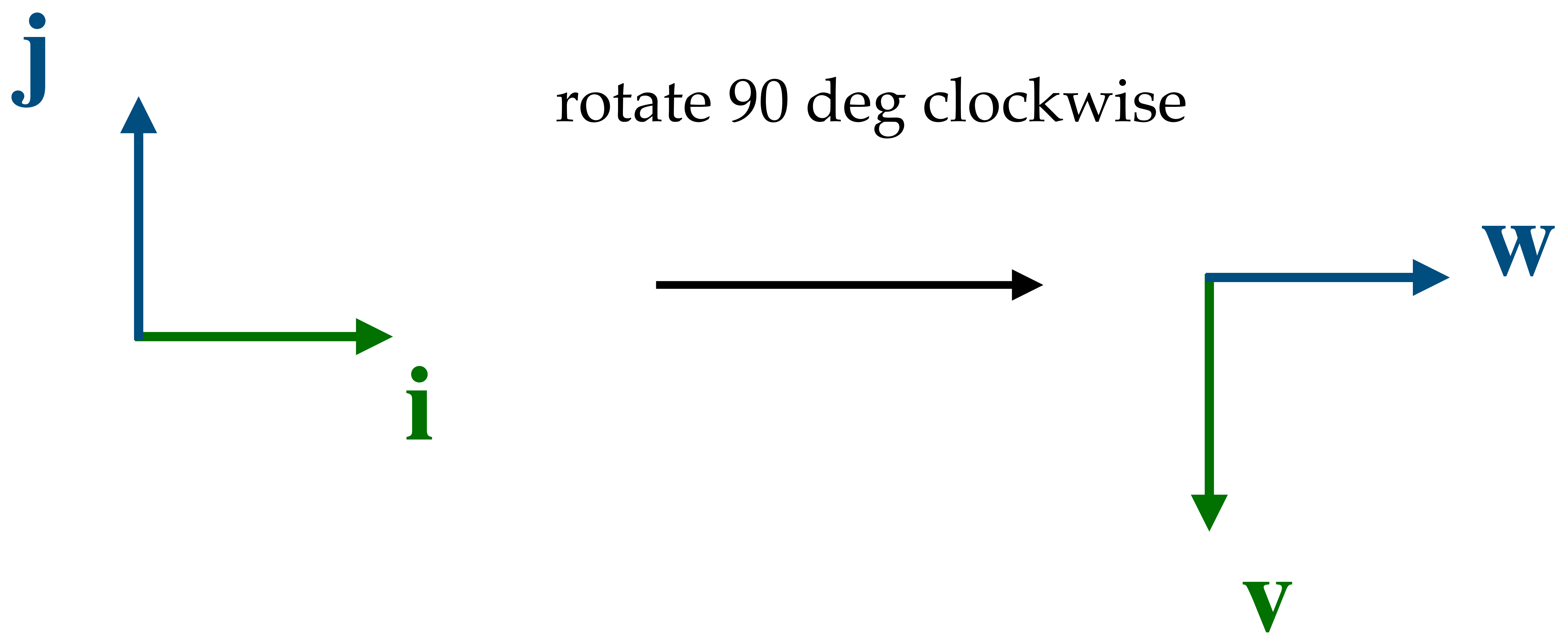
$\mathbf{v} = 2\mathbf{i}$

$\mathbf{w} = \mathbf{i} + \mathbf{j}$

$$\begin{array}{c} \begin{array}{cc} \mathbf{v} & \mathbf{w} \end{array} \\ \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix} \end{array} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2x + y \\ y \end{bmatrix}$$

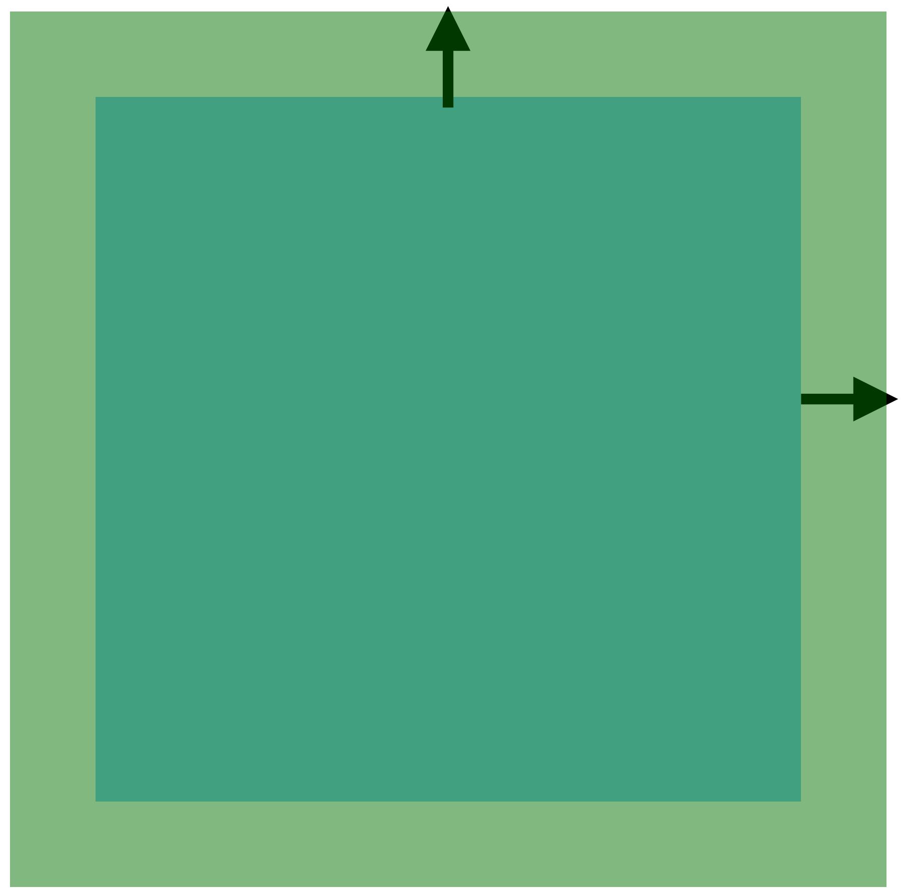


Quiz: what is the matrix form of this transformation?



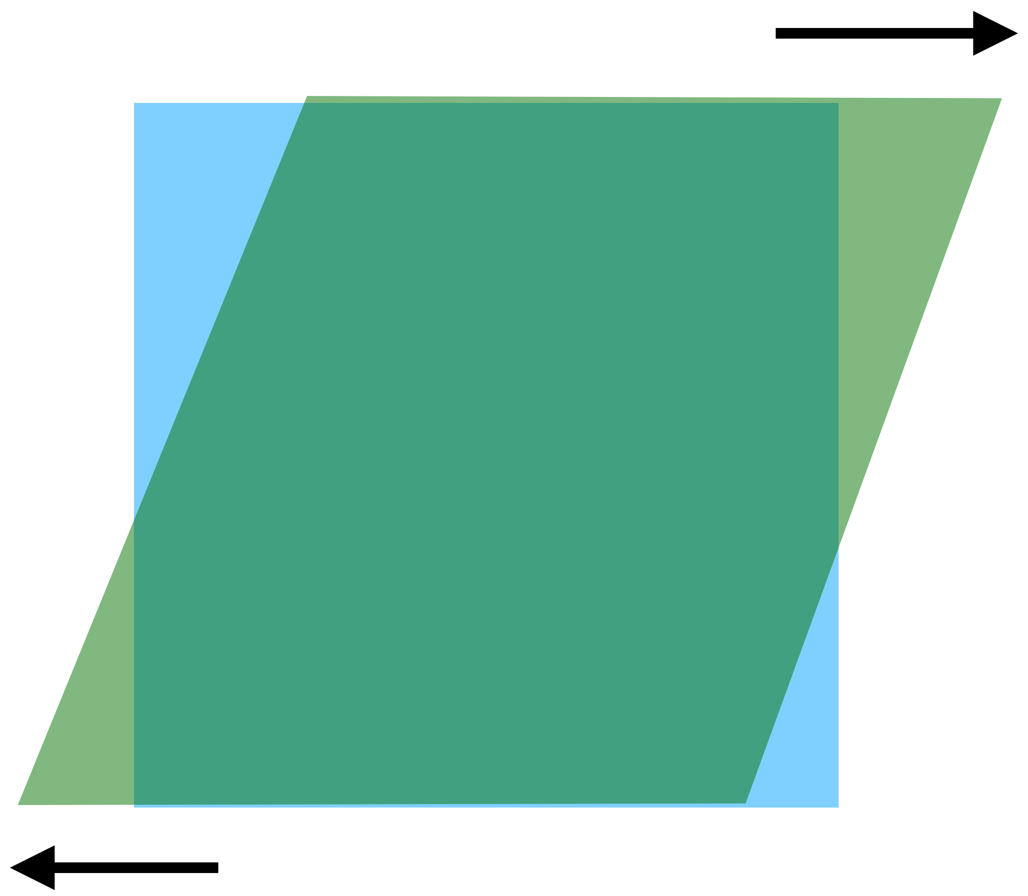
Common 2D linear transformations

scaling



$$\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

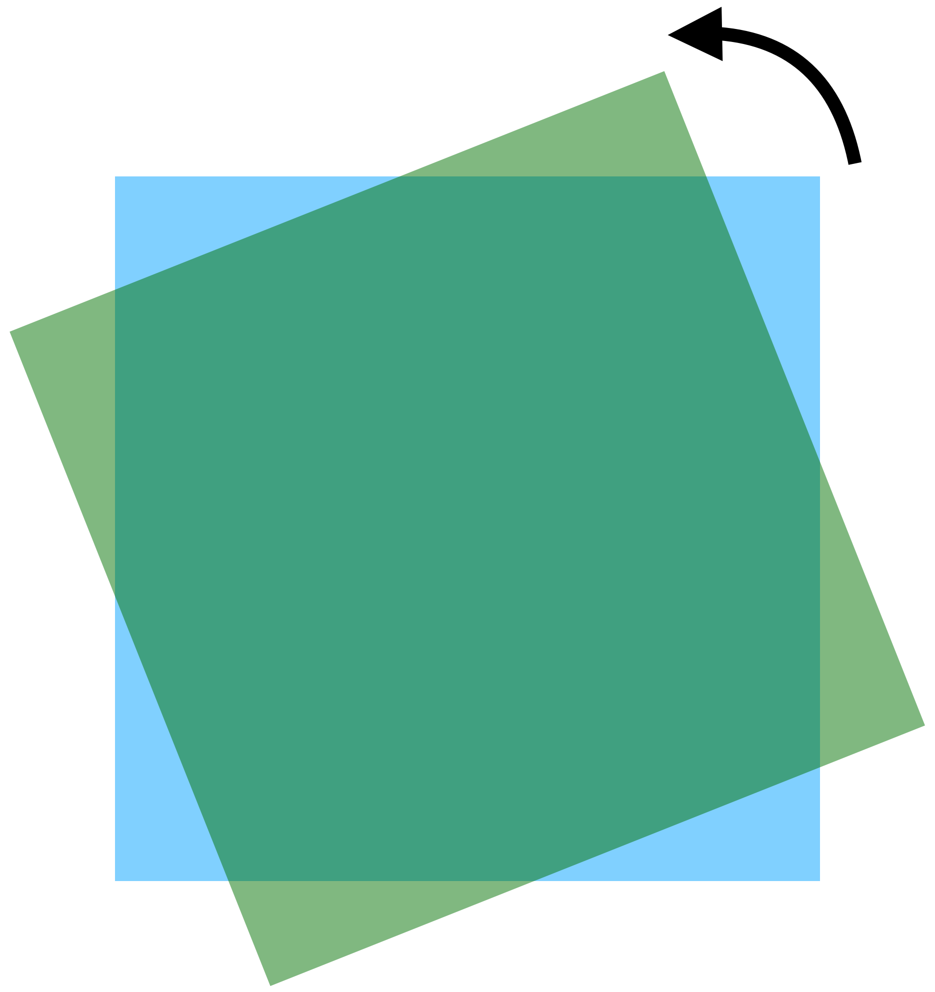
Common 2D linear transformations



x shearing

$$\begin{bmatrix} 1 & \lambda_x \\ 0 & 1 \end{bmatrix}$$

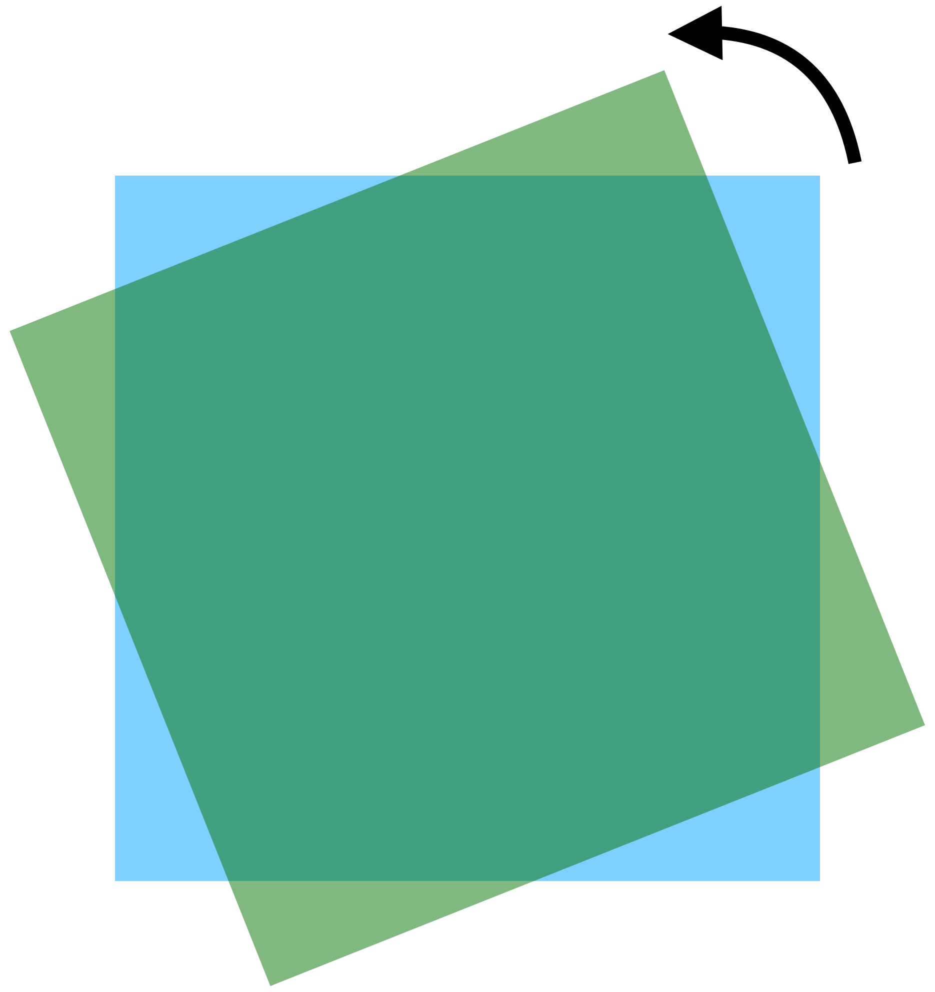
Common 2D linear transformations



rotation

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

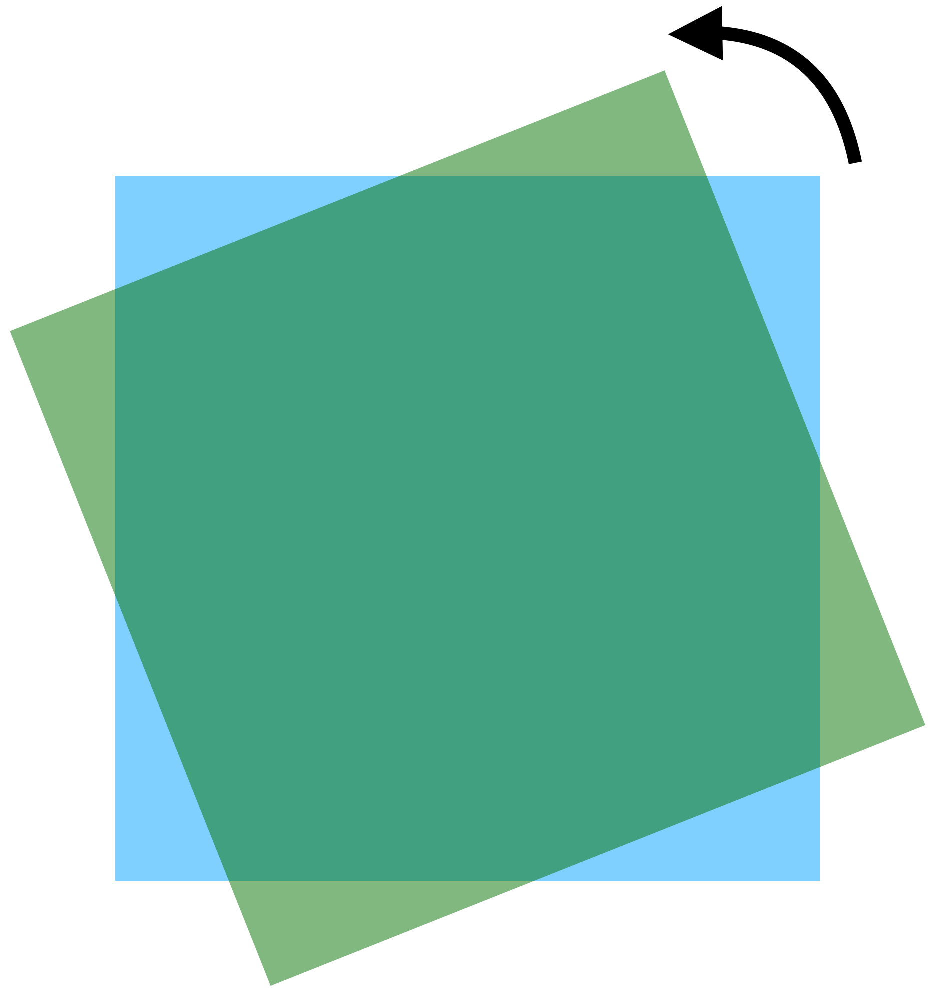
Common 2D linear transformations



rotation

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

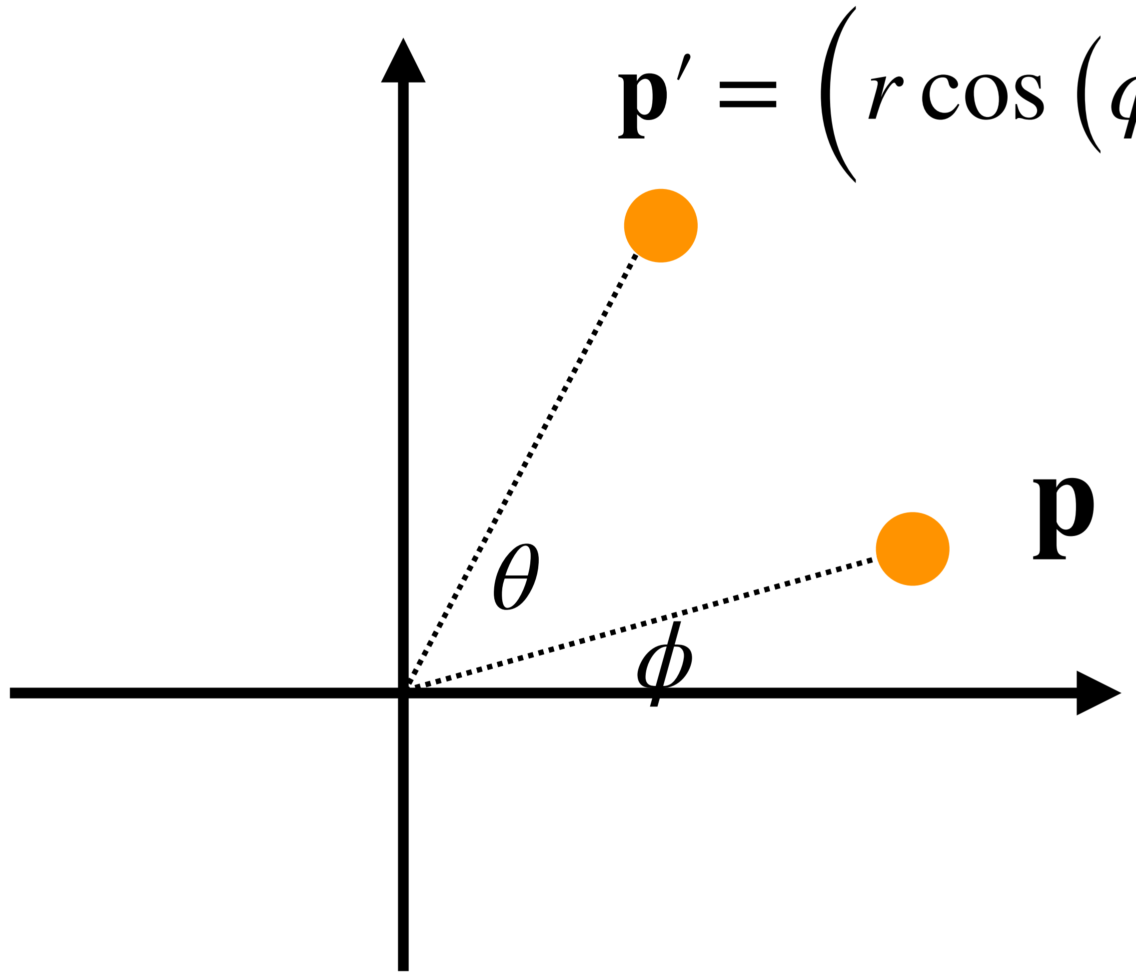
Common 2D linear transformations



rotation

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

$$\mathbf{p}' = \left(r\cos\left(\phi+\theta\right), r\sin\left(\phi+\theta\right)\right)$$

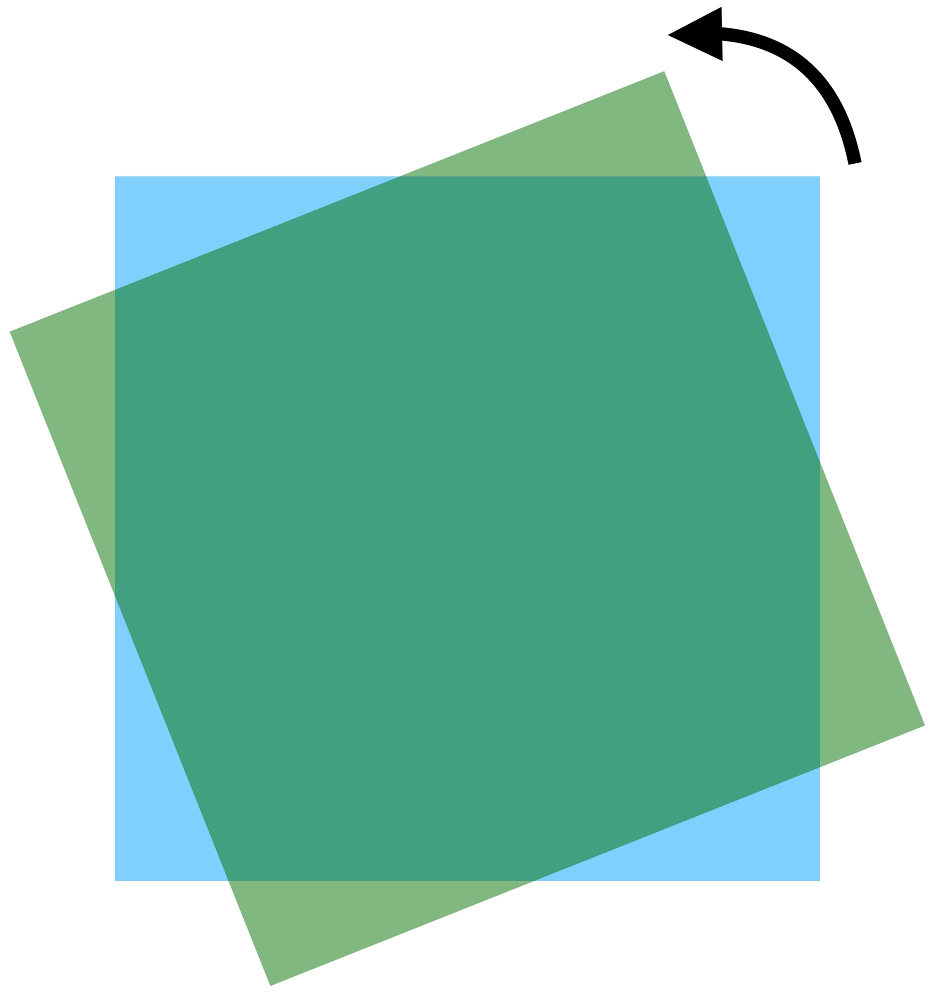


$$\mathbf{p} = \left(r\cos\phi, r\sin\phi\right)$$

Common 2D linear transformations

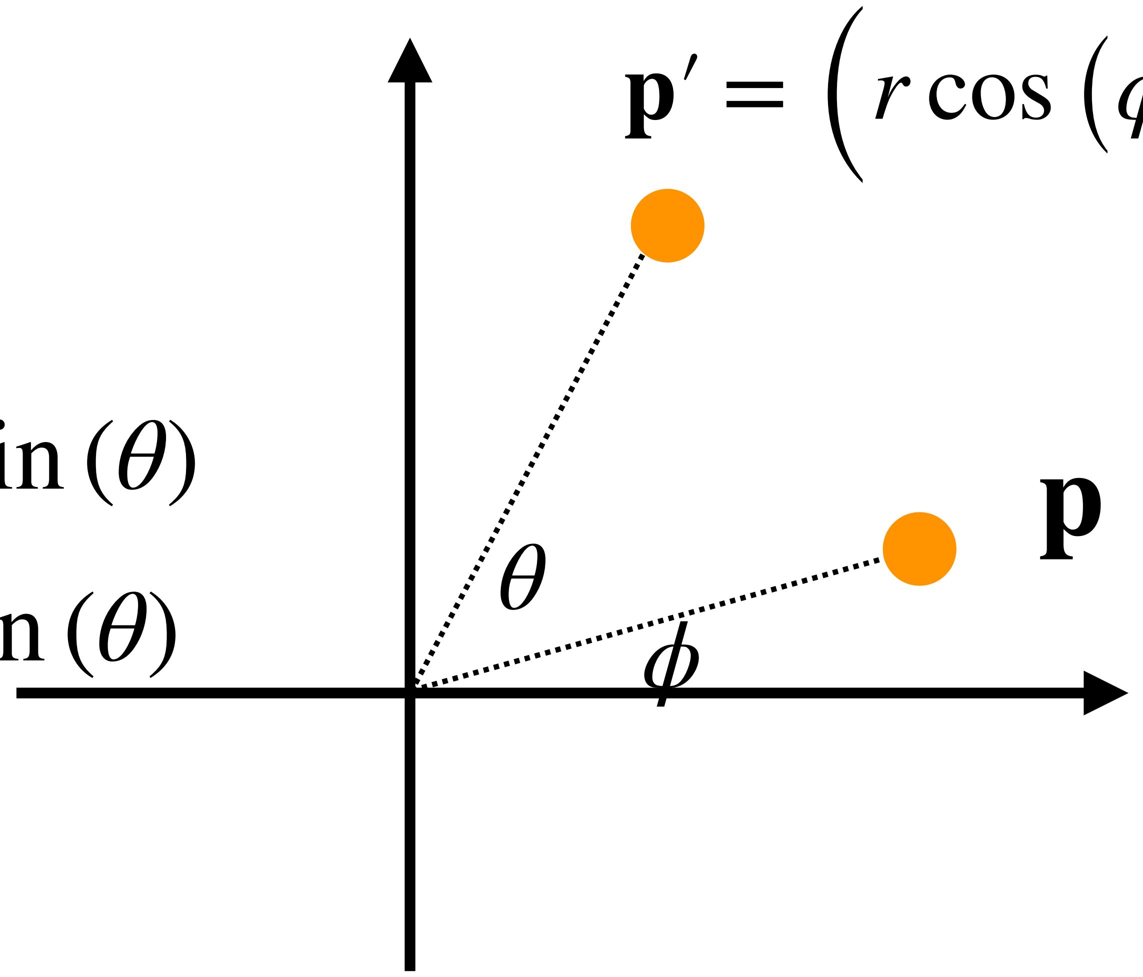
rotation

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



$$\cos(\phi+\theta) = \cos(\phi)\cos(\theta) - \sin(\phi)\sin(\theta)$$
$$\sin(\phi+\theta) = \sin(\phi)\cos(\theta) + \cos(\phi)\sin(\theta)$$

$$\mathbf{p}' = \left(r\cos(\phi+\theta), r\sin(\phi+\theta)\right)$$



$$\mathbf{p} = \left(r\cos\phi, r\sin\phi\right)$$

What about translation?

we can’t write translation as a 2x2 matrix!
(the basis vectors don’t encode the information of where is the origin)

-> translation is not a 2D linear transformation!

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Trick: augmented matrix

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

$$\downarrow$$

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

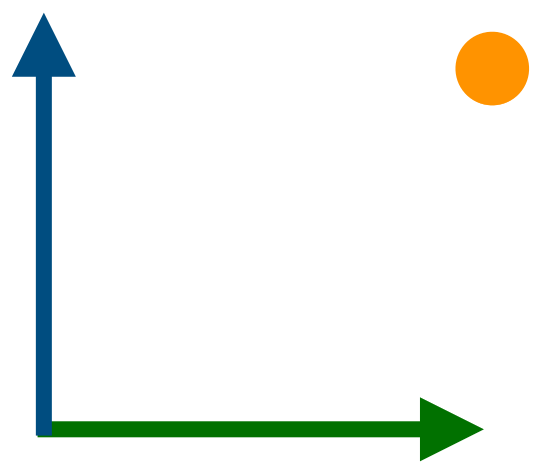
Trick: augmented matrix

$$\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

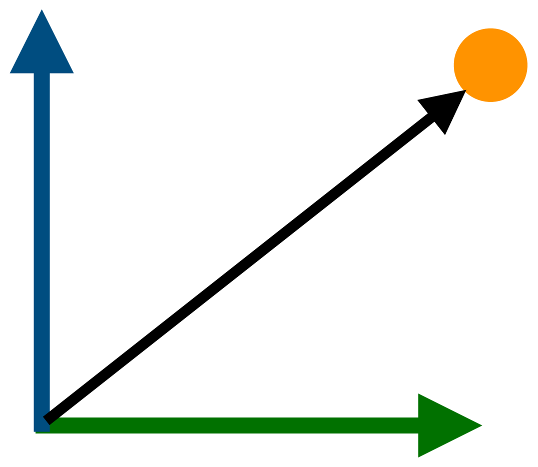
the 2D transformations represented by
this matrix are called **affine transformation**

Augmented matrices allow us to distinguish **points** and **vectors**

points: locations in space

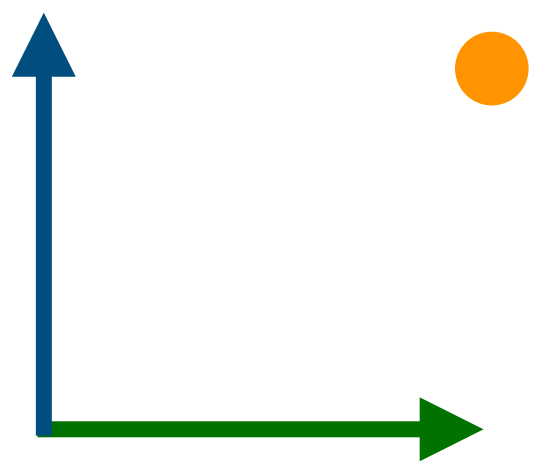


vectors: offsets in space

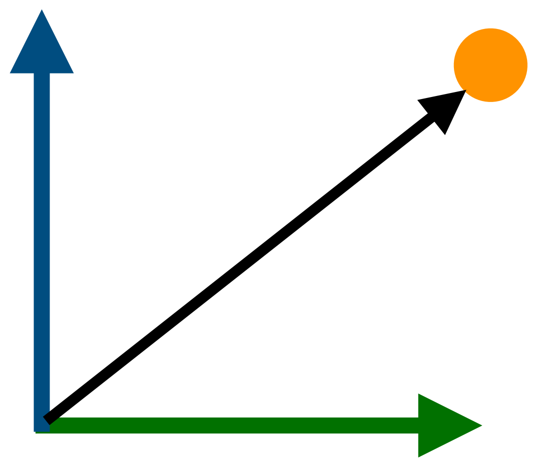


Augmented matrices allow us to distinguish **points** and **vectors**

points: locations in space



vectors: offsets in space



points are affected by translation

vectors are **not** affected by translation

Augmented matrices allow us to distinguish **points** and **vectors**

setting the third coordinate to 0 allows us to "turn off" the translation feature

$$\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

points are affected by translation

$$\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$$

vectors are **not** affected by translation

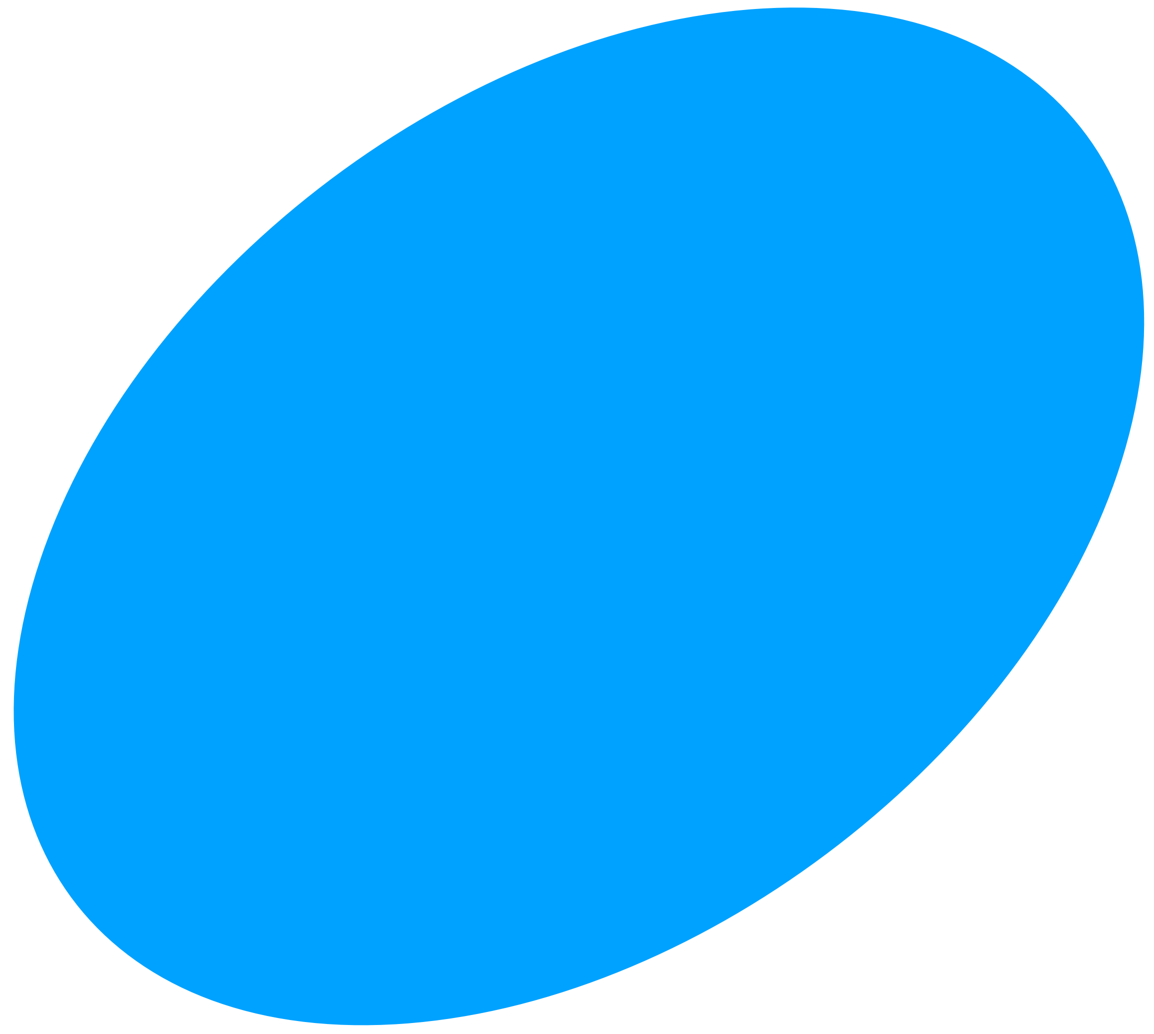
Combining multiple transformations = multiplying matrices

e.g., first scale (S), then rotate (R), then translate (T)

$$\text{translate}\left(\text{rotate}\left(\text{scale}\left(\mathbf{p}\right)\right)\right) = TRS\mathbf{p}$$

we can premultiply the TRS matrices if we want to apply them to many **p**s

How do we use linear transformation to render an ellipse?

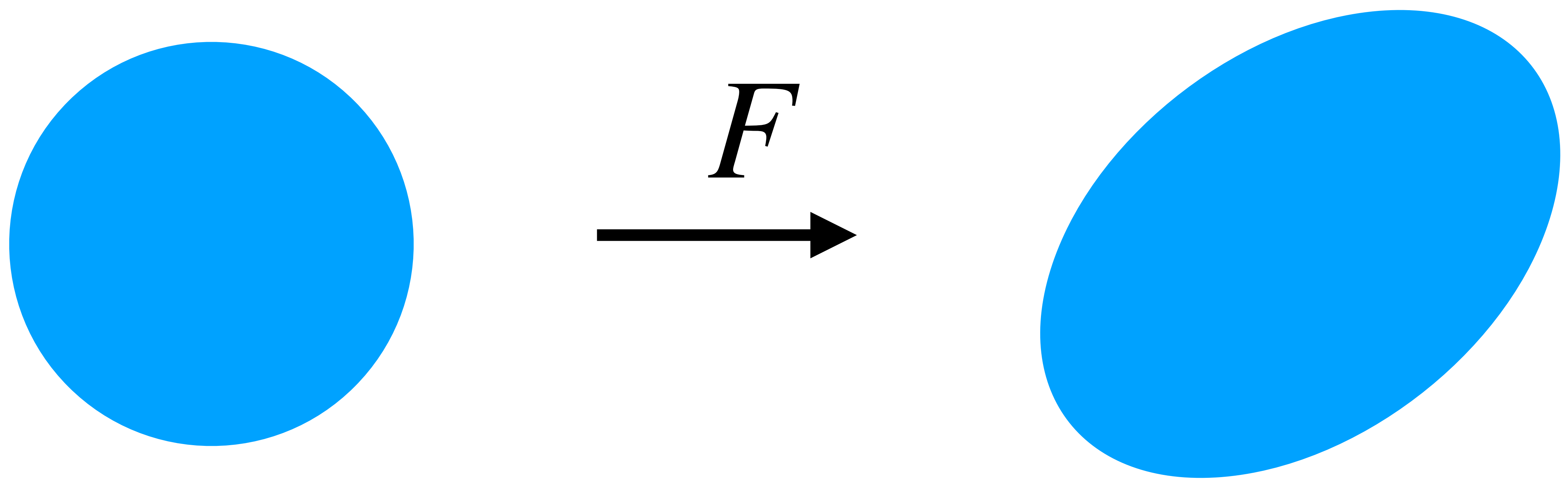


Object space vs "canvas space"

I invented this term

object space

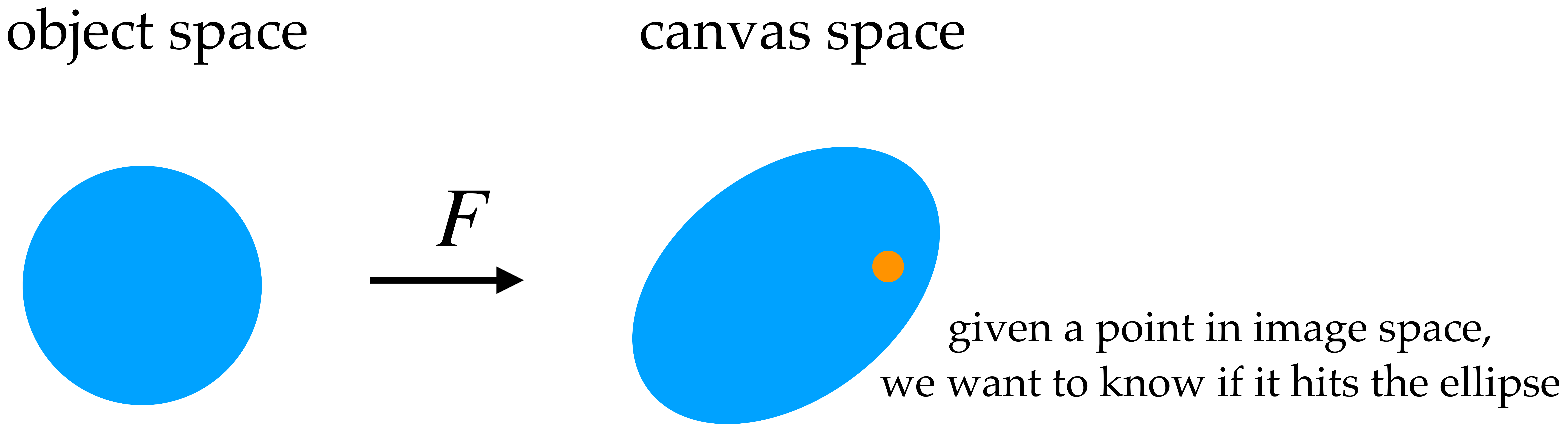
canvas space



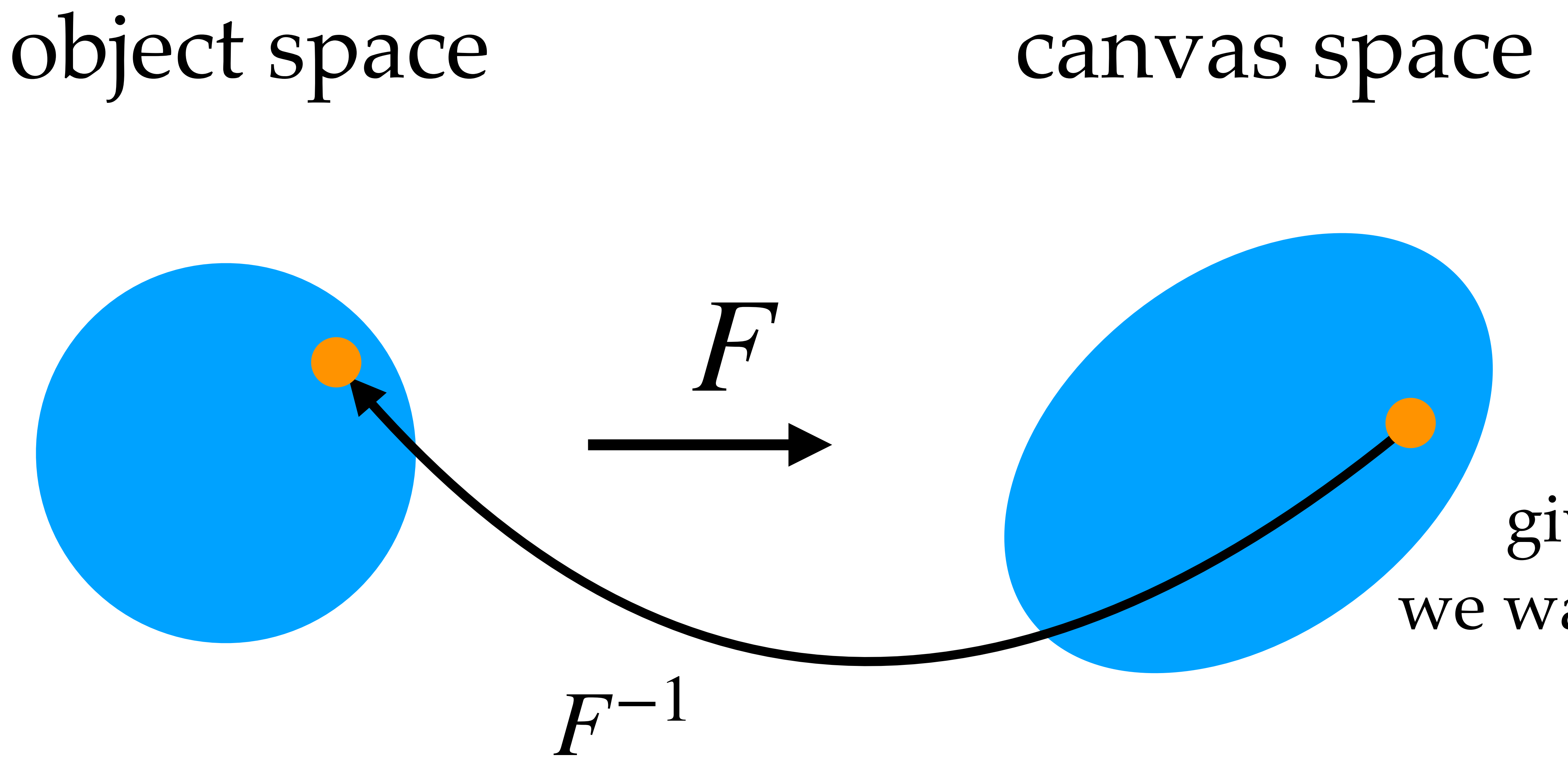
the space we define
the object and test whether
a point is inside the circle

the space we actually
see the transformed circle

Rendering by testing in object space



Rendering by testing in object space

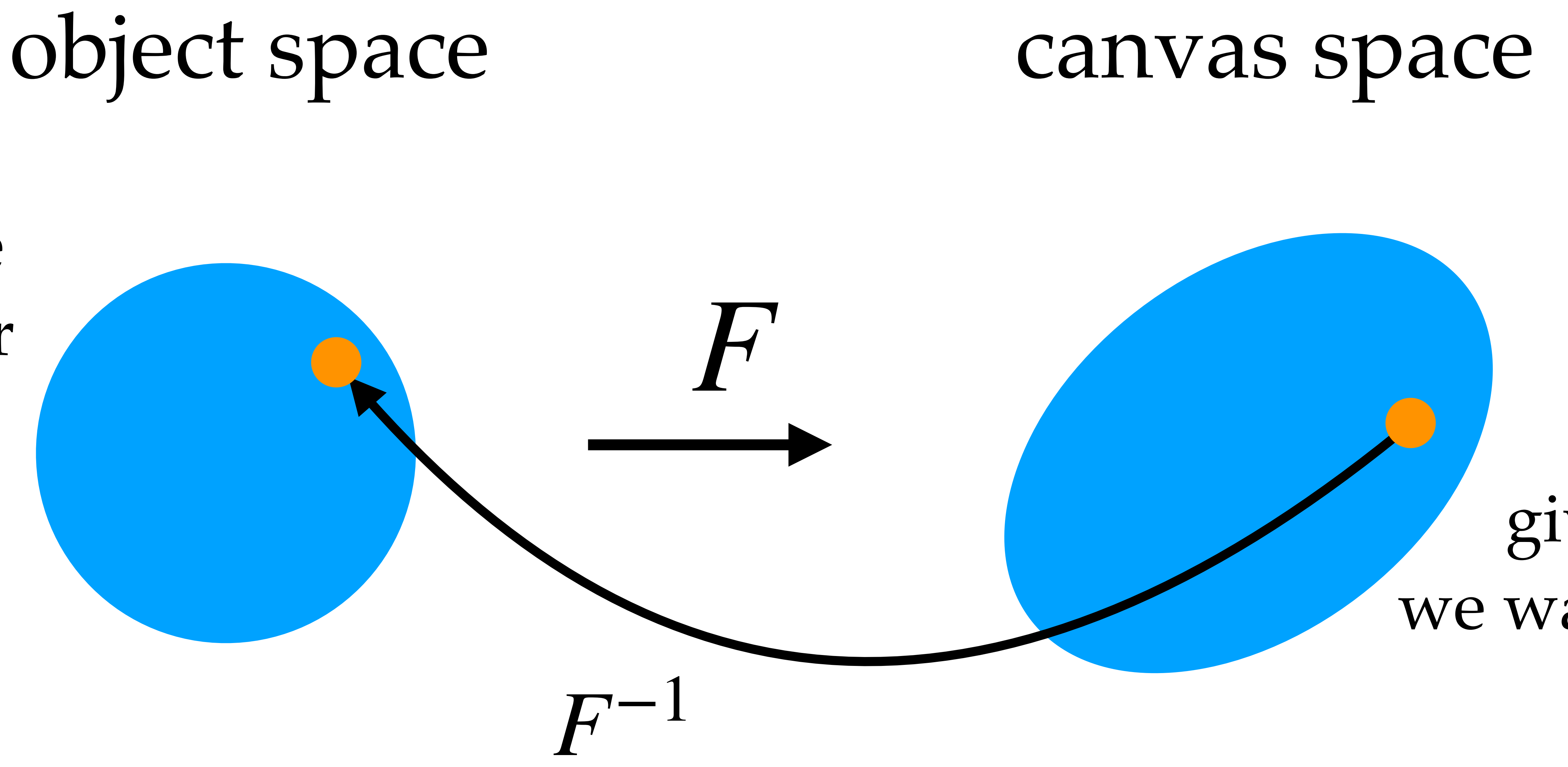


given a point in image space,
we want to know if it hits the ellipse

we apply the inverse transform F^{-1} to the point,
converting it from image space to object space

Rendering by testing in object space

we test in object space
which is much simpler

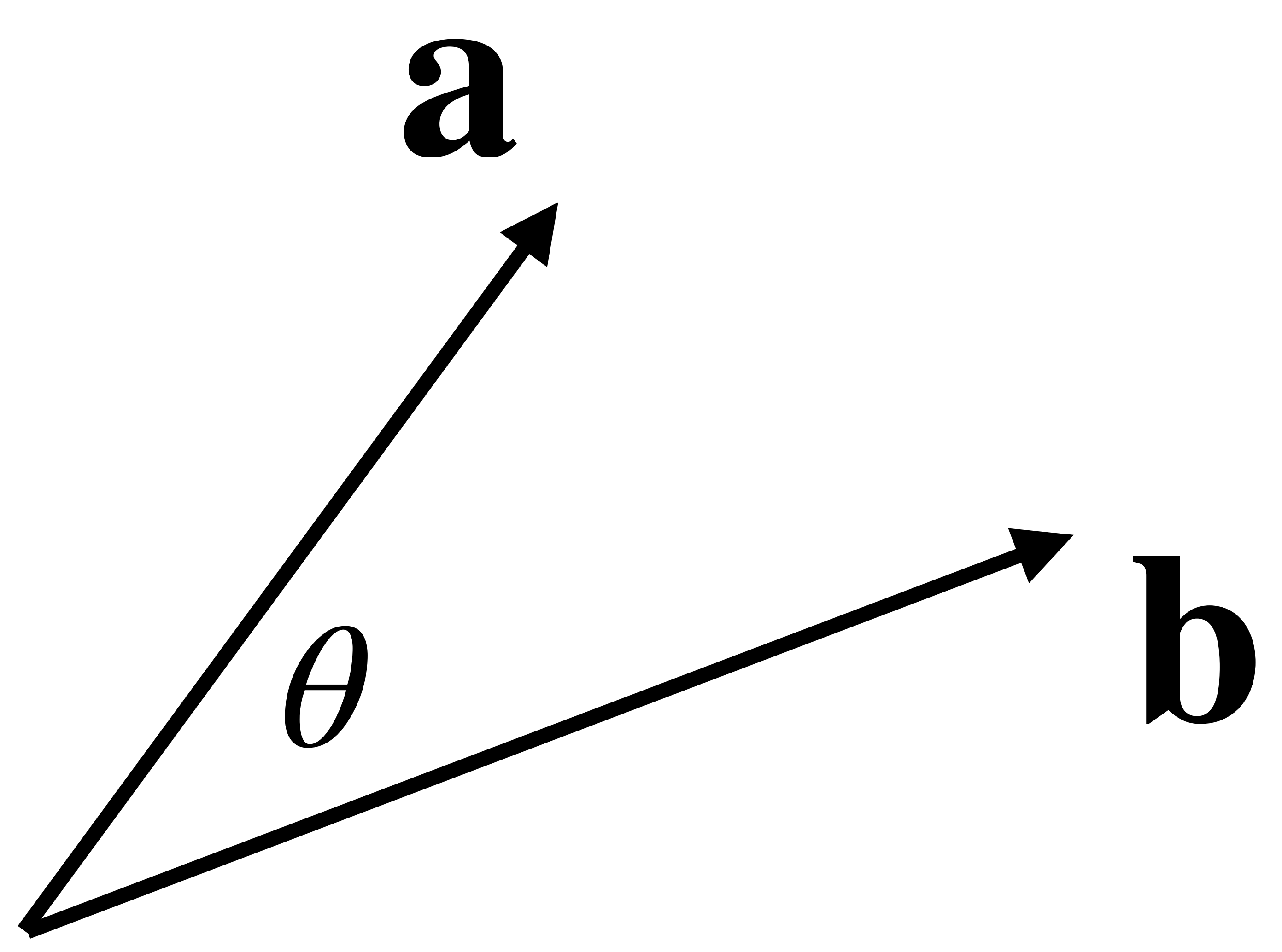


given a point in image space,
we want to know if it hits the ellipse

we apply the inverse transform F^{-1} to the point,
converting it from canvas space to object space

Dot product

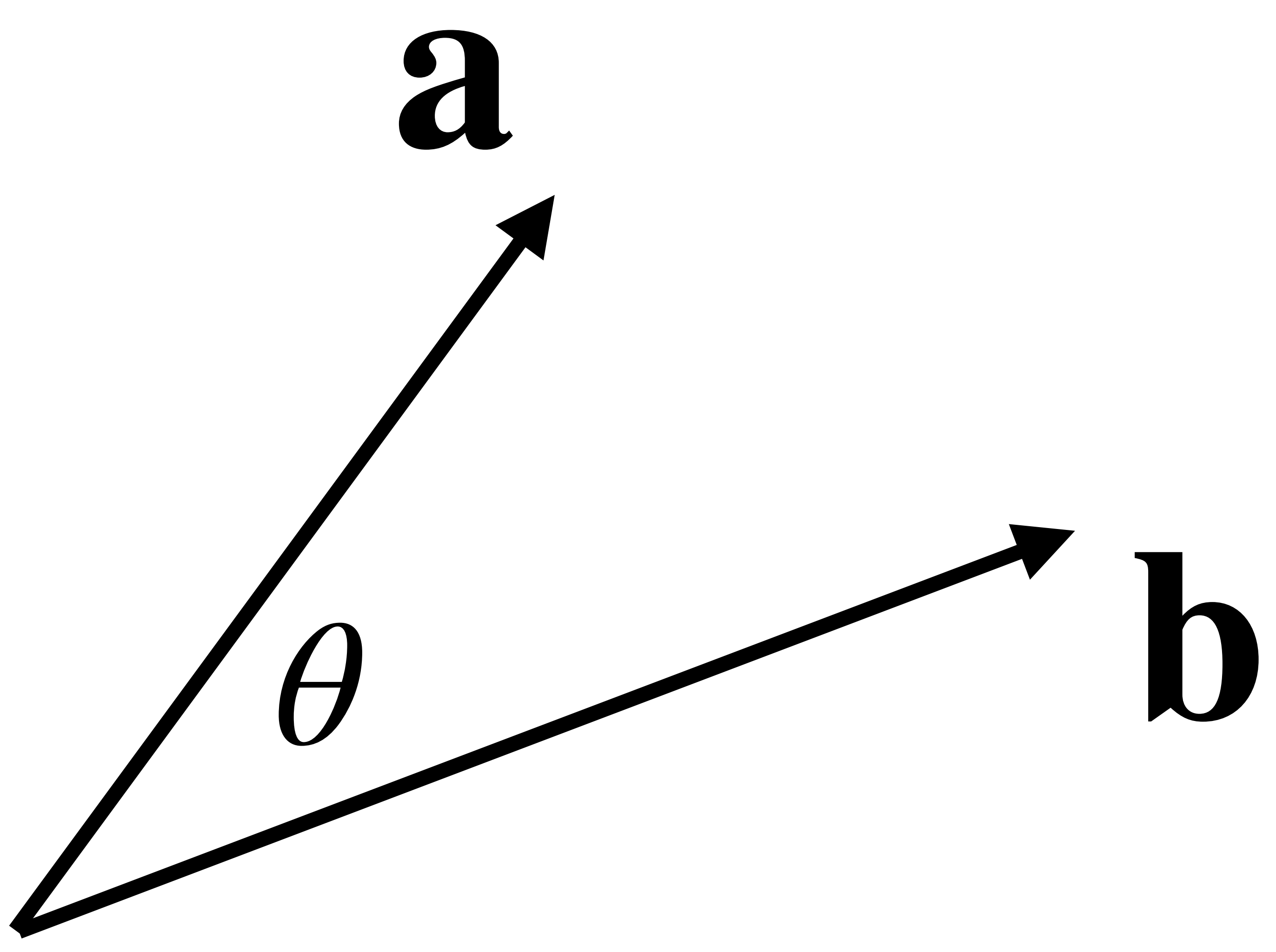
dot products measure projected length of vectors



$$\mathbf{a} \cdot \mathbf{b} = \| a \| \, \| b \| \cos\theta$$

Dot product

dot products measure projected length of vectors



$$\mathbf{a} \cdot \mathbf{b} = \| a \| \ \| b \| \cos\theta$$

but we can also compute it by element-wise products — why?

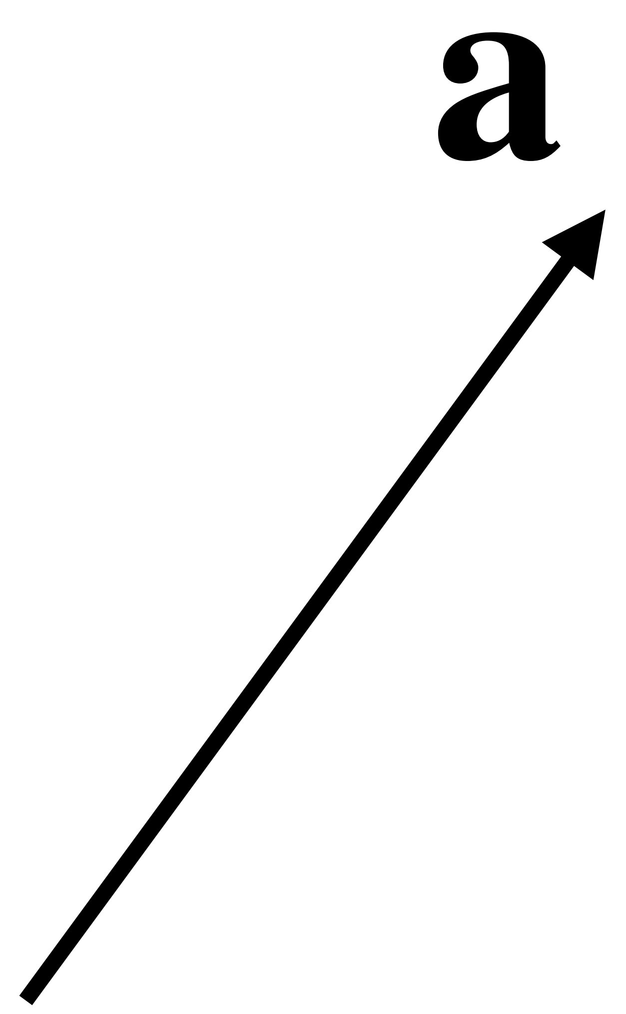
$$\mathbf{a} \cdot \mathbf{b} = \mathbf{a}_x \mathbf{b}_x + \mathbf{a}_y \mathbf{b}_y$$

Dot product as a linear transformation

we can see $\mathbf{a} \cdot \mathbf{b}$ as applying a linear transformation $\mathbf{a}^T$ to $\mathbf{b}$

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{a}^T \mathbf{b}$$

$$\mathbf{a}^T = \begin{bmatrix} \mathbf{a}_x & \mathbf{a}_y \end{bmatrix}$$



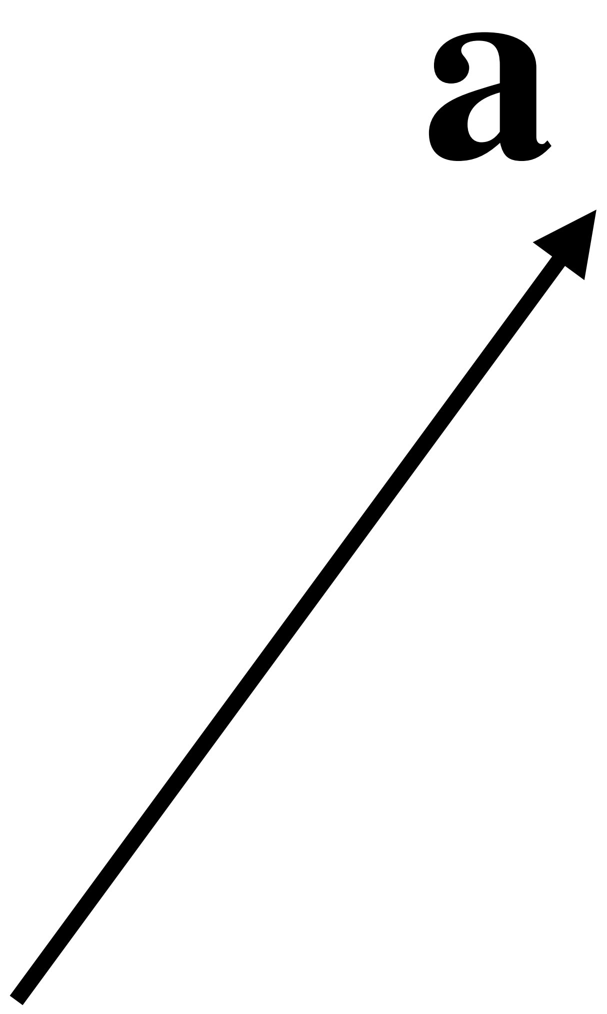
Dot product as a linear transformation

we can see $\mathbf{a} \cdot \mathbf{b}$ as applying a linear transformation $\mathbf{a}^T$ to $\mathbf{b}$

let's for now assume $\mathbf{a}$ is a unit vector

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{a}^T \mathbf{b}$$

$$\mathbf{a}^T = \begin{bmatrix} \mathbf{a}_x & \mathbf{a}_y \end{bmatrix}$$



Dot product as a linear transformation

we can see $\mathbf{a} \cdot \mathbf{b}$ as applying a linear transformation $\mathbf{a}^T$ to $\mathbf{b}$

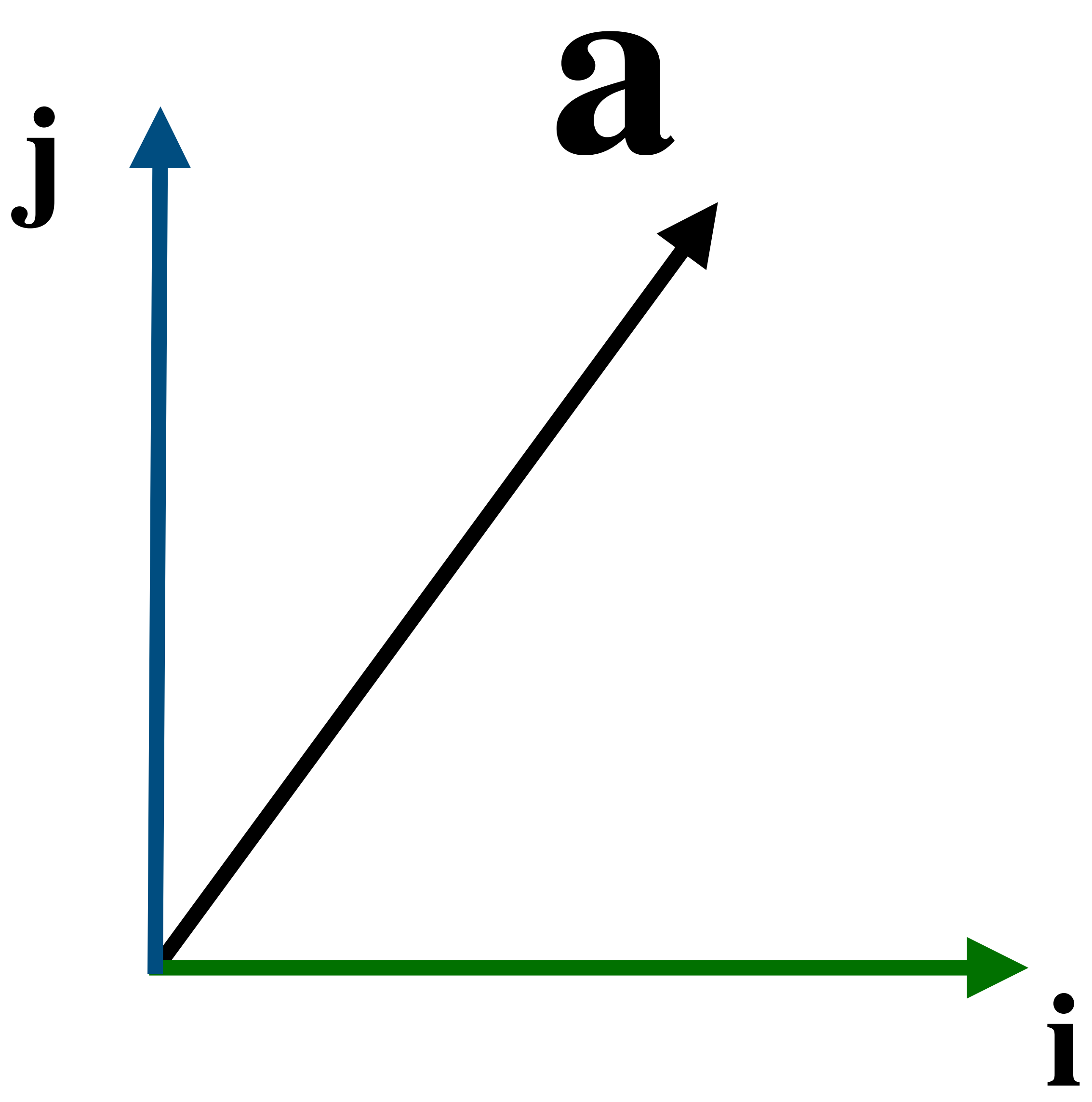
let's for now assume $\mathbf{a}$ is a unit vector

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{a}^T \mathbf{b}$$

where $\mathbf{j}$ goes

$$\mathbf{a}^T = \begin{bmatrix} \mathbf{a}_x & \mathbf{a}_y \end{bmatrix}$$

where $\mathbf{i}$ goes



Dot product as a linear transformation

we can see $\mathbf{a} \cdot \mathbf{b}$ as applying a linear transformation $\mathbf{a}^T$ to $\mathbf{b}$

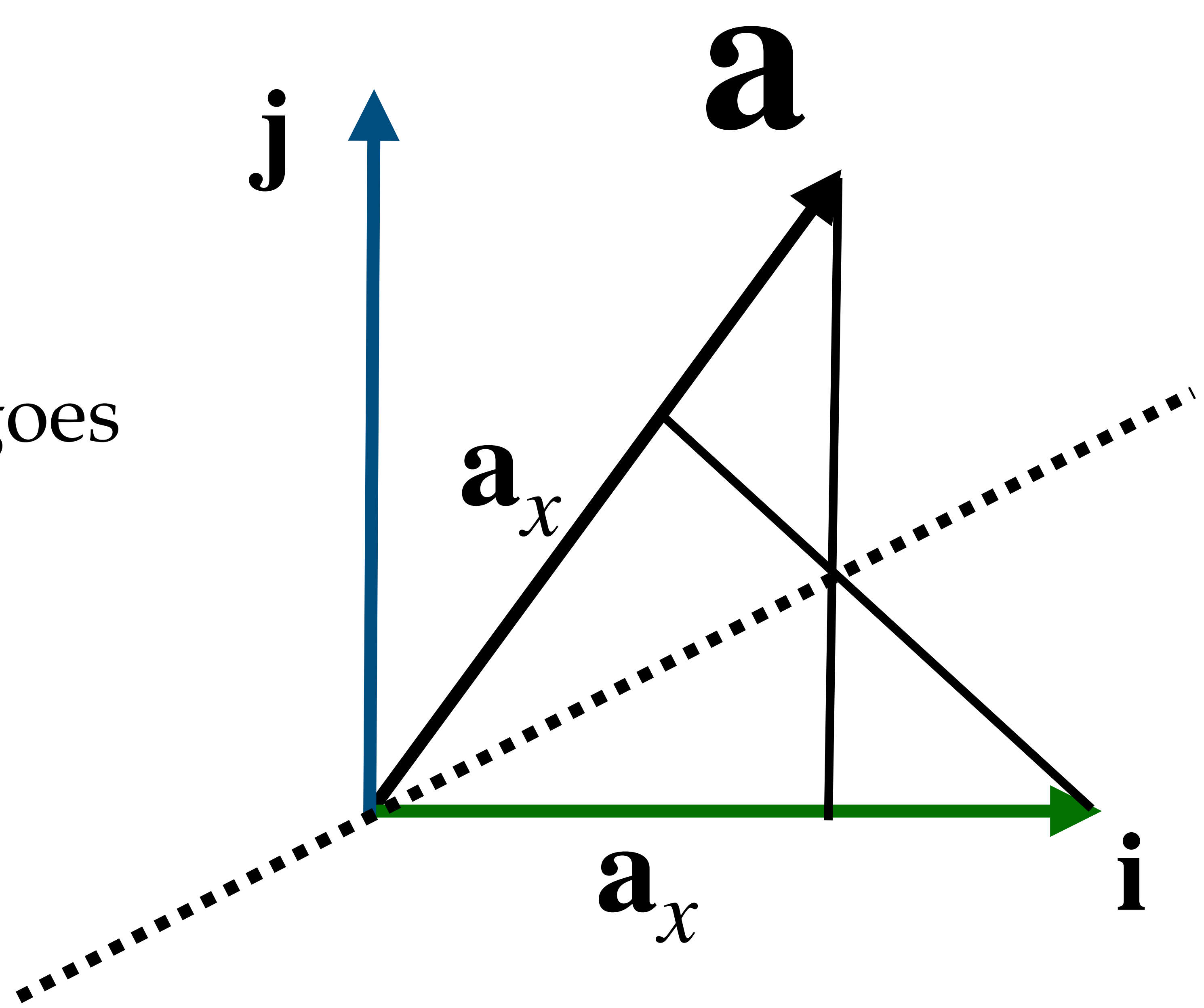
let's for now assume $\mathbf{a}$ is a unit vector

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{a}^T \mathbf{b}$$

where $\mathbf{j}$ goes

$$\mathbf{a}^T = \begin{bmatrix} \mathbf{a}_x & \mathbf{a}_y \end{bmatrix}$$

where $\mathbf{i}$ goes



$\mathbf{a}_x$ happens to be
the projection length of $\mathbf{i}$!

Dot product as a linear transformation

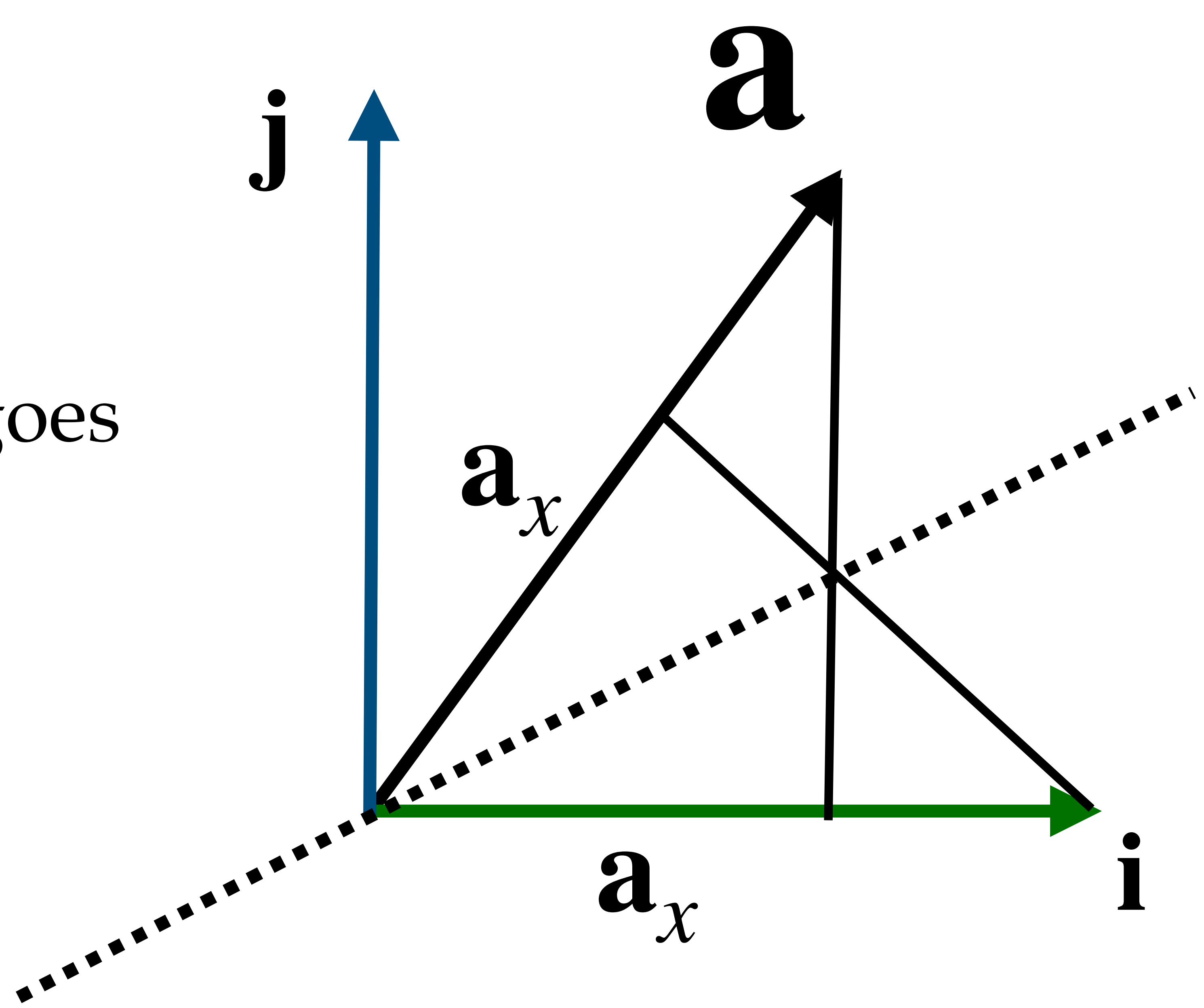
we can see $\mathbf{a} \cdot \mathbf{b}$ as applying a linear transformation $\mathbf{a}^T$ to $\mathbf{b}$

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{a}^T \mathbf{b}$$

where $\mathbf{j}$ goes

$$\mathbf{a}^T = \begin{bmatrix} \mathbf{a}_x & \mathbf{a}_y \end{bmatrix}$$

where $\mathbf{i}$ goes



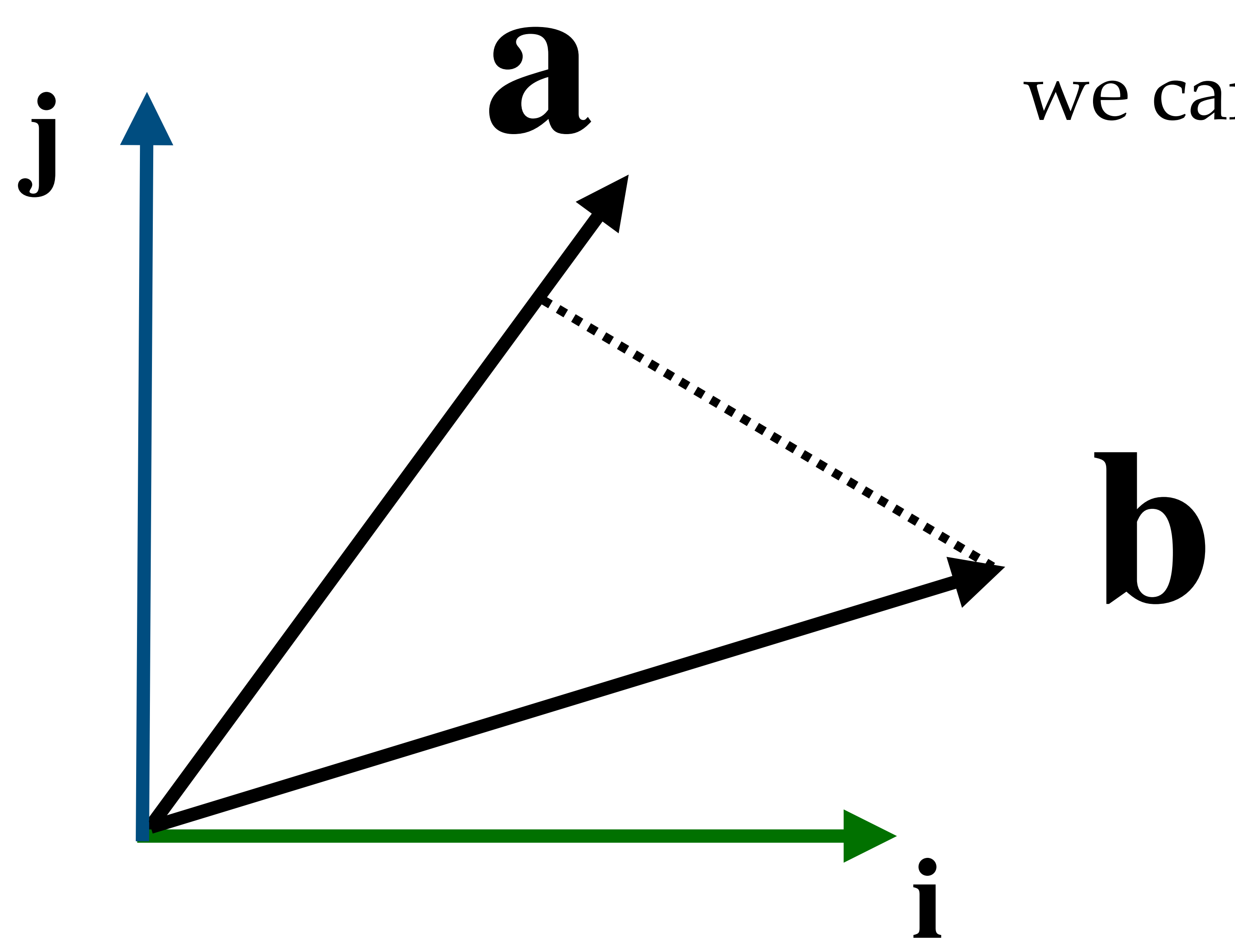
$\mathbf{a}_x$ happens to be the projection length of $\mathbf{i}$!

if $\mathbf{a}$ is not a unit vector, just need to scale the result by its length

Dot product as a linear transformation

we can see $\mathbf{a} \cdot \mathbf{b}$ as applying a linear transformation $\mathbf{a}^T$ to $\mathbf{b}$

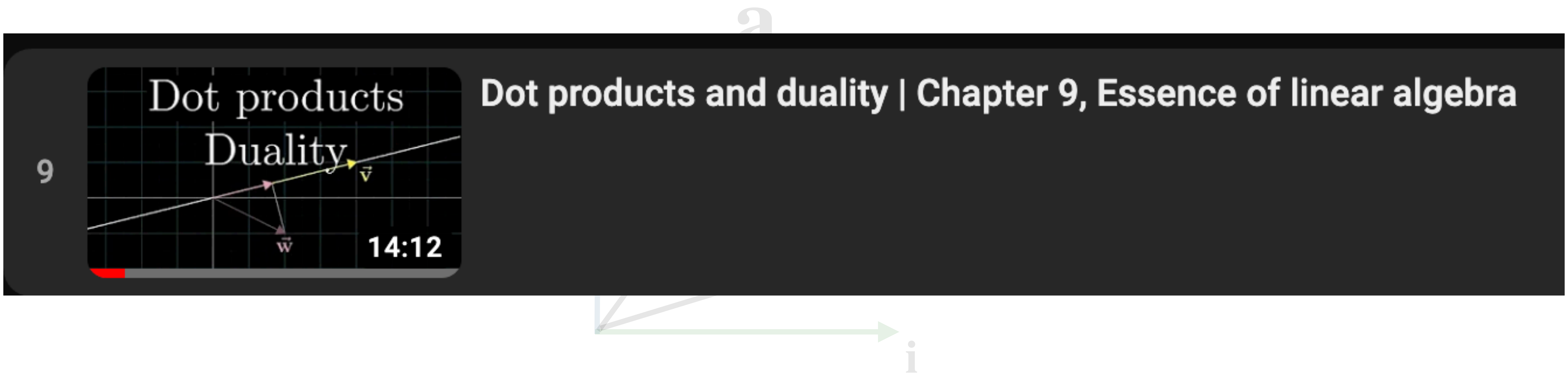
$$\mathbf{a} \cdot \mathbf{b} = \mathbf{a}^T\mathbf{b}$$



we can decompose $\mathbf{b}$ into linear combination of $\mathbf{i}$ and $\mathbf{j}$

therefore,
$\mathbf{a} \cdot \mathbf{b}$ = projecting $\mathbf{b}$ to $\mathbf{a}$ and scale by $\mathbf{a}$'s length

See the 3Blue1Brown video!



https://www.youtube.com/watch?v=LyGKycYT2v0

Next: Cameras