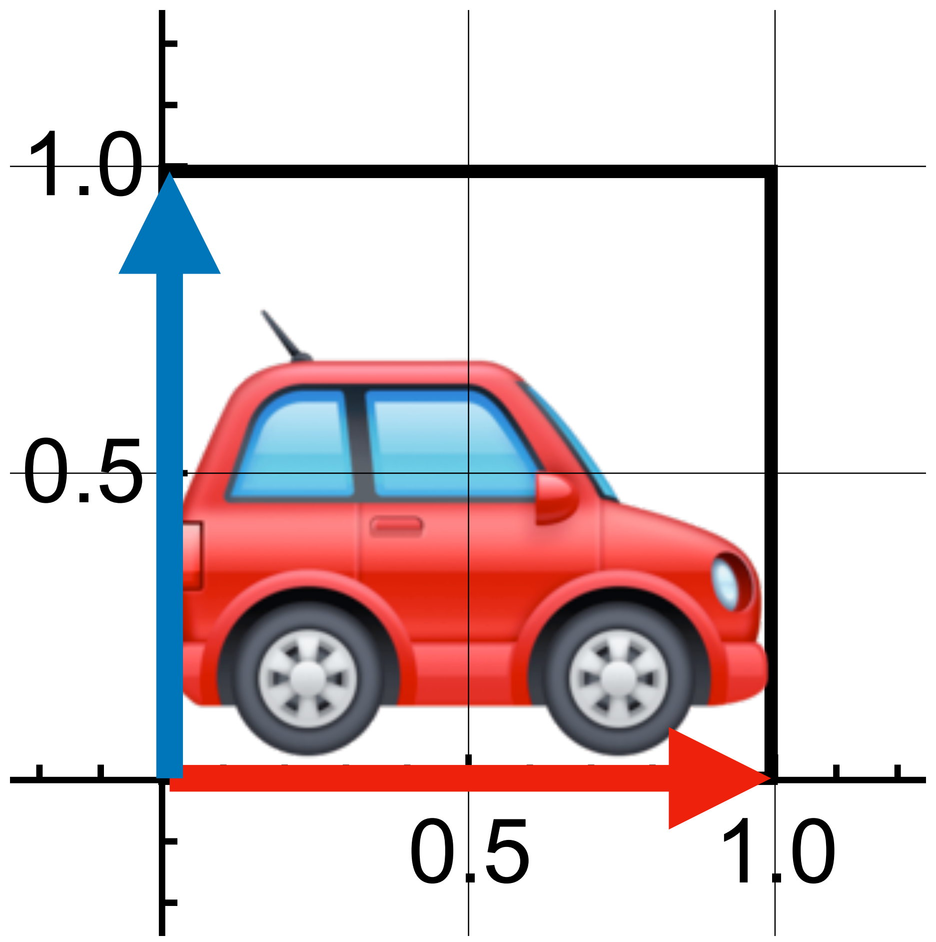
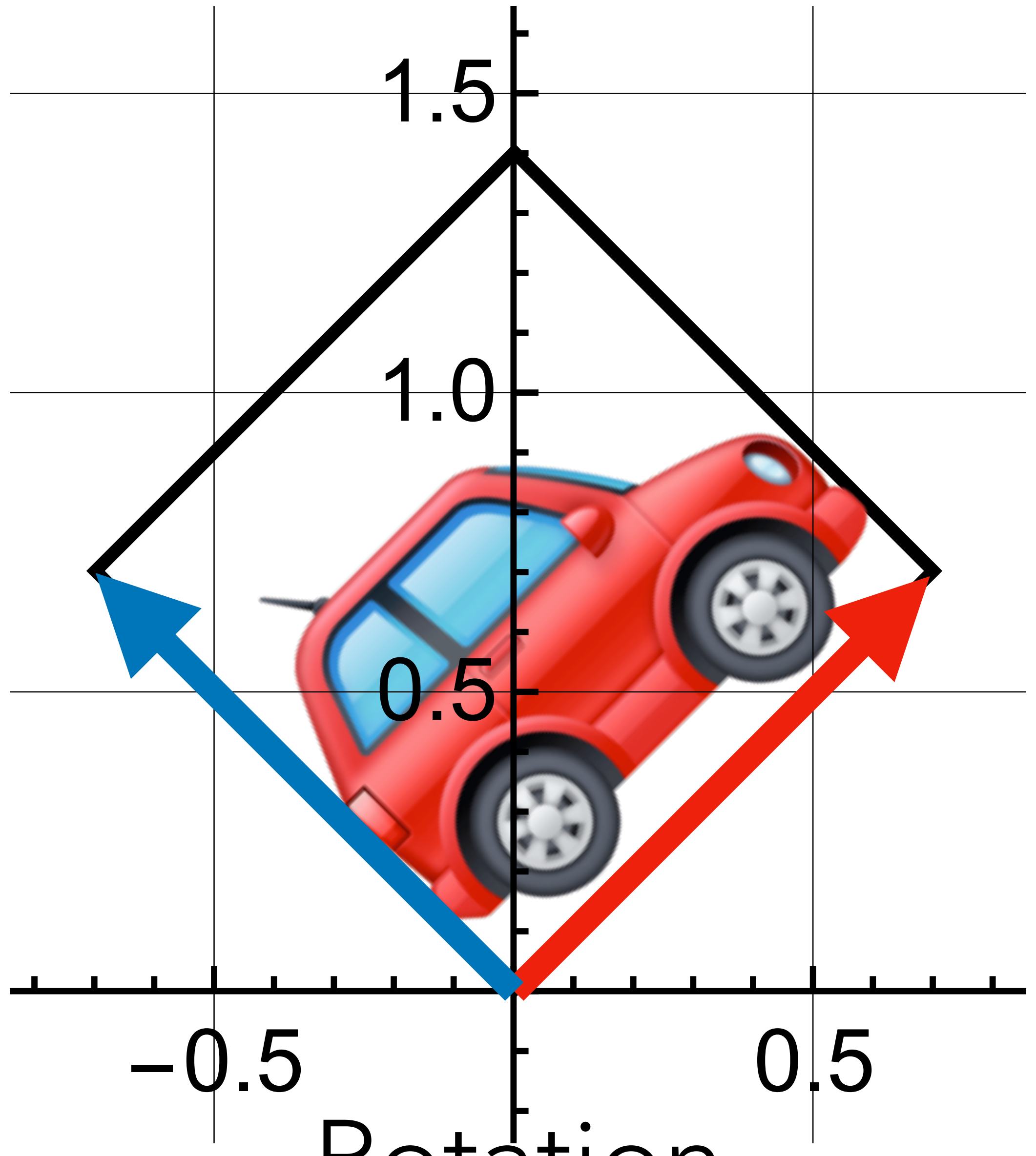
COL781: Computer Graphics

5. Affine Transformations

Blender manual

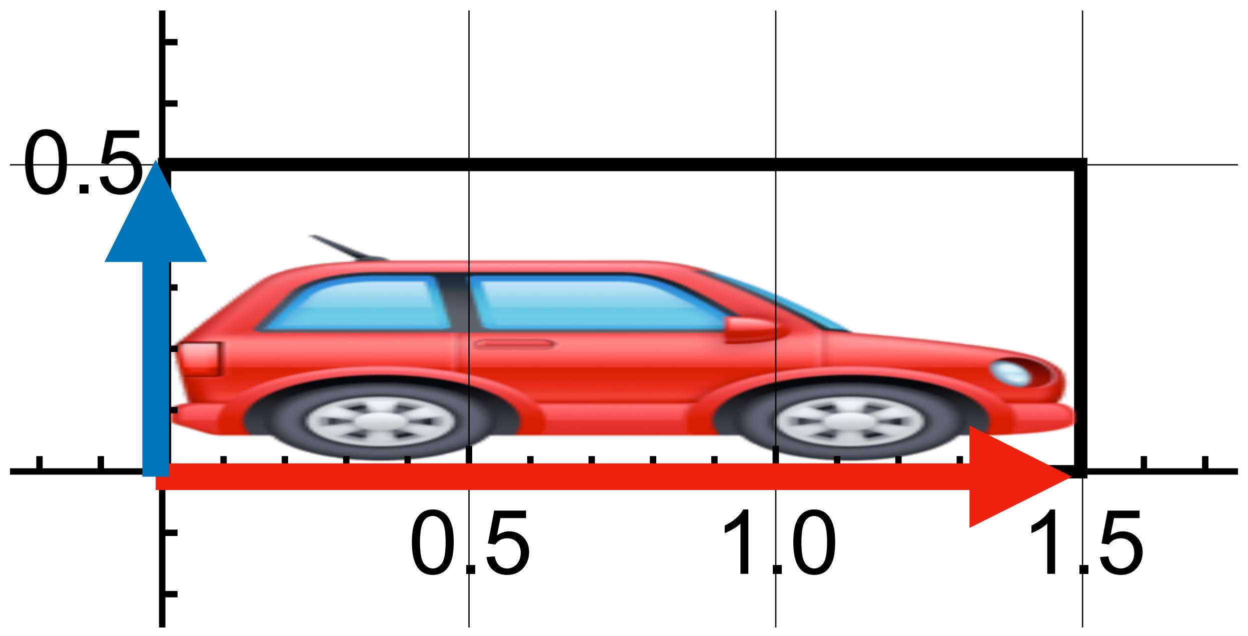
Continuing from last class…





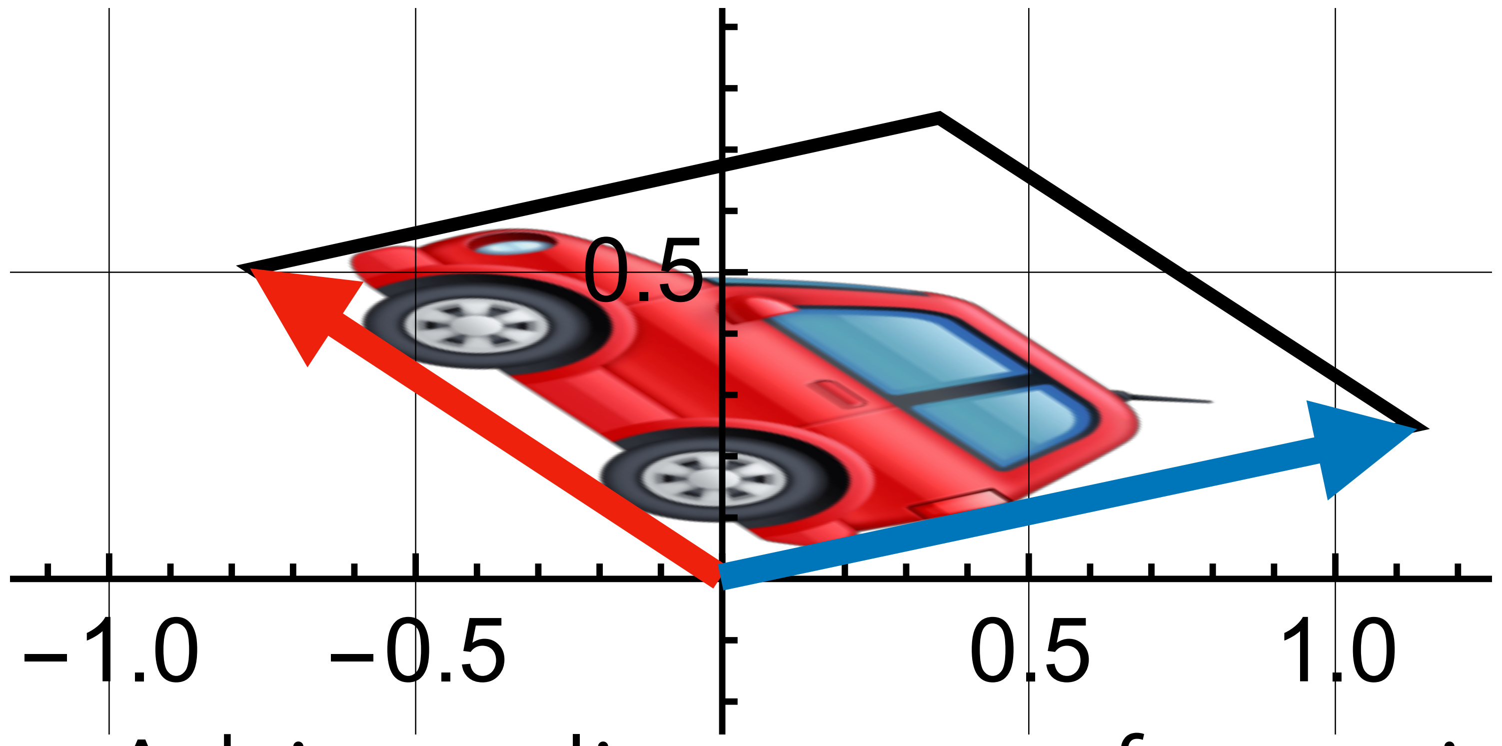
Rotation

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



Nonuniform scaling

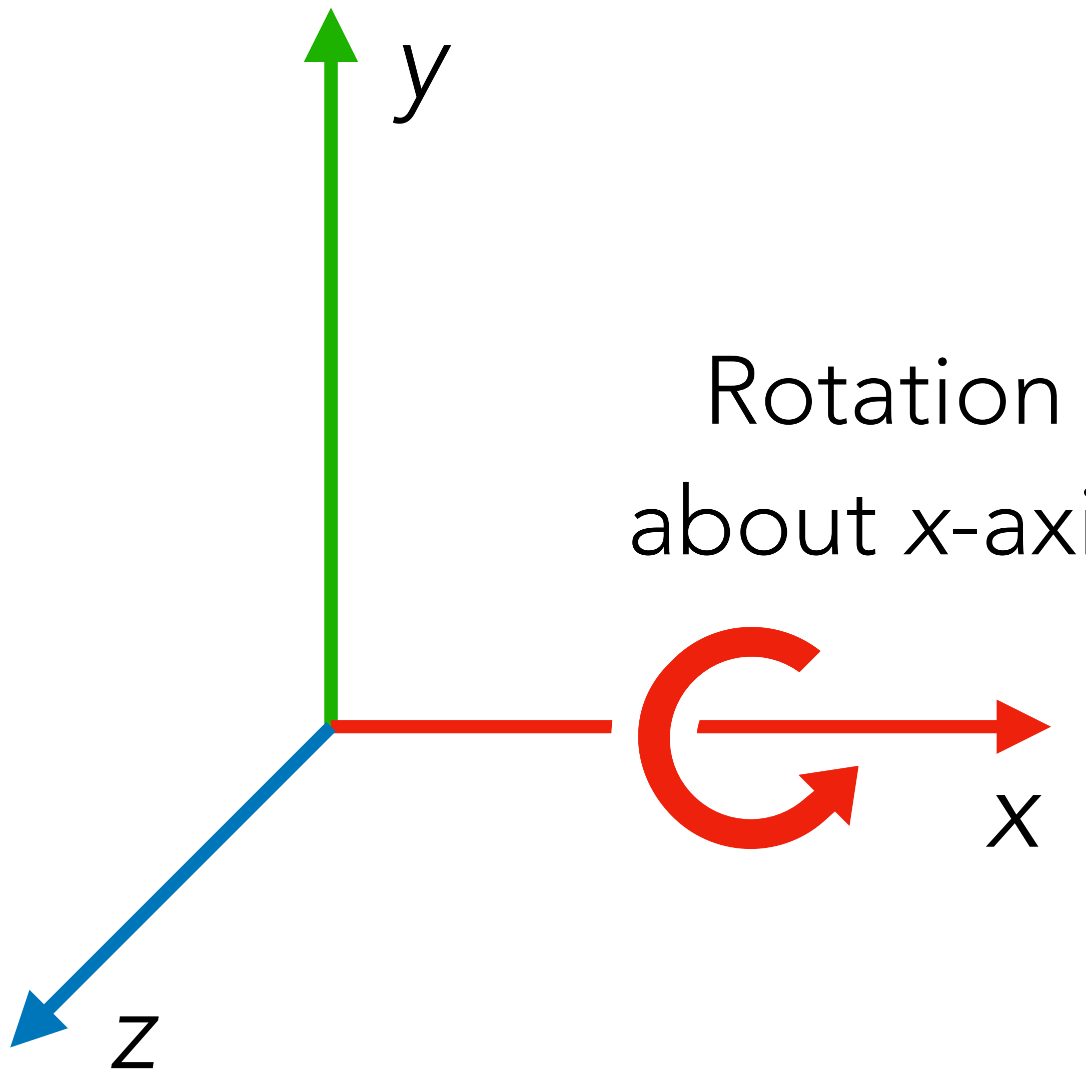
$$\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$



Arbitrary linear transformation

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

Rotations in 3D



Rotations about the coordinate axes:

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

Rotation about *x*-axis
= Rotation in *yz*-plane

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

Rotation about *y*-axis
= Rotation in ***zx***-plane

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

Rotation about *z*-axis
= Rotation in *xy*-plane

Are these all the possible rotations?

Rotations in 3D

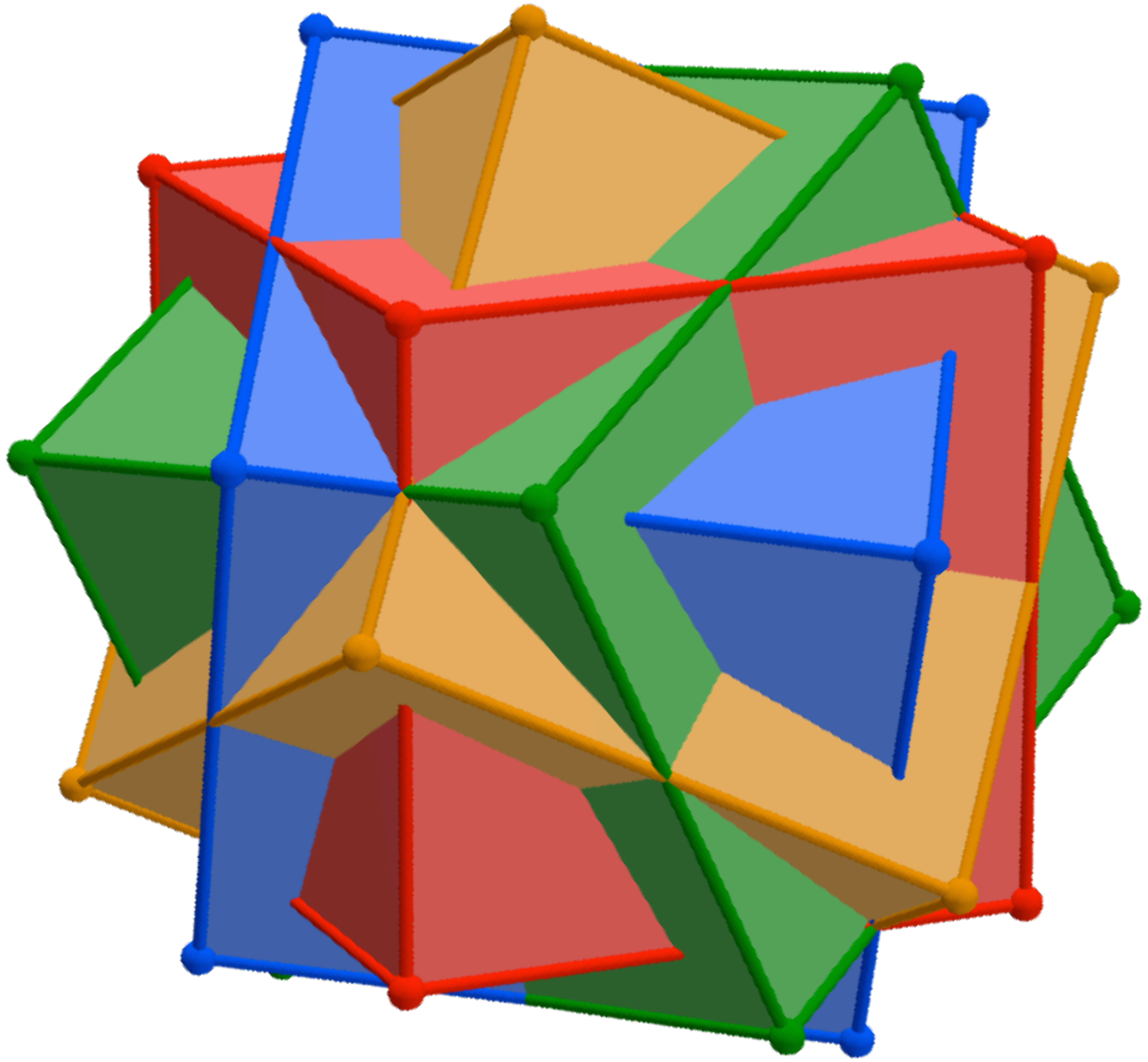
Are these all possible rotations?

Not at all!

A rotation is any transformation which:

- preserves distances and angles
- preserves orientation

Equivalently, $\mathbf{R}^\mathsf{T}\mathbf{R} = \mathbf{I}$, and $\det \mathbf{R} = 1$

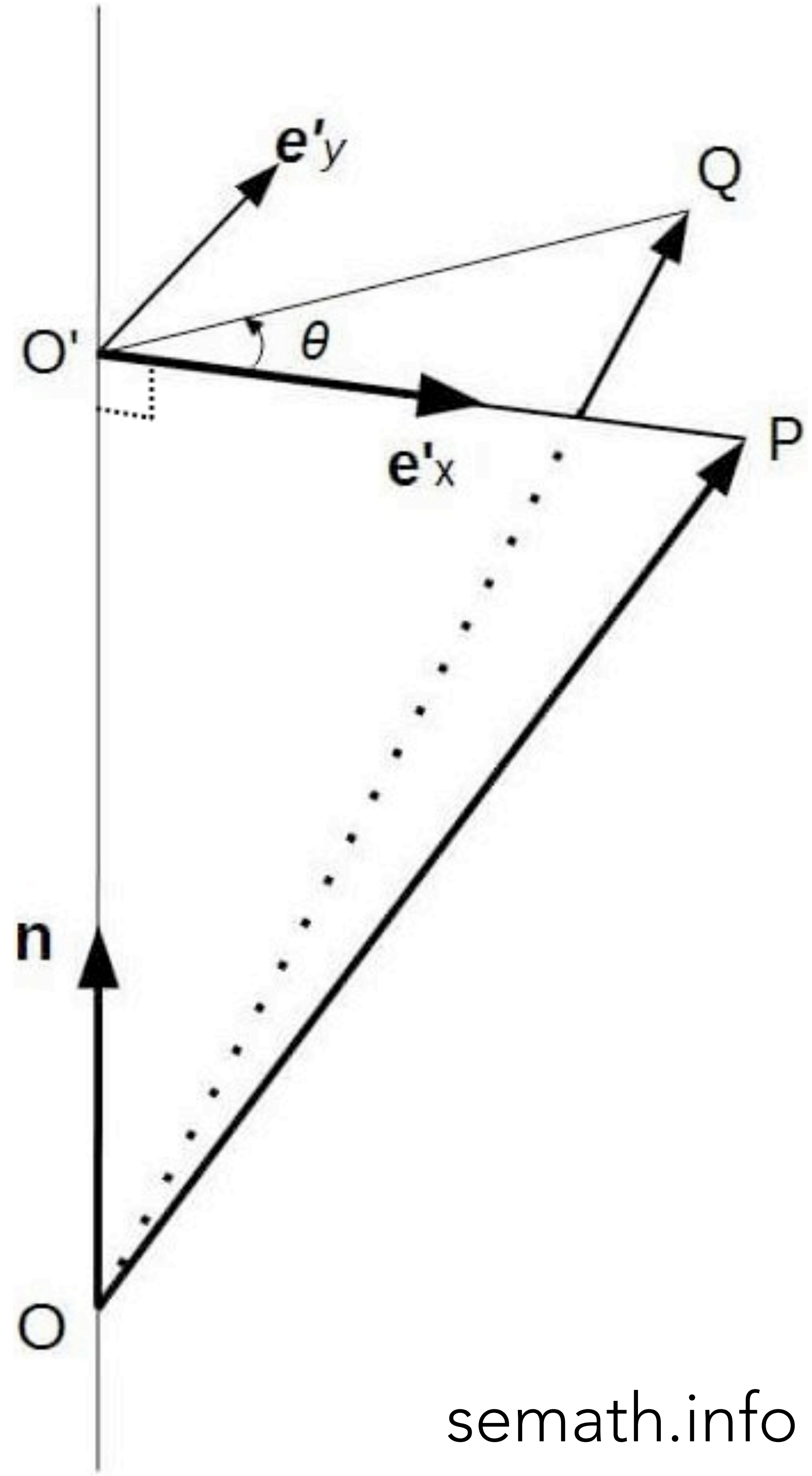


Rodrigues' rotation formula

Rotation around an axis **n** by angle θ:

$$\mathbf{R} = \mathbf{I}\cos\theta + [\mathbf{n}]_\times \sin\theta + \mathbf{n}\,\mathbf{n}^\mathsf{T}(1 - \cos\theta)$$

$$\text{where } [\mathbf{n}]_\times = \begin{bmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{bmatrix}$$



semath.info

How? Hints:

- $[\mathbf{n}]_\times$ is the "cross-product matrix": $[\mathbf{n}]_\times \mathbf{v} = \mathbf{n} \times \mathbf{v}$
- Assume an orthogonal basis **n**, $\mathbf{e}_1$, $\mathbf{e}_2$ and see what **R** does to it

Euler angles

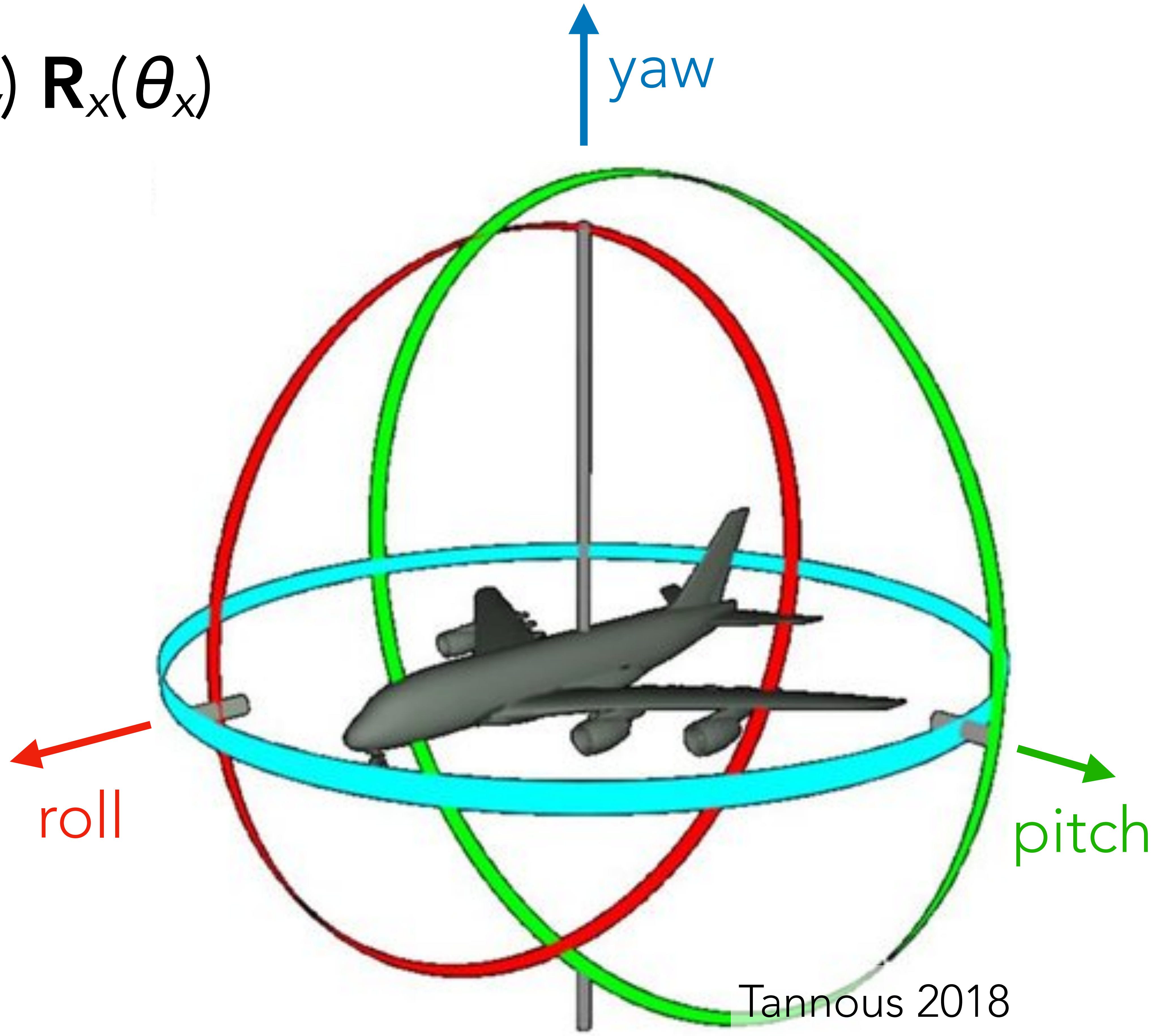
Any 3D rotation can also be expressed using 3 rotations about coordinate axes:

e.g. $\mathbf{R} = \mathbf{R}_z(\theta_z)\ \mathbf{R}_y(\theta_y)\ \mathbf{R}_x(\theta_x)$

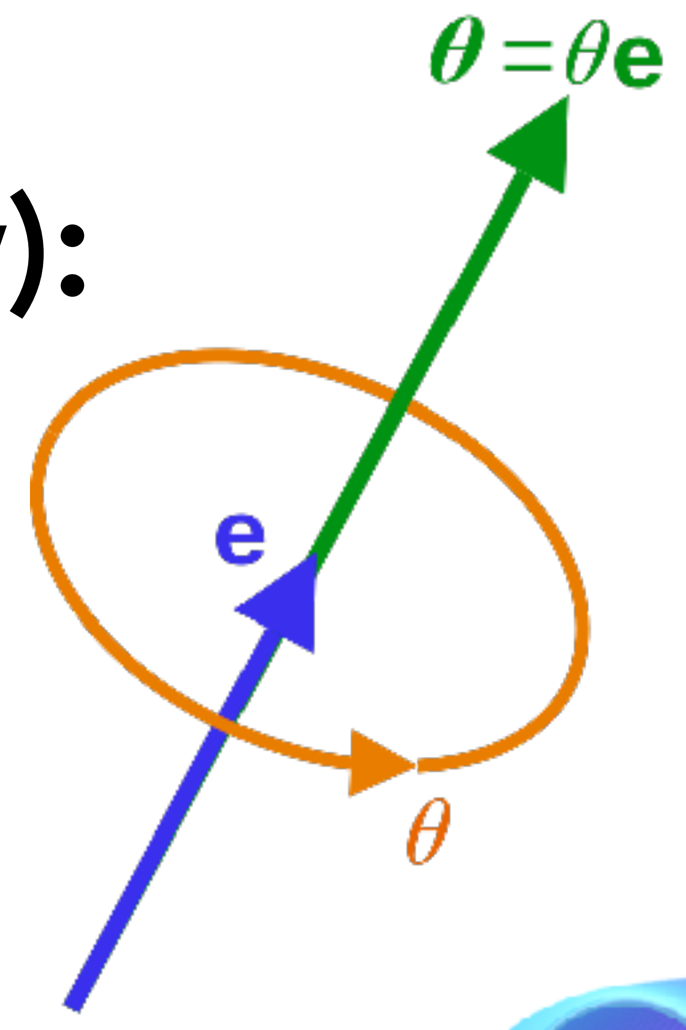
θ_x, θ_y, θ_z are called Euler angles

Also called "roll, pitch, yaw" in aircraft

Note: Order of rotation matters! Need to know which angle for which axis, and also which order to multiply them.

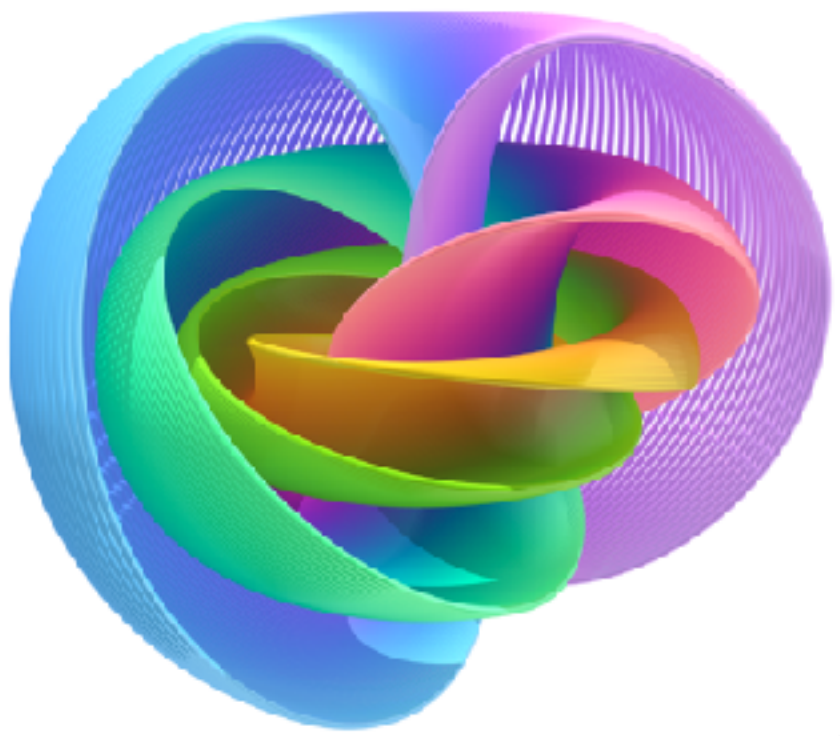


Other rotation representations (not covering now):



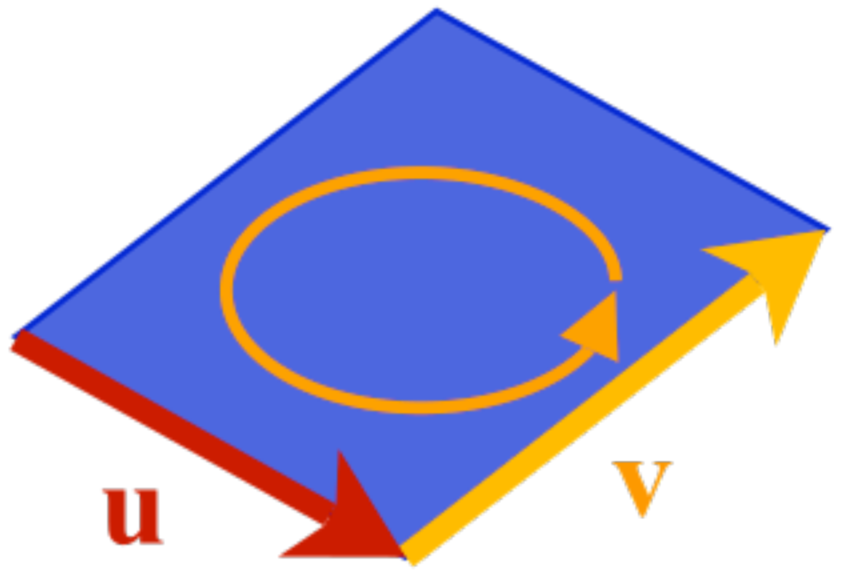
- Angle vector / exponential map

$$\boldsymbol{\theta} = \theta\mathbf{e}$$



- Quaternions

$$\mathbf{q} = s + ix + jy + kz$$

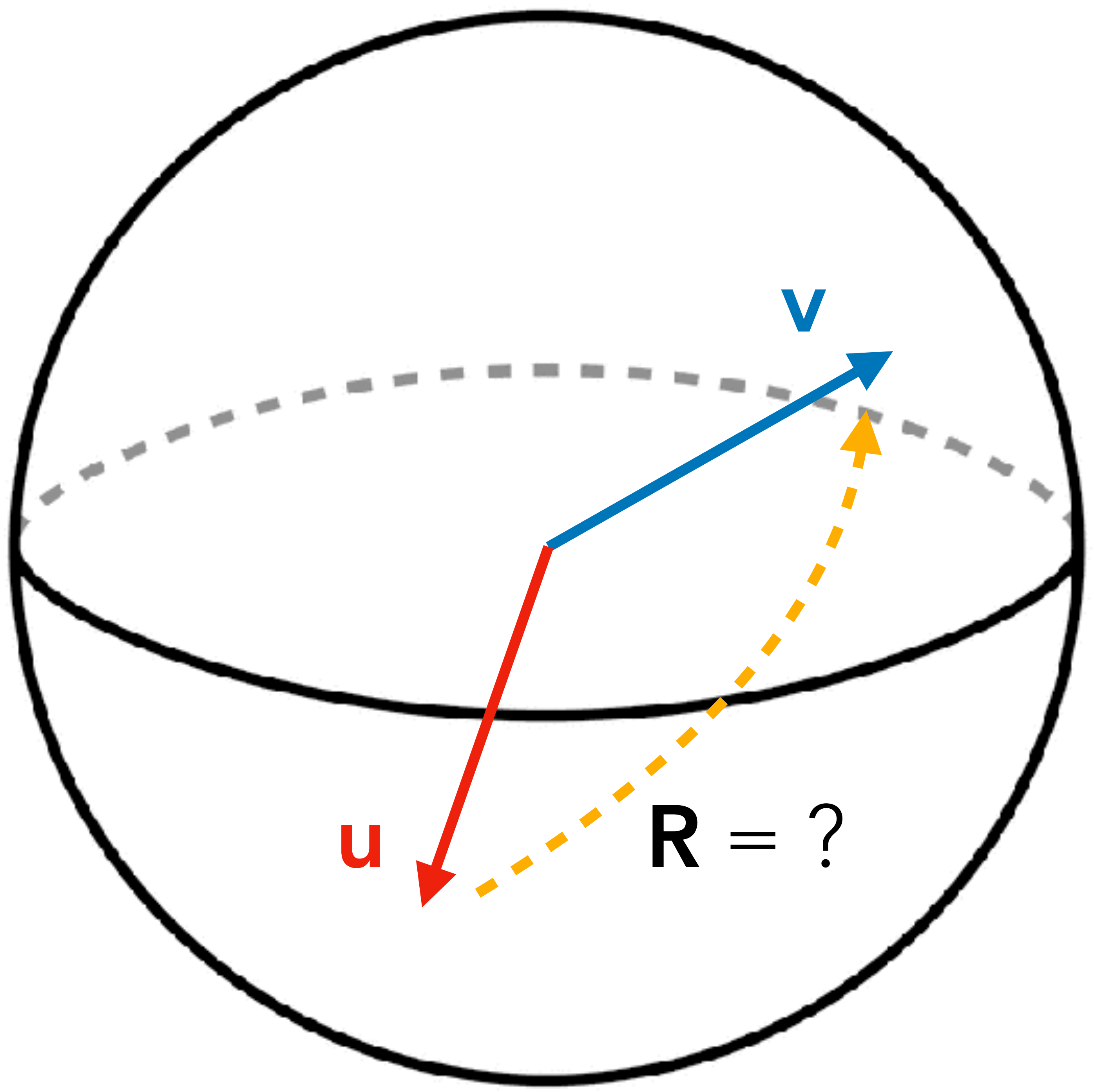


- Rotors

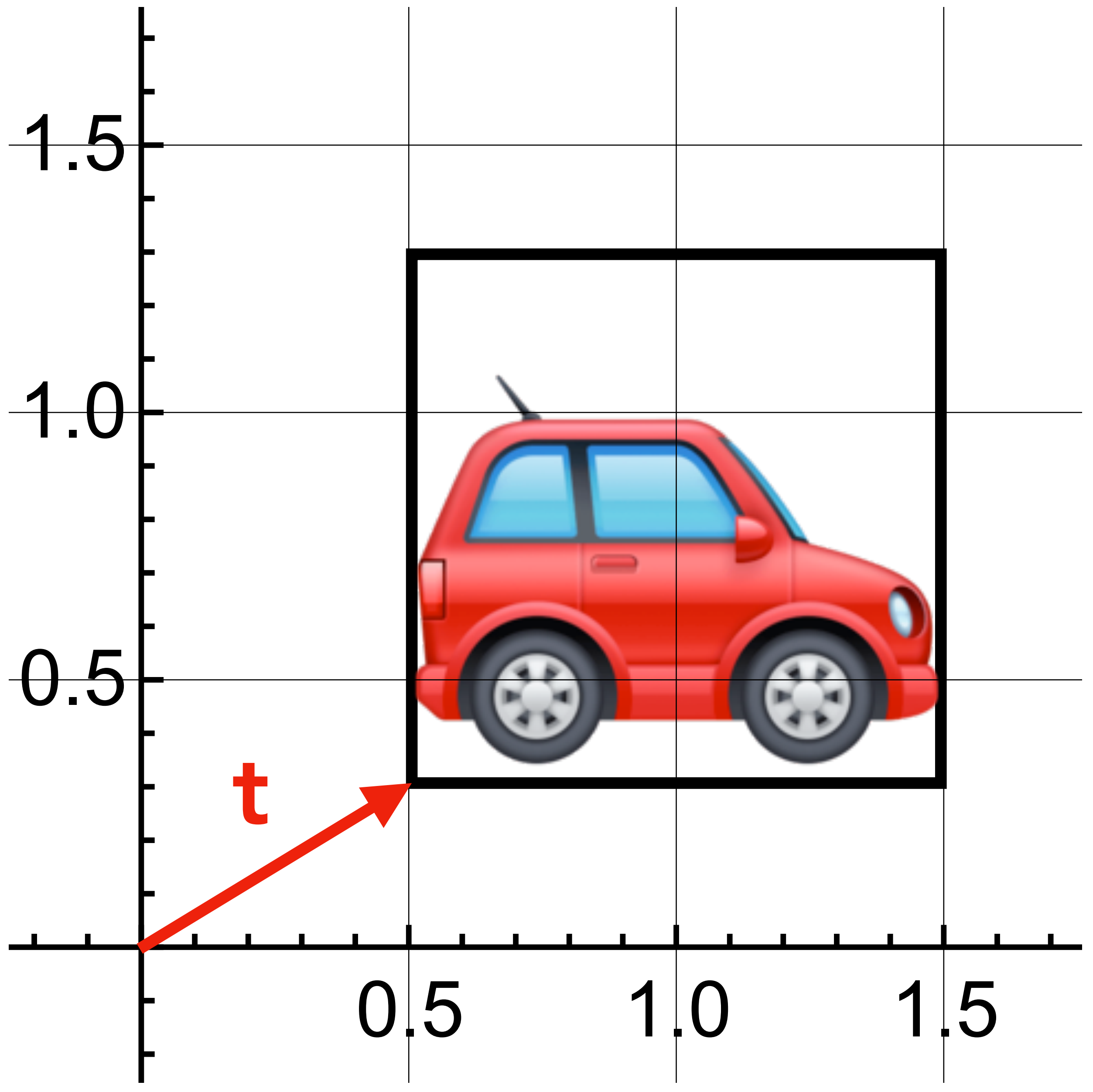
$$\mathbf{uv} = \mathbf{u} \cdot \mathbf{v} + \mathbf{u} \wedge \mathbf{v}$$

Homework exercise

Given unit vectors **u** and **v**, find a way to construct a rotation matrix **R** which maps **u** to **v**, i.e. **Ru** = **v**. Is it unique, or are there many different such rotations?



Translations



Move all points by a constant displacement

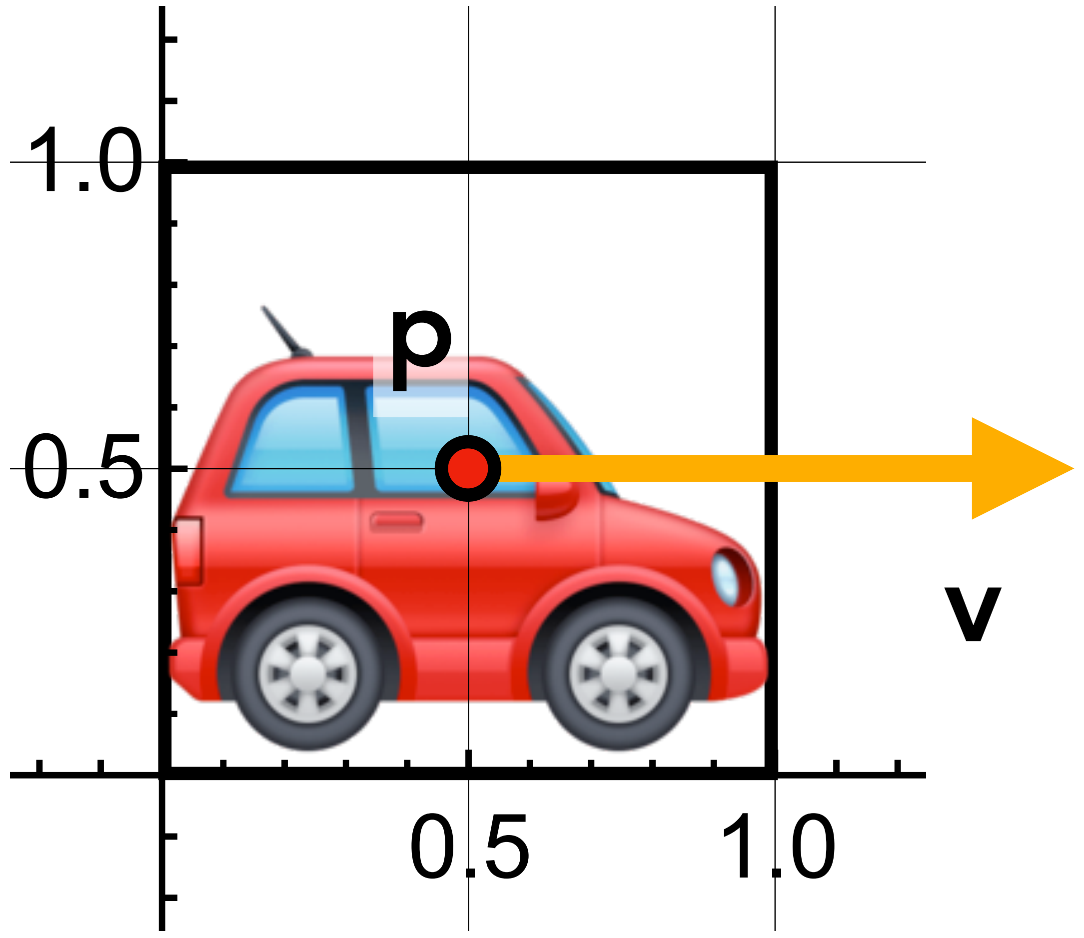
$$T(\mathbf{p}) = \mathbf{p} + \mathbf{t}$$

So a linear transformation followed by a translation will be of the form $T(\mathbf{p}) = \mathbf{A}\mathbf{p} + \mathbf{b}$

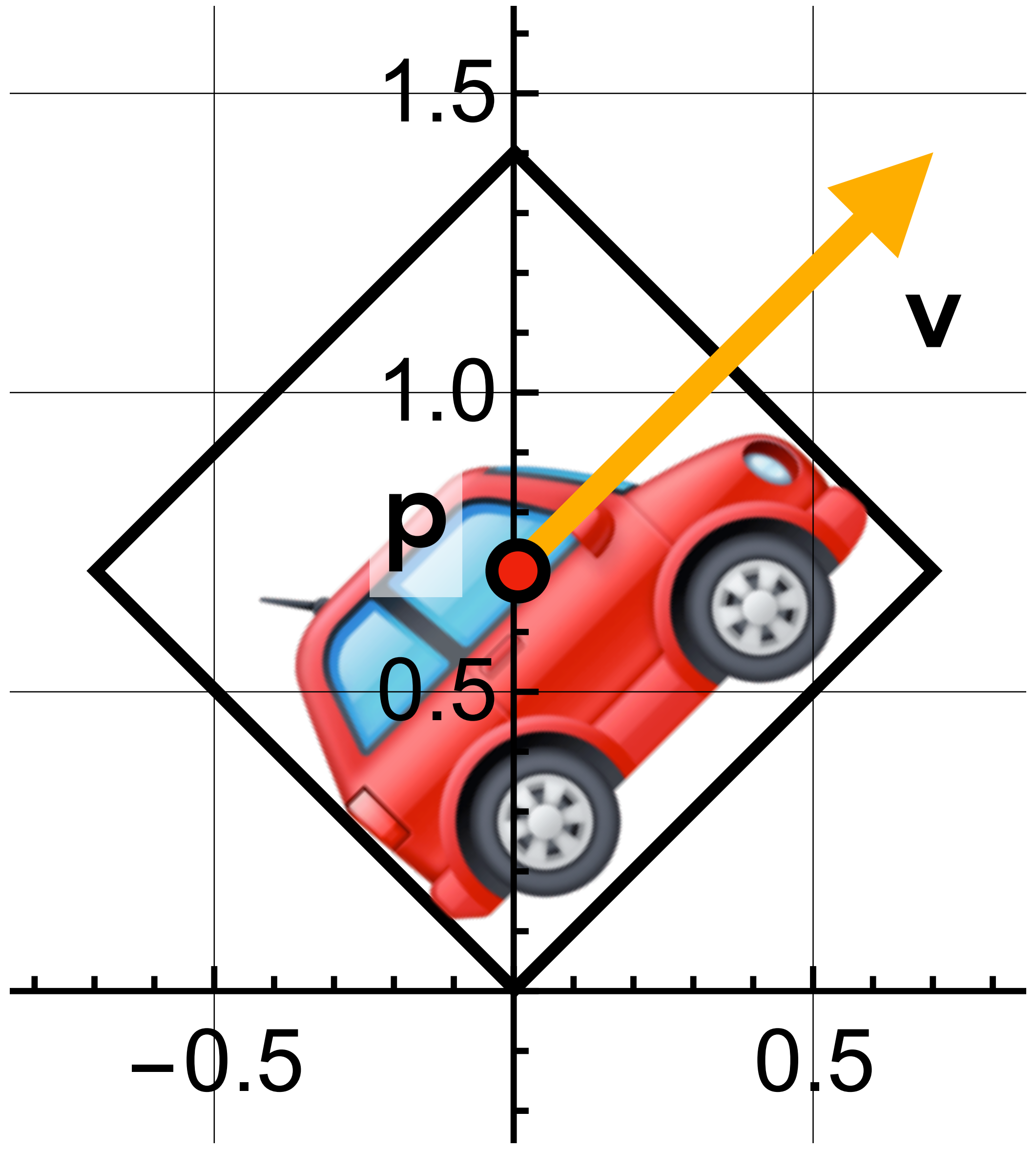
A bit tedious to compose:

$$T_2(T_1(\mathbf{p})) = \mathbf{A}_2(\mathbf{A}_1\mathbf{p} + \mathbf{b}_1) + \mathbf{b}_2 = (\mathbf{A}_2\mathbf{A}_1)\mathbf{p} + (\mathbf{A}_2\mathbf{b}_1 + \mathbf{b}_2)$$

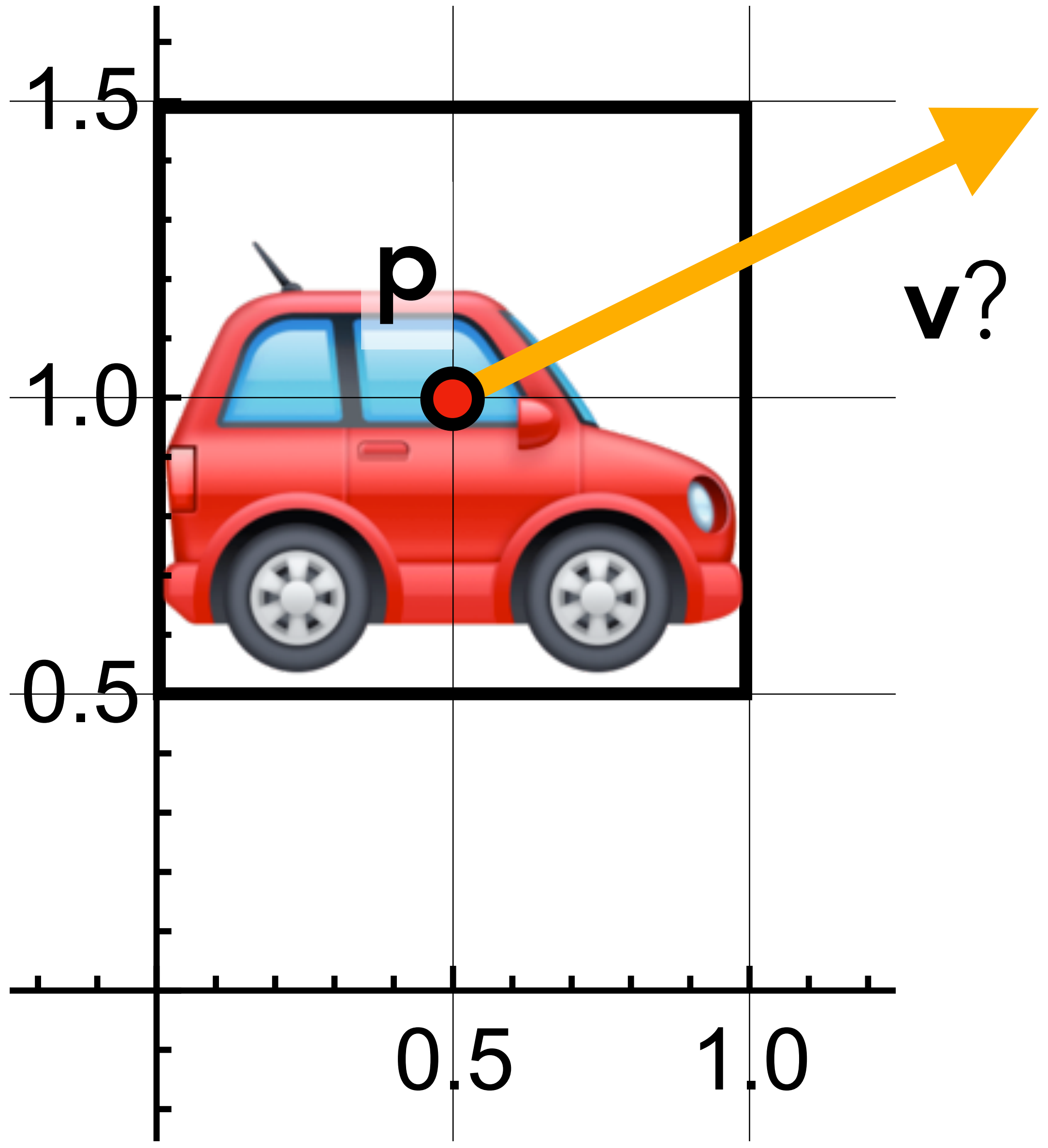
Suppose I have both points and directions/velocities/etc. to transform.



Original:
p = (0.5, 0.5)
v = (1, 0)



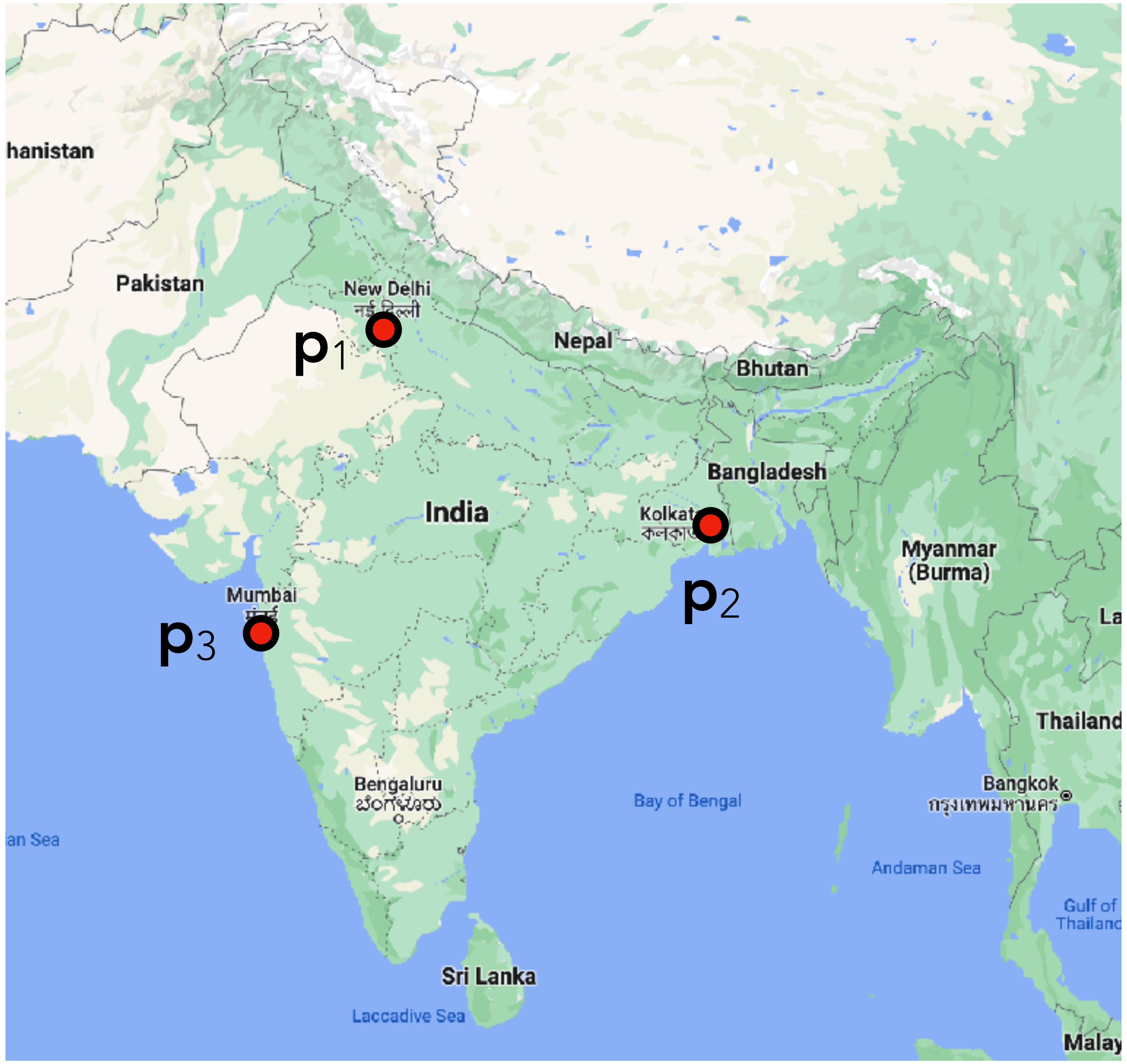
Rotation by 45°:
p = (0, 0.7)
v = (0.7, 0.7)



Translation by (0, 0.5):
p = (0.5, 1)
v = (1, 0.5)?

It seems translation should only affect some things, not others. But why?

Are points really vectors?



$\mathbf{p}_1 + \mathbf{p}_2 = ?$

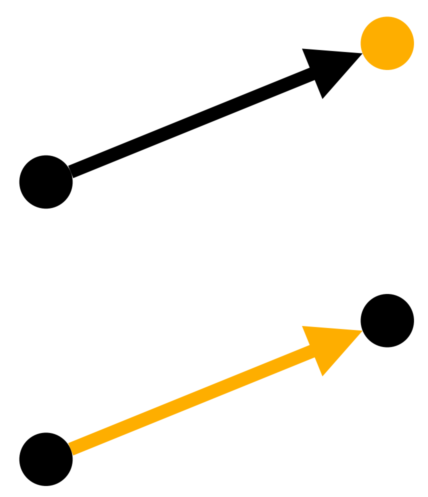
$5\mathbf{p}_3 = ?$

How about I just choose an origin and then add the displacement vectors?

Points vs. vectors

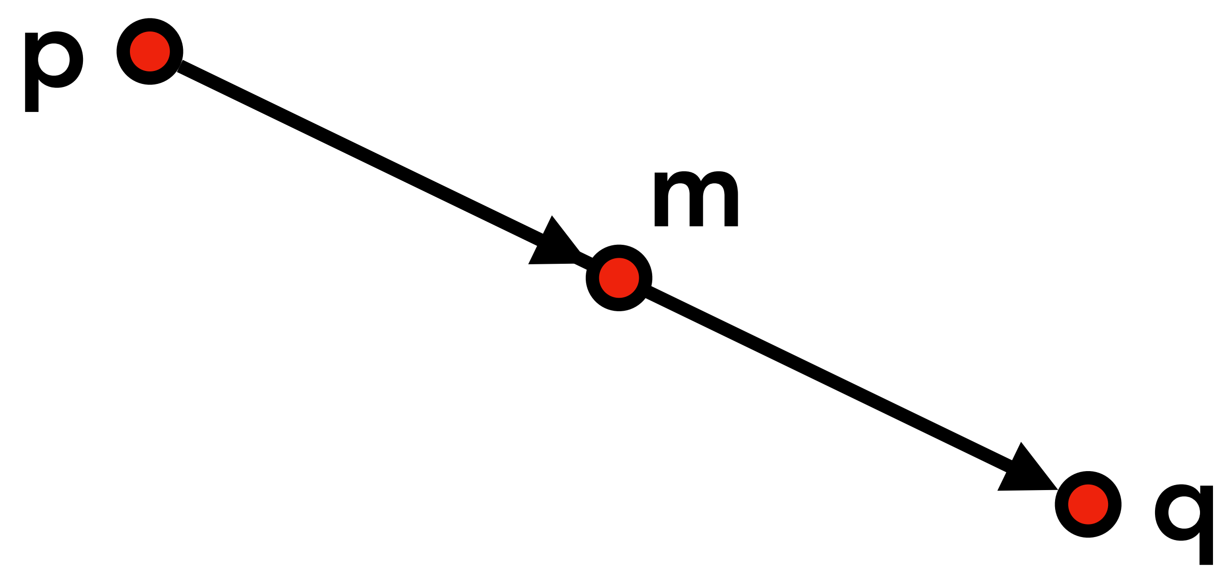
Points form an **affine space** A over the vector space V.

- Point-vector addition: $A \times V \to A$
- Point subtraction: $A \times A \to V$



with the obvious properties e.g. $(\mathbf{p} + \mathbf{u}) + \mathbf{v} = \mathbf{p} + (\mathbf{u} + \mathbf{v})$, $\mathbf{p} + (\mathbf{q} - \mathbf{p}) = \mathbf{q}$, etc.

Example: midpoint of two points $\mathbf{p}$ and $\mathbf{q}$



$$\mathbf{m} = \tfrac{1}{2}(\mathbf{p} + \mathbf{q})?$$

Not allowed! But can rewrite as

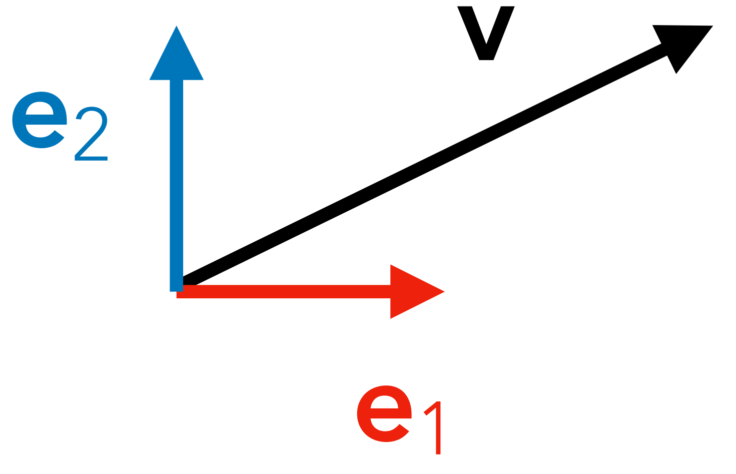
$$\mathbf{m} = \mathbf{p} + \tfrac{1}{2}(\mathbf{q} - \mathbf{p}) = \mathbf{q} + \tfrac{1}{2}(\mathbf{p} - \mathbf{q})$$

In fact it's valid to take any **affine combination** $w_1\mathbf{p}_1 + w_2\mathbf{p}_2 + \cdots + w_n\mathbf{p}_n$ as long as $w_1 + w_2 + \cdots + w_n = 1$. So we will allow this too.

(**Exercise:** Check that this can be done using only the affine space operations)

Coordinate frames

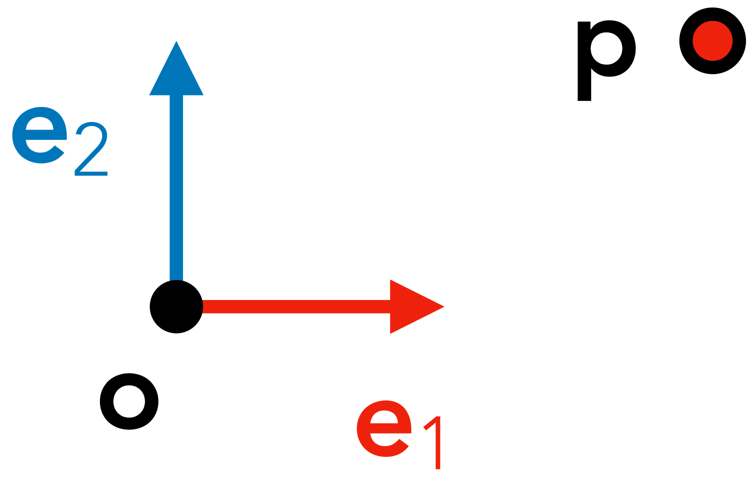
To specify a vector numerically, we need a basis



$$\mathbf{v} = v_1\mathbf{e}_1 + v_2\mathbf{e}_2 + \cdots \quad \Leftrightarrow \quad \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \end{bmatrix} \text{ in the basis}$$

To specify a point numerically, we need a **coordinate frame**: origin and basis

$$\mathbf{p} = p_1\mathbf{e}_1 + p_2\mathbf{e}_2 + \cdots + \mathbf{o} \quad \text{so maybe} \quad \mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ 1 \end{bmatrix} ?$$



Write a point as an $(n+1)$-tuple $\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ 1 \end{bmatrix}$ to mean $\mathbf{p} = p_1\mathbf{e}_1 + p_2\mathbf{e}_2 + \cdots + \mathbf{o}$.

Linear transformations are now $\begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}$, mapping $\mathbf{e}_i \to \mathbf{A}\mathbf{e}_i$ and $\mathbf{o} \to \mathbf{o}$

e.g. $\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x p_x \\ s_y p_y \\ 1 \end{bmatrix}$

Translation by a vector $\mathbf{t}$: $\begin{bmatrix} \mathbf{I} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$, mapping $\mathbf{e}_i \to \mathbf{e}_i$ but $\mathbf{o} \to \mathbf{o} + \mathbf{t}$

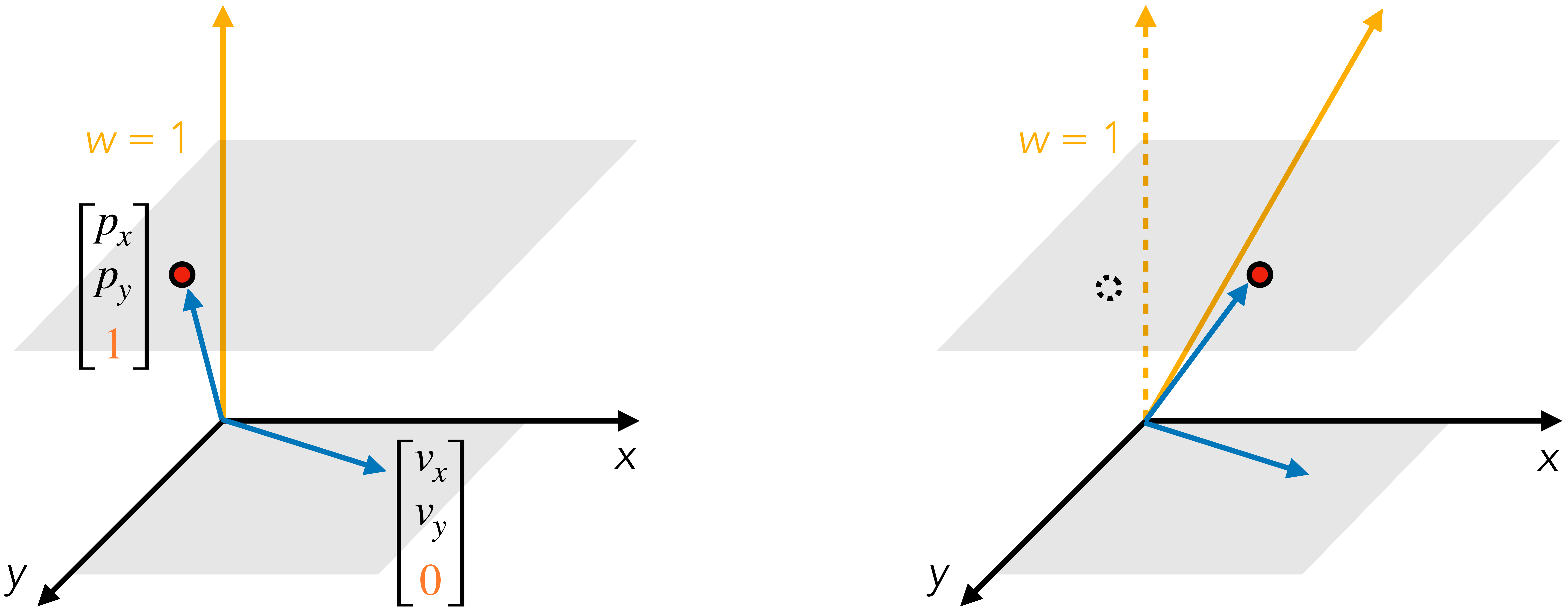
$$\text{e.g.} \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} = \begin{bmatrix} p_x + t_x \\ p_y + t_y \\ 1 \end{bmatrix}$$

What about vectors?

$$\mathbf{v} = v_1\mathbf{e}_1 + v_2\mathbf{e}_2 + \cdots + 0\mathbf{o} \quad \Leftrightarrow \quad \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ 0 \end{bmatrix}$$

Apply a translation:

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ 0 \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ 0 \end{bmatrix}$$



w = 1
$\begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix}$
$\begin{bmatrix} v_x \\ v_y \\ 0 \end{bmatrix}$
x
y
w = 1
x
y

Homogeneous coordinates

Add an extra coordinate *w* at the end.

- Points: $w = 1$
- Vectors: $w = 0$

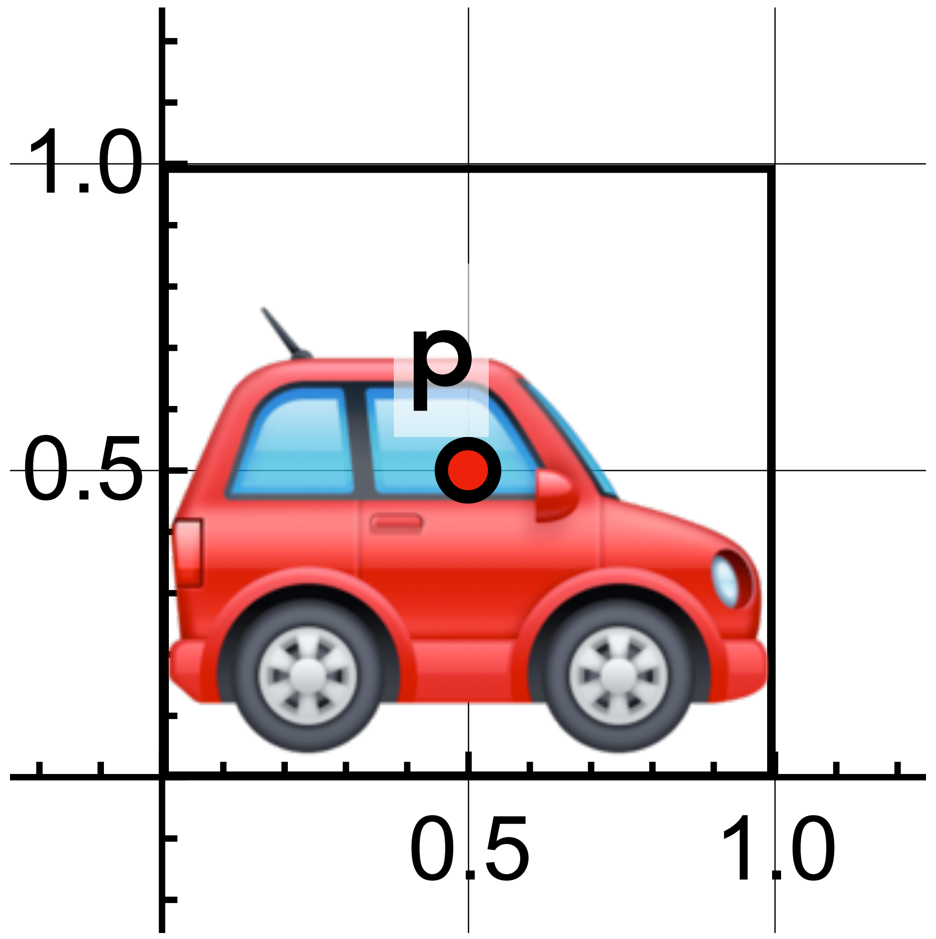
Transformations become $(n+1)\times(n+1)$ matrices

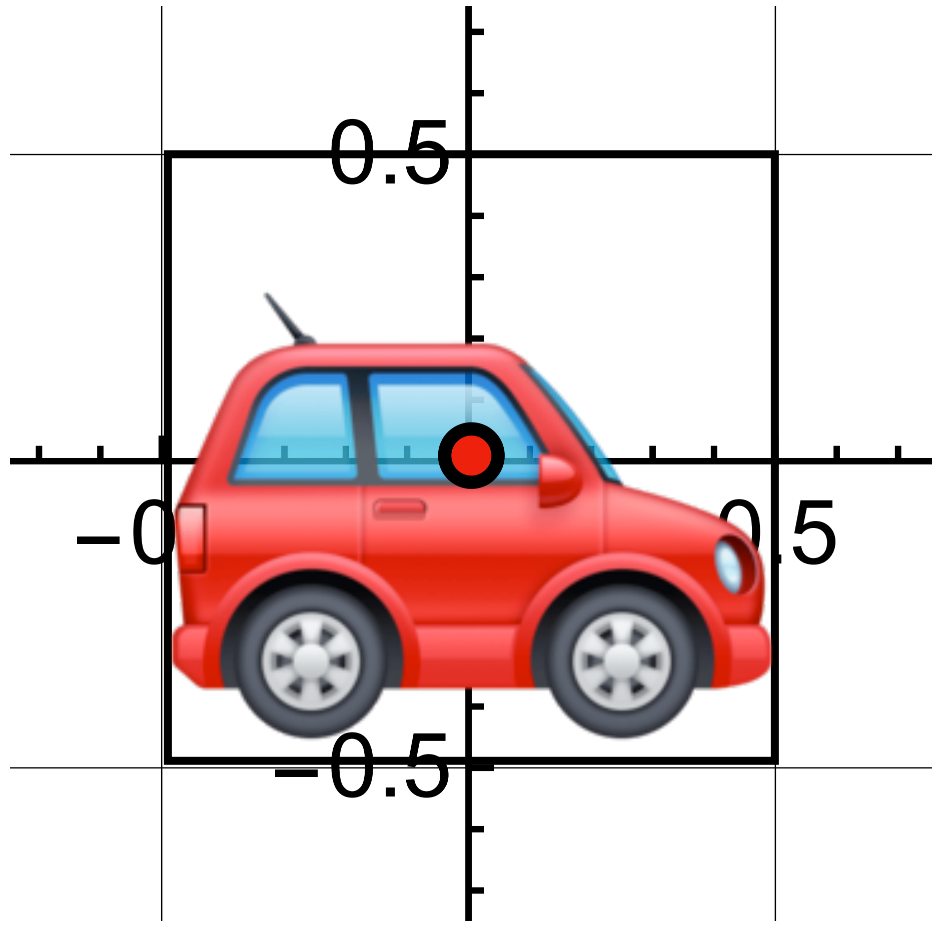
- Linear transformations: $\begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}$
- Translations: $\begin{bmatrix} \mathbf{I} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$

General **affine transformation**: $\begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$

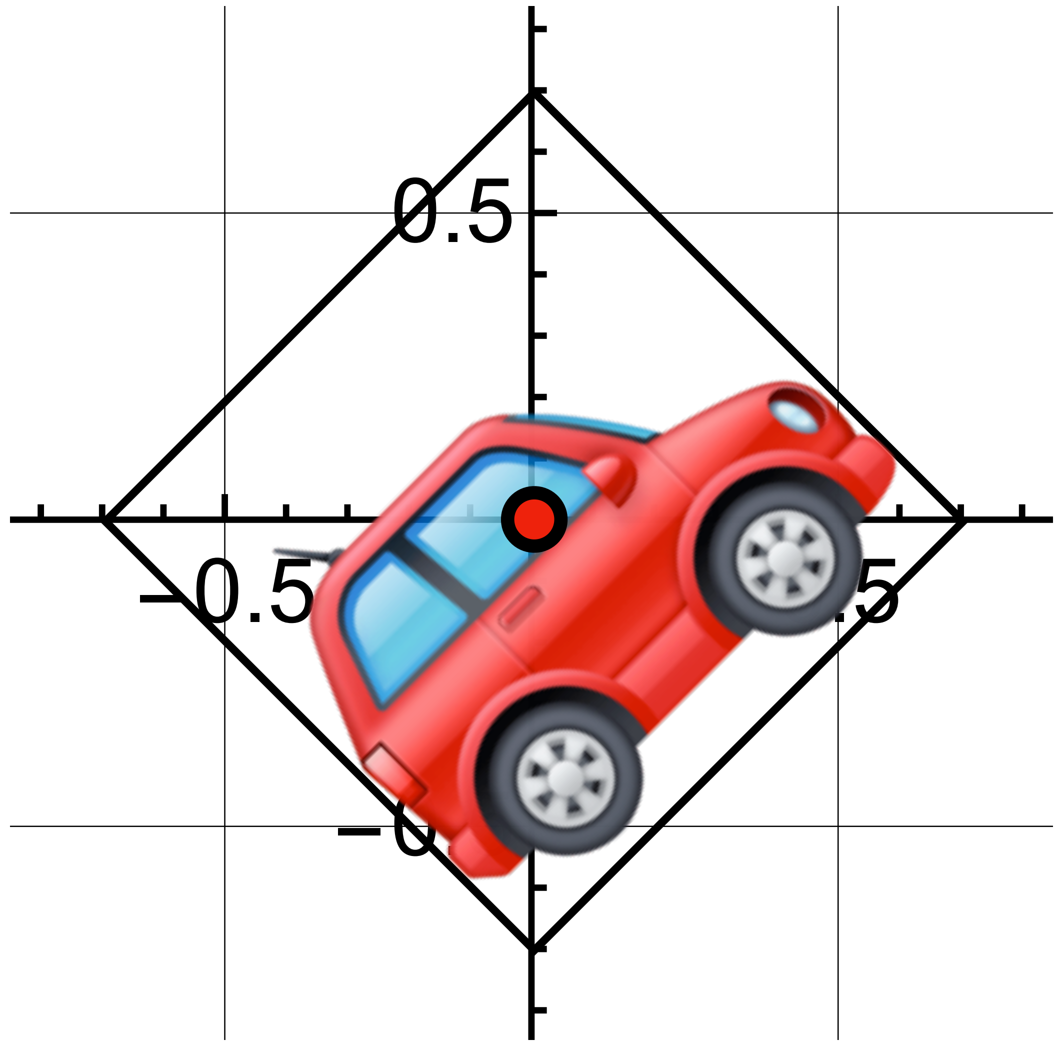
- Corresponds to linearly transforming basis vectors $\mathbf{e}_i \rightarrow \mathbf{A}\mathbf{e}_i$ and translating origin $\mathbf{o} \rightarrow \mathbf{o} + \mathbf{t}$
- Maps parallel lines to parallel lines, but does not preserve the origin
- Composition: just matrix multiplication again.

Example: Rotate by given angle θ about given point $\mathbf{p}$ (instead of about origin)

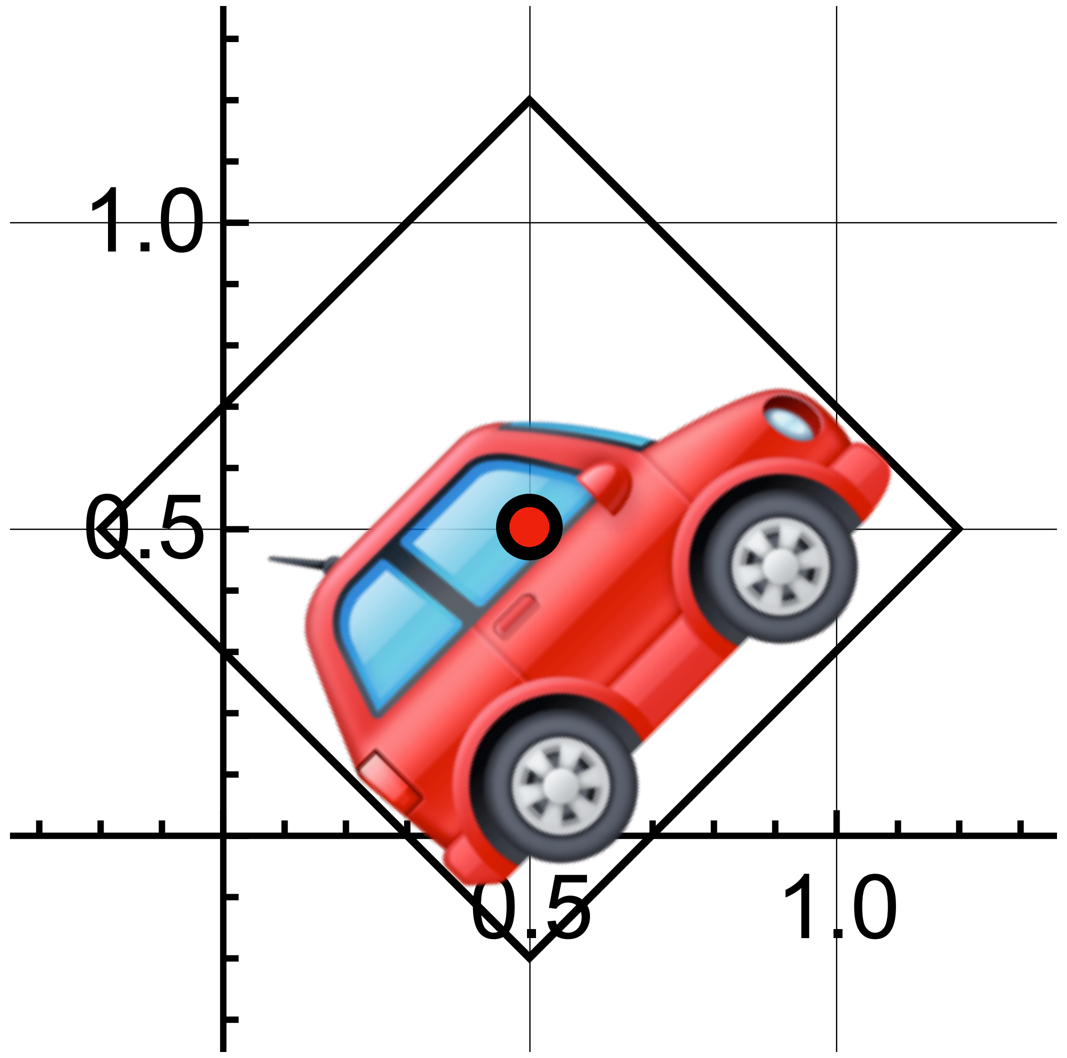




Translate by $-\mathbf{p}$



Rotate by θ about origin



Translate by $\mathbf{p}$

$$\mathbf{M} = \mathbf{T}(\mathbf{p})\ \mathbf{R}(\theta)\ \mathbf{T}(-\mathbf{p})$$

Given coordinates of $\mathbf{p}$ in frame 1, what are its coordinates in frame 2?

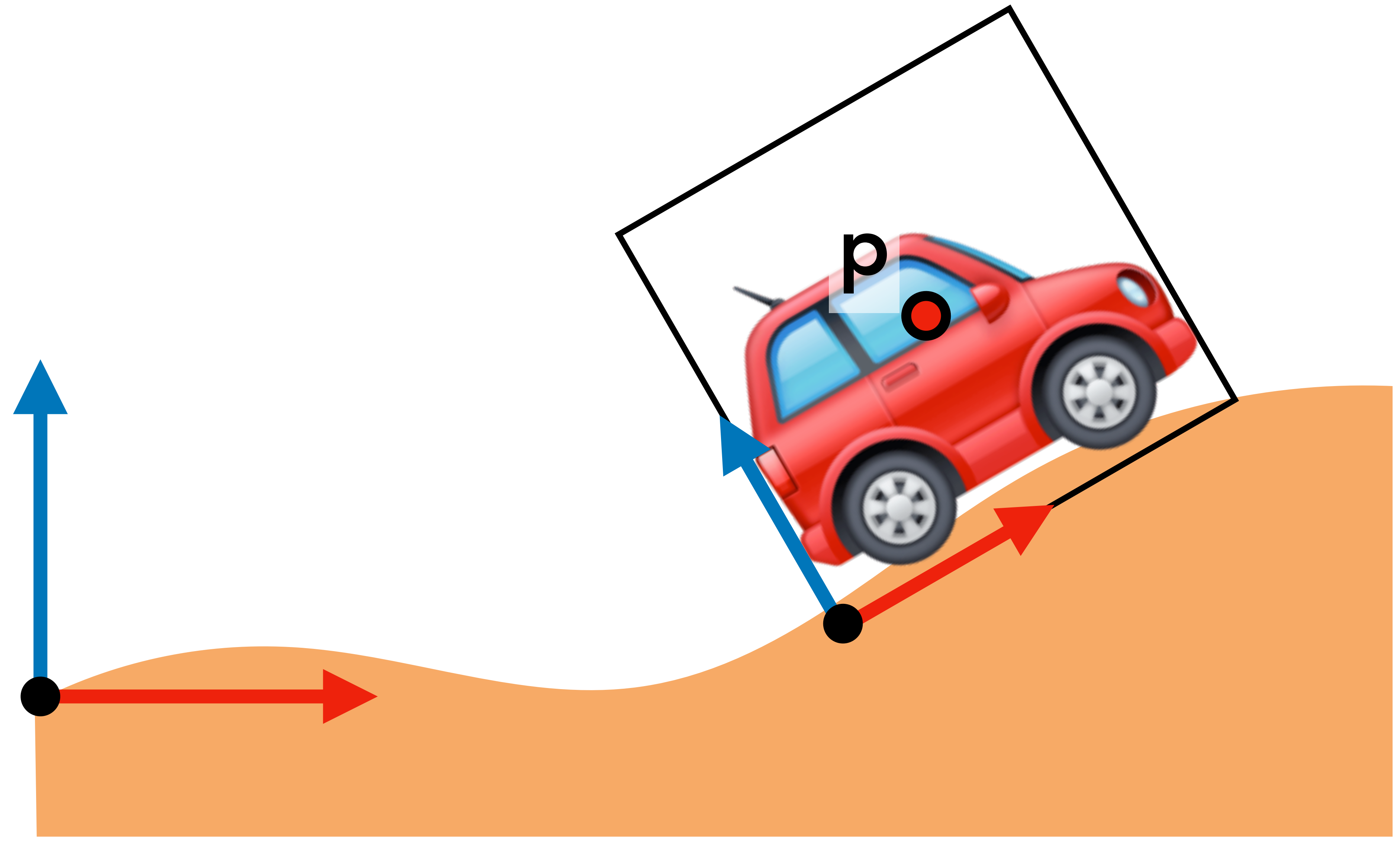
$$\mathbf{p} = p_1\mathbf{e}_1 + p_2\mathbf{e}_2 + \cdots + \mathbf{o}$$

Write coords of $\mathbf{e}_1$, $\mathbf{e}_2$, … and $\mathbf{o}$ in frame 2:

$$\mathbf{e}_i = \begin{bmatrix} \bullet \\ \bullet \\ \vdots \\ 0 \end{bmatrix}, \quad \mathbf{o} = \begin{bmatrix} \bullet \\ \bullet \\ \vdots \\ 1 \end{bmatrix}$$

Then $\mathbf{p} =$

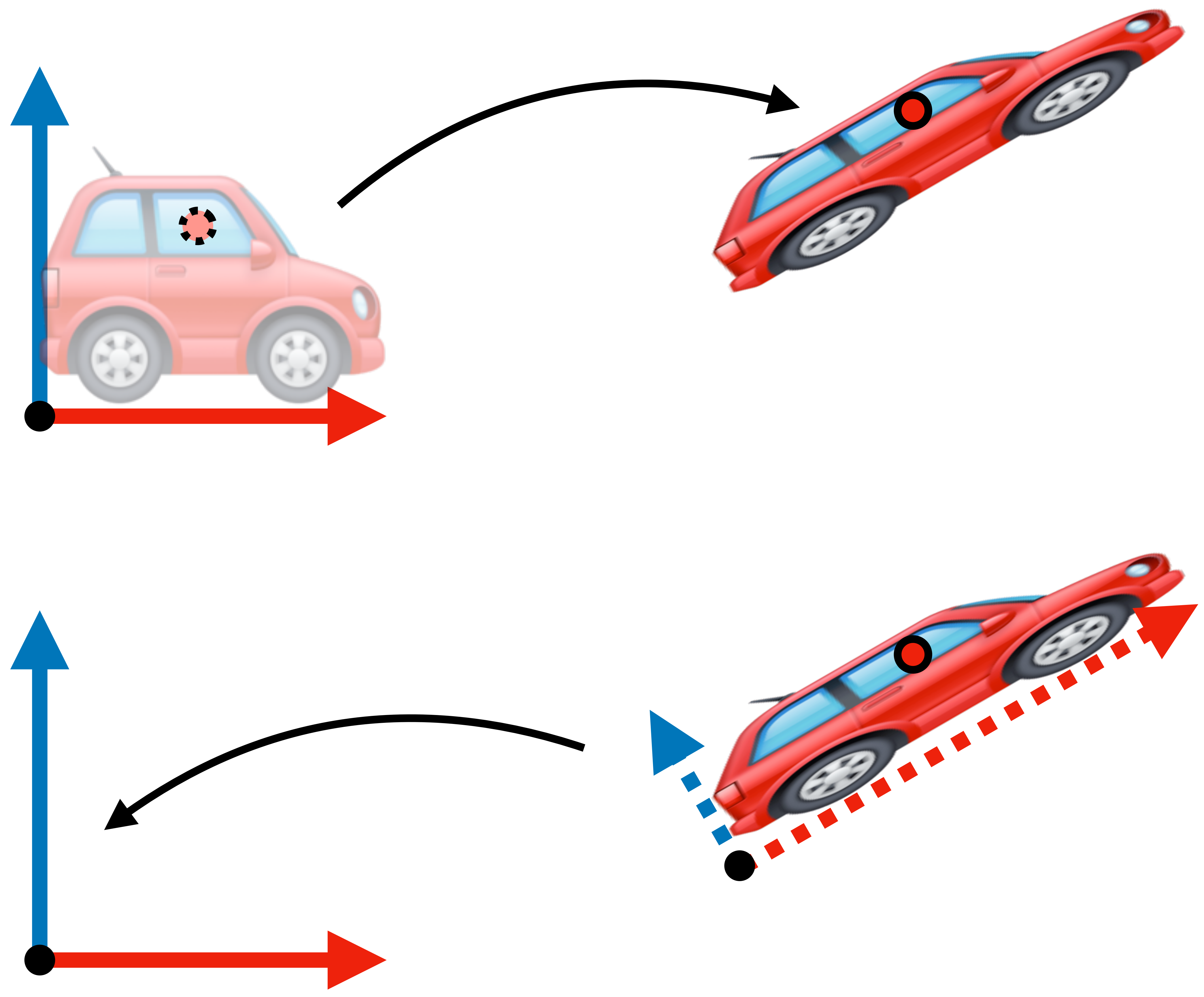
$$\begin{array}{c} \begin{bmatrix} \bullet & \bullet & \cdots & \bullet \\ \bullet & \bullet & \cdots & \bullet \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \\ \begin{array}{cccc} \mathbf{e}_1 & \mathbf{e}_2 & \quad & \mathbf{o} \end{array} \end{array} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ 1 \end{bmatrix}$$



Change of coordinates looks exactly like a transformation matrix!

Active transformation: Moves points to new locations in the same frame

Change of coordinates (passive transformation): Gives coordinates of the same point in a different frame



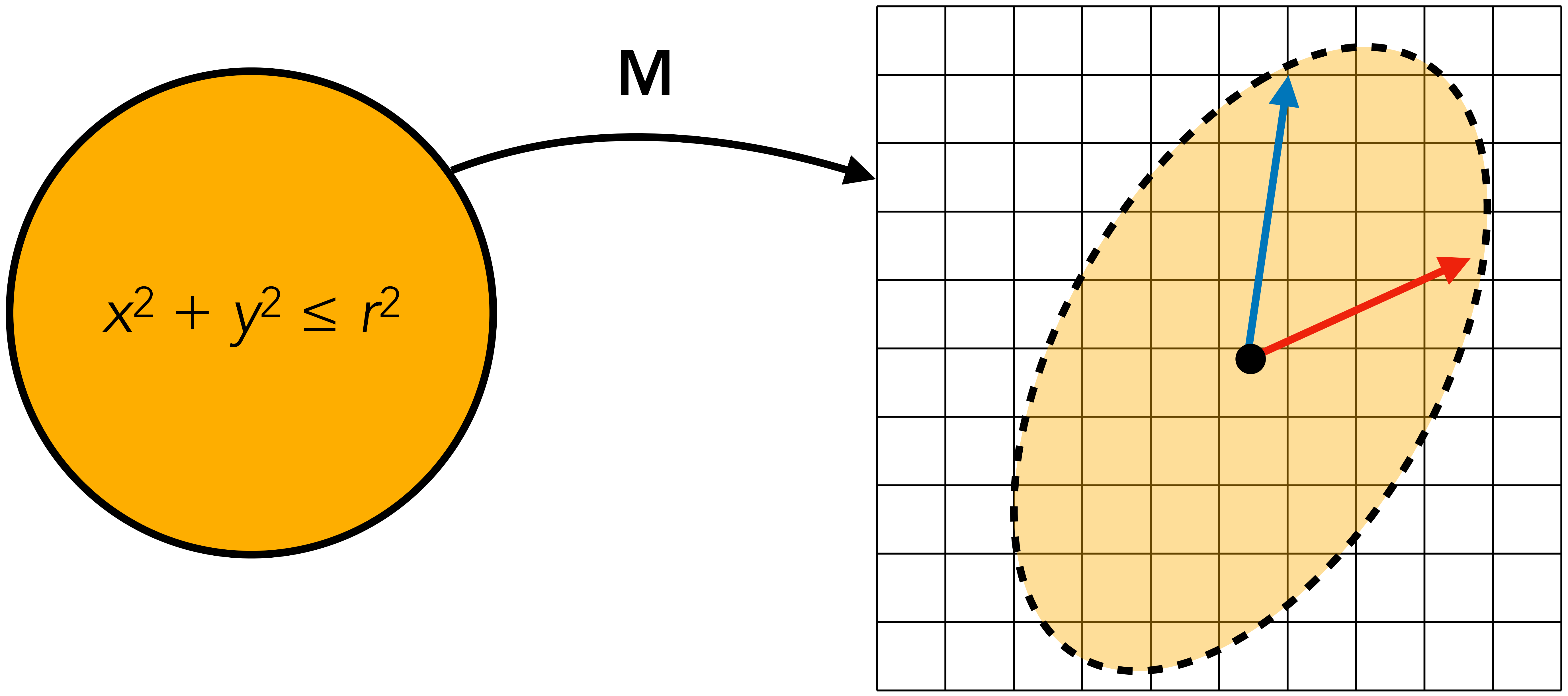
Matrices are the same but the meaning is different! You have to keep track.

e.g. `world_driver = world_from_car * car_driver`

`Vec3` `Mat3x3` `Vec3`

Puzzle:

- To draw a transformed polygon, I can just transform the vertices.
- To draw a shape specified by a function $f(x,y) \leq 0$, I can just test each pixel (x,y).



How can I draw a transformed version of a shape specified by a function?