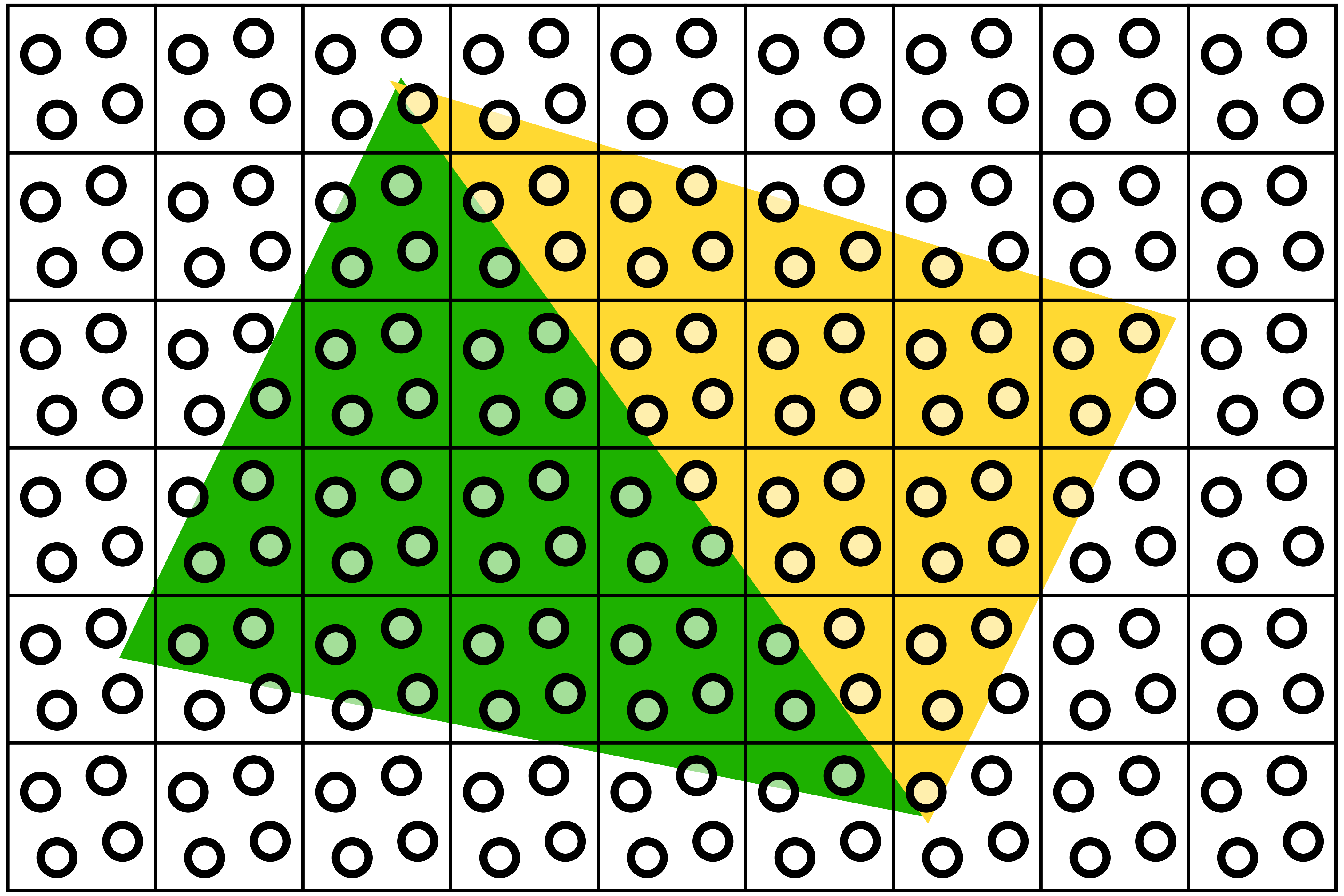
COL781: Computer Graphics

4. Transformations

End of last class

Suppose you want to draw multiple triangles. When should you average a pixel's sample values down to a single colour?

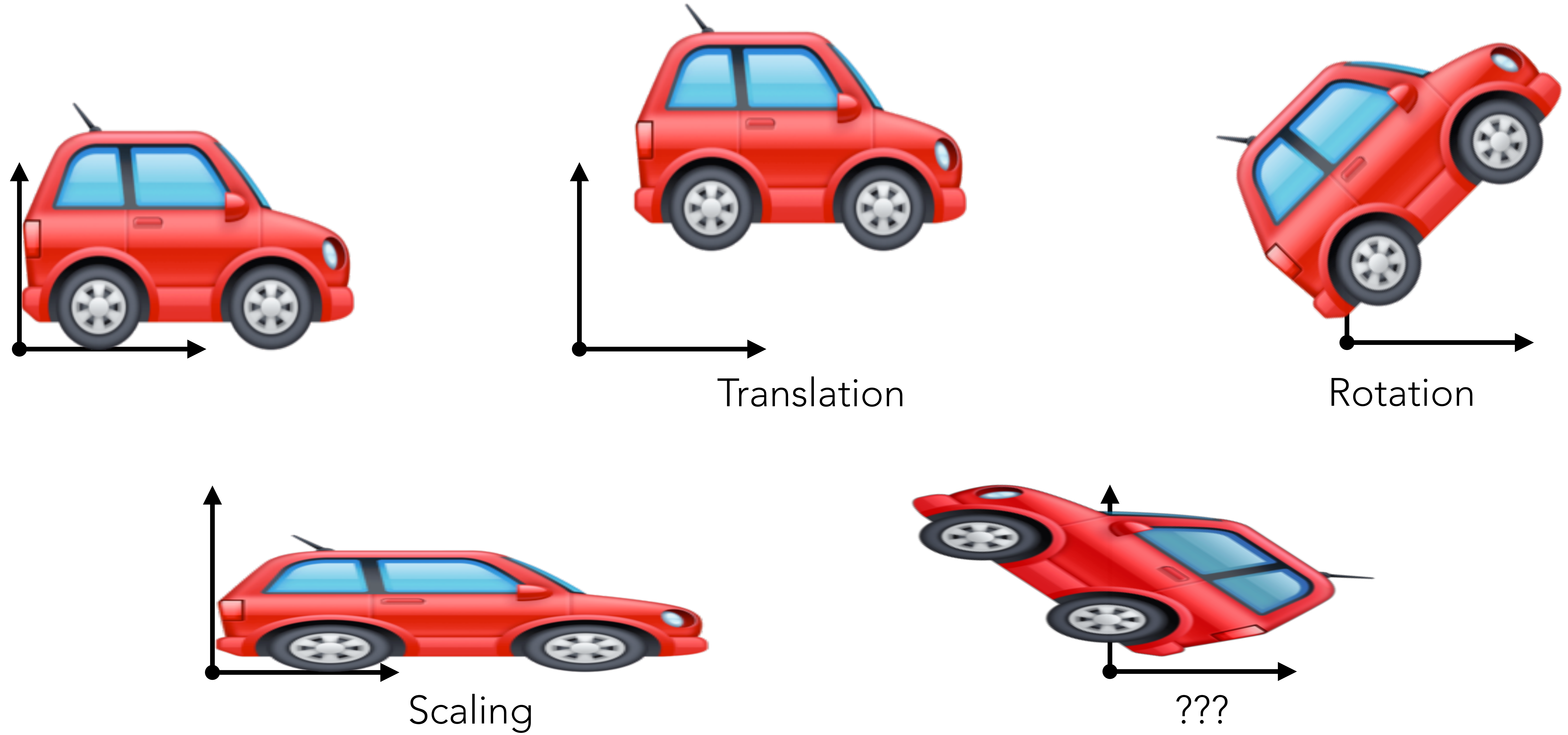
- After drawing each triangle?
- Only in the end?



How do these choices affect the image quality and the memory usage?

Cover image: Matthew Wagner

Transformations



Applications: Instancing

Star Wars: Episode II – Attack of the Clones (2002)

Applications: Posing

Applications: Viewing

Cristian Goga

Transformation matrices

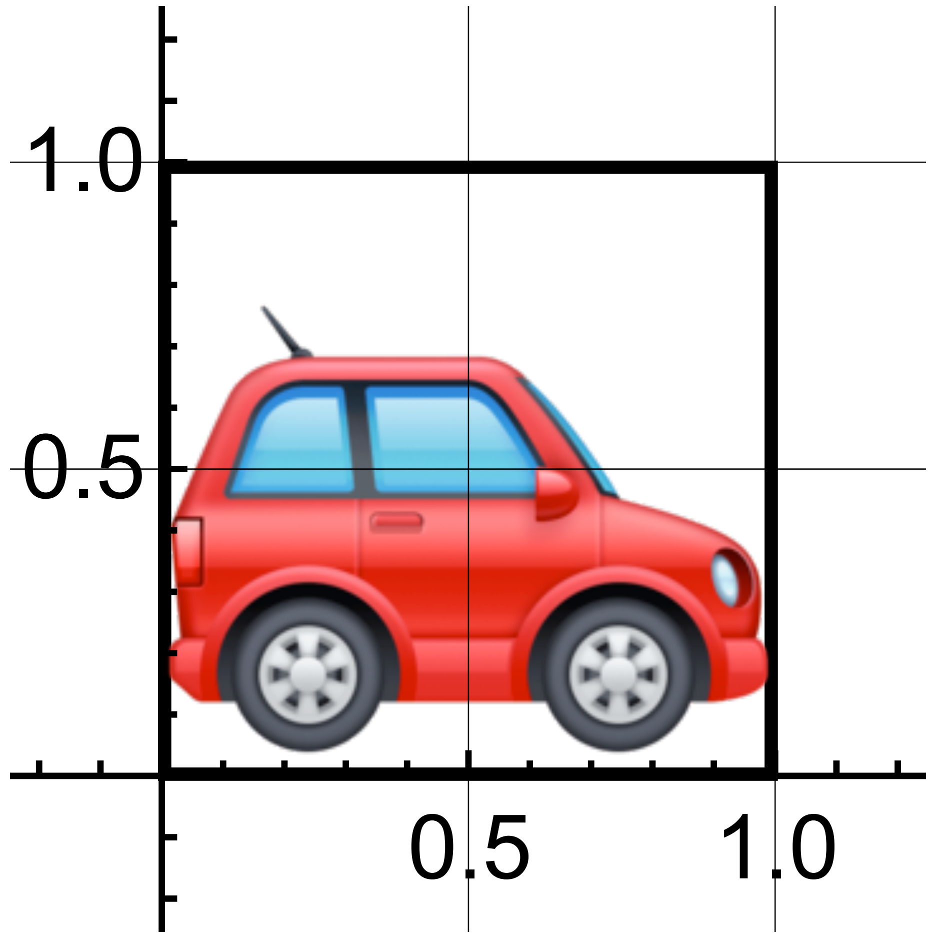
As you probably know, we can represent many transformations by matrices:

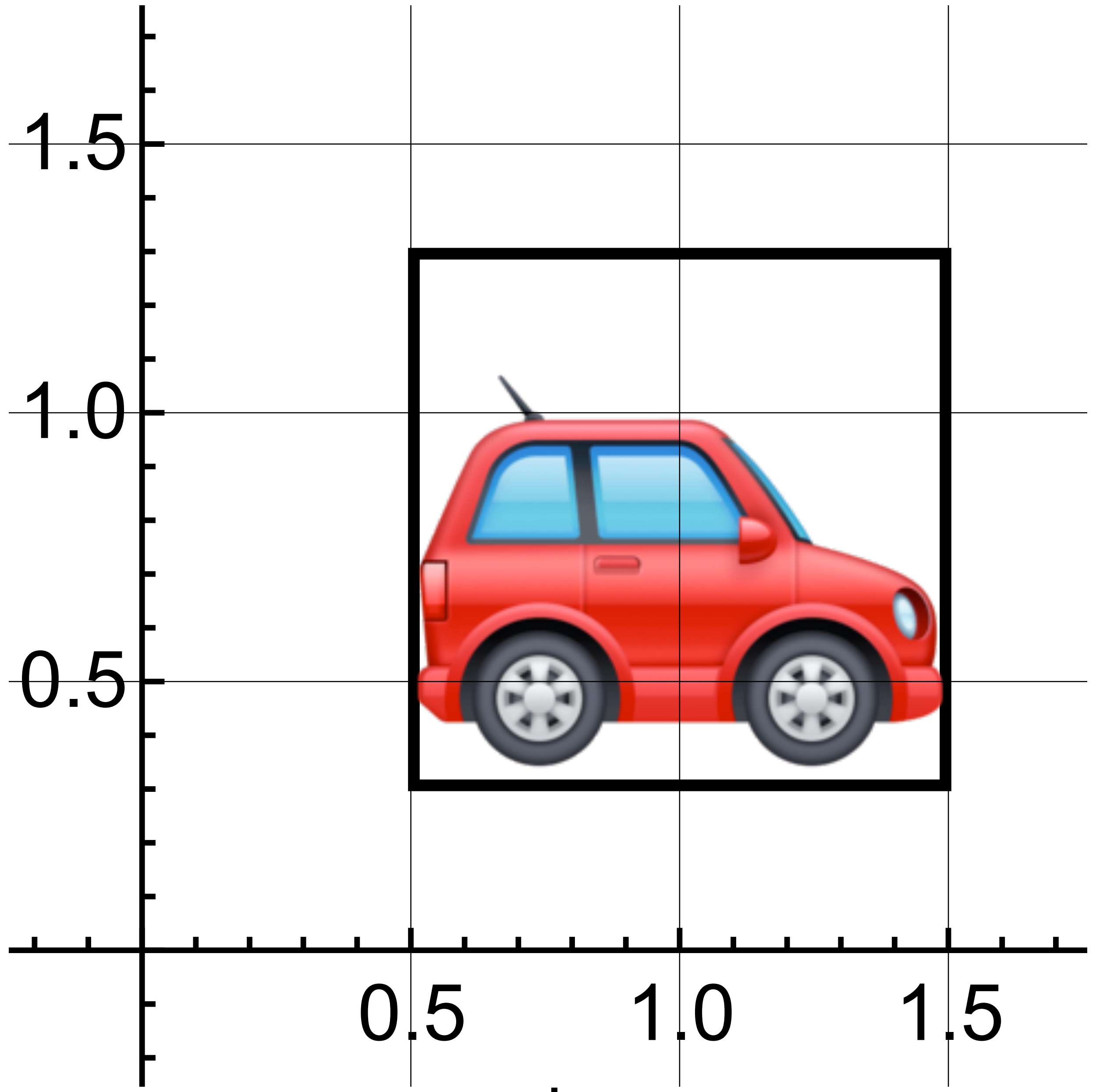
$$\mathbf{v} = \begin{bmatrix} v_x \\ v_y \end{bmatrix} \qquad \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \qquad \mathbf{Av} = \begin{bmatrix} a_{11}v_x + a_{12}v_y \\ a_{21}v_x + a_{22}v_y \end{bmatrix}$$

and similarly in 3D:

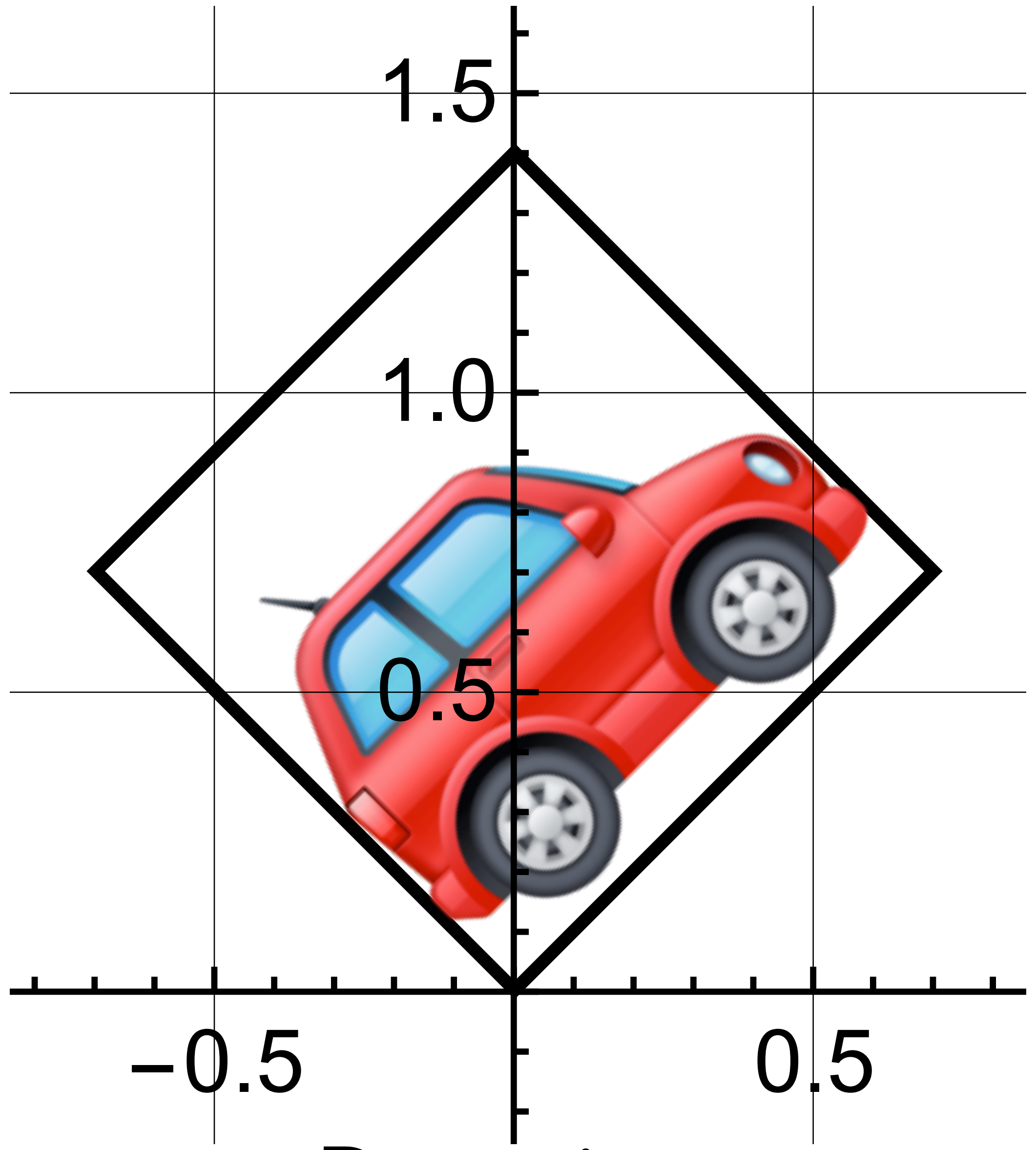
$$\mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \qquad \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \qquad \mathbf{Av} = \begin{bmatrix} a_{11}v_x + a_{12}v_y + a_{13}v_z \\ a_{21}v_x + a_{22}v_y + a_{23}v_z \\ a_{31}v_x + a_{32}v_y + a_{33}v_y \end{bmatrix}$$

What are the matrices for these transformations?

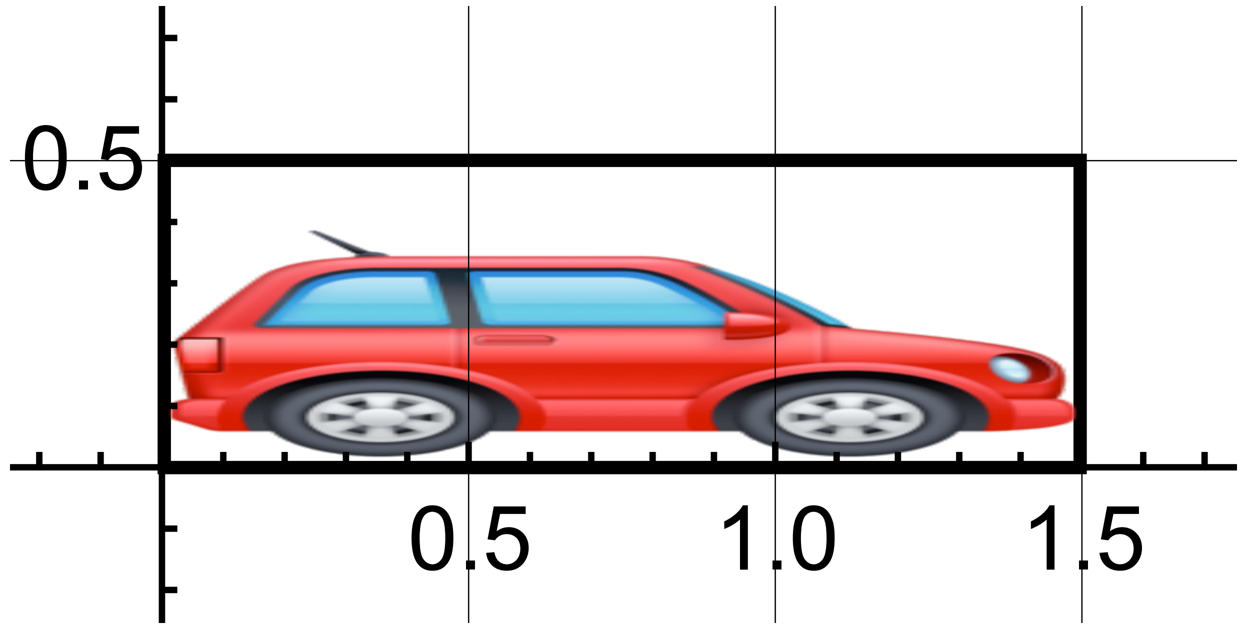




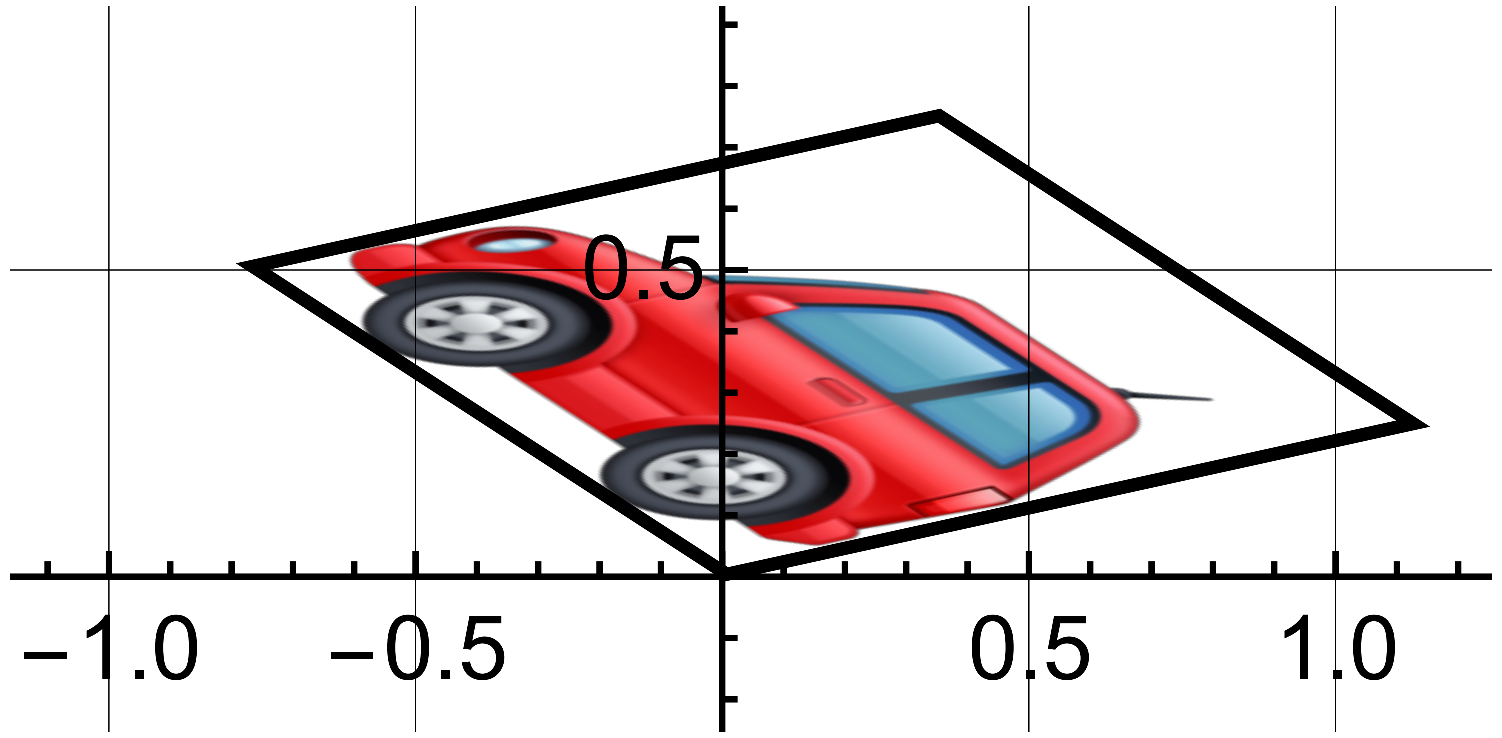
Translation



Rotation



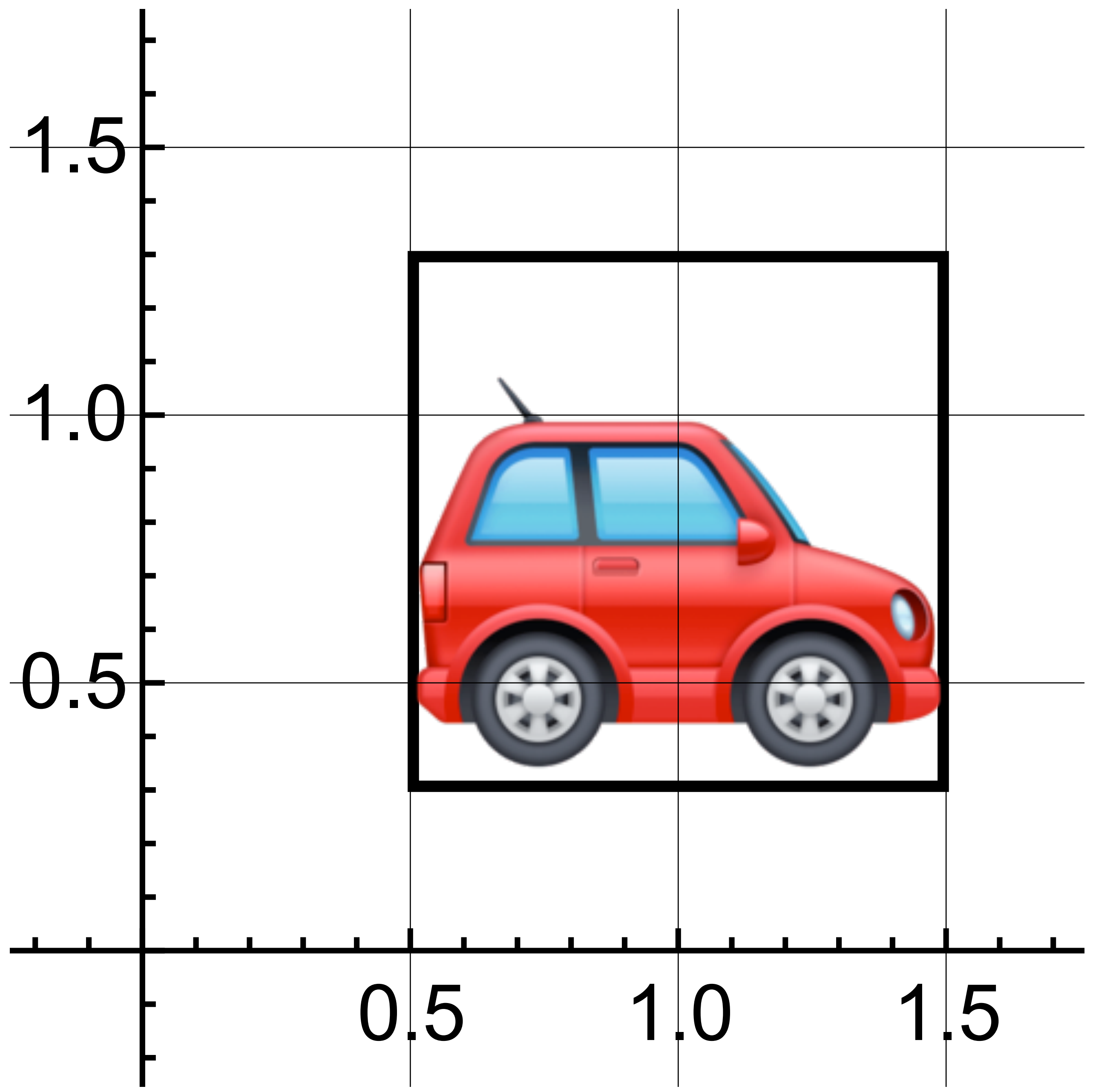
Scaling



???

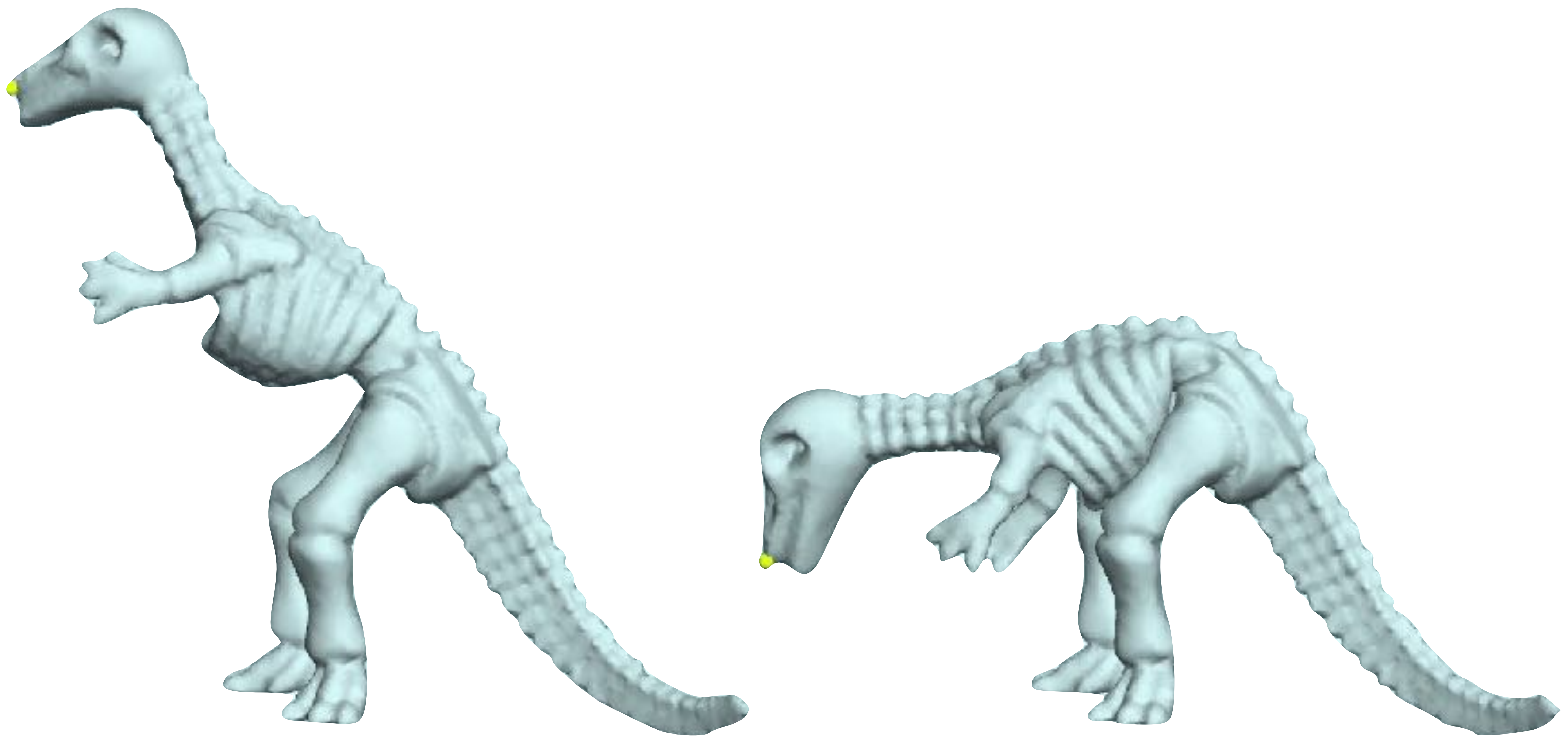
What can't matrices do?

$\mathbf{v}_{new} \neq \mathbf{A}\mathbf{v}_{old}$



Translation

(not yet at least…)



Sorkine & Alexa 2007

Nonlinear deformation

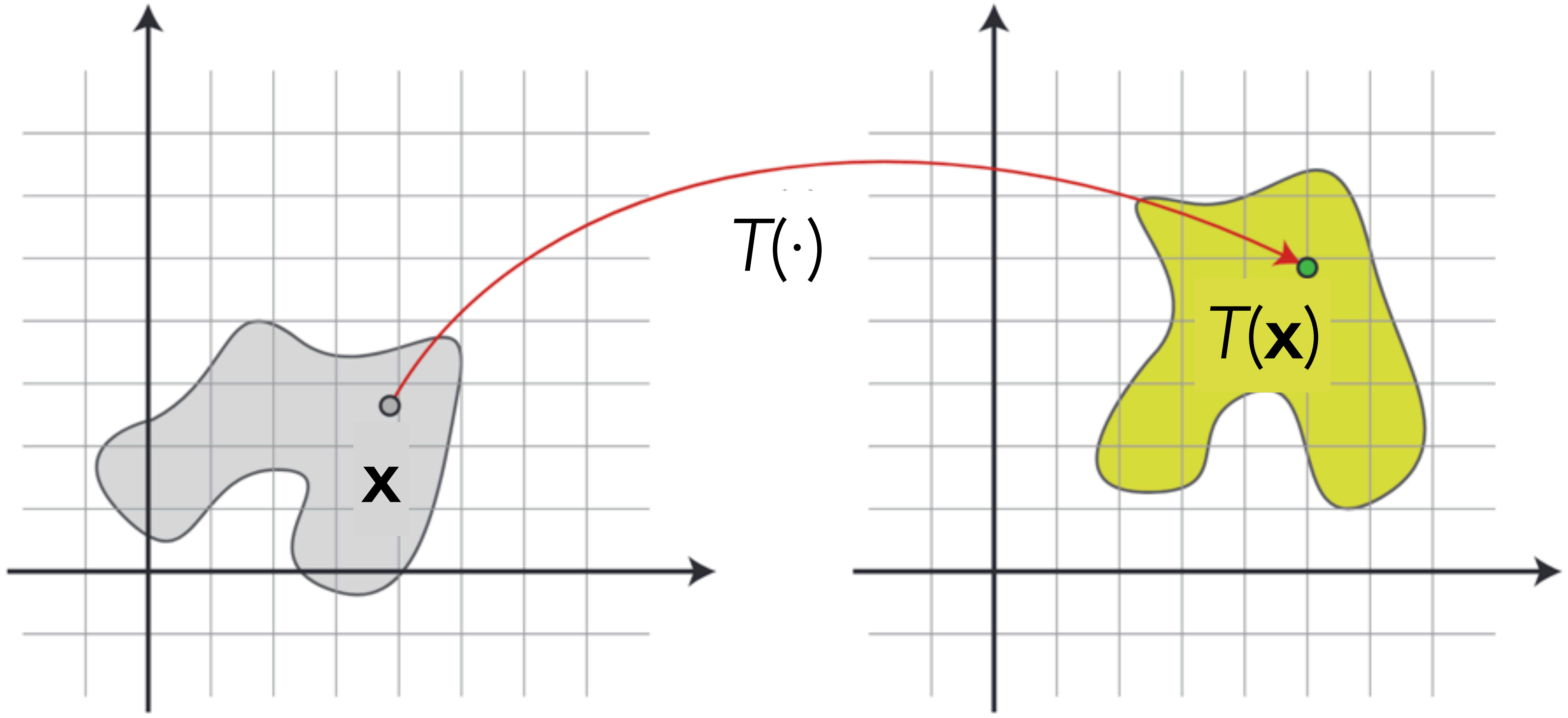
Transformations

Transformations are just functions that map points to points

$$T : \mathbb{R}^n \rightarrow \mathbb{R}^n$$

Today: linear transformations
(easy to represent with matrices)

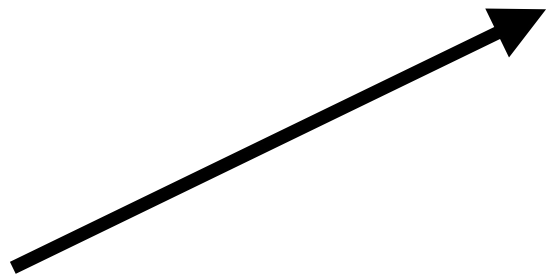
Next class: affine transformations
(linear transformations + translation)



Linear algebra

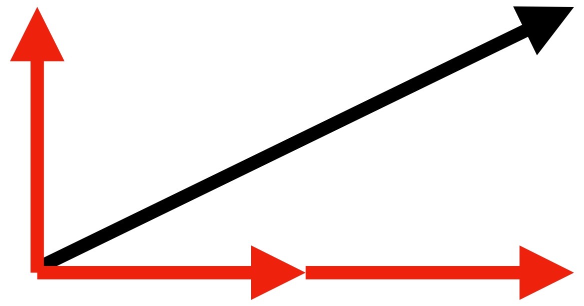
Linear algebra

Linear algebra is not about little lists of numbers!

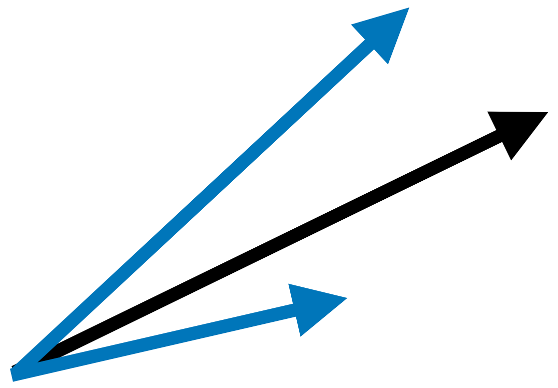


$$\neq \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

A vector only has coordinates once you make an (arbitrary) choice of basis

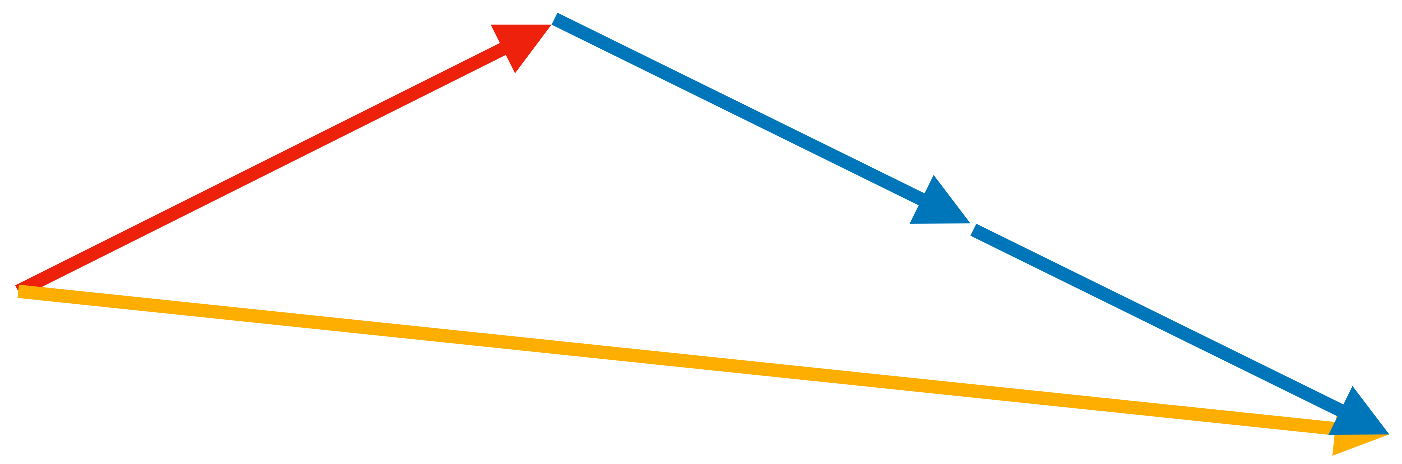


$$\begin{bmatrix} 2 \\ 1 \end{bmatrix}$$



$$\begin{bmatrix} 1 \\ 0.5 \end{bmatrix}$$

Outcomes of operations should be independent of arbitrary choices!



If **a** + 2**b** = **c** in my basis, then it should be true in your basis as well.

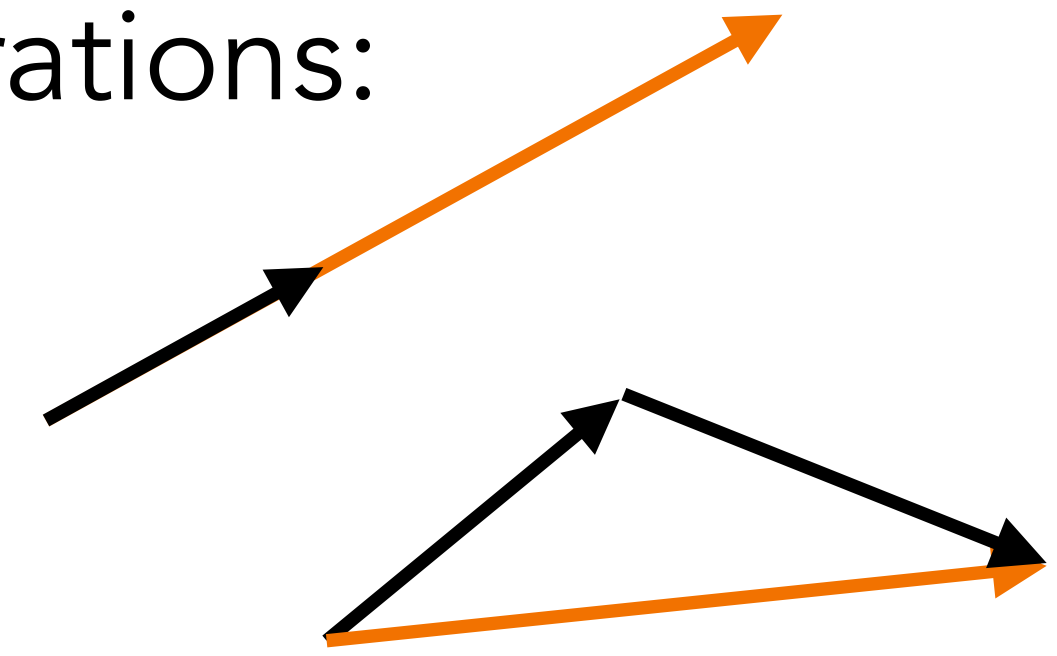
Best to think in a basis-independent way as much as possible

Though, to compute anything we will always need a basis in the end…

What are vectors, really?

A vector is an element of a vector space.

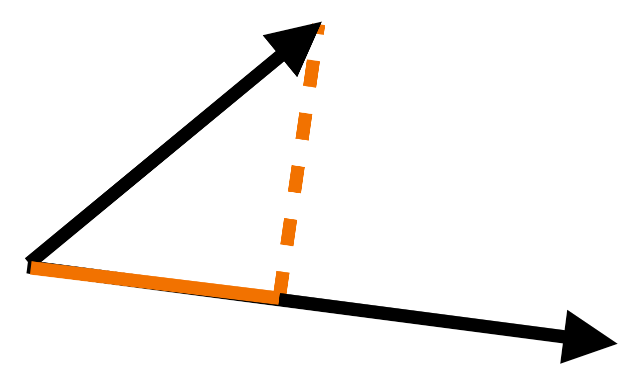
A vector space over $\mathbb{R}$ is any set V equipped with two operations:



- scalar multiplication: $\mathbb{R} \times V \to V$
- vector addition: $V \times V \to V$

satisfying various identities, e.g. $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$, $a(\mathbf{u} + \mathbf{v}) = a\mathbf{u} + a\mathbf{v}$, etc.

To do geometry, we also need a third operation:



- dot product / inner product: $V \times V \to \mathbb{R}$

satisfying identities like $\mathbf{u} \cdot \mathbf{v} = \mathbf{v} \cdot \mathbf{u}$, $(a\mathbf{u} + b\mathbf{v}) \cdot \mathbf{w} = a(\mathbf{u} \cdot \mathbf{w}) + b(\mathbf{v} \cdot \mathbf{w})$, etc.

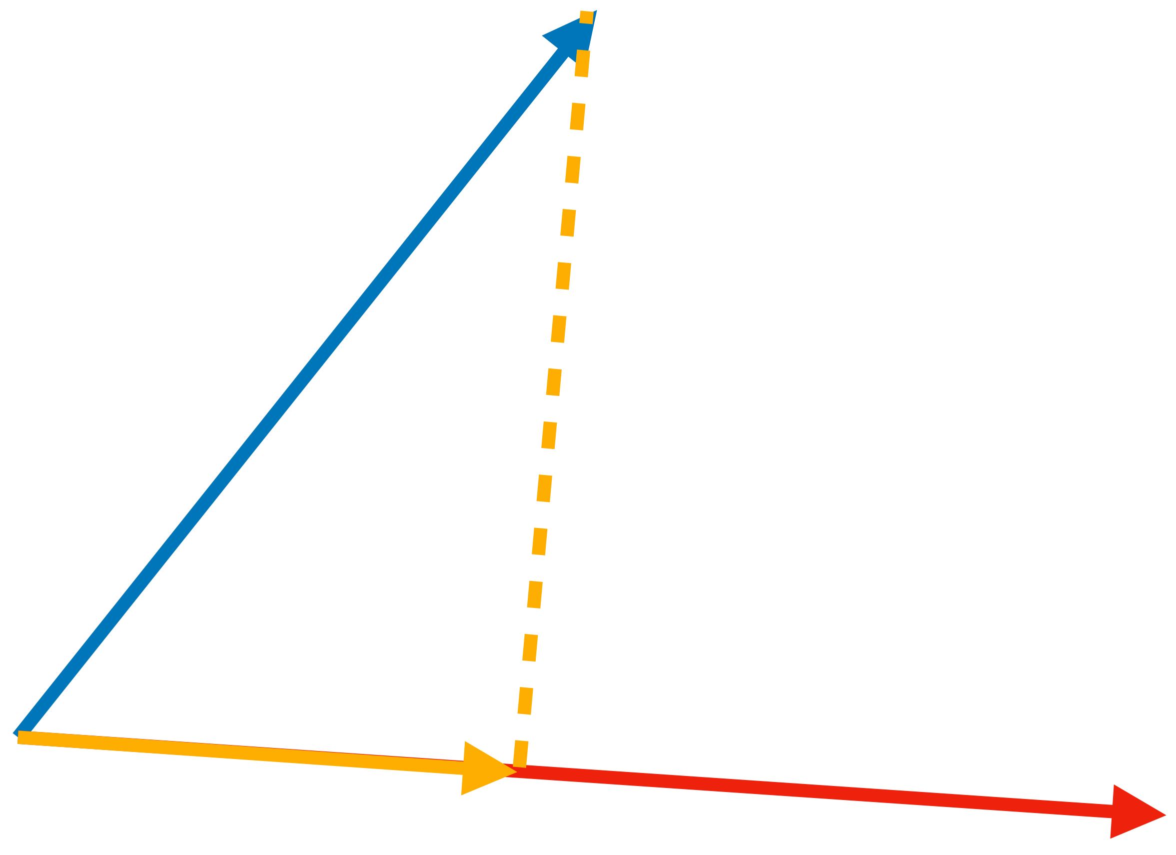
Think of these three operations as the public API of the “vector” data type.

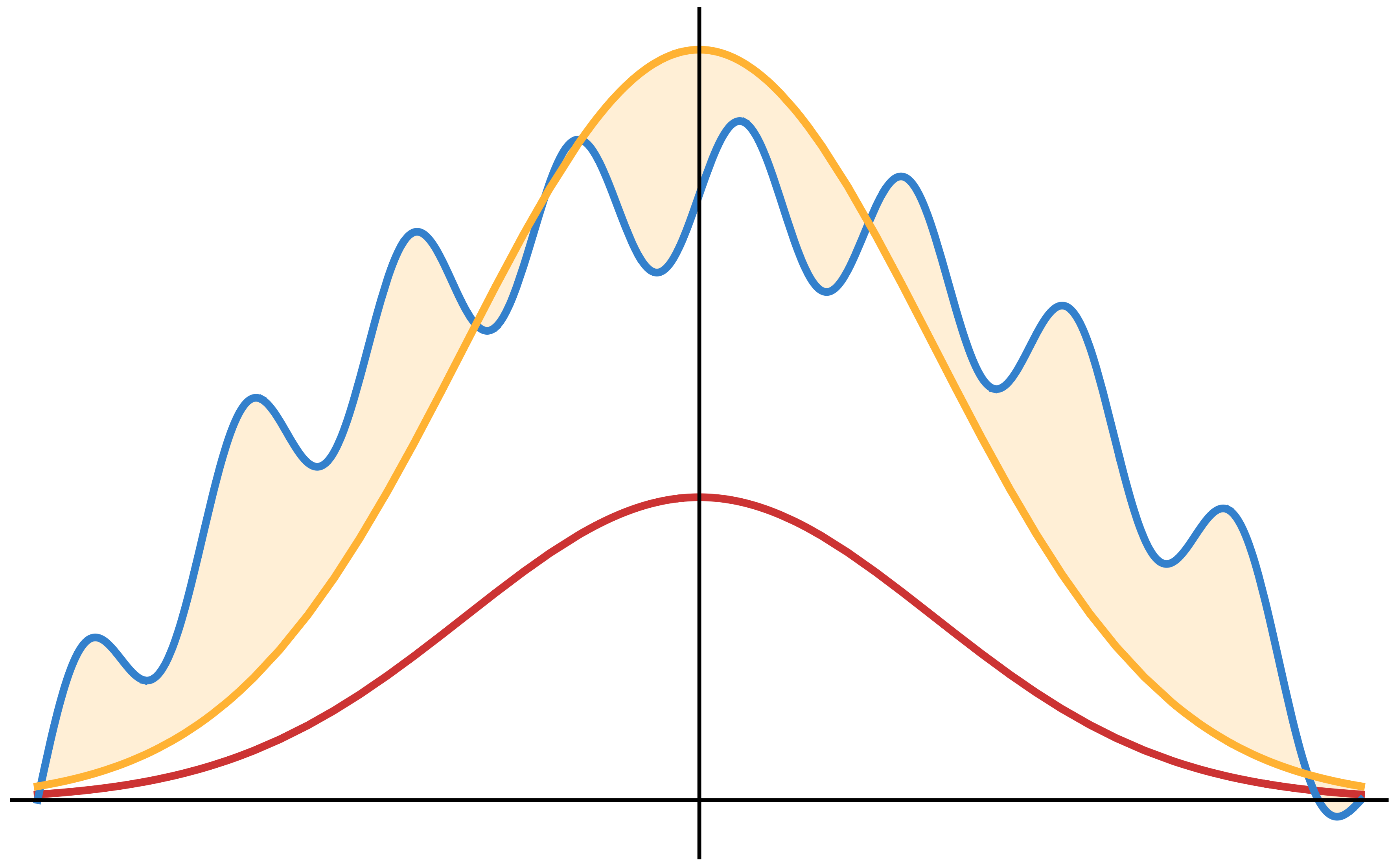
Write your algorithm and code in terms of these, and it will work in 2D, in 3D, and in any n dimensions!

(It will also work for other vector spaces: functions, images, etc. …)

Example: Find the multiple of **u** that minimizes distance to **v**.

$$\left(\frac{\mathbf{u} \cdot \mathbf{v}}{\mathbf{u} \cdot \mathbf{u}}\right)\mathbf{u}$$

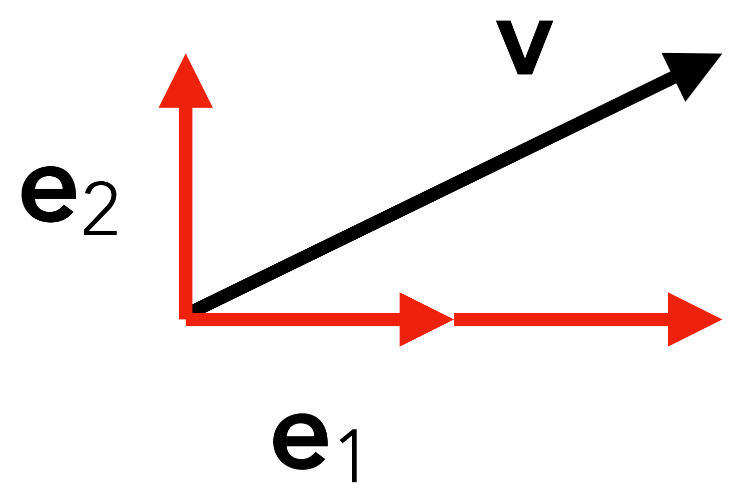




Bases

A basis is just a set of vectors $\{\mathbf{e}_1, \mathbf{e}_2, \ldots\}$ such that any vector can be written uniquely as a linear combination of them.

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



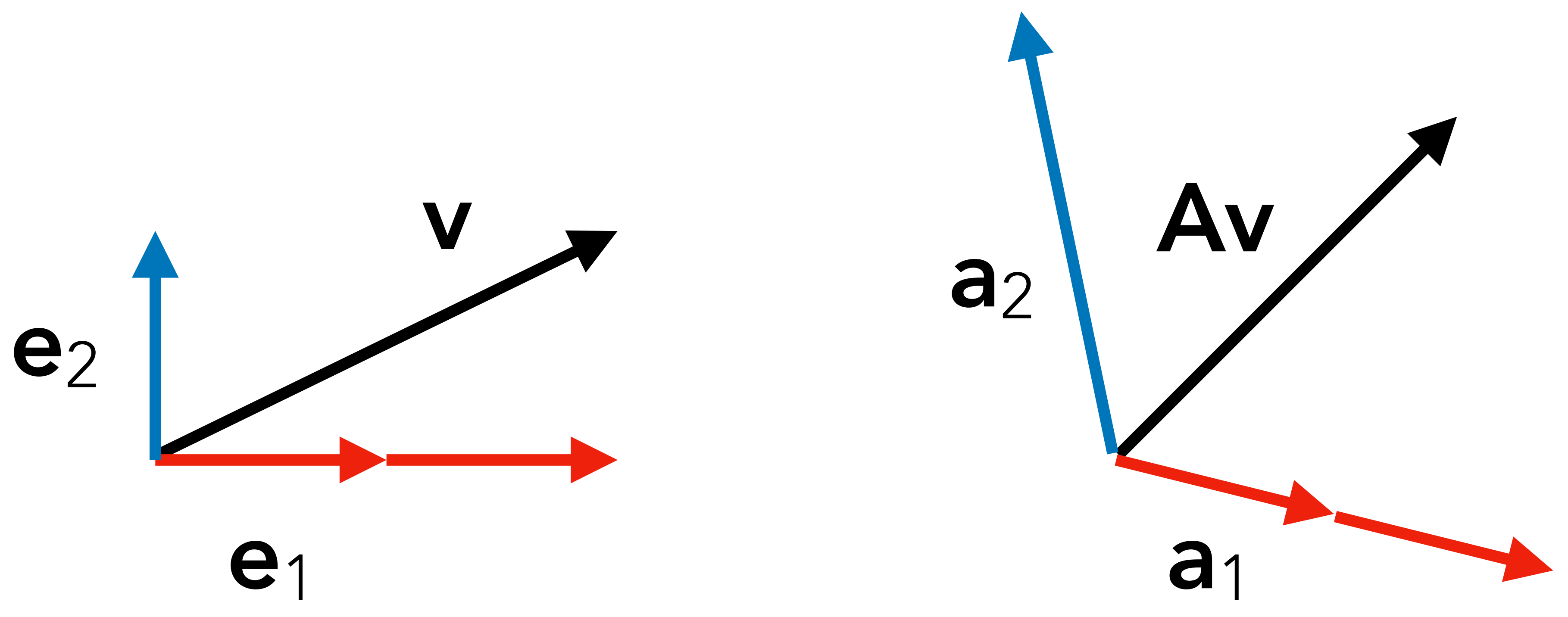
$$\begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

What happens when you apply a matrix **A** to the basis vectors?

$$\mathbf{A}\mathbf{e}_1 = \begin{bmatrix} a_{11} & a_{12} & \cdots \\ a_{21} & a_{22} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ \vdots \end{bmatrix} = \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \end{bmatrix} = \text{1st column of } \mathbf{A}$$

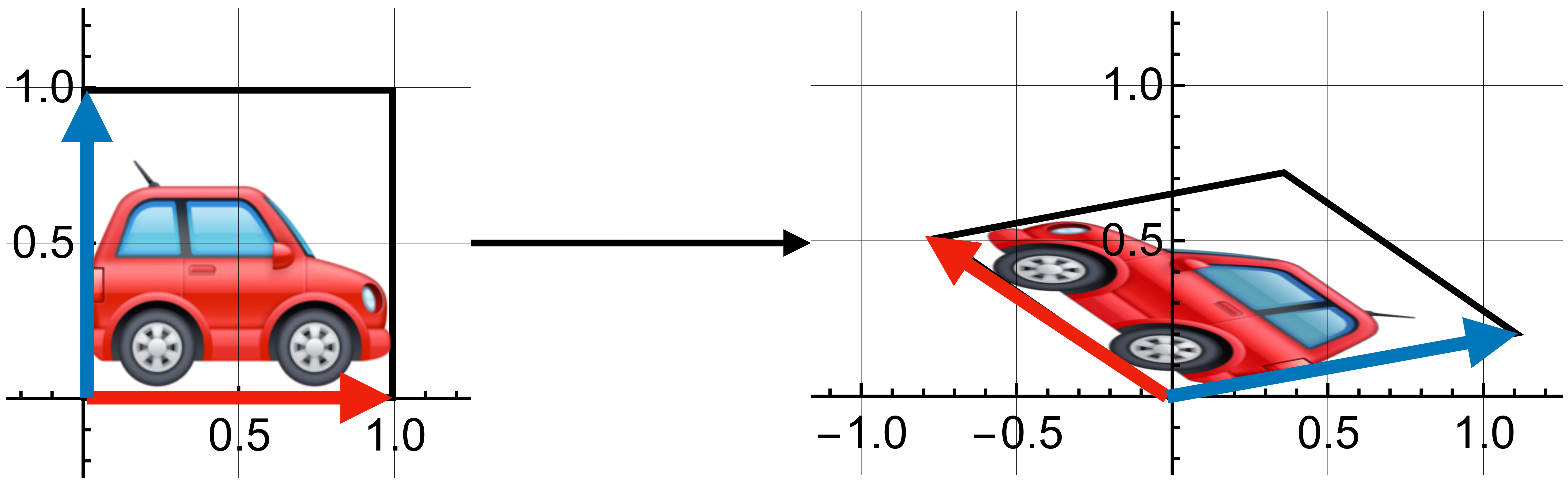
This determines the action of **A** on all other vectors!

$$\mathbf{A}\mathbf{v} = \mathbf{A}(v_1\mathbf{e}_1 + v_2\mathbf{e}_2 + \cdots) = v_1(\mathbf{A}\mathbf{e}_1) + v_2(\mathbf{A}\mathbf{e}_2) + \cdots = v_1\mathbf{a}_1 + v_2\mathbf{a}_2 + \cdots$$



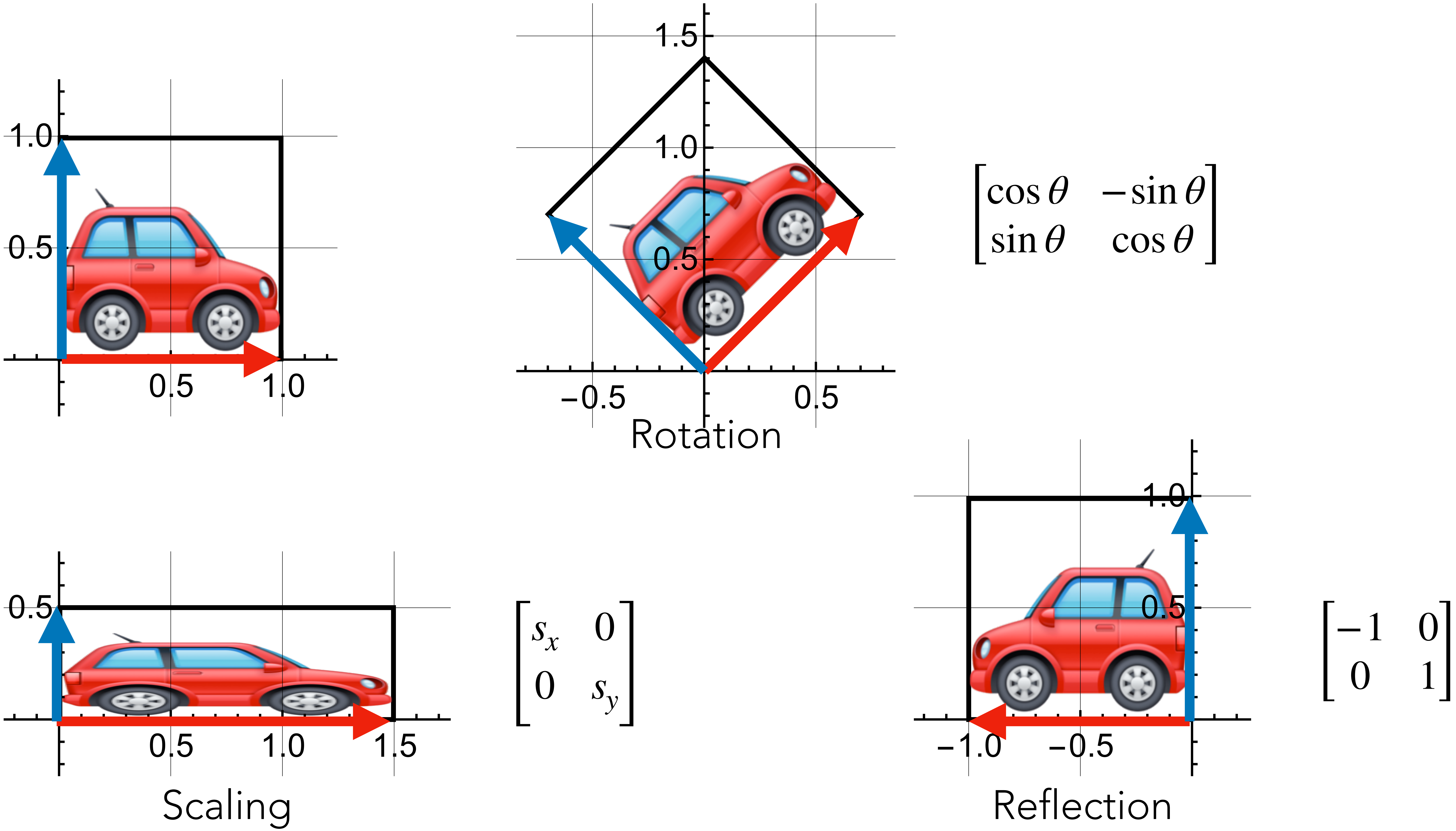
- **Interpretation 1:** A matrix transforms the basis vectors to its columns; all other vectors follow.
- **Interpretation 2:** Matrix-vector multiplication **Av** produces a linear combination of the columns of **A**, weighted by the components v_1, v_2, …

Now, what is the matrix for this transformation?



$$\mathbf{a}_1 = \text{image of } \mathbf{e}_1 \approx \begin{bmatrix} -0.8 \\ 0.5 \end{bmatrix}, \quad \mathbf{a}_2 = \text{image of } \mathbf{e}_2 \approx \begin{bmatrix} 1.1 \\ 0.2 \end{bmatrix}$$

$$\mathbf{A} \approx \begin{bmatrix} -0.8 & 1.1 \\ 0.5 & 0.2 \end{bmatrix}$$



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

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

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

Puzzle:

It’s just as easy to scale *x* and *y* by different amounts as it is to scale by the same amount.

So why do video players show black bars instead of scaling the image to fill the screen?

Invariants

Different types of transformations preserve different quantities:

- **Rotations**: distances, angles, orientation
- **Reflections**: distances, angles
- **Uniform scaling** ($s_x = s_y$): relative distances, angles, directions
- **Nonuniform scaling** ($s_x \neq s_y$): ?

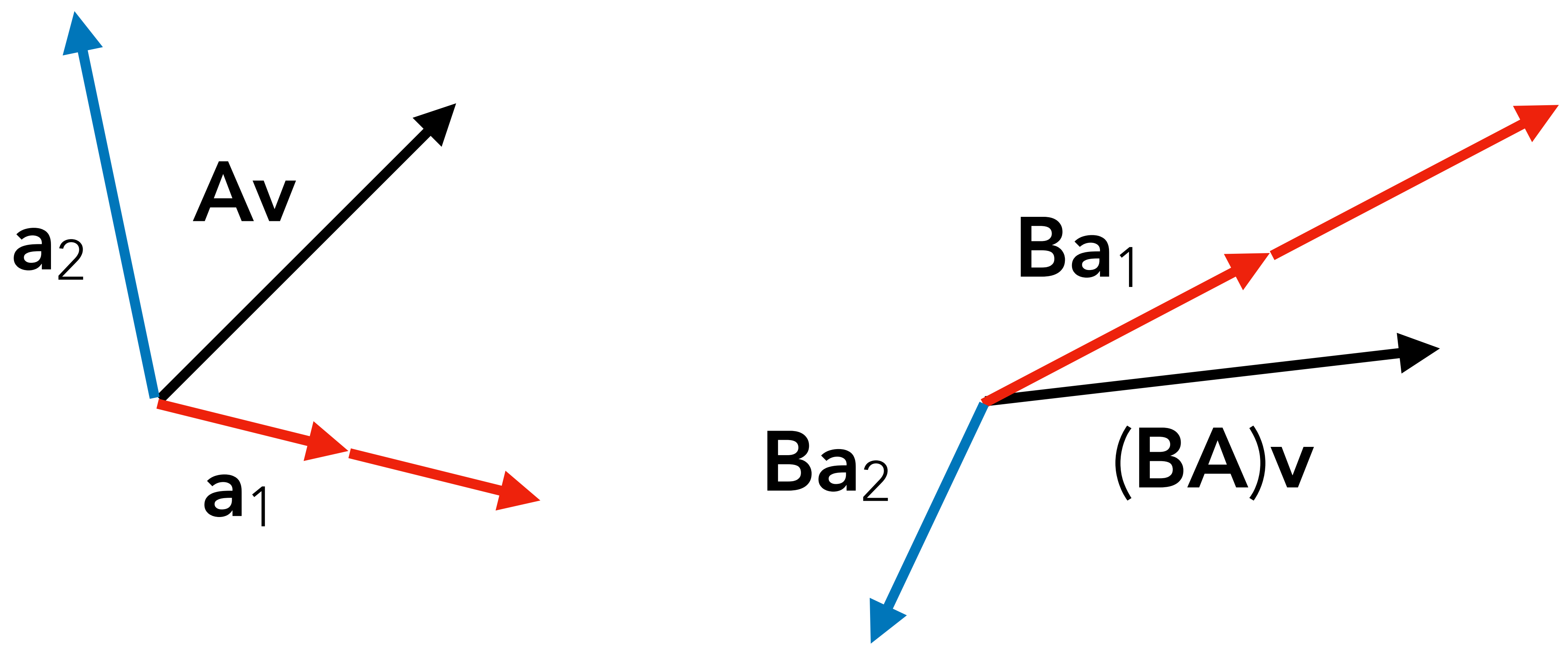
Composition of transformations

Apply transformation **A** then transformation **B**:

$$\mathbf{v} \;\rightarrow\; \mathbf{Av} \;\rightarrow\; \mathbf{B(Av)} = \mathbf{(BA)v}$$

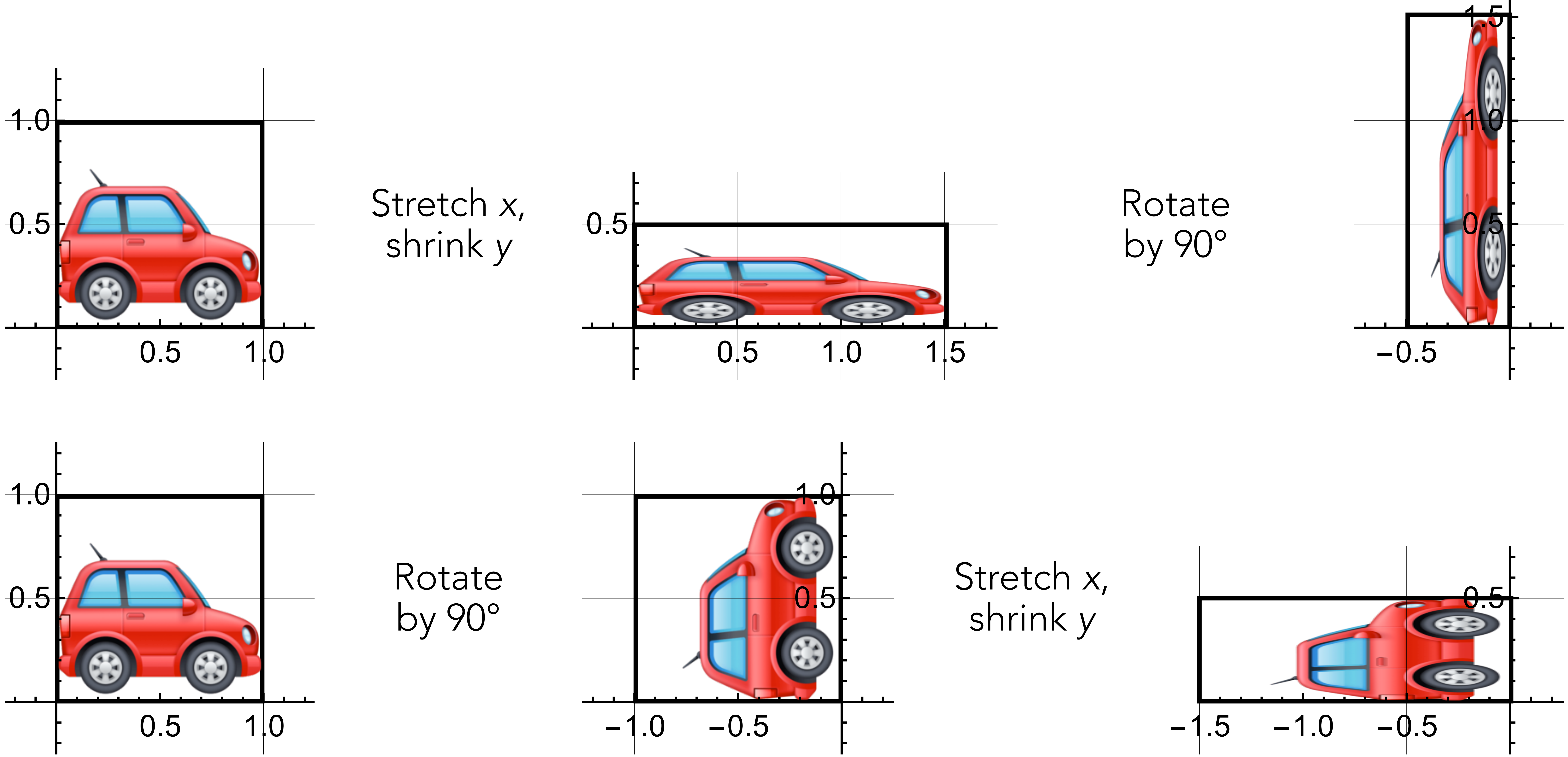
Column interpretation:

$$\mathbf{B}\begin{bmatrix}\mathbf{a}_1 & \mathbf{a}_2 & \cdots\end{bmatrix} = \begin{bmatrix}\mathbf{Ba}_1 & \mathbf{Ba}_2 & \cdots\end{bmatrix}$$



Often, want to apply a sequence of *n* transformations on millions of vertices.
Just compute the product: then only 1 matrix-vector multiplication per vertex.

AB ≠ BA


1.0
0.5
0.5
1.0
Stretch x,
shrink y
0.5
0.5
1.0
1.5
Rotate
by 90°
1.5
1.0
0.5
−0.5
1.0
0.5
0.5
1.0
Rotate
by 90°
1.0
0.5
−1.0
−0.5
Stretch x,
shrink y
0.5
−1.5
−1.0
−0.5