
Computer Graphics

4 - Affine Space / Frame / Matrix

Yoonsang Lee
Hanyang University

Spring 2026

Outline

- Affine Spaces and Coordinate-Free Formulation
- Coordinate System & Reference Frame
- Affine Transformation Matrix
- Interpretation of Composite Transformations

Affine Spaces and Coordinate-Free Formulation

Points

Point **p**



Point **q**



- What does it mean to “add” two points?

If we assume coordinates...

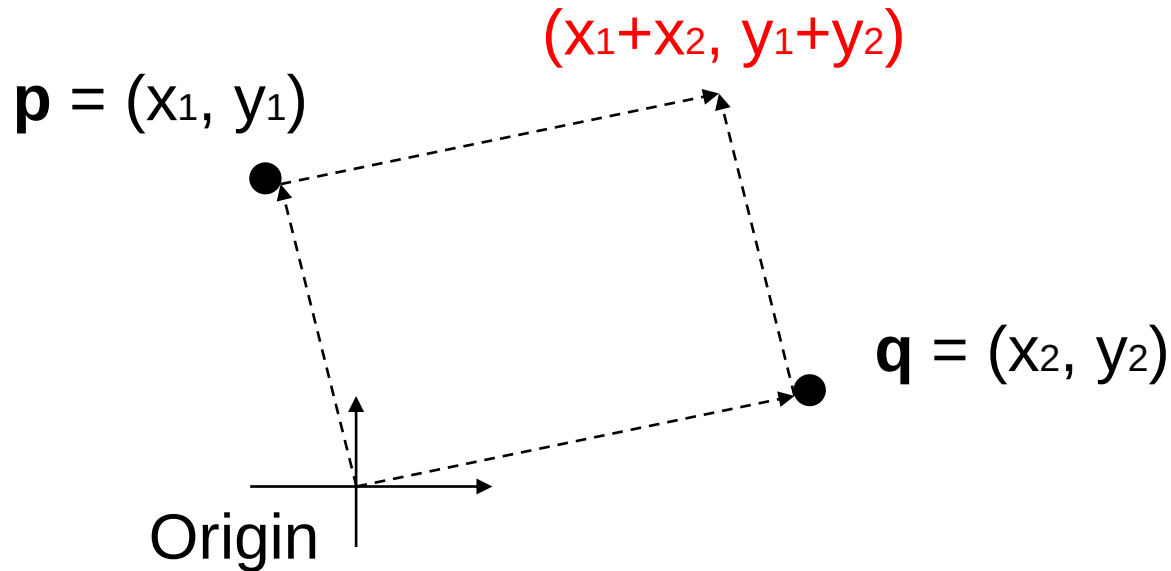
$$\mathbf{p} = (x_1, y_1)$$



$$\mathbf{q} = (x_2, y_2)$$

- We can compute: $(x_1 + x_2, y_1 + y_2)$
- Is this correct ?
- Is this meaningful ?

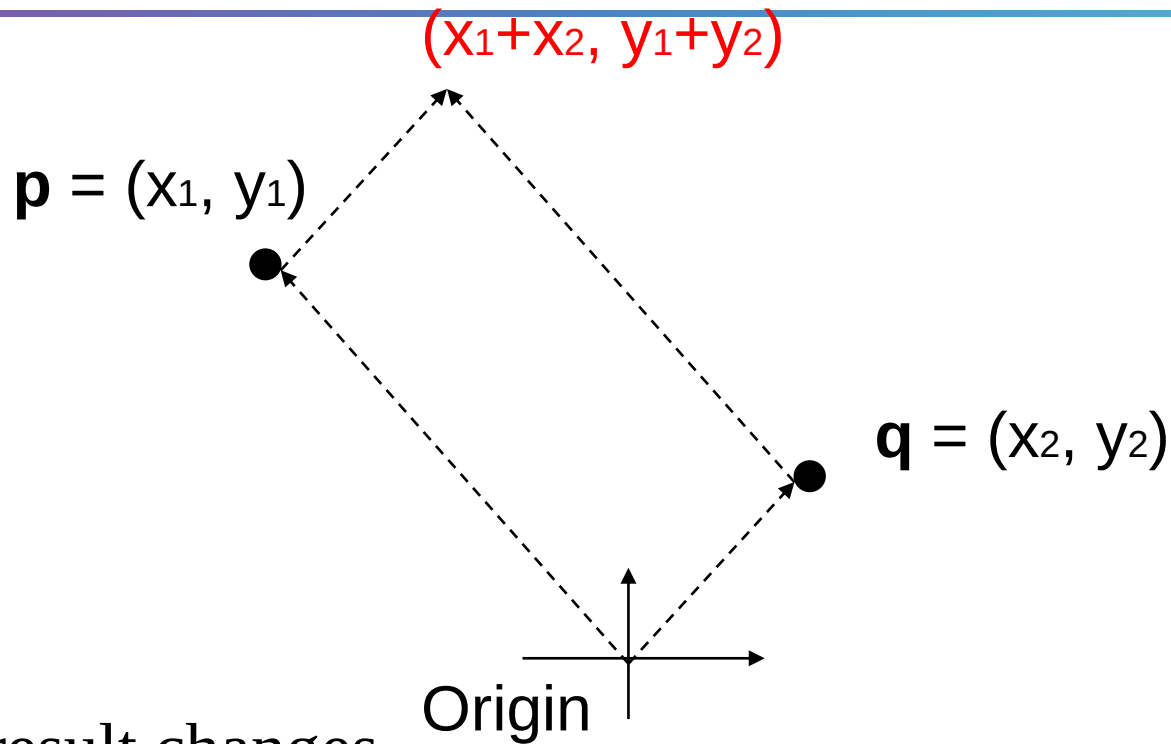
What are we really doing?



→ Vector addition from the origin

- $\mathbf{p}, \mathbf{q} \rightarrow$ vectors from the origin
- **Requires an origin**

If we change the origin...



→ The result changes

- This operations depend on the coordinate system (undesirable)

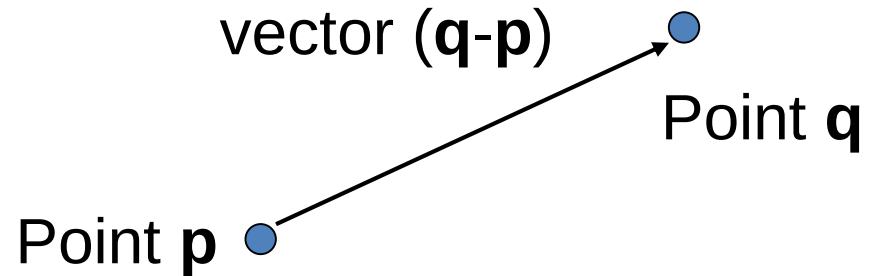
What's the problem?

- We tried: adding two points
- Problem:
 - The result depends on the choice of origin
 - Different coordinate systems → different results
- This is undesirable because:
 - Geometry should not depend on arbitrary choices
 - The same geometric configuration should not produce different results under different coordinate systems
- ➔ Adding two points is not *coordinate invariant*

What do we want?

- Goal:
Describe geometry independent of coordinate system.
- ➔ ***Coordinate-free (coordinate invariant)***
formulation

Key Idea: Points $\neq$ Vectors



- **Point** = position
- **Vector** = displacement
→ Mixing them caused the problem we saw
- If an **origin** is chosen:
→ a point can be represented as a vector
- But:
A point $\neq$ a vector (in **coordinate-free** formulation).

Analogy: Time vs Geometry

- **datetime** = position in time
 - **duration** = displacement in time
 - You cannot:
 - datetime + datetime
 - But you can:
 - datetime + duration
 - datetime – datetime
- **point** = position
 - **vector** = displacement
 - You cannot:
 - point + point
 - But you can:
 - point + vector
 - point - point

Vector and Affine Spaces: A Structure for Geometry

- We need a mathematical structure where these rules are built in
- **Vector space**
 - Consists of vectors with associated operations
 - No points (so it cannot represent positions)
- **Affine space**
 - Consists of points & an associated vector space
 - Supports operations involving points and vectors
- Affine space is exactly the structure we need

Vector Spaces

- A **vector space** consists of:
 - A set of vectors
 - Closed under two operations:
 - Vector addition
 - Scalar multiplication
- *Linear combinations* remain in the space.

$$\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_N \in V \quad \Rightarrow \quad c_0 \mathbf{u}_0 + c_1 \mathbf{u}_1 + \dots + c_N \mathbf{u}_N \in V$$

(Linear combination = a weighted sum of vectors)

Affine Spaces

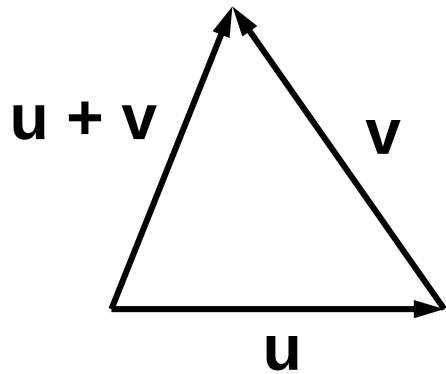
- A **affine space** consists of:
 - A set of points
 - An associated vector space
- Supports:
 - Vector space operations (on vectors)
 - Additional operations involving points:
 - $\text{point} - \text{point}$
 - $\text{point} + \text{vector}$
 - But: $\text{point} + \text{point}$ is not defined

Coordinate-Free Geometric Operations

- These are exactly the operations we just discussed.
- We can categorize these operations into three basic types:
 - Addition
 - Subtraction
 - Scalar multiplication
- They are defined without relying on coordinates, and behave differently for points and vectors.

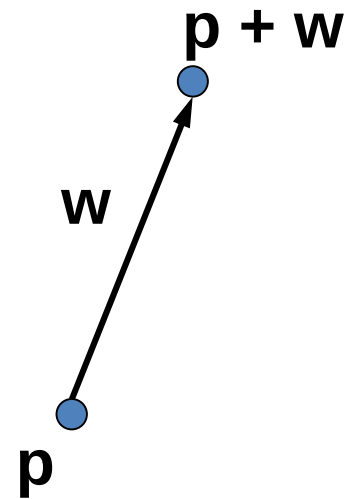
Addition

Vector addition:



$u + v \rightarrow$ vector

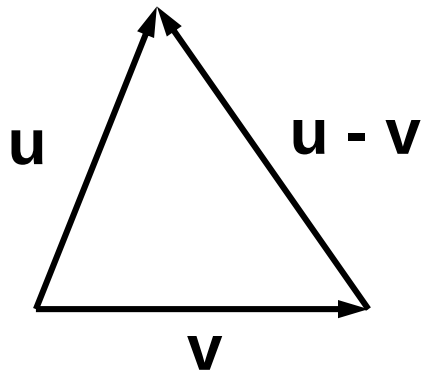
Point–vector addition:



$p + w \rightarrow$ point

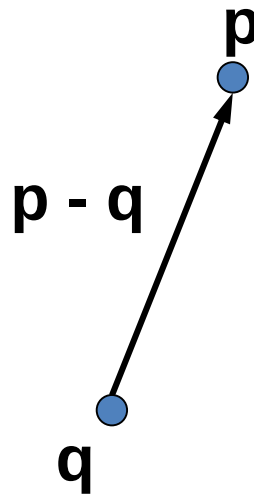
Subtraction

Vector subtraction:



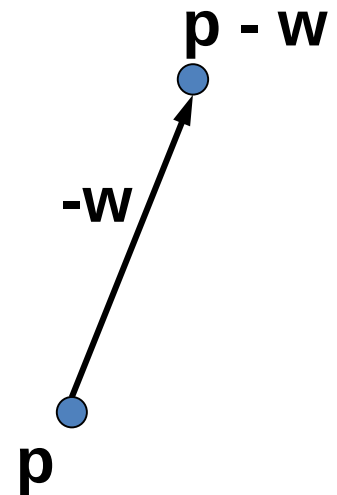
$u - v \rightarrow$ vector

Point–point subtraction:



$p - q \rightarrow$ vector

Point–vector subtraction:



$p - w \rightarrow$ point

Scalar Multiplication

Vector scaling:

$$c \cdot \mathbf{v} = c\mathbf{v} \text{ (vector)}$$

Point scaling:

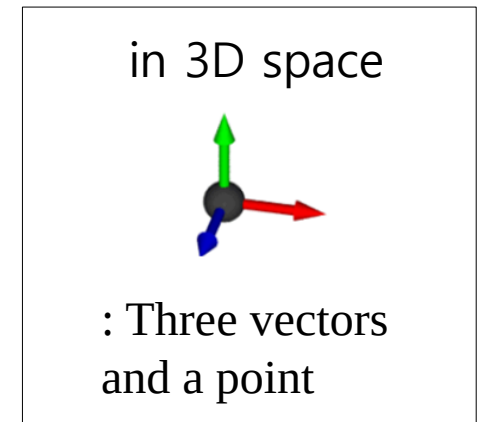
$$1 \cdot \mathbf{p} = \mathbf{p} \text{ (point)}$$

$$0 \cdot \mathbf{p} = \mathbf{0} \text{ (zero vector)}$$

$$c \cdot \mathbf{p} \rightarrow \text{undefined (otherwise)}$$

Affine Frame

- Coordinate-free is conceptually clean, but to compute, we need coordinates.
- *A frame* defines a coordinate system.
- To define a coordinate system, we choose:
 - An origin point $\mathbf{o}$
 - Basis vectors $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$
- Then every point and vector can be represented by coordinates $(c_1, \dots, c_n)$.
 - $\mathbf{p} = \mathbf{o} + c_1\mathbf{v}_1 + \dots + c_n\mathbf{v}_n$ (point)
 - $\mathbf{v} = c_1\mathbf{v}_1 + \dots + c_n\mathbf{v}_n$ (vector)



Summary: Points $\neq$ Vectors

- **point + point $\rightarrow$ undefined**
- **point - point $\rightarrow$ vector**
- **point $\pm$ vector $\rightarrow$ point**
- **vector $\pm$ vector $\rightarrow$ vector**
- **scalar $\cdot$ vector $\rightarrow$ vector**
- **scalar $\cdot$ point:**
 - = point (c = 1)**
 - = vector (c = 0)**
 - = undefined (otherwise)**
- So how do we represent this in a coordinate system?

Points & Vectors in Homogeneous Coordinates

- We can encode the difference between points and vectors using homogeneous coordinates.
- A 3D **point** is represented: $(x, y, z, \mathbf{1})$
- A 3D **vector** is represented: $(x, y, z, \mathbf{0})$
- The last coordinate (1 vs 0) encodes the difference between points and vectors.
 - This lets us treat points and vectors in a unified way.

Geometric Operations in Homogeneous Coordinates

$$\begin{matrix} (x_1, y_1, z_1, 1) & + & (x_2, y_2, z_2, 1) & = & (x_1+x_2, y_1+y_2, z_1+z_2, 2) \\ \textit{point} & & \textit{point} & & \textit{undefined} \end{matrix}$$

$$\begin{matrix} (x_1, y_1, z_1, 1) & - & (x_2, y_2, z_2, 1) & = & (x_1-x_2, y_1-y_2, z_1-z_2, 0) \\ \textit{point} & & \textit{point} & & \textit{vector} \end{matrix}$$

$$\begin{matrix} (x_1, y_1, z_1, 1) & + & (x_2, y_2, z_2, 0) & = & (x_1+x_2, y_1+y_2, z_1+z_2, 1) \\ \textit{point} & & \textit{vector} & & \textit{point} \end{matrix}$$

$$\begin{matrix} (x_1, y_1, z_1, 0) & + & (x_2, y_2, z_2, 0) & = & (x_1+x_2, y_1+y_2, z_1+z_2, 0) \\ \textit{vector} & & \textit{vector} & & \textit{vector} \end{matrix}$$

$$\begin{matrix} c * (x_1, y_1, z_1, 0) & = & (cx_1, cy_1, cz_1, 0) \\ \textit{vector} & & \textit{vector} \end{matrix}$$

$$\begin{matrix} c * (x_1, y_1, z_1, 1) & = & (cx_1, cy_1, cz_1, c) \\ \textit{point} & & \textit{undefined} & \textit{(if } c \neq 0, 1) \end{matrix}$$

Affine Transformations in Homogeneous Coordinates

- Applying an affine transformation:

$$\begin{bmatrix} M & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ \boxed{1} \end{bmatrix} = \begin{bmatrix} M\mathbf{p} + \mathbf{t} \\ \boxed{1} \end{bmatrix} \quad \begin{bmatrix} M & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \boxed{0} \end{bmatrix} = \begin{bmatrix} \boxed{M\mathbf{v}} \\ \boxed{0} \end{bmatrix}$$

point $\longrightarrow$ point vector $\longrightarrow$ vector

- Translation affects **points**, but **not vectors**.

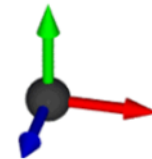
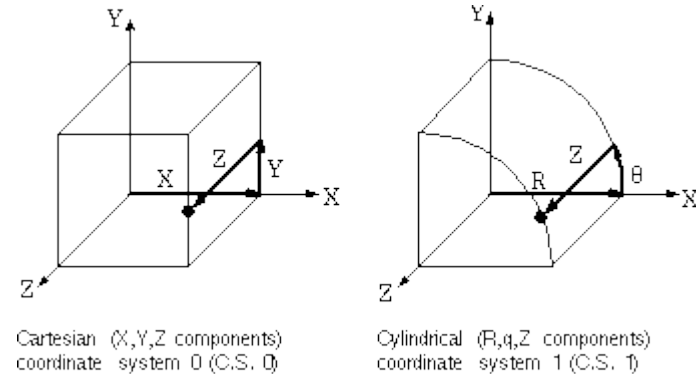
Quiz 1

- Go to <https://www.slido.com/>
- Join #cg-ys
- Click "Polls"
- Submit your answer in the following format:
 - **Student ID: Your answer**
 - e.g. 2024123456: 4.0
- Note that your quiz answer must be submitted **in the above format** to be counted for attendance.

Coordinate System & Reference Frame

Coordinate System & Reference Frame

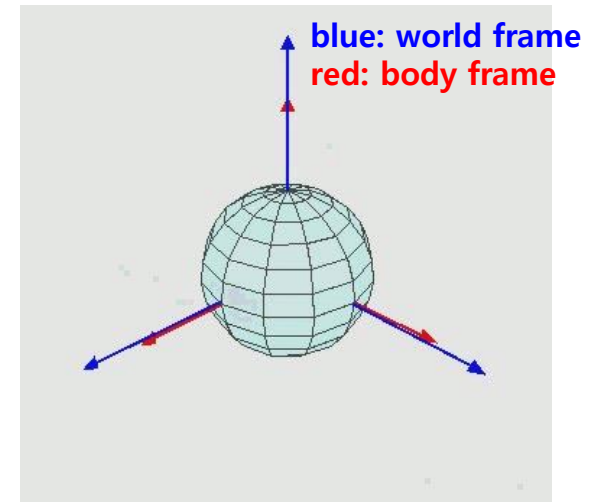
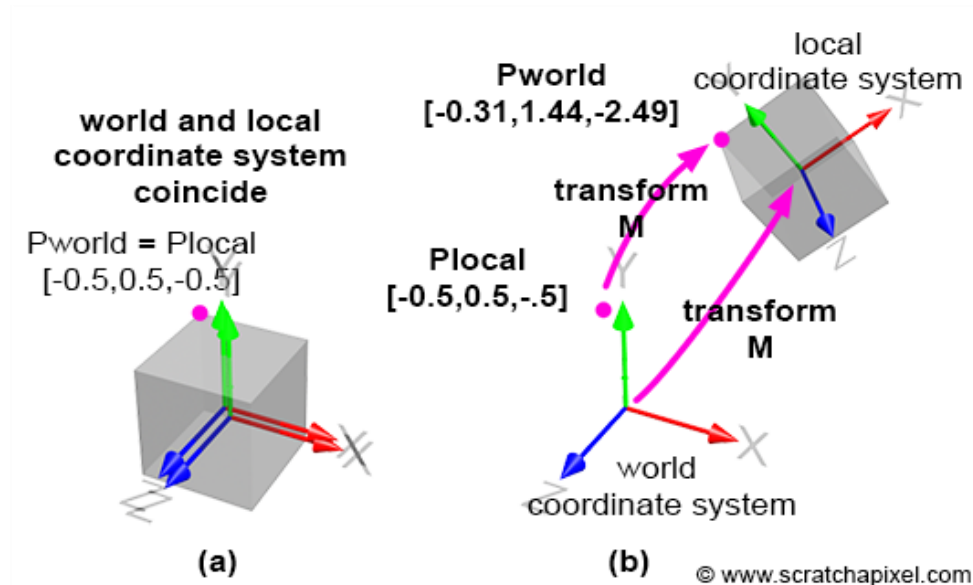
- Coordinate system
 - A way to assign numbers to points
- Reference frame
 - A coordinate system anchored to a specific origin and basis
- These two concepts are slightly different, but are often used interchangeably in practice.



: Three vectors
and a point

World Frame vs Body Frame

- **World frame** (or coordinate system)
 - A frame (or coordinate system) attached to the **world**.
 - a.k.a. **global** frame, **fixed** frame
- **Body frame** (or coordinate system)
 - A frame (or coordinate system) attached to a **moving object**.
 - a.k.a. **local** frame



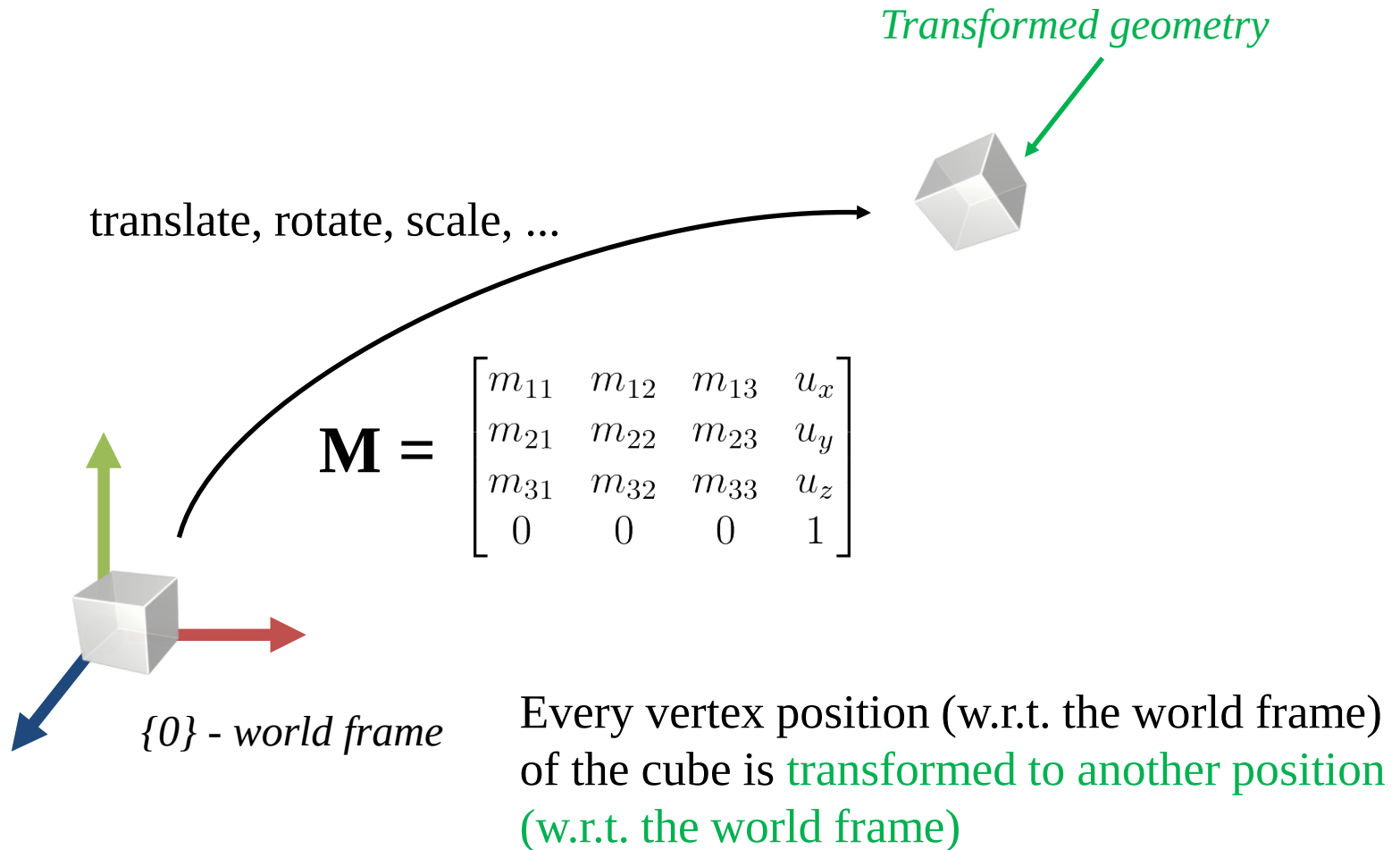
<https://commons.wikimedia.org/wiki/File:Euler2a.gif>

Affine Transformation Matrix

Three Interpretations of an Affine Transformation Matrix

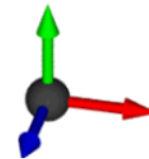
- An affine transformation matrix can be understood in three ways:
- 1) **Transforming a Geometry** (expressed in World Frame)
- 2) **Defining a New Frame** (w.r.t. World Frame)
 - w.r.t. = with respect to
 - expressed in that frame
- 3) Changing the Representation of a Point (from **Body Frame** to **World Frame**)

1) Transforming a Geometry (expressed in World Frame)



Review: Affine Frame

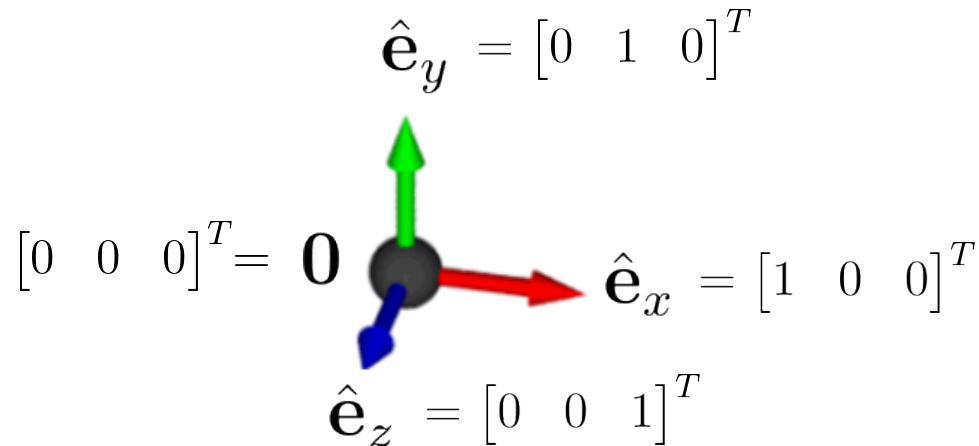
- An **affine frame** (for a 3D space) is defined by three vectors and one point
 - Three vectors for x, y, z axes
 - One point for origin



: Three vectors
and a point

World Frame

- The **world frame** is usually represented by
 - Standard basis vectors for axes : $\hat{\mathbf{e}}_x, \hat{\mathbf{e}}_y, \hat{\mathbf{e}}_z$
 - Origin point : $\mathbf{0}$



Let's transform a "world frame"

- Apply M to this "world frame", that is,
 - Multiply M with the x, y, z axis *vectors* and the origin *point* of the world frame:

x axis *vector*

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & u_x \\ m_{21} & m_{22} & m_{23} & u_y \\ m_{31} & m_{32} & m_{33} & u_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} m_{11} \\ m_{21} \\ m_{31} \\ 0 \end{bmatrix}$$

y axis *vector*

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & u_x \\ m_{21} & m_{22} & m_{23} & u_y \\ m_{31} & m_{32} & m_{33} & u_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} m_{12} \\ m_{22} \\ m_{32} \\ 0 \end{bmatrix}$$

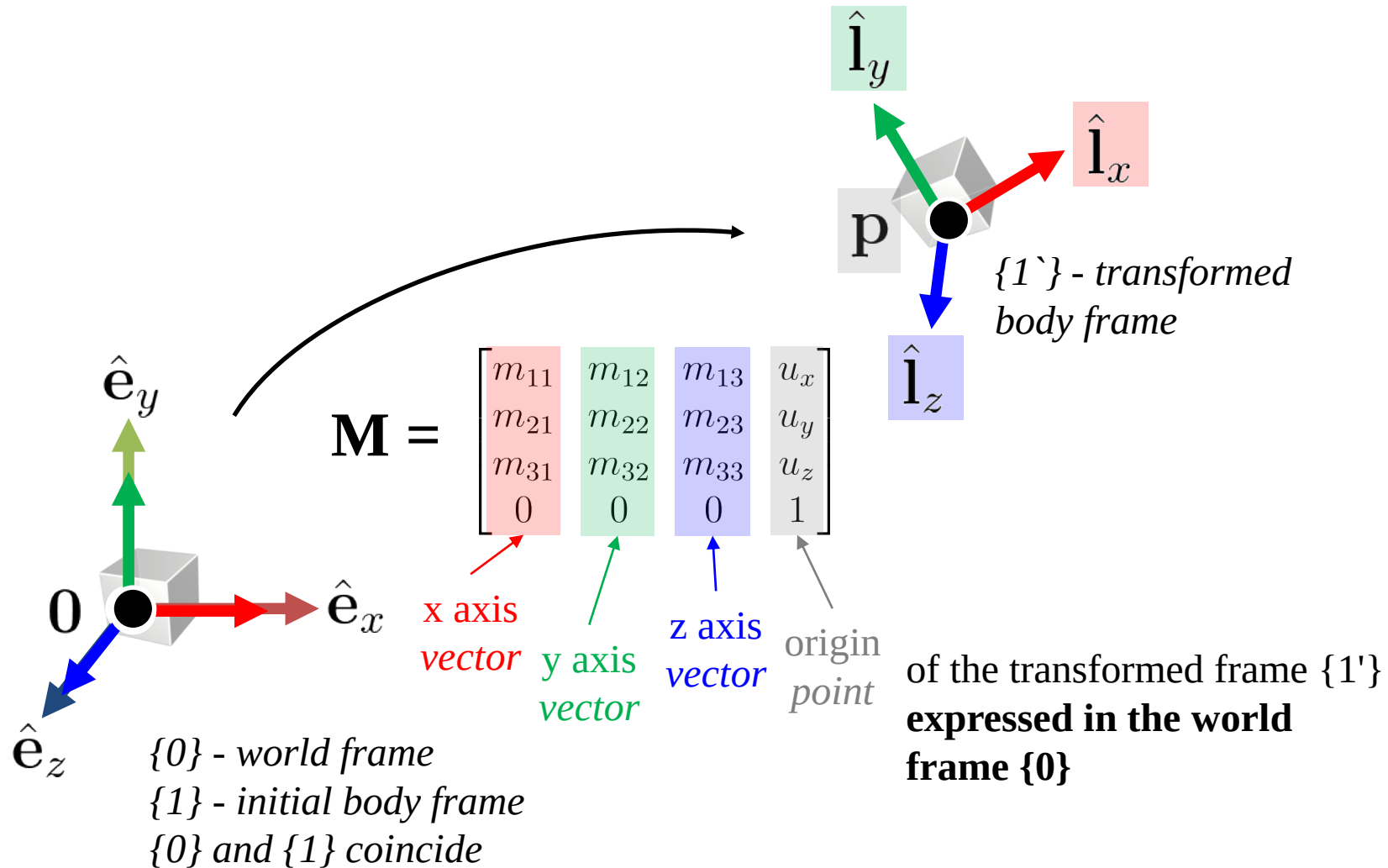
z axis *vector*

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & u_x \\ m_{21} & m_{22} & m_{23} & u_y \\ m_{31} & m_{32} & m_{33} & u_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} m_{13} \\ m_{23} \\ m_{33} \\ 0 \end{bmatrix}$$

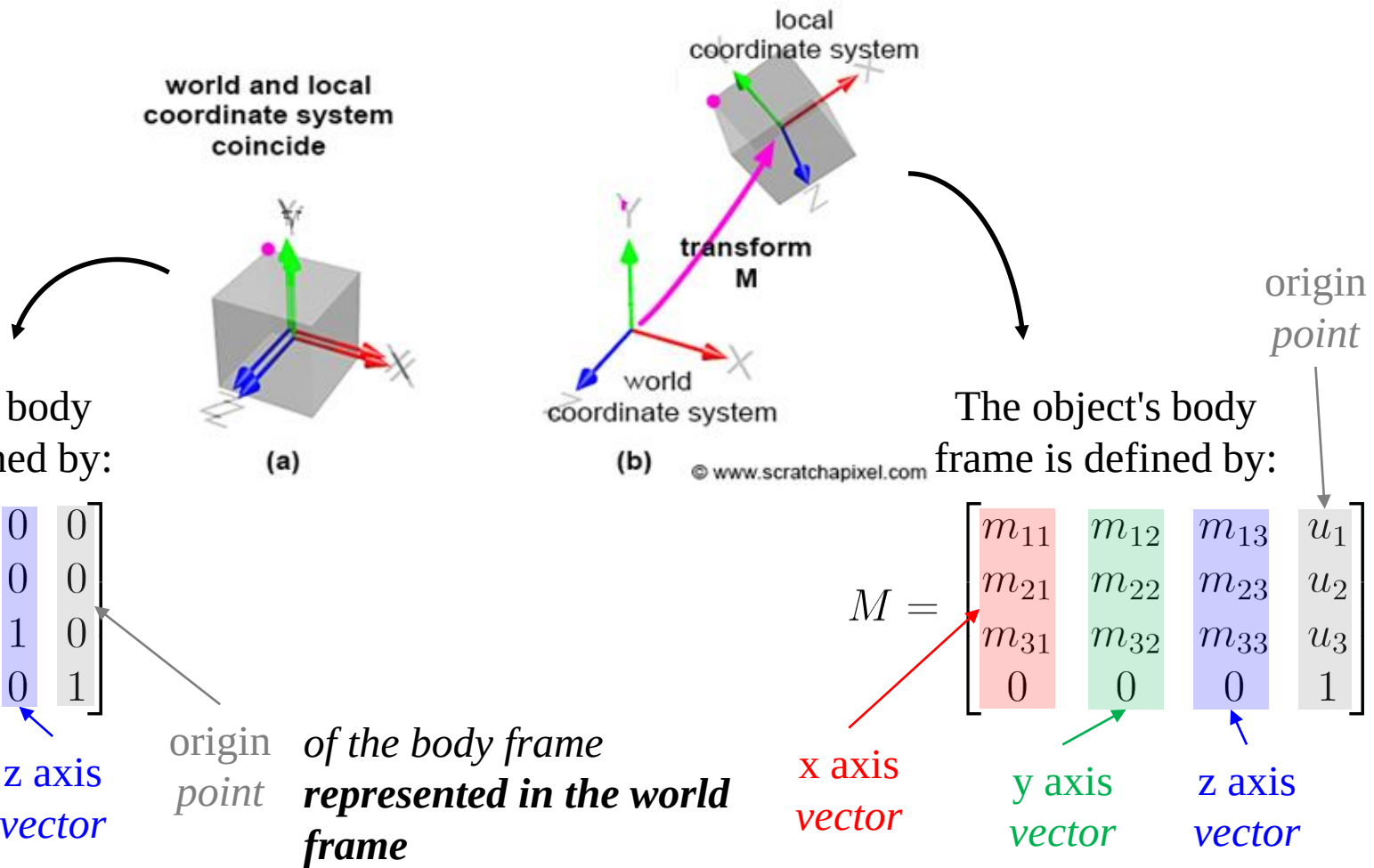
origin *point*

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & u_x \\ m_{21} & m_{22} & m_{23} & u_y \\ m_{31} & m_{32} & m_{33} & u_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} u_x \\ u_y \\ u_z \\ 1 \end{bmatrix}$$

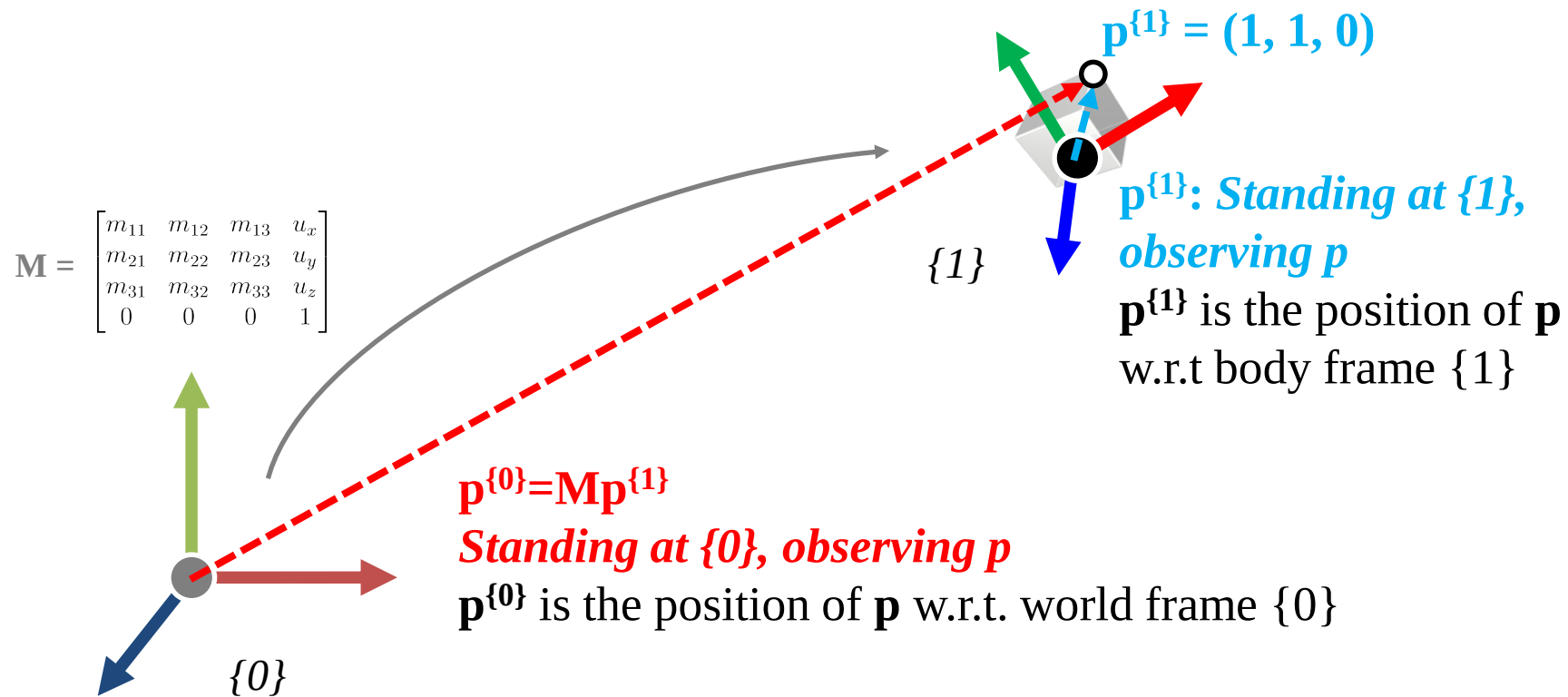
2) Defining a New Frame (w.r.t. World Frame)



Examples



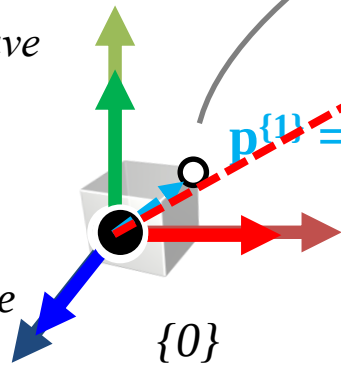
3) Changing the Representation of a Point (from **Body Frame** to **World Frame**)



3) Changing the Representation of a Point (from **Body Frame** to **World Frame**) Because...

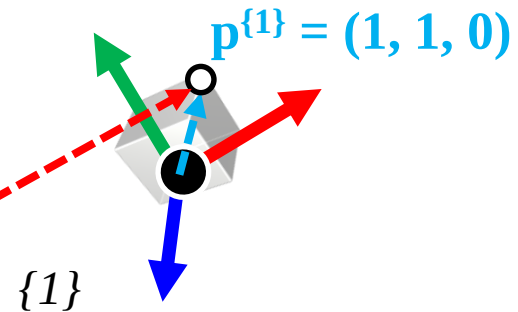
$$\mathbf{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & u_x \\ m_{21} & m_{22} & m_{23} & u_y \\ m_{31} & m_{32} & m_{33} & u_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

*Let's say we have
the same cube
object whose
body frame
coincides with
the world frame*



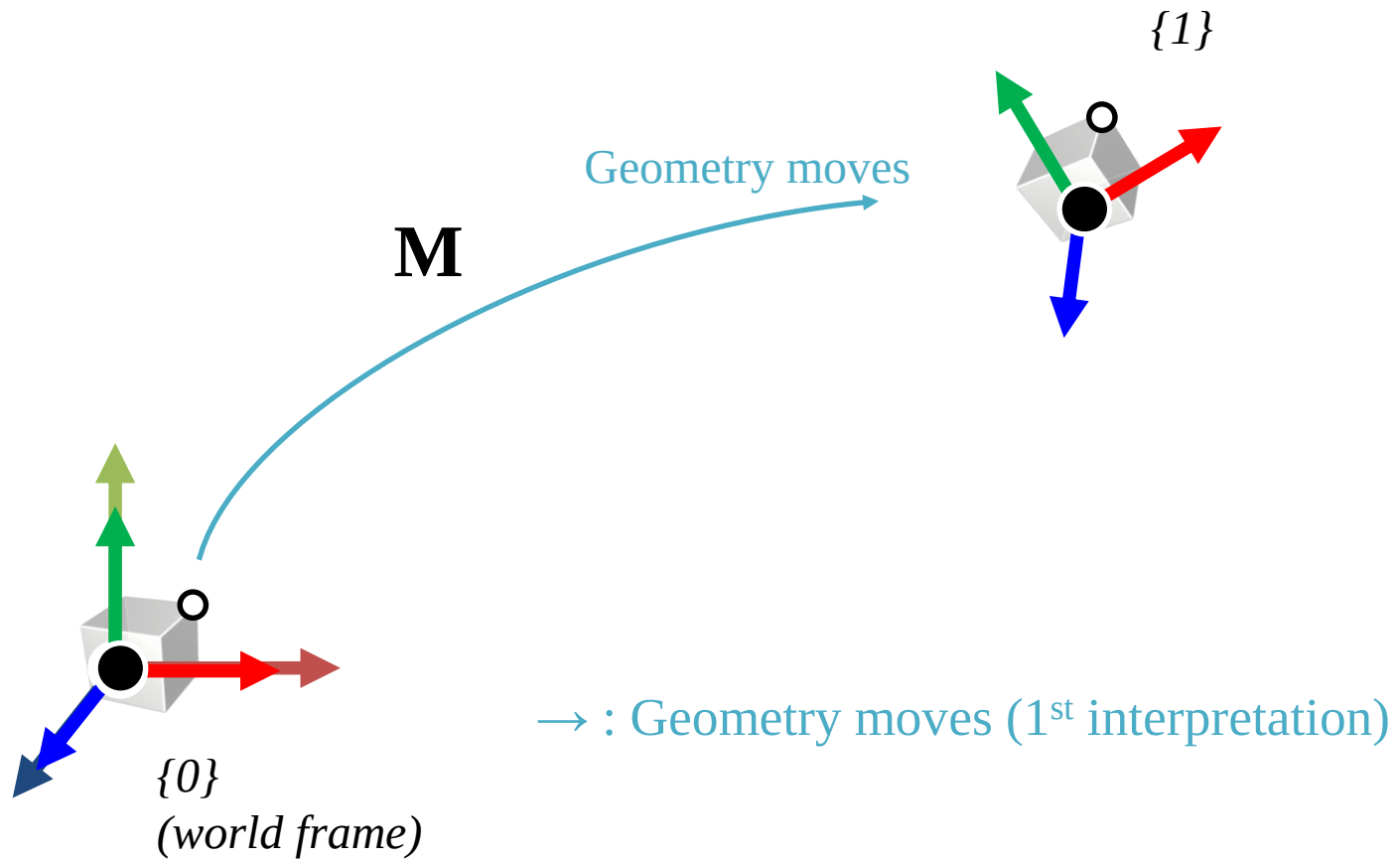
$$p^{\{1\}} = (1, 1, 0)$$

$$p^{\{0\}} = \mathbf{M}p^{\{1\}}$$

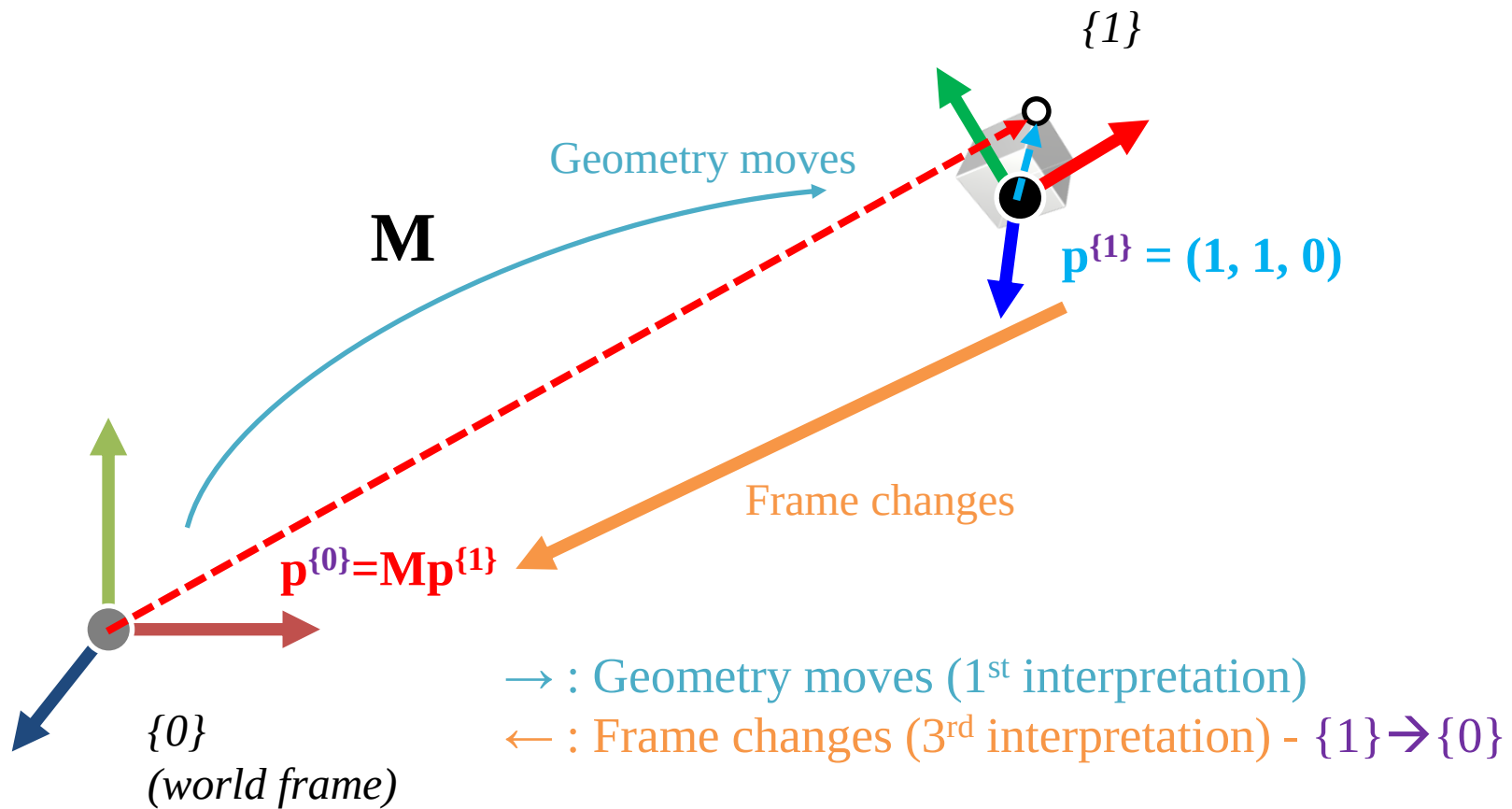


*Then, it's a just story of
transforming a geometry!*

Direction Depends on Interpretation



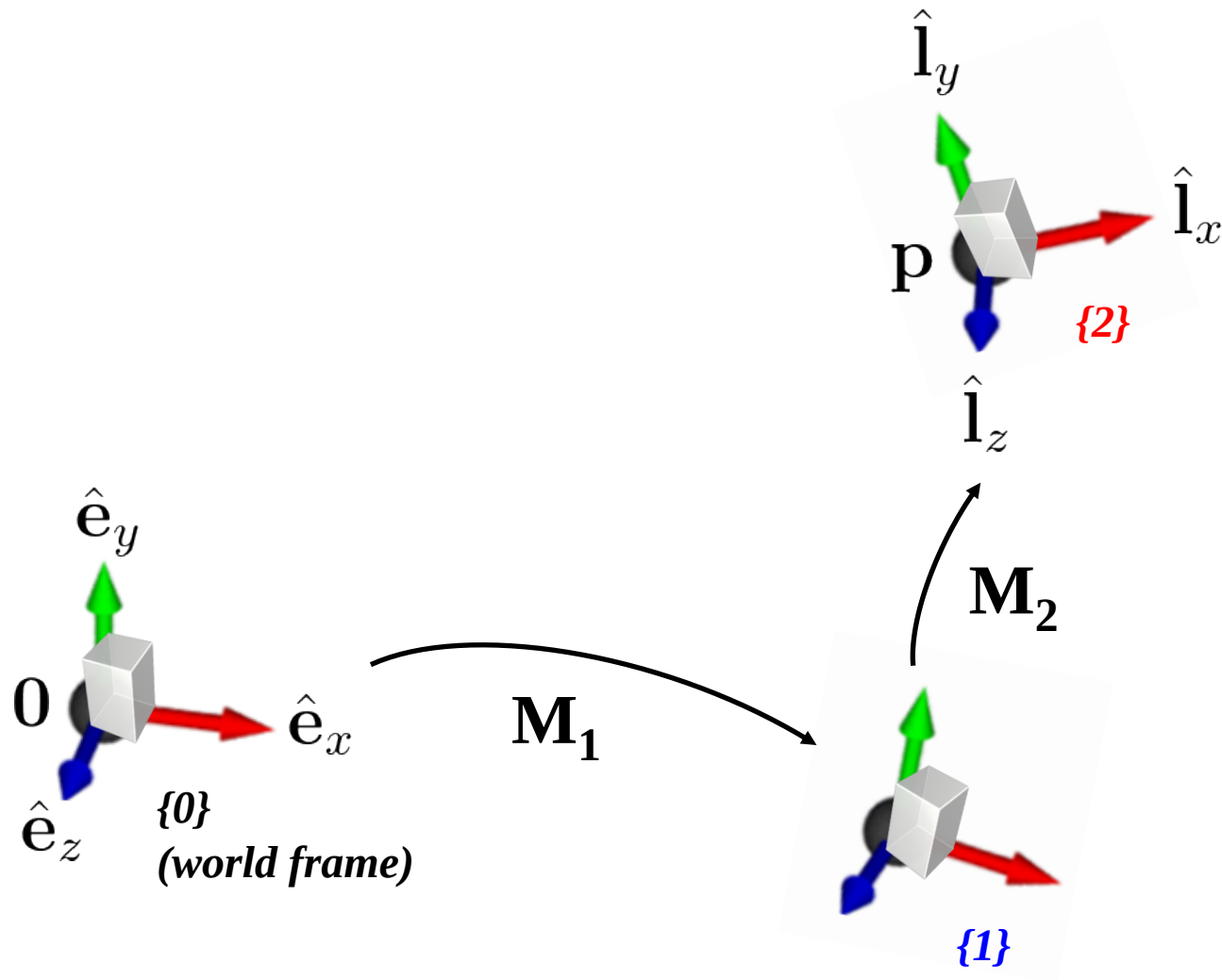
Direction Depends on Interpretation



Quiz 2

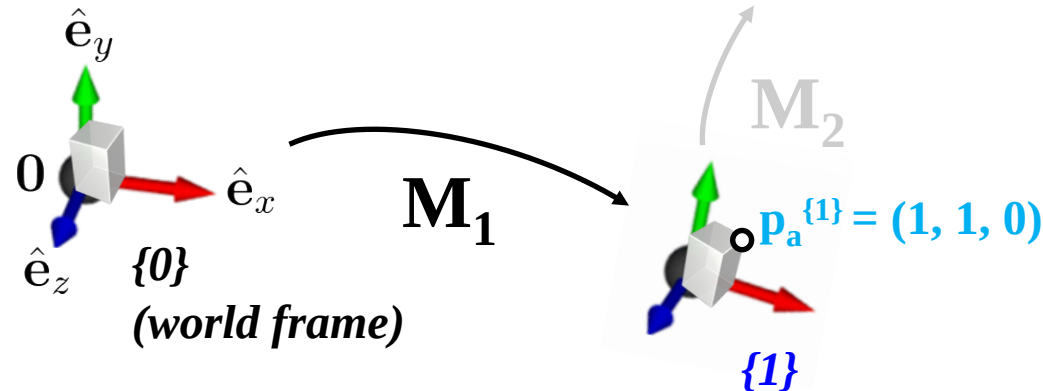
- Go to <https://www.slido.com/>
- Join #cg-ys
- Click "Polls"
- Submit your answer in the following format:
 - **Student ID: Your answer**
 - e.g. **2024123456: 4.0**
- Note that your quiz answer must be submitted **in the above format** to be counted for attendance.

These Concepts Apply to Any Reference Frame!



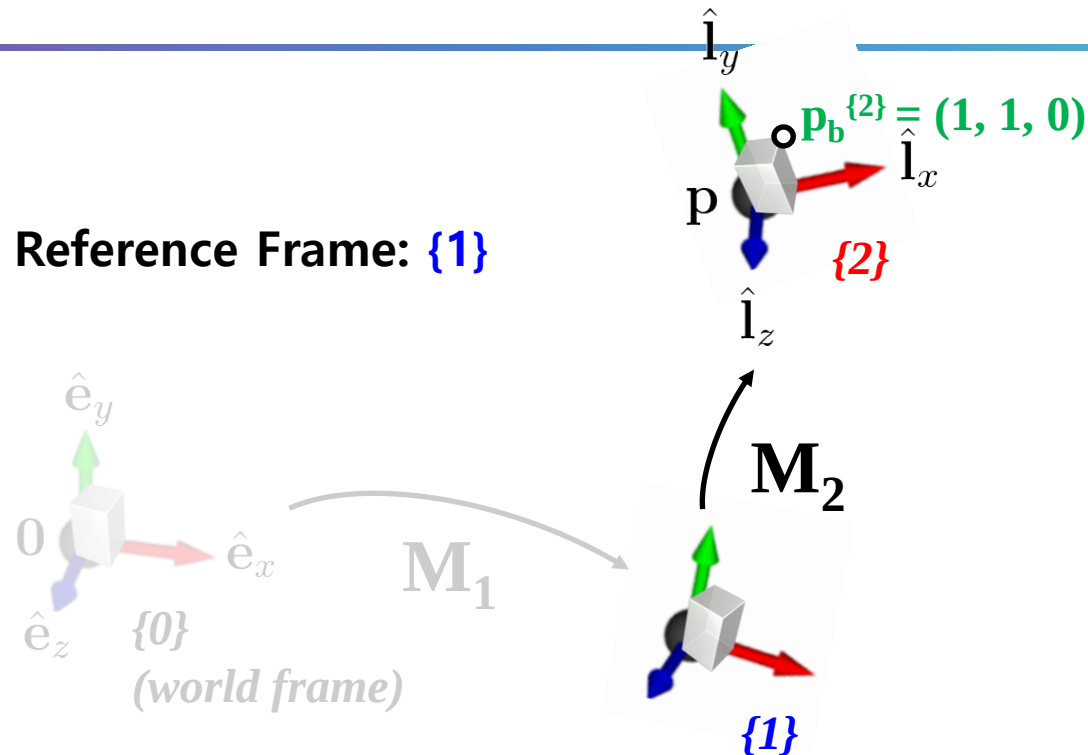
$\{0\}$ to $\{1\}$

Reference Frame: $\{0\}$



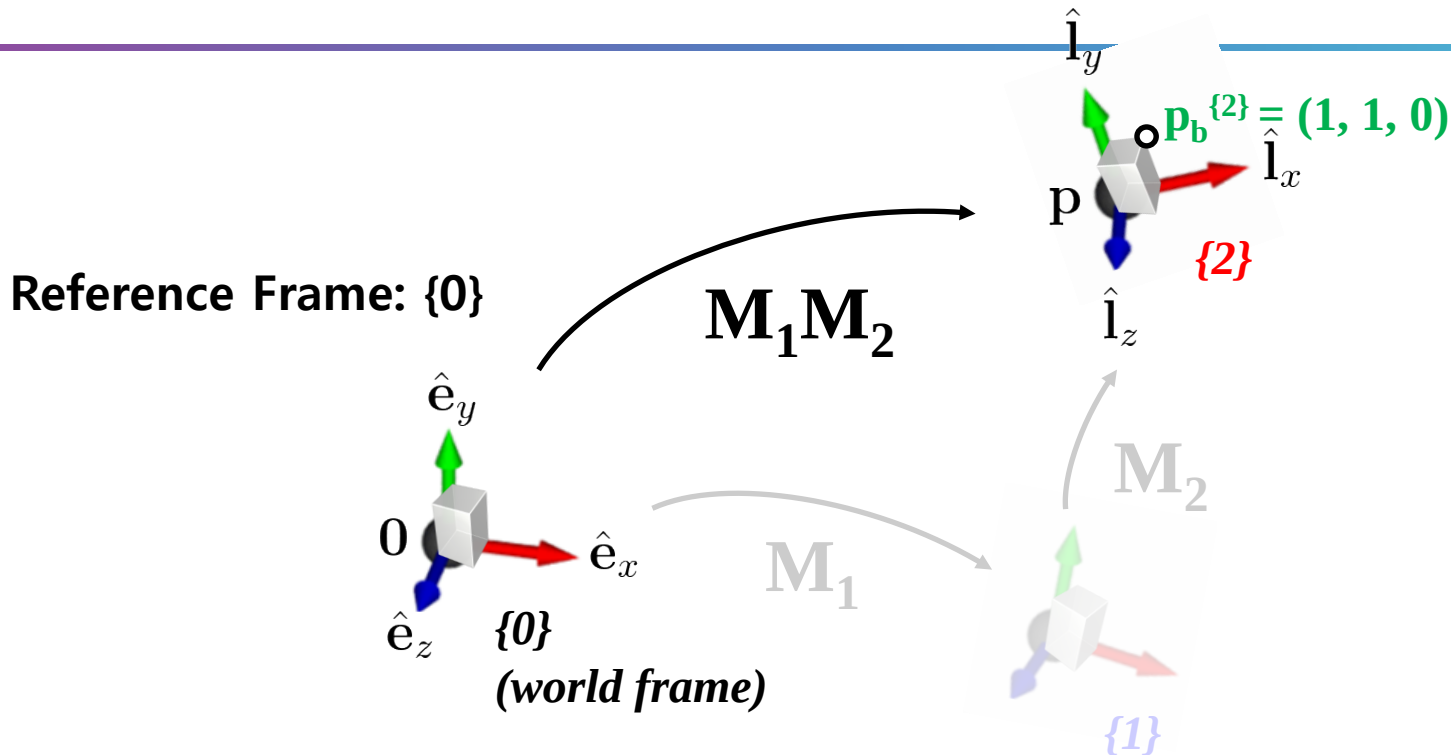
- 1) M_1 transforms the geometry expressed in $\{0\}$
- 2) M_1 defines frame $\{1\}$ w.r.t. $\{0\}$
- 3) M_1 changes the representation of the point from $\{1\}$ to $\{0\}$
 - $p_a^{\{0\}} = M_1 p_a^{\{1\}}$

{1} to {2}



- 1) M_2 transforms the geometry expressed in {1}
- 2) M_2 defines frame {2} w.r.t. {1}
- 3) M_2 changes the representation of the point from {2} to {1}
 - $\mathbf{p}_b^{\{1\}} = M_2 \mathbf{p}_b^{\{2\}}$

{0} to {2}

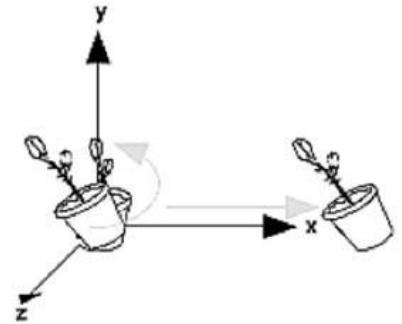


- 1) M_1M_2 transforms the geometry expressed in {0}
- 2) M_1M_2 defines frame {2} w.r.t. {0}
- 3) M_1M_2 changes the representation of the point from {2} to {0}
 - $\mathbf{p}_b^{\{1\}} = M_2\mathbf{p}_b^{\{2\}}$, $\mathbf{p}_b^{\{0\}} = M_1\mathbf{p}_b^{\{1\}} = M_1M_2\mathbf{p}_b^{\{2\}}$

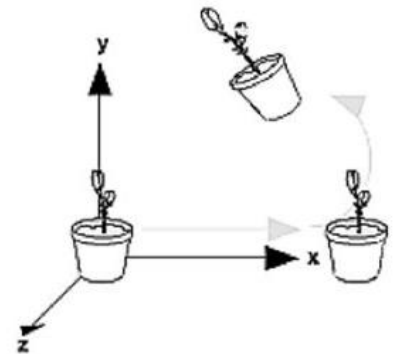
Interpretation of Composite Transformations

Revisit: Order Matters!

- If T and R are matrices representing affine transformations,
- $\mathbf{p}' = \mathbf{T}\mathbf{R}\mathbf{p}$
 - First apply transformation R to point $\mathbf{p}$, then apply transformation T to transformed point $\mathbf{R}\mathbf{p}$
- $\mathbf{p}' = \mathbf{R}\mathbf{T}\mathbf{p}$
 - First apply transformation T to point $\mathbf{p}$, then apply transformation R to transformed point $\mathbf{T}\mathbf{p}$
- *w.r.t. world frame!*



Rotate then Translate



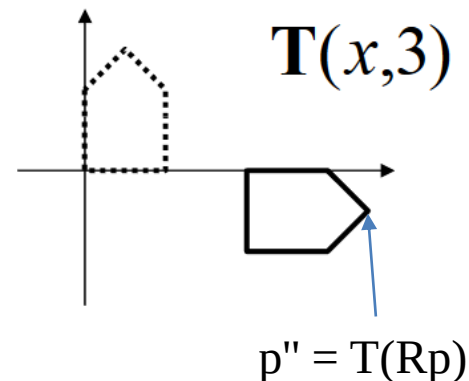
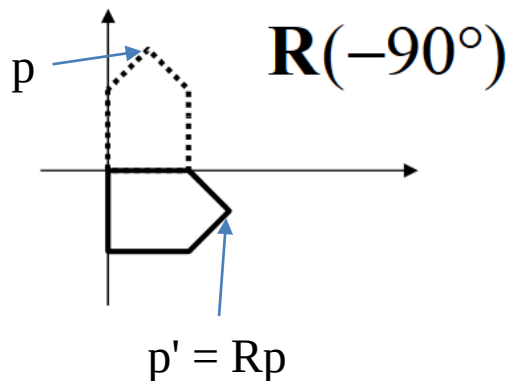
Translate then Rotate

Interpretation of Composite Transformations #1

- An example transformation:

$$M = \mathbf{T}(x,3) \cdot \mathbf{R}(-90^\circ)$$

- This is how we've interpreted so far:
 - R-to-L: Apply in *world frame*

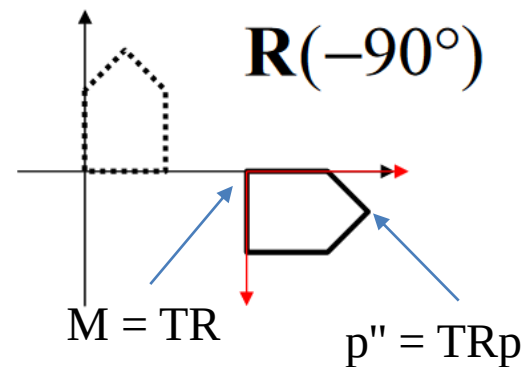
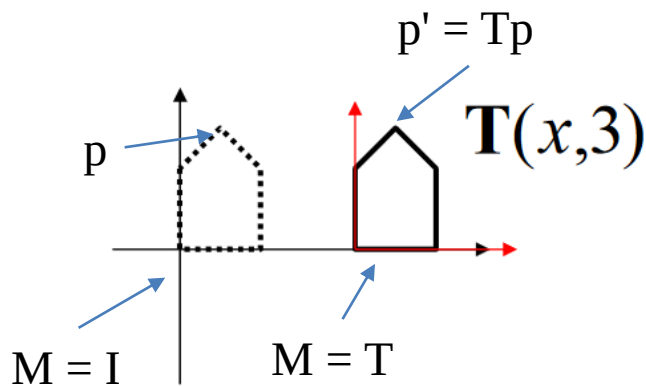


Interpretation of Composite Transformations #2

- An example transformation:

$$M = \mathbf{T}(x,3) \cdot \mathbf{R}(-90^\circ)$$

- **Another interpretation of the SAME matrix:**
 - L-to-R: Apply in *body frame*



(M defines the *body frame* w.r.t. world frame)

Pre-(left) & Post-(right) Multiplication

- $\mathbf{p}' = \mathbf{M}_1\mathbf{M}_2\mathbf{p}$ (pre-multiplication by $\mathbf{M}_1$)
 - (R-to-L)
 - 1) $\mathbf{M}_2\mathbf{p}$: Apply $\mathbf{M}_2$ w.r.t. world frame to a point $\mathbf{p}$ represented in world frame.
 - 2) $\mathbf{M}_1\mathbf{M}_2\mathbf{p}$: Apply $\mathbf{M}_1$ w.r.t. world frame to a point $\mathbf{M}_2\mathbf{p}$ represented in world frame.
- $\mathbf{p}' = \mathbf{M}_1\mathbf{M}_2\mathbf{p}$ (post-multiplication by $\mathbf{M}_2$)
 - (L-to-R)
 - 1') $\mathbf{M}_1$: Apply $\mathbf{M}_1$ w.r.t. body frame $\mathbf{I}$ (world frame) to update body frame to $\mathbf{M}_1$
 - 2') $\mathbf{M}_1\mathbf{M}_2$: Apply $\mathbf{M}_2$ w.r.t. body frame $\mathbf{M}_1$ to update body frame to $\mathbf{M}_1\mathbf{M}_2$
 - 3') $\mathbf{M}_1\mathbf{M}_2\mathbf{p}$: Locate $\mathbf{p}$ in body frame $\mathbf{M}_1\mathbf{M}_2$

→ Same formula, different interpretation

[Demo] L-to-R & R-to-L Interpretation

Transformation demo

An interactive demo for experimenting with 2D transformation matrix composition.

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



<https://observablehq.com/@esperanc/transformation-demo>

- Add translation and linear transforms in various orders with '+' buttons.
- Drag the slider to see the matrix value change and the shape transform.
- Interpret the meaning of composite transformations in L-to-R and R-to-L order.

Lab Session

- Now, let's start the lab today.