Computer Graphics

5 - Affine Space, Rendering Pipeline

Yoonsang Lee
Spring 2019

Topics Covered

- Affine Space & Coordinate-Free Concepts
- Meanings of an Affine Matrix
- Rendering Pipeline
  - Vertex Processing
    - Modeling transformation

Affine Space & Coordinate-Free Concepts

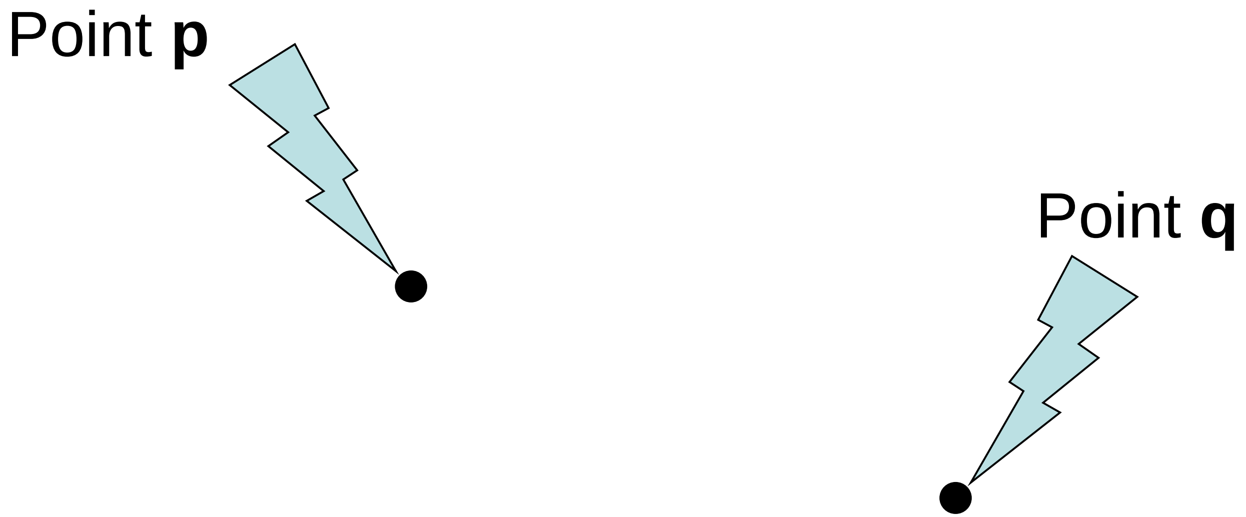
Coordinate-invariant (Coordinate-free)

- Traditionally, computer graphics packages are implemented using *homogeneous coordinates*.

- We will see *affine space* and *coordinate-invariant geometric programming* concepts and their relationship with the homogeneous coordinates.

- Because of historical reasons, it has been called *"coordinate-free"* geometric programming.

Points



- What is the “sum” of these two "points" ?

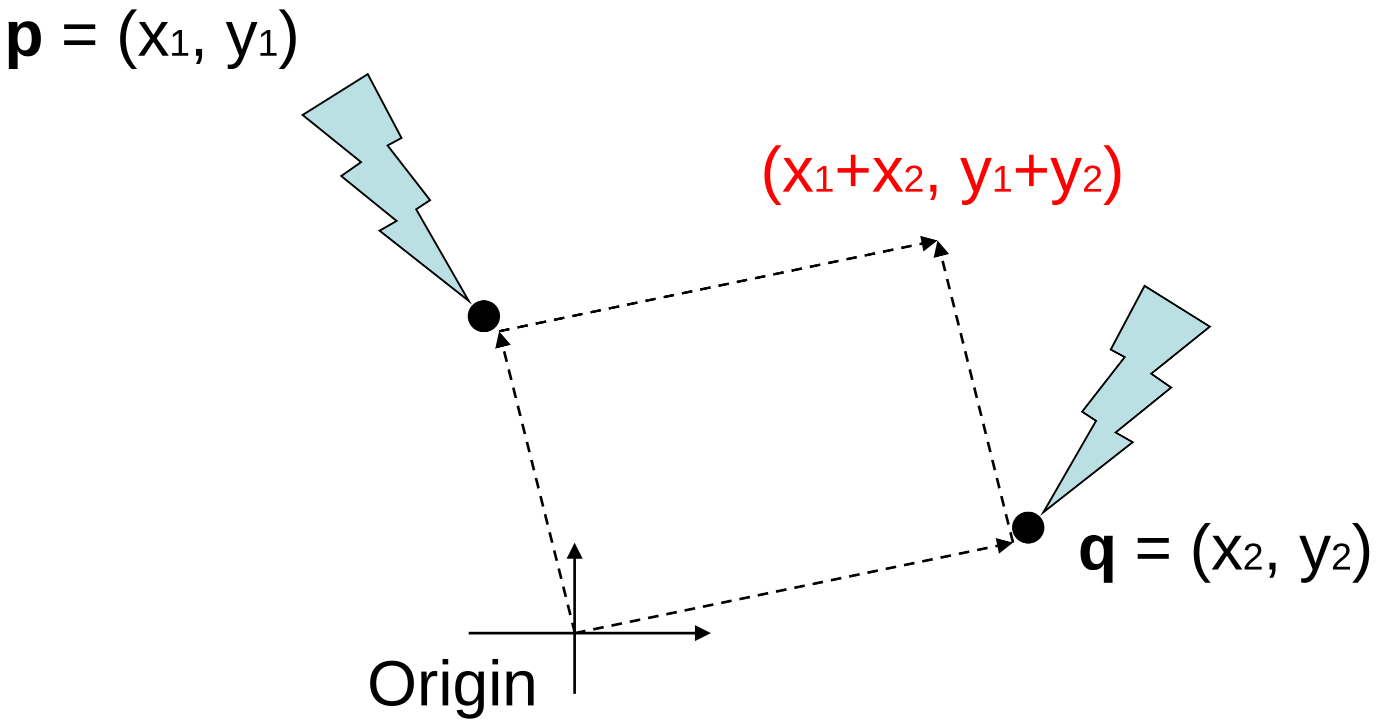
If you assume coordinates, …

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

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

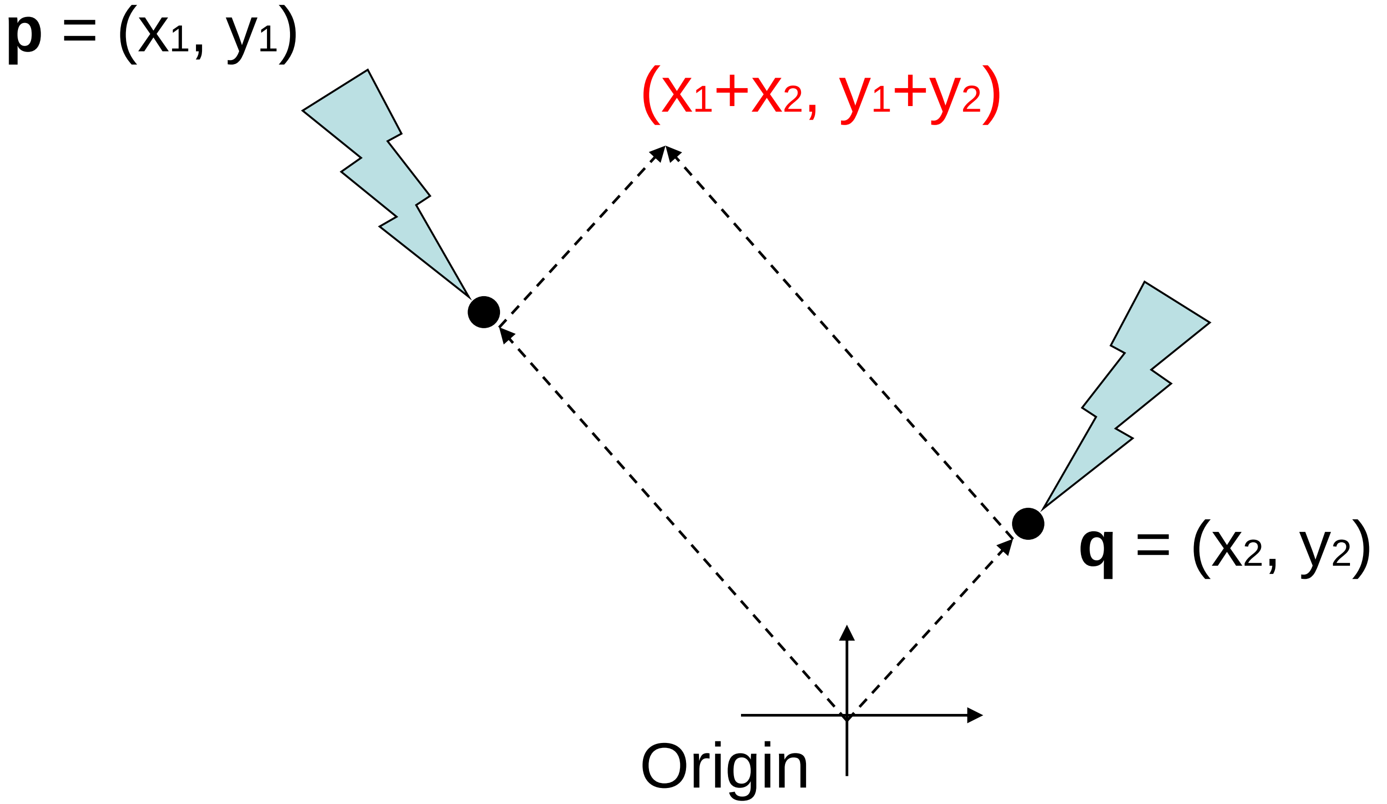
- The sum is (x_1+x_2, y_1+y_2)
  - Is it correct ?
  - Is it geometrically meaningful ?

If you assume coordinates, …



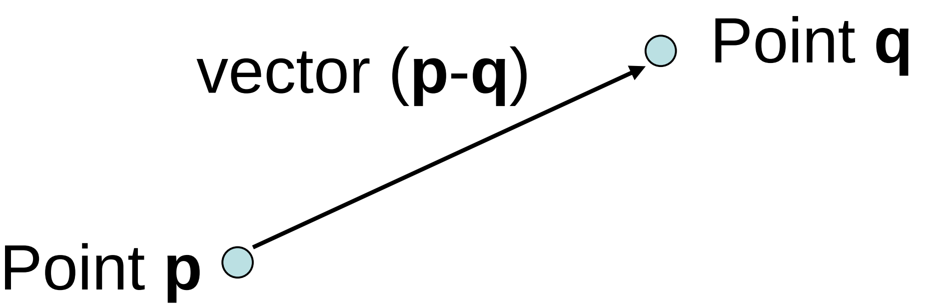
- Vector sum
  - (x_1, y_1) and (x_2, y_2) are considered as vectors from the origin to **p** and **q**, respectively.

If you select a different origin, …



- If you choose a different coordinate frame, you will get a different result

Points and Vectors



- A ***point*** is a position specified with coordinate values.
- A ***vector*** is specified as the difference between two points.
- If an ***origin*** is specified, then a **point** can be represented by a **vector from the origin.**
- But, a point is still not a vector in ***coordinate-free*** concepts.

Points & Vectors are Different!

- Mathematically (and physically),
- *Points* are **locations in space**.
- *Vectors* are **displacements in space**.

- An analogy with time:
- *Times* (or datetimes) are **locations in time**.
- *Durations* are **displacements in time**.

Vector and Affine Spaces

- ***Vector space***
  - Includes vectors and related operations
  - No points

- ***Affine space***
  - Superset of vector space
  - Includes vectors, points, and related operations

Vector spaces

- A ***vector space*** consists of
  - Set of vectors, together with
  - Two operations: addition of vectors and multiplication of vectors by scalar numbers

- A ***linear combination*** of vectors is also a vector

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

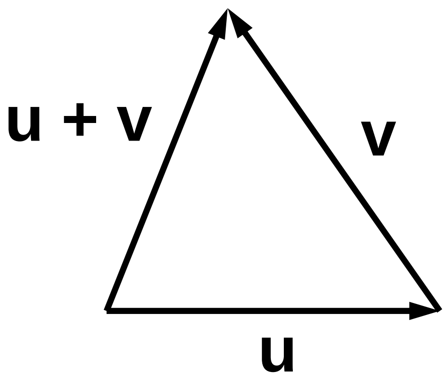
Affine Spaces

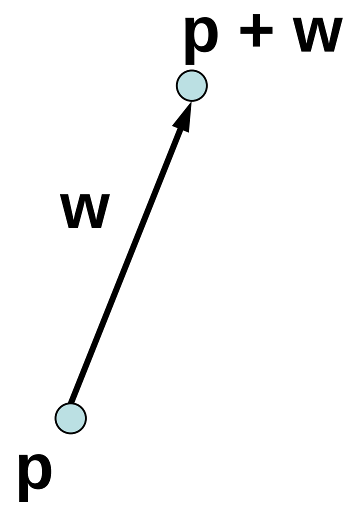
- An ***affine space*** consists of
  - Set of points, an associated vector space, and
  - Two operations: the difference between two points and the addition of a vector to a point

Coordinate-Free Geometric Operations

- Addition
- Subtraction
- Scalar multiplication

Addition



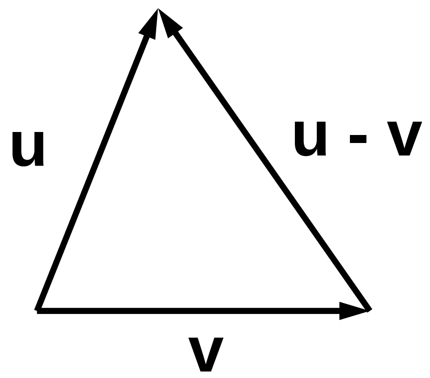


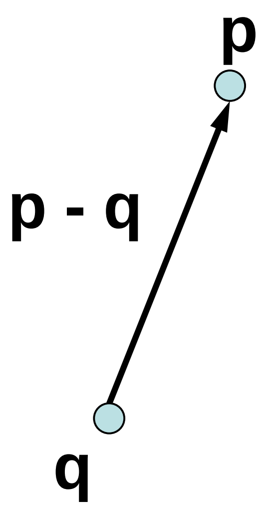
u + **v** is a vector

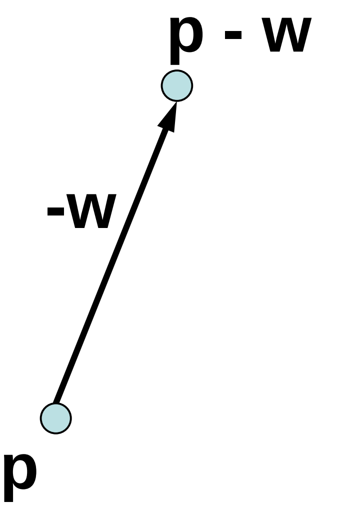
p + **w** is a point

u, v, w : vectors
p, q : points

Subtraction







u - **v** is a vector

p - **q** is a vector

p - **w** is a point

u, v, w : vectors
p, q : points

Scalar Multiplication

scalar • vector = vector

1 • point = point

0 • point = vector

c • point = (undefined)   if ($c \neq 0,1$)

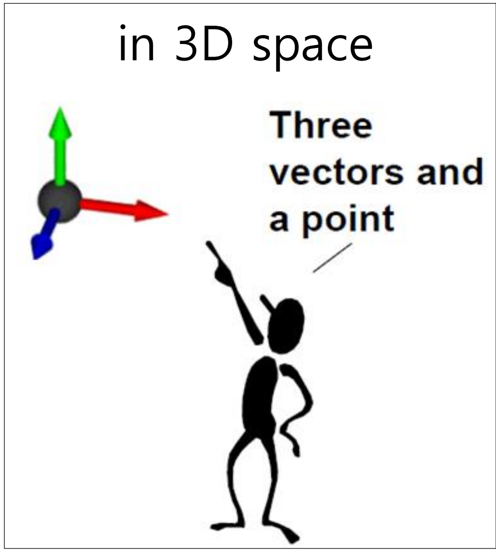
Affine Frame

- A ***frame*** is defined as a set of vectors $\{\mathbf{v}_i \mid i=1, \ldots, N\}$ and a point **o**
  - Set of vectors $\{\mathbf{v}_i\}$ are bases of the associate vector space
  - **o** is an origin of the frame
  - *N* is the dimension of the affine space
  - Any point **p** can be written as

$$\mathbf{p} = \mathbf{o} + c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_N\mathbf{v}_N$$

  - Any vector **v** can be written as

$$\mathbf{v} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_N\mathbf{v}_N$$



Summary

- In an affine space,

point + point = undefined		
point − point = vector		
point ± vector = point		
vector ± vector = vector		
scalar • vector = vector		
scalar • point = point		iff scalar = 1
	= vector	iff scalar = 0
	= undefined	otherwise

Points & Vectors in Homogeneous Coordinates

- In 3D spaces,
- A **point** is represented: (x, y, z, **1**)
- A **vector** can be represented: (x, y, z, **0**)

$(x_1, y_1, z_1, 1) + (x_2, y_2, z_2, 1) = (x_1+x_2, y_1+y_2, z_1+z_2, 2)$

point ***point*** ***undefined***

$(x_1, y_1, z_1, 1) - (x_2, y_2, z_2, 1) = (x_1-x_2, y_1-y_2, z_1-z_2, 0)$

point ***point*** ***vector***

$(x_1, y_1, z_1, 1) + (x_2, y_2, z_2, 0) = (x_1+x_2, y_1+y_2, z_1+z_2, 1)$

point ***vector*** ***point***

A Consistent Model

- **Behavior of affine frame coordinates is completely consistent with our intuition**
  - **Subtracting two points yields a vector**
  - **Adding a vector to a point produces a point**
  - **If you multiply a vector by a scalar you still get a vector**
  - **Scaling points gives a nonsense 4th coordinate element in most cases**

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ 1 \end{bmatrix} - \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 - b_1 \\ a_2 - b_2 \\ a_3 - b_3 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ 1 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{bmatrix} = \begin{bmatrix} a_1 + v_1 \\ a_2 + v_2 \\ a_3 + v_3 \\ 1 \end{bmatrix}$$

Points & Vectors in Homogeneous Coordinates

- Multiplying affine transformation matrix to a point and a vector:

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

point → point

vector → vector

- Note that translation is not applied to a vector!

Quiz #1

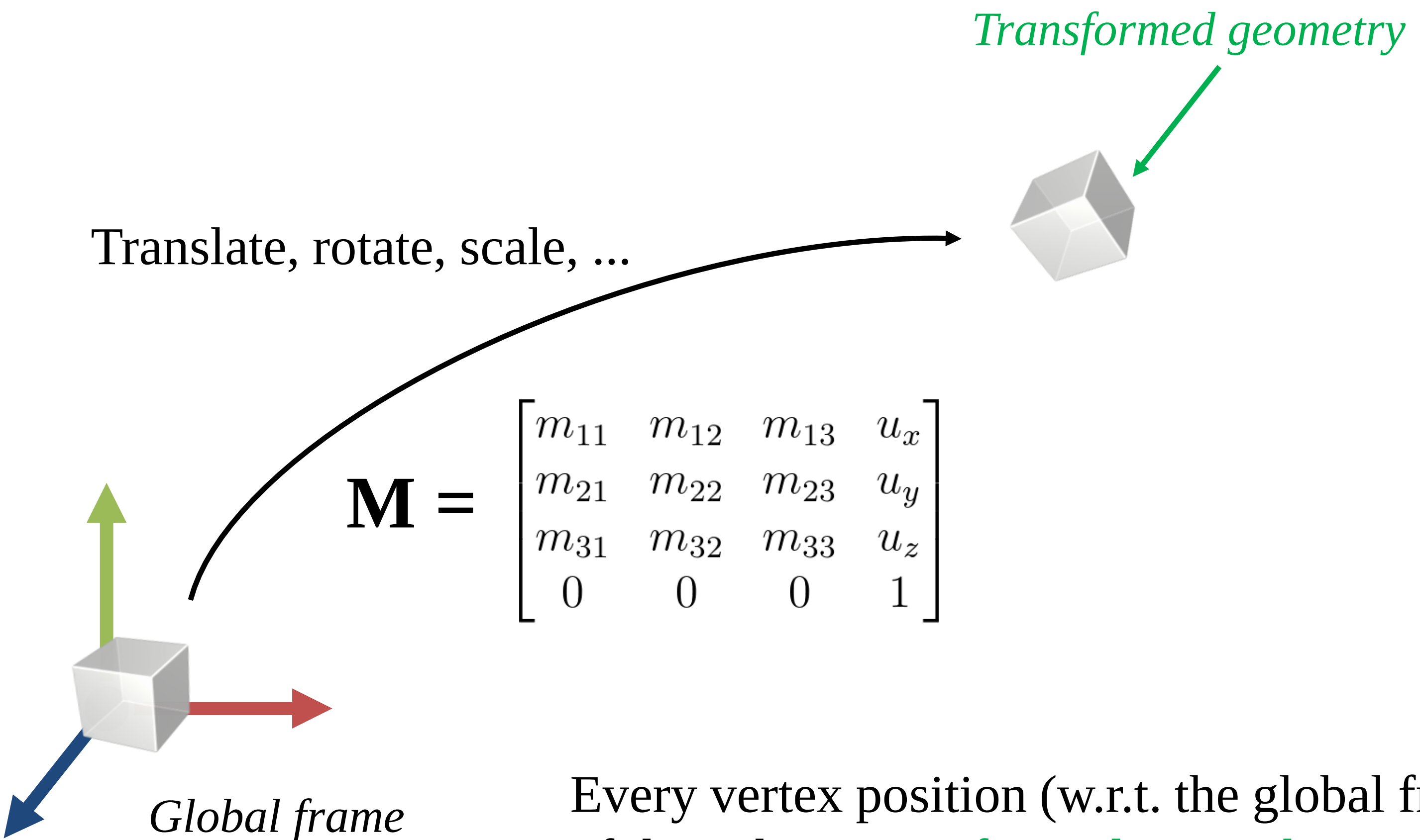
- Go to https://www.slido.com/
- Join #cg-hyu
- Click “Polls”

- Submit your answer in the following format:
  - **Student ID: Your answer**
  - **e.g. 2017123456: 4)**

- Note that you must submit all quiz answers in the above format to be checked for “attendance”.

Meanings of an Affine Matrix

1) A 4x4 Affine Transformation Matrix transforms a Geometry

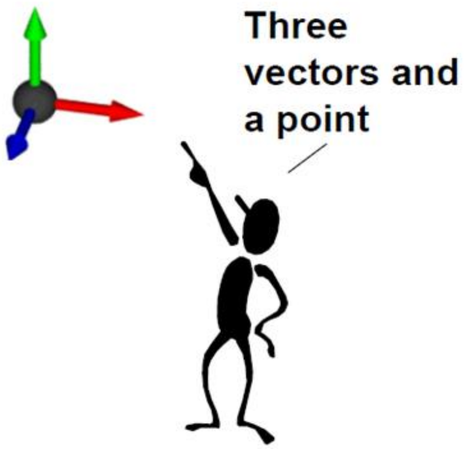


$$\mathbf{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & u_x \\ m_{21} & m_{22} & m_{23} & u_y \\ m_{31} & m_{32} & m_{33} & u_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Every vertex position (w.r.t. the global frame) of the cube is transformed to another position (w.r.t. the global frame)

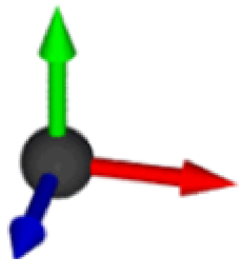
Review: Affine Frame

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



Global Frame

- A **global 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 "global frame"

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

x axis *vector*

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

y axis *vector*

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

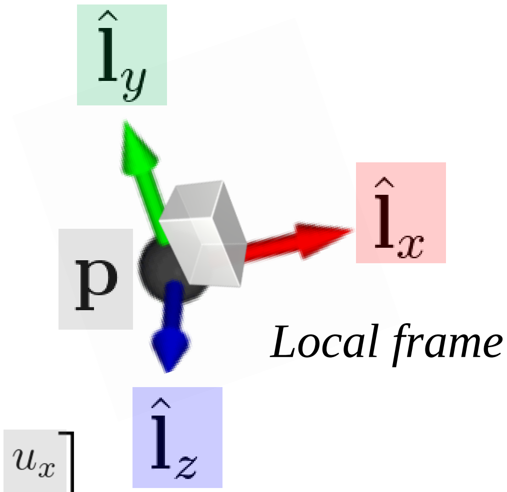
z axis *vector*

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

origin *point*

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

2) A 4x4 Affine Transformation Matrix defines an Affine Frame w.r.t. Global Frame



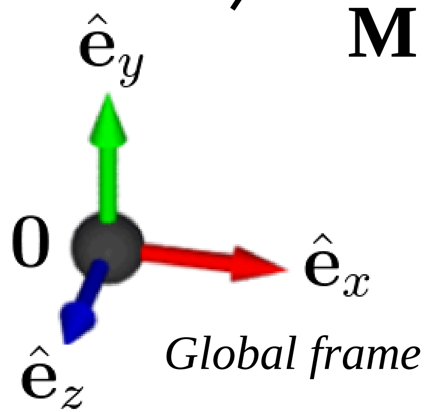
$$\mathbf{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & u_x \\ m_{21} & m_{22} & m_{23} & u_y \\ m_{31} & m_{32} & m_{33} & u_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

x axis *vector*

y axis *vector*

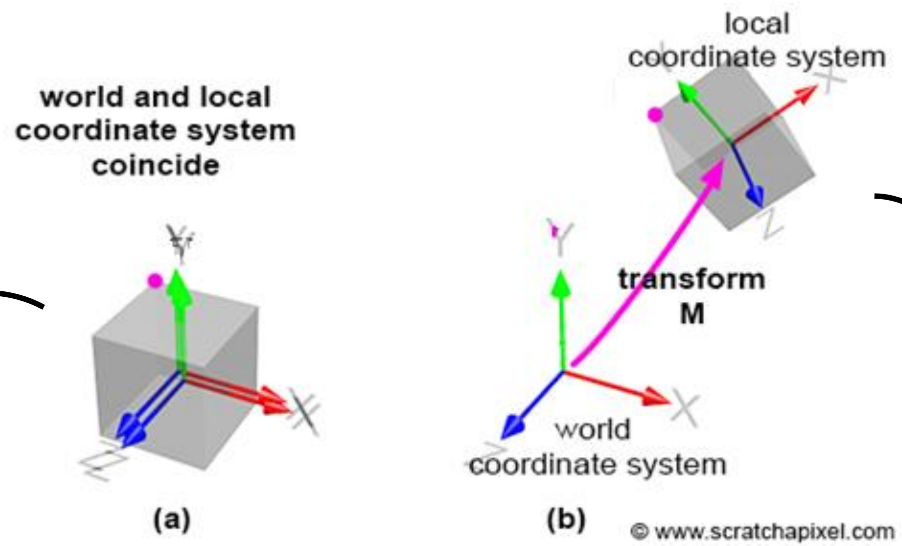
z axis *vector*

origin *point*



: The axis vectors and origin point of the **object's local frame represented in the global frame**

Examples



This local frame is defined by:

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

x axis *vector*

y axis *vector*

z axis *vector*

origin *point*

of the local frame ***represented in the global frame***

This local frame is defined by:

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} & u_1 \\ m_{21} & m_{22} & m_{23} & u_2 \\ m_{31} & m_{32} & m_{33} & u_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

origin *point*

x axis *vector*

y axis *vector*

z axis *vector*

Quiz #2

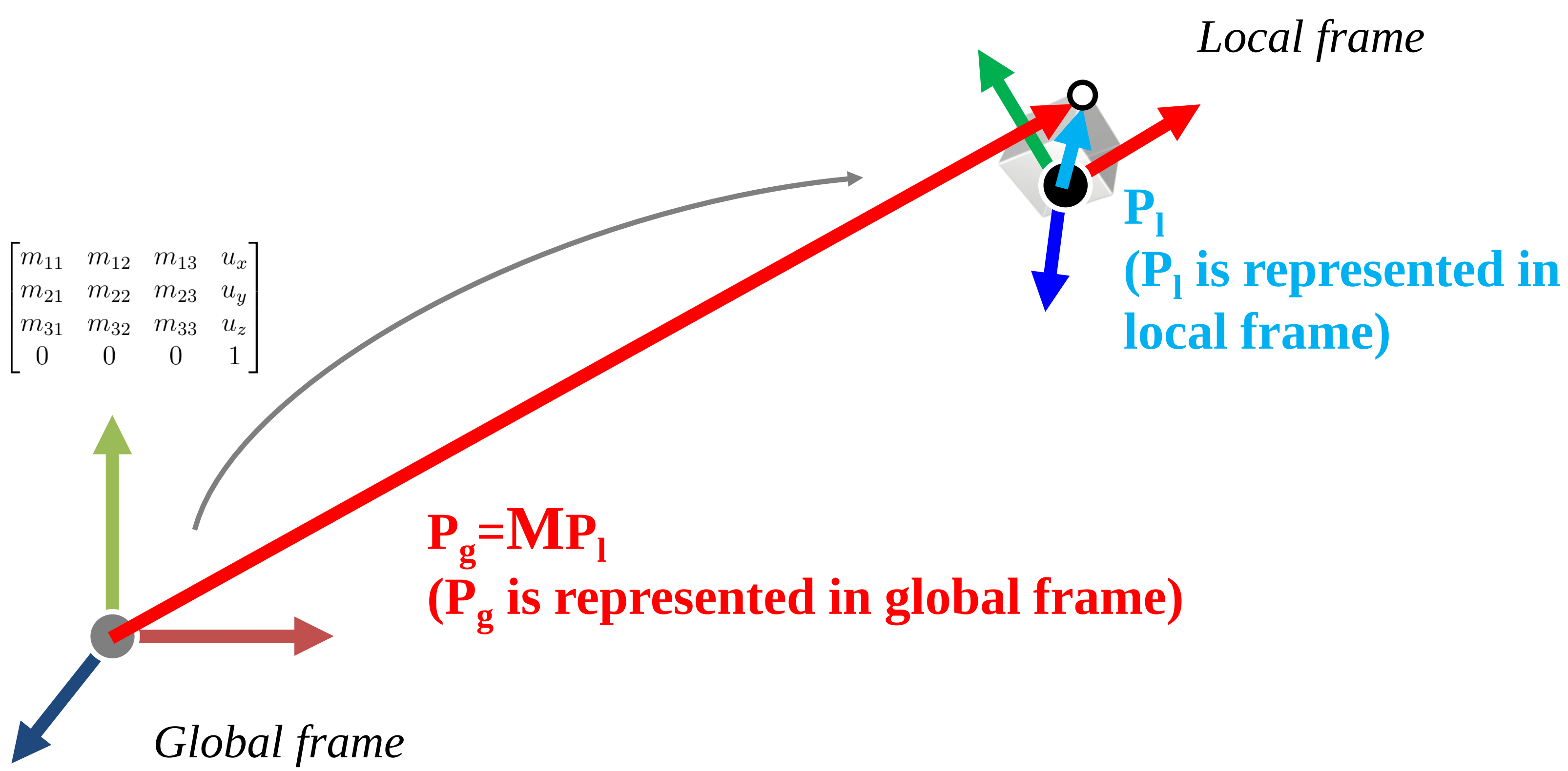
- Go to https://www.slido.com/
- Join #cg-hyu
- Click “Polls”

- Submit your answer in the following format:
  - **Student ID: Your answer**
  - **e.g. 2017123456: 4)**

- Note that you must submit all quiz answers in the above format to be checked for “attendance”.

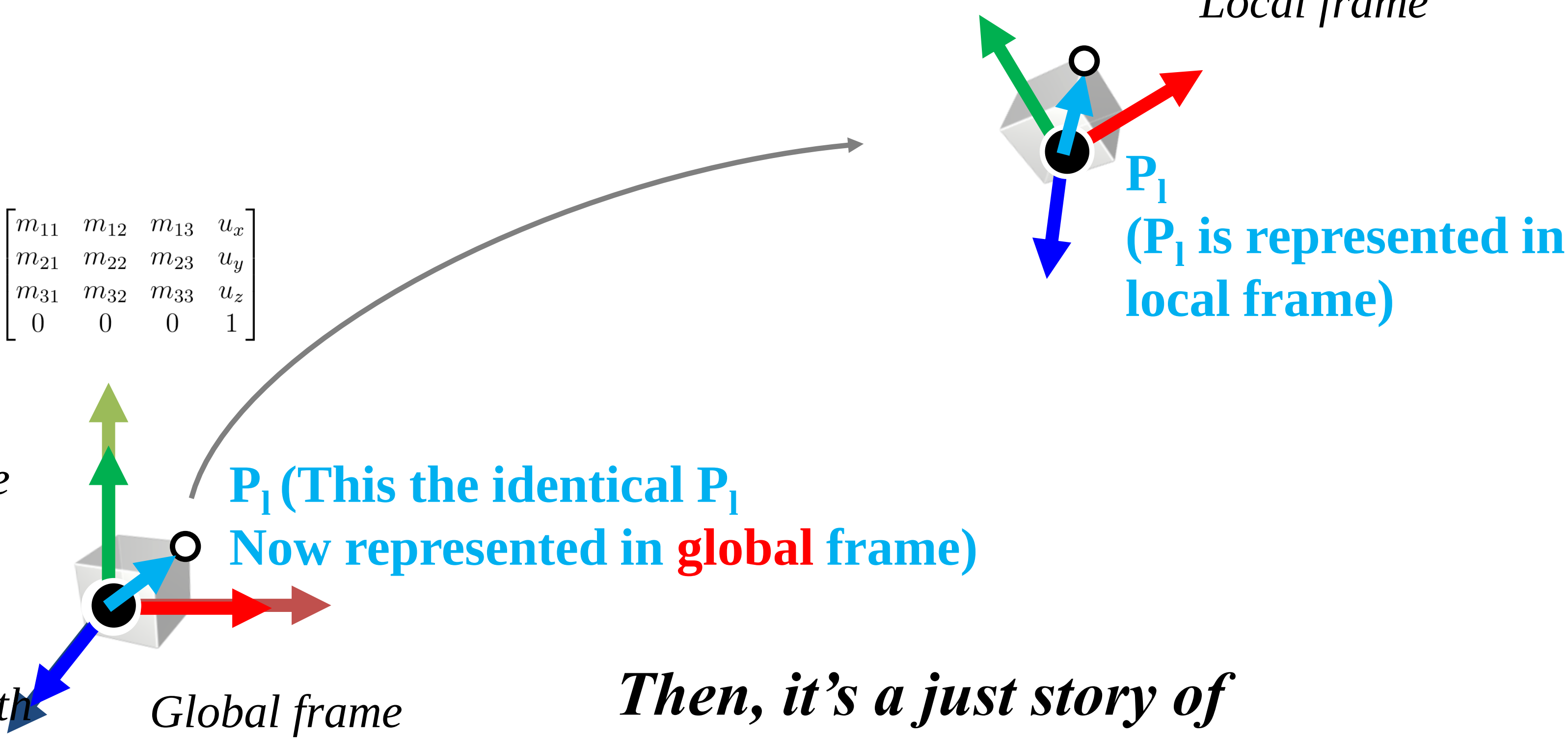
3) A 4x4 Affine Transformation Matrix transforms a Point Represented in an Affine Frame to a Point Represented in Global Frame

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



3) A 4x4 Affine Transformation Matrix transforms a Point Represented in an Affine Frame to a Point Represented in Global 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 and its local frame coincident with the global frame

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

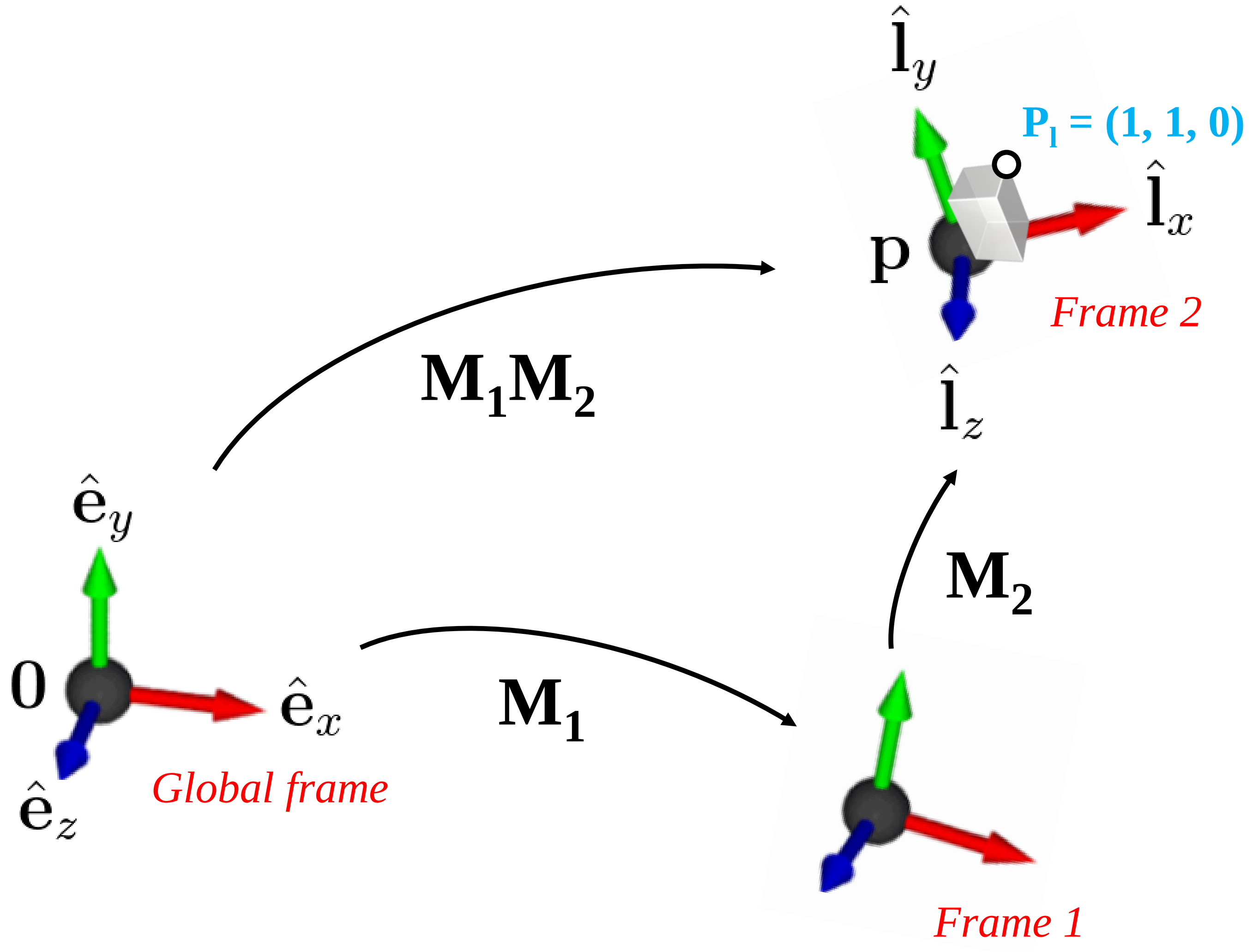
Quiz #3

- Go to https://www.slido.com/
- Join #cg-hyu
- Click “Polls”

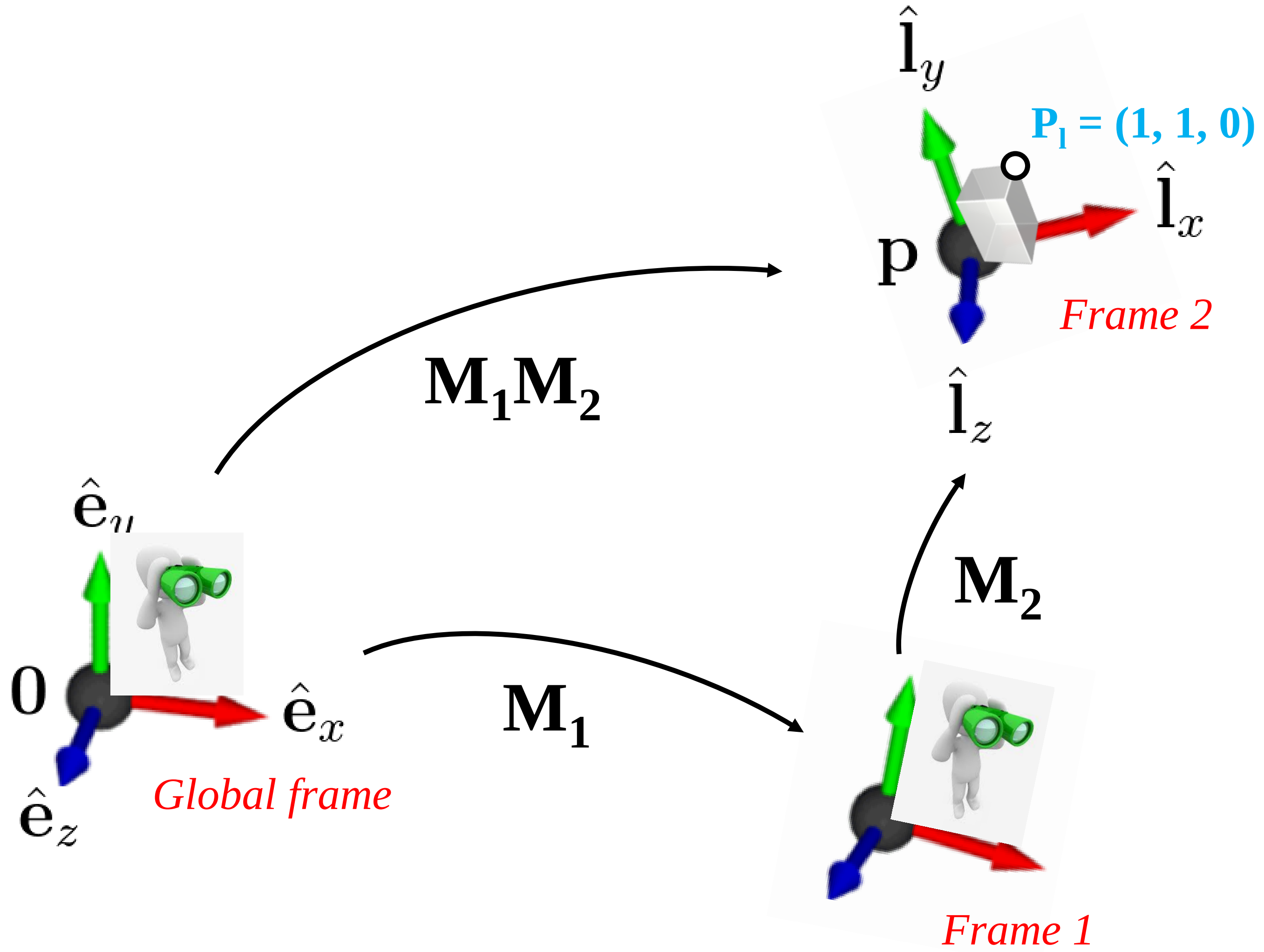
- Submit your answer in the following format:
  - **Student ID: Your answer**
  - **e.g. 2017123456: 4)**

- Note that you must submit all quiz answers in the above format to be checked for “attendance”.

All these concepts works if the original frame is not global frame!



Think it as: Standing at a frame and observing the object



Left & Right Multiplication

- p' = **R**Tp (left-multiplication by **R**)
  - Apply transformation **R** to point Tp w.r.t. global coordinates
  - **Standing at global frame and applying R then T to point p**

- p' = T**R**p (right-multiplication by **R**)
  - Apply transformation **R** to point Tp w.r.t. local coordinates
  - **Standing at frame T and applying R to point p**

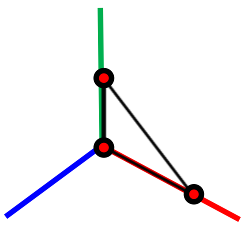
Rendering Pipeline

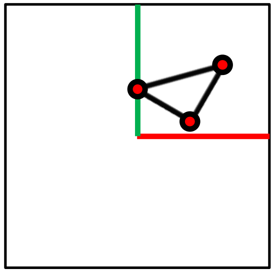
Rendering Pipeline

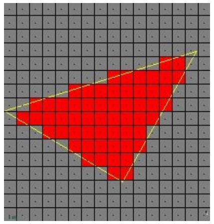
- A conceptual model that describes what steps a graphics system needs to perform to render a 3D scene to a 2D image.

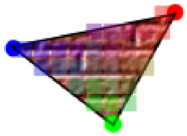
- Also known as **graphics pipeline**.

Rendering Pipeline









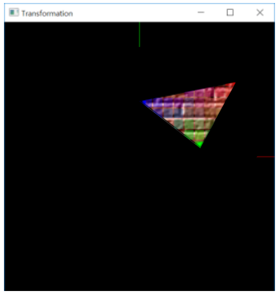
vertex processing → **rasterization** → **fragment processing** → output merging

vertex processing: performs a sequence of vertex transformations

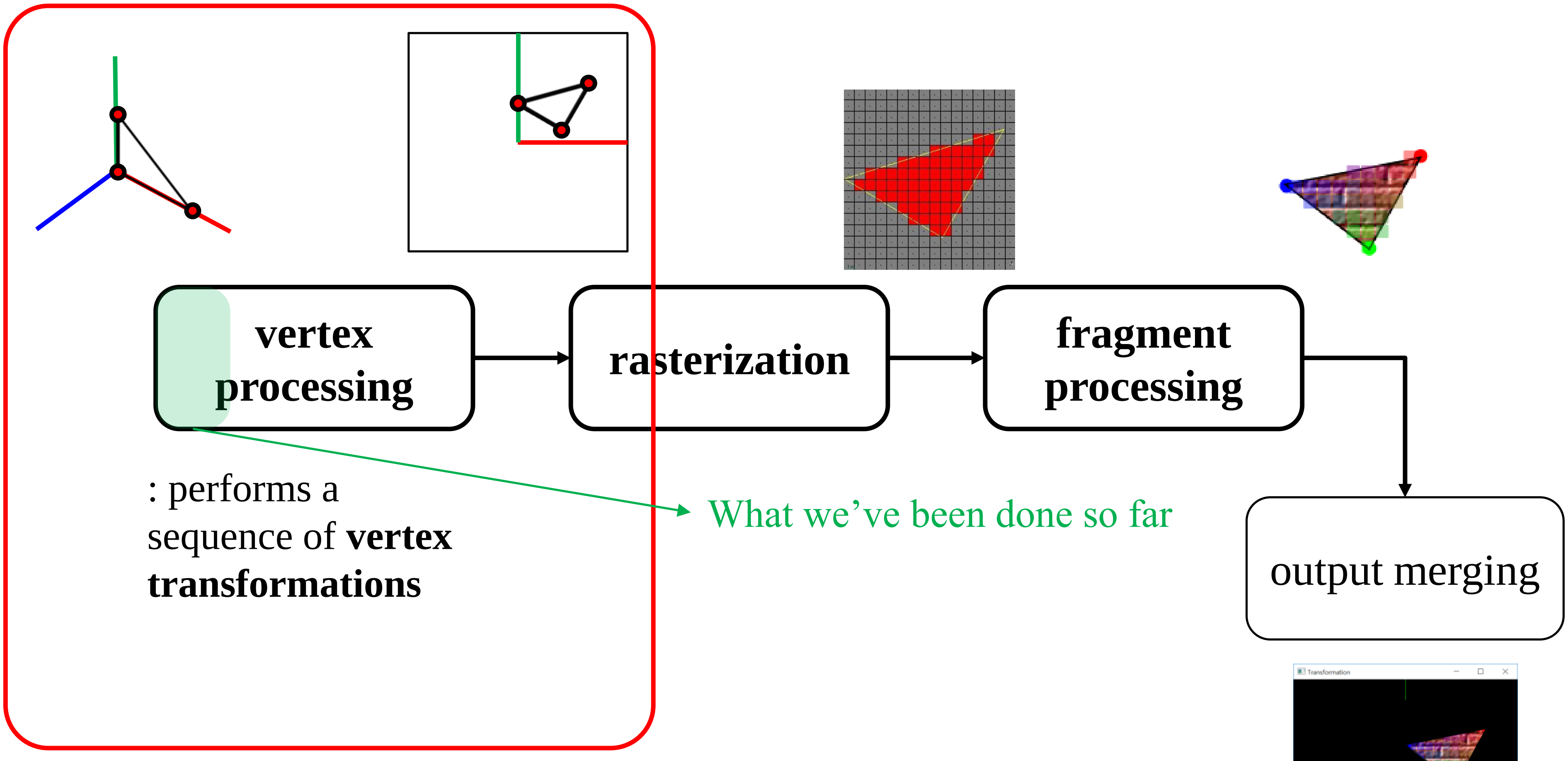
rasterization: assembles polygons & converts each polygon into a set of fragments (pixels)

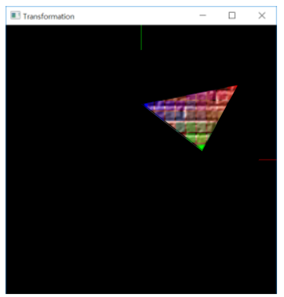
fragment processing: determines color of each fragment with light & texture

output merging



Rendering Pipeline





→ We'll see today & next lecture

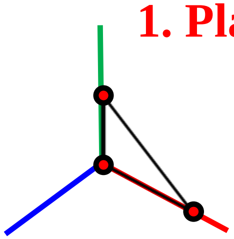
Vertex Processing

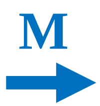
Set vertex positions

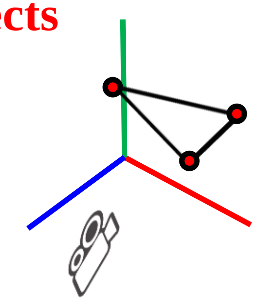
Transformed vertices

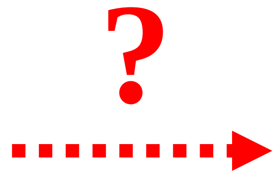
Vertex positions in 2D viewport

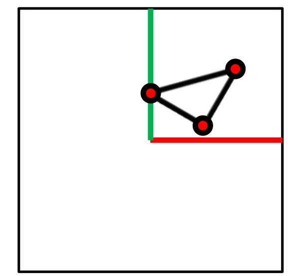
1. Placing objects











Let's think a "camera" is watching the "scene".

glVertex3fv($\mathbf{p}_1$)
glVertex3fv($\mathbf{p}_2$)
glVertex3fv($\mathbf{p}_3$)

glMultMatrixf($\mathbf{M}^T$)
glVertex3fv($\mathbf{p}_1$)
glVertex3fv($\mathbf{p}_2$)
glVertex3fv($\mathbf{p}_3$)

…or

glVertex3fv($\mathbf{M}\mathbf{p}_1$)
glVertex3fv($\mathbf{M}\mathbf{p}_2$)
glVertex3fv($\mathbf{M}\mathbf{p}_3$)

Then what we have to do are…

2. Placing the "camera"
3. Selecting a "lens"
4. Displaying on a "cinema screen"

In Terms of CG Transformation,

- 1. Placing objects

→ **Modeling transformation**

- 2. Placing the “camera”

→ **Viewing transformation**

- 3. Selecting a “lens”

→ **Projection transformation**

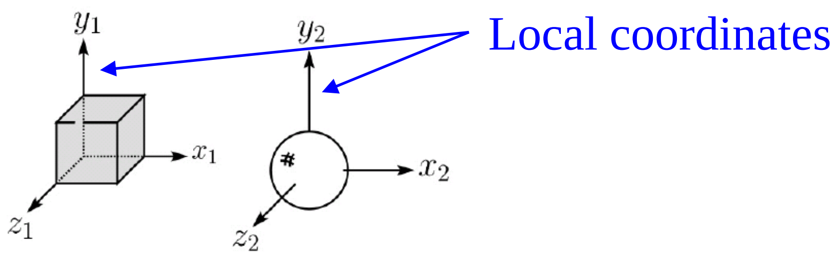
- 4. Displaying on a “cinema screen”

→ **Viewport transformation**

- All these transformations just work by **matrix multiplications**!

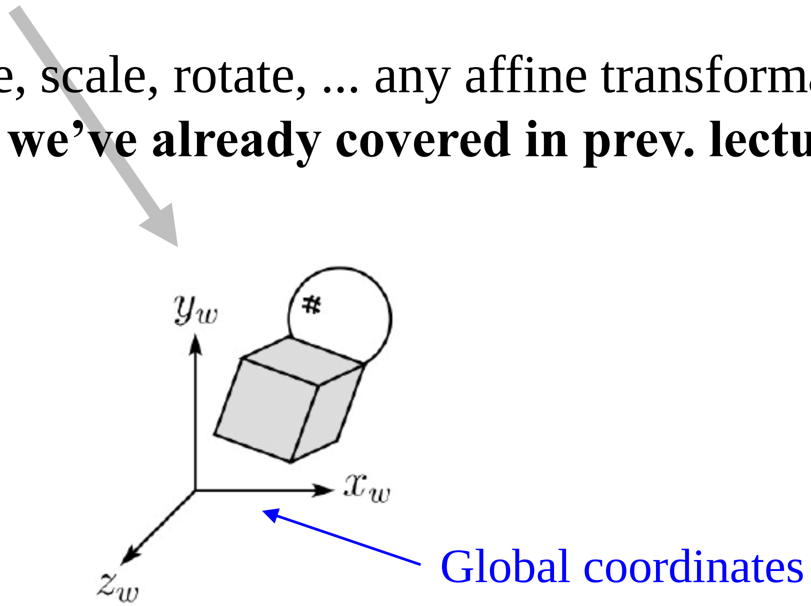
Vertex Processing (Transformation Pipeline)

Object space



Translate, scale, rotate, ... any affine transformations

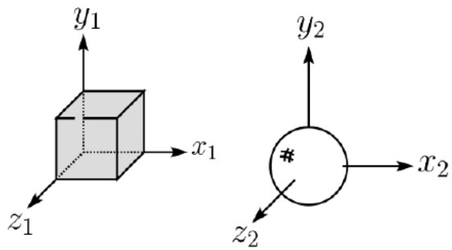
(What we've already covered in prev. lectures)



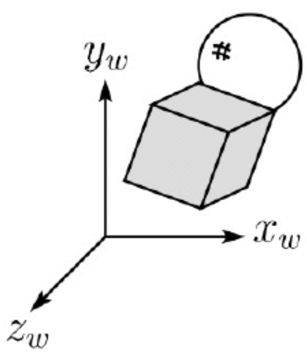
World space

Vertex Processing (Transformation Pipeline)

Object space

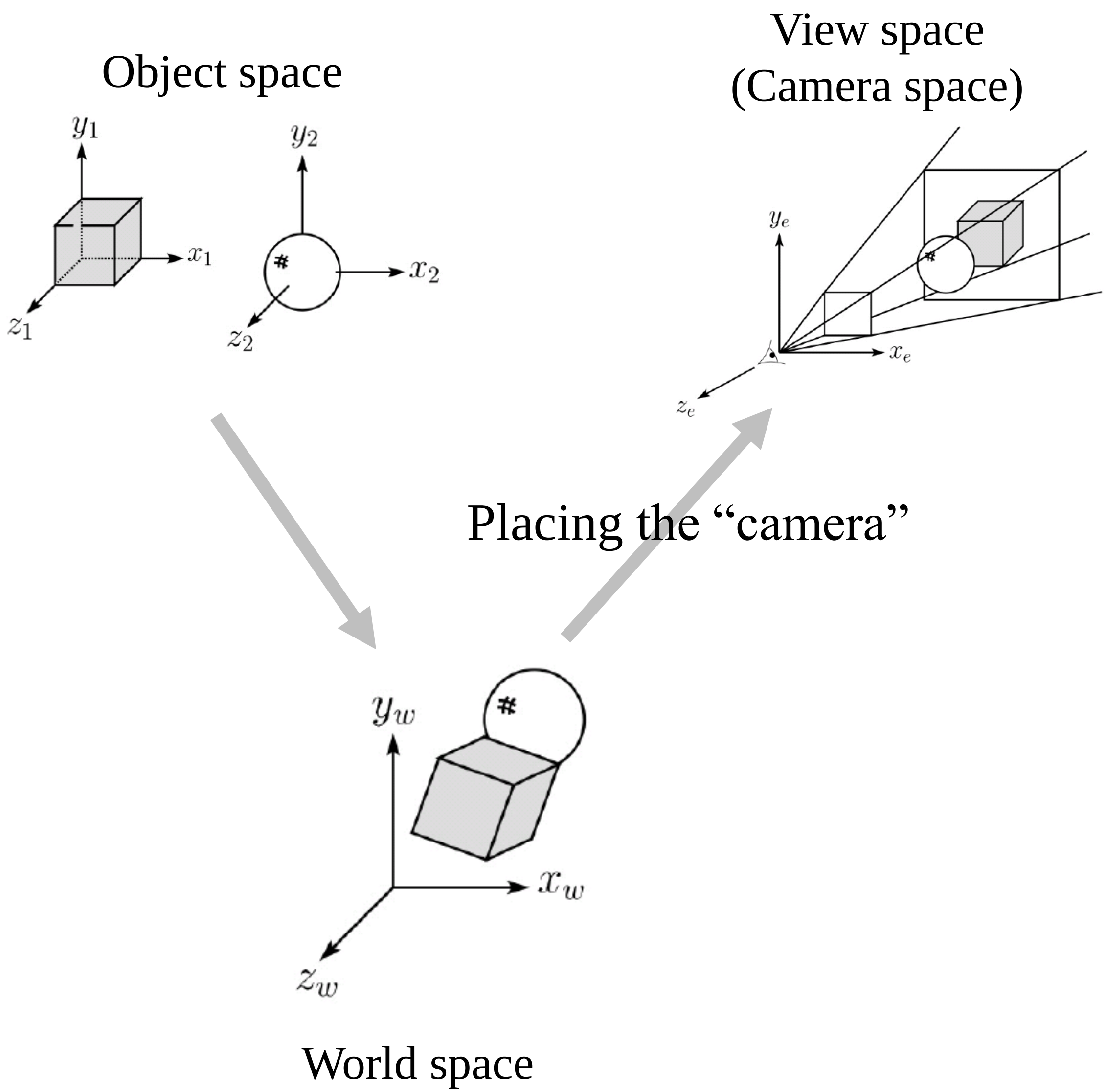


Modeling transformation

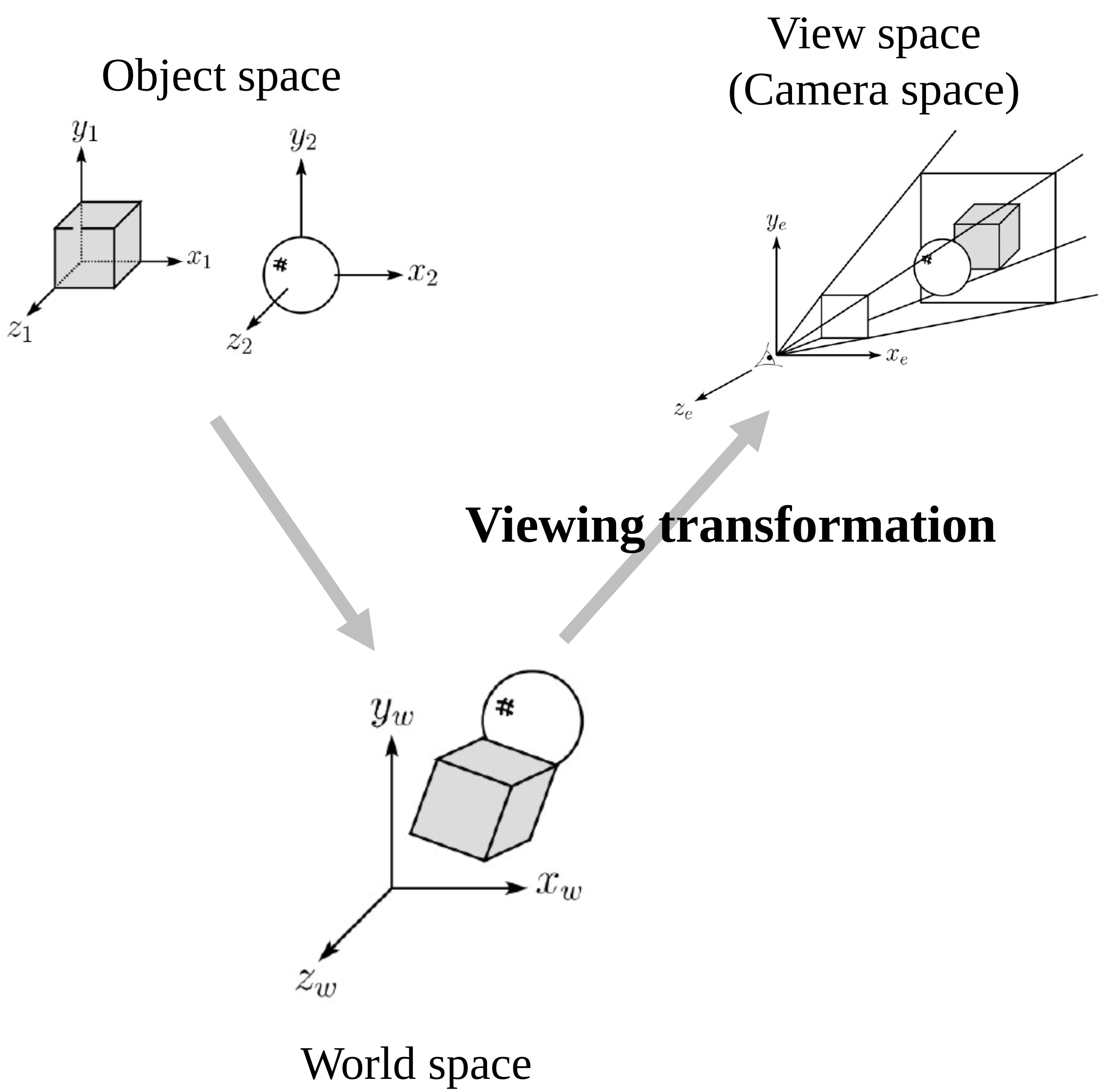


World space

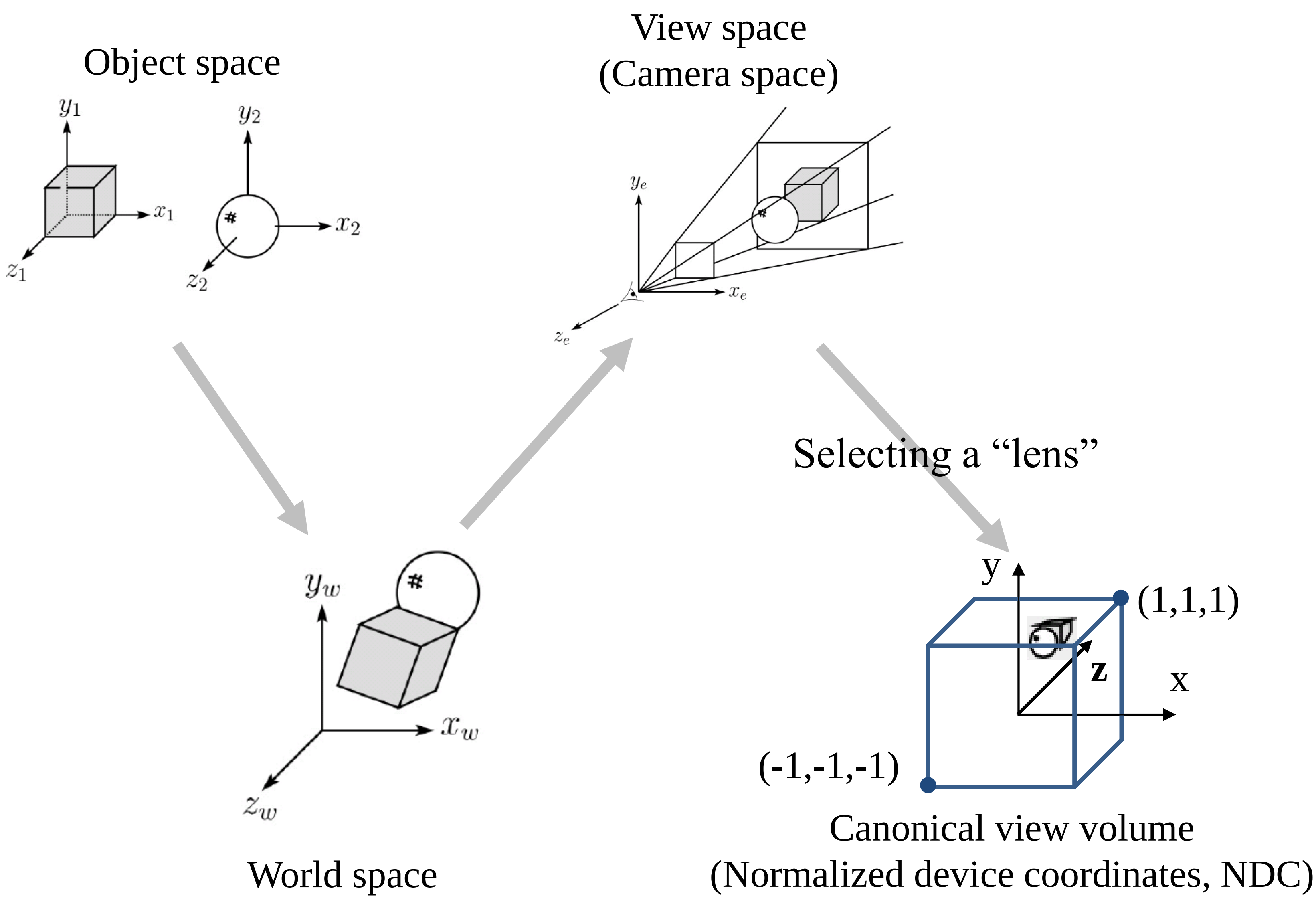
Vertex Processing (Transformation Pipeline)



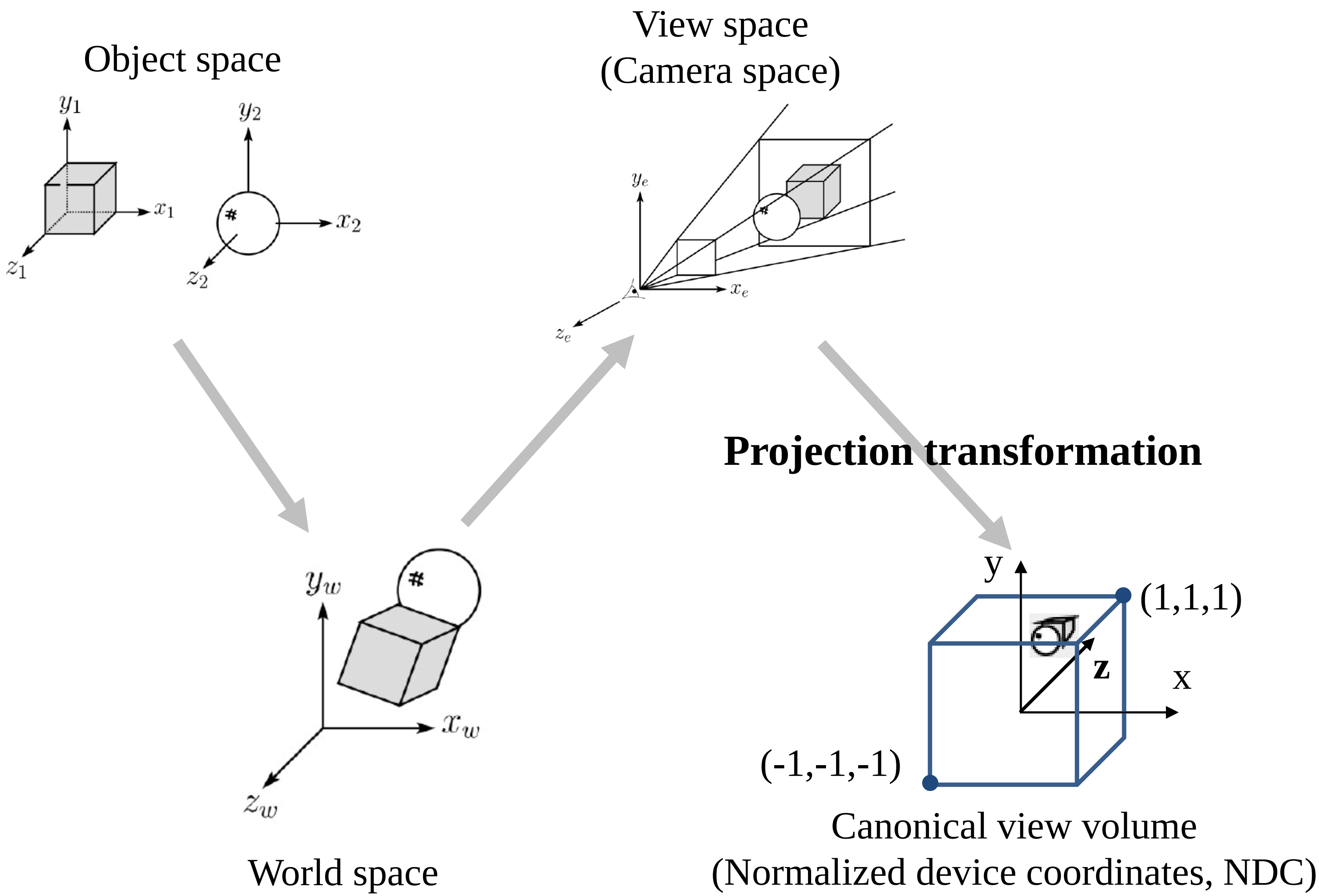
Vertex Processing (Transformation Pipeline)



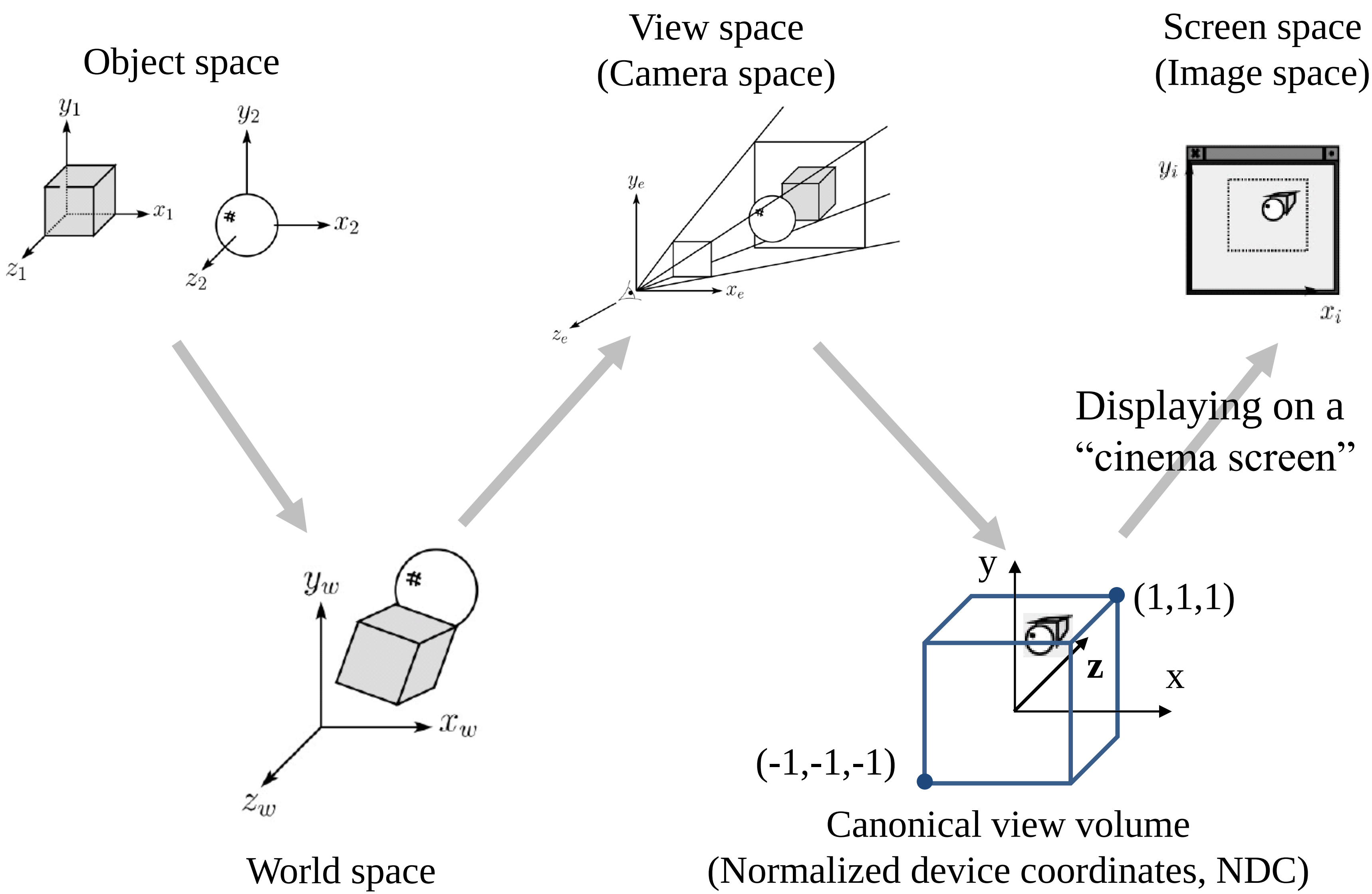
Vertex Processing (Transformation Pipeline)



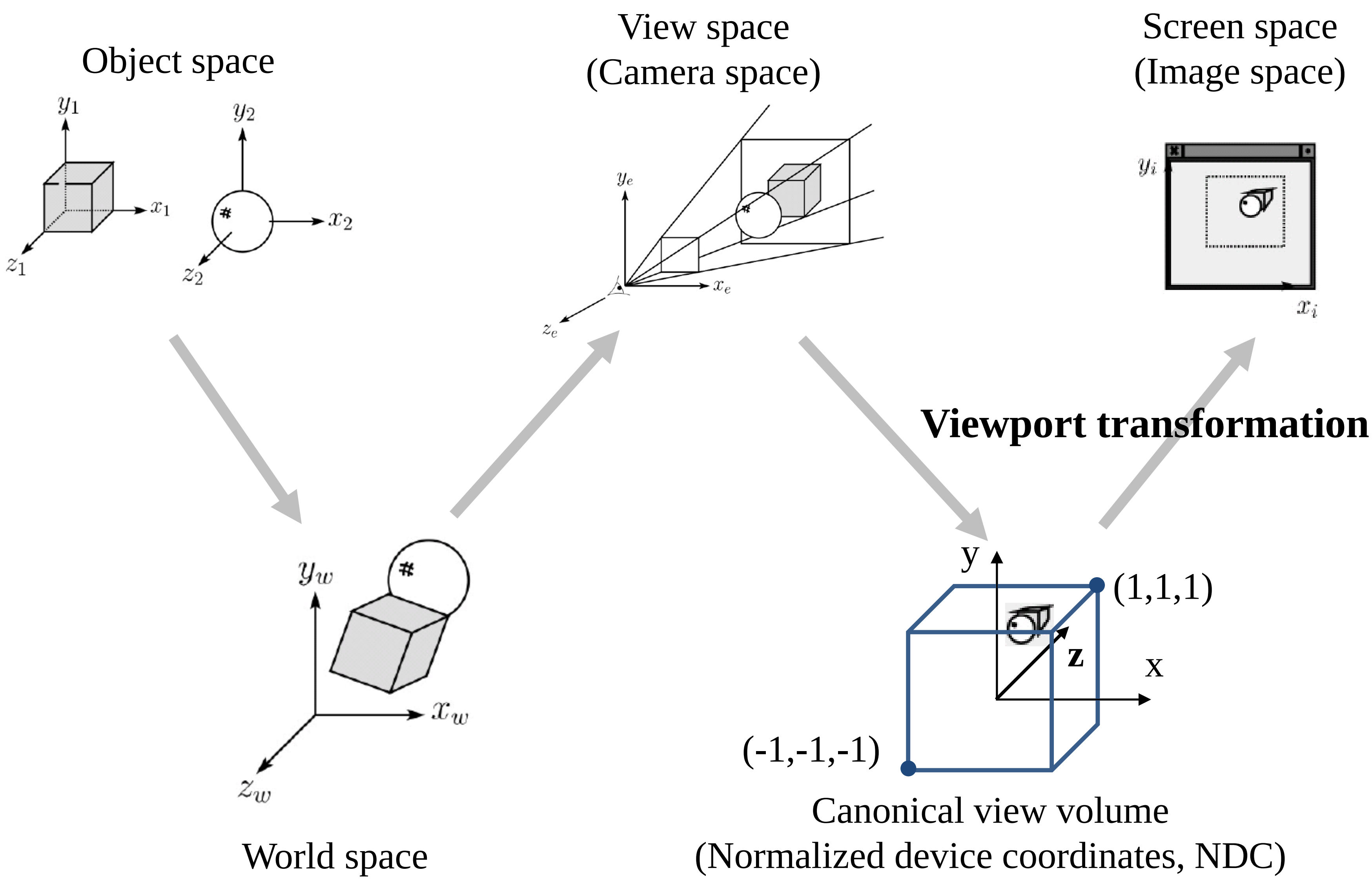
Vertex Processing (Transformation Pipeline)



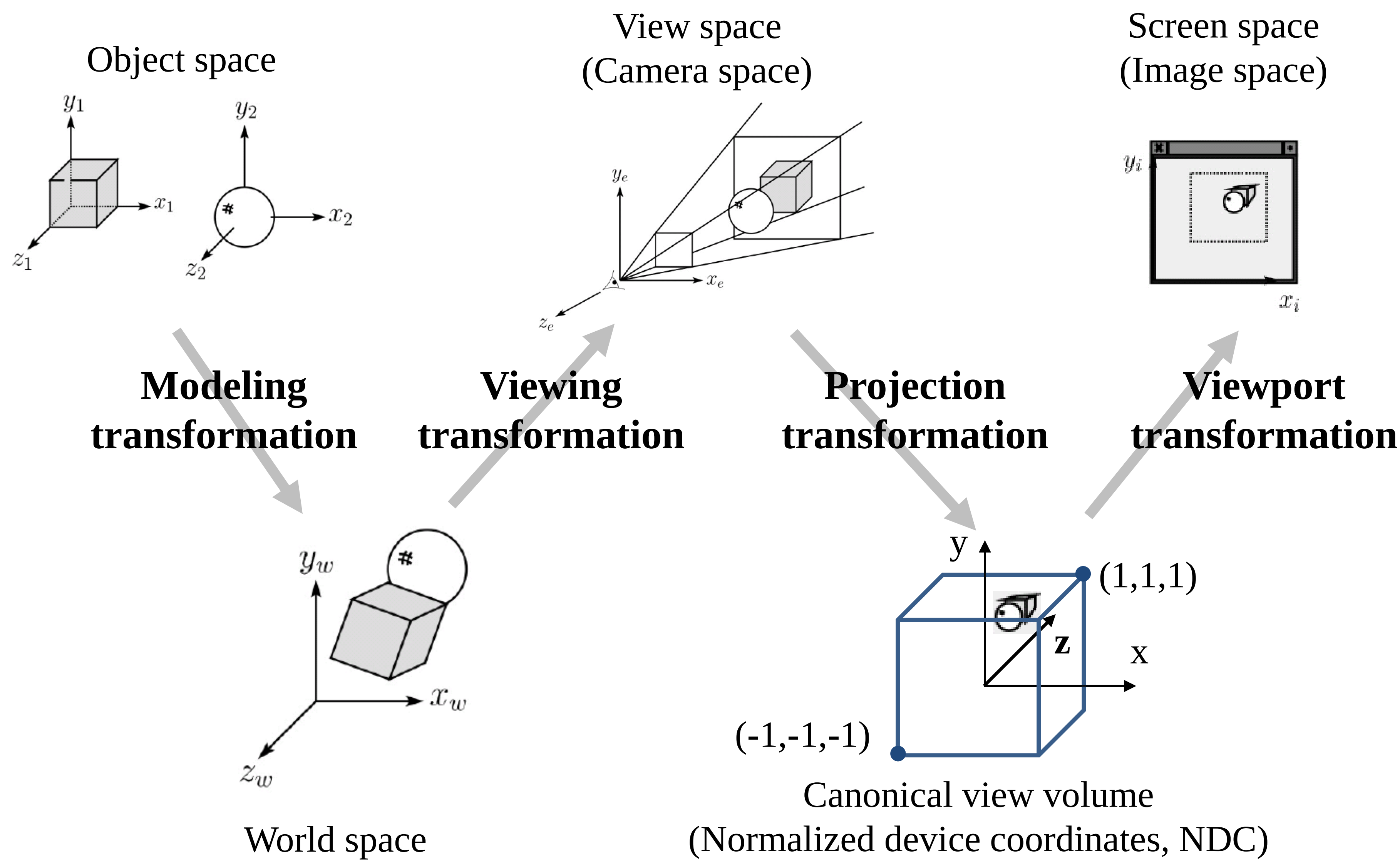
Vertex Processing (Transformation Pipeline)



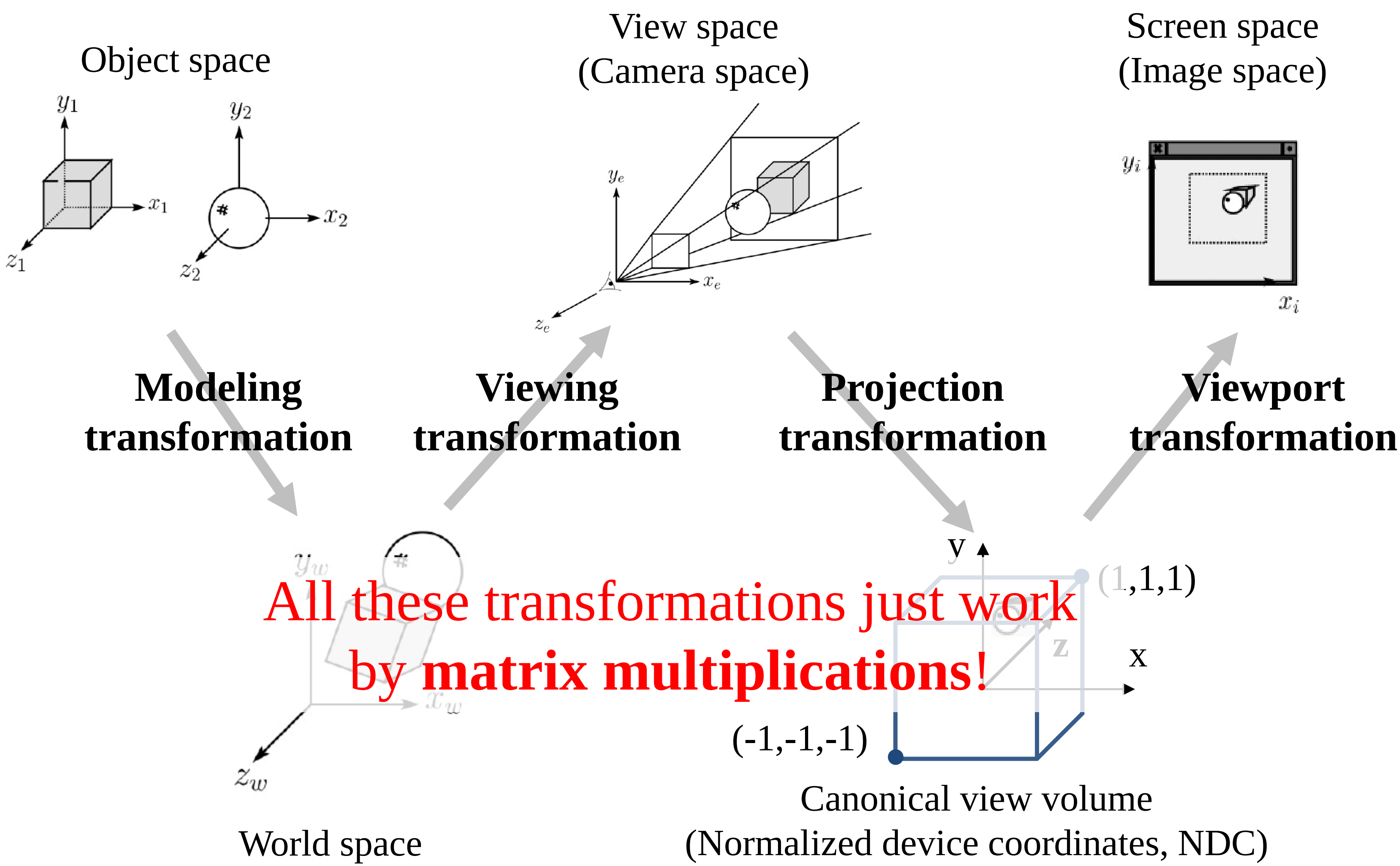
Vertex Processing (Transformation Pipeline)



Vertex Processing (Transformation Pipeline)

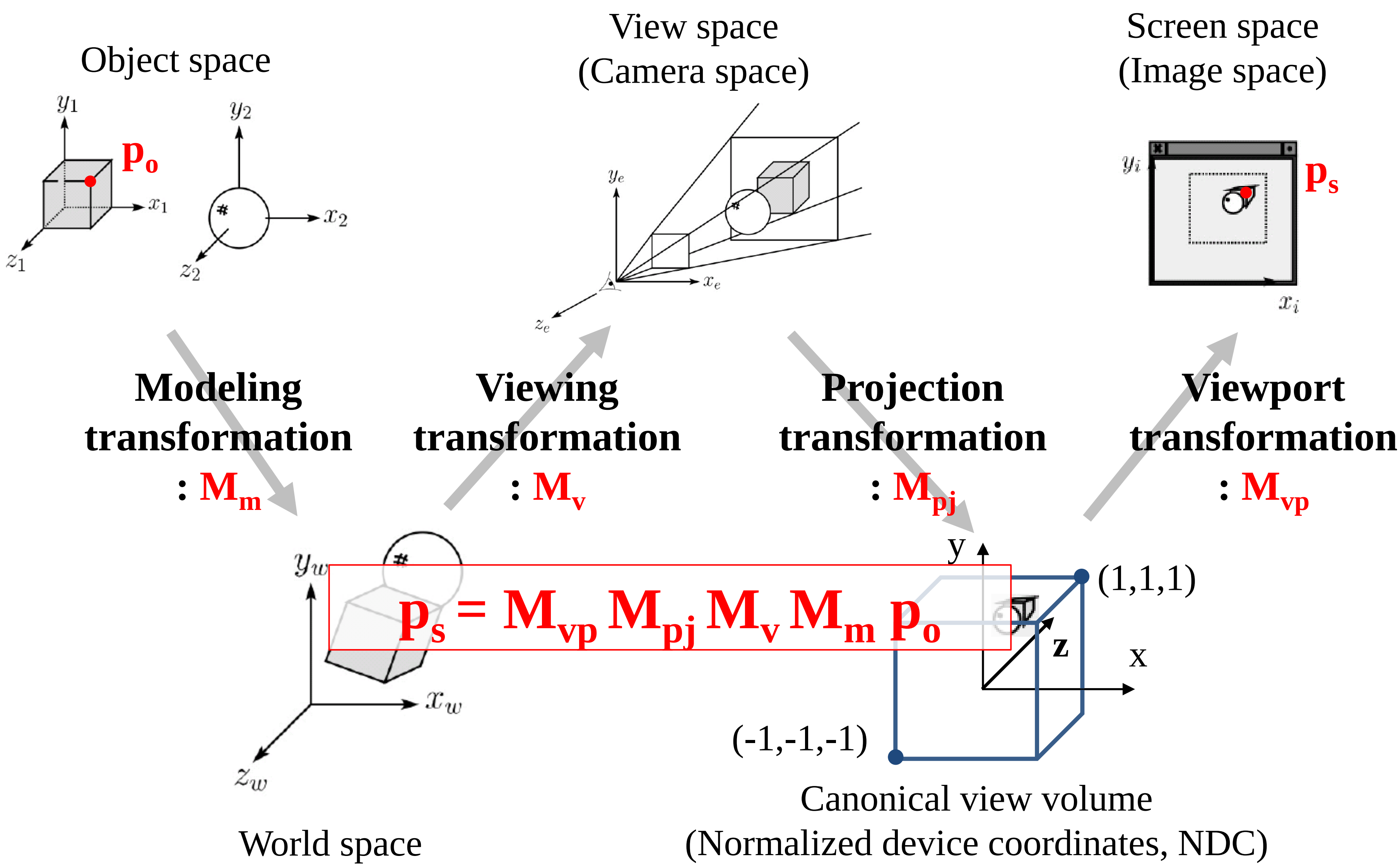


Vertex Processing (Transformation Pipeline)

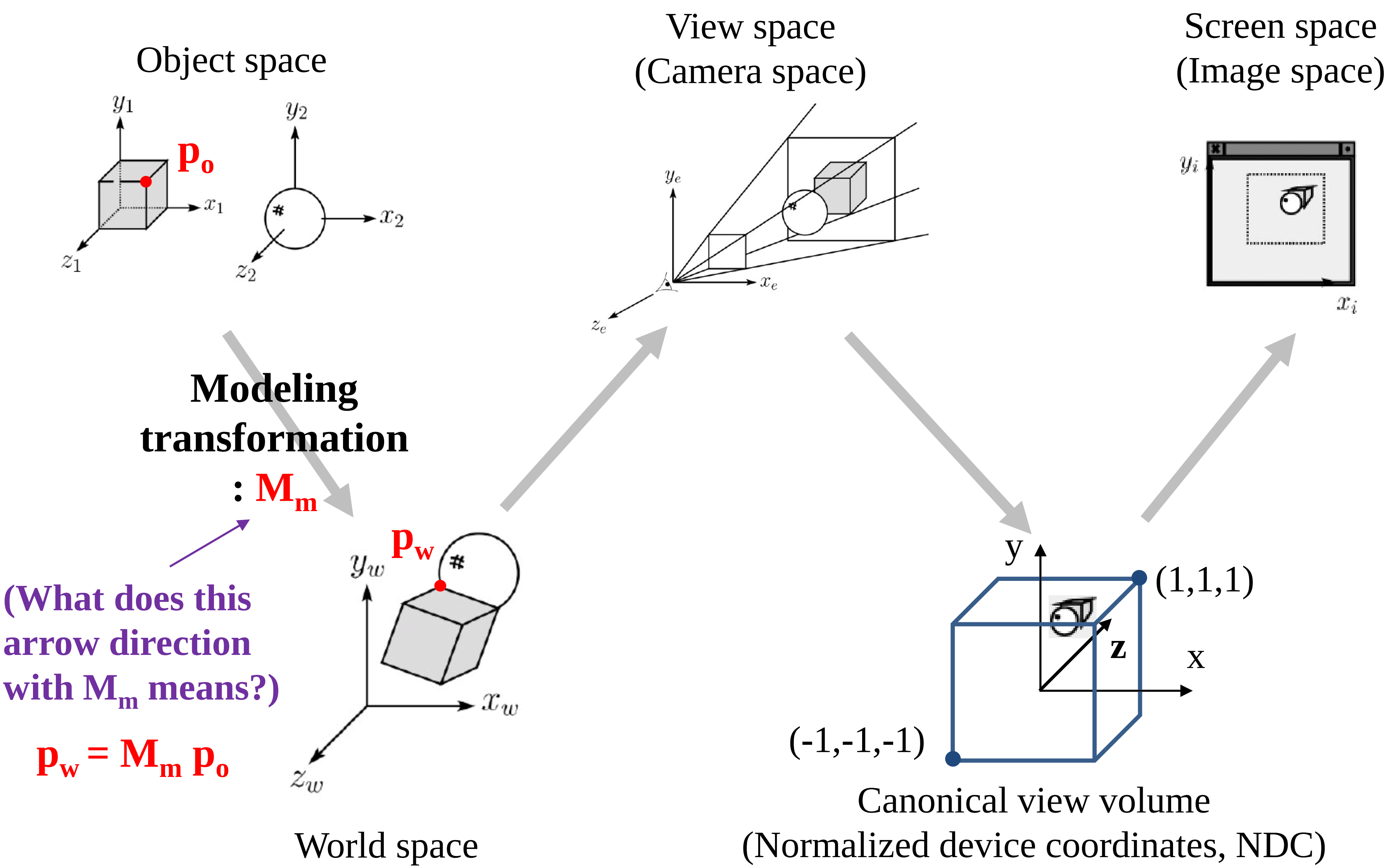


All these transformations just work by **matrix multiplications!**

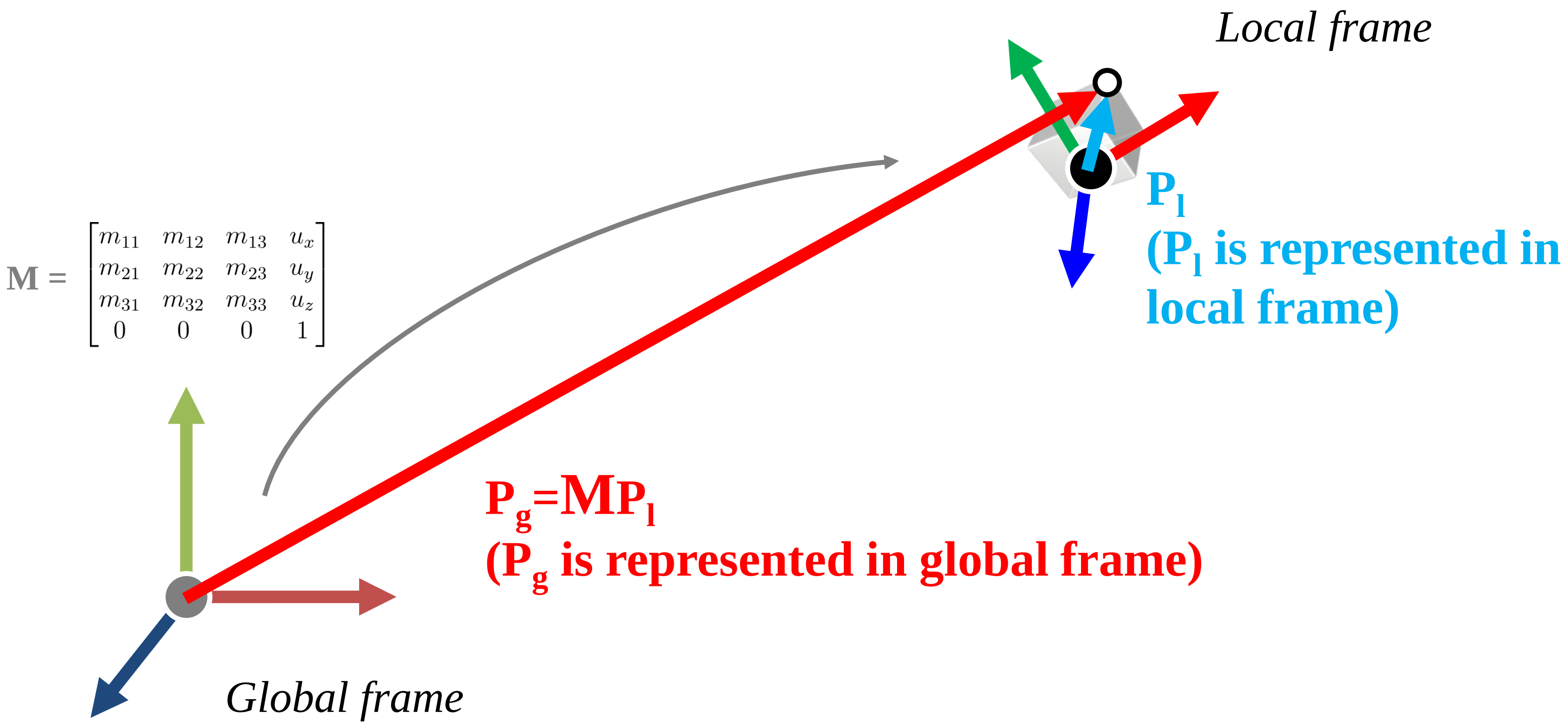
Vertex Processing (Transformation Pipeline)



Modeling Transformation



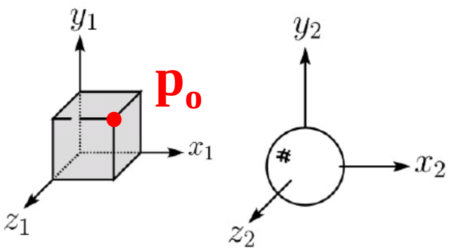
3) Review: A 4x4 Affine Transformation Matrix transforms a Point Represented in One Frame to a Point Represented in Global Frame



Modeling Transformation

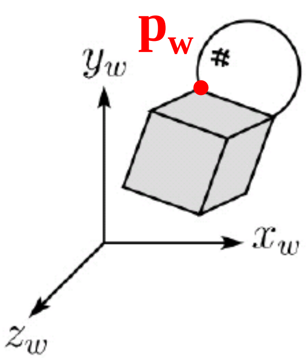
- Geometry would originally have been in the **object's local coordinates**;
- Transform into world coordinates is called the *modeling matrix*, M_m
- Composite affine transformations
- (What we've covered so far!)

Object space

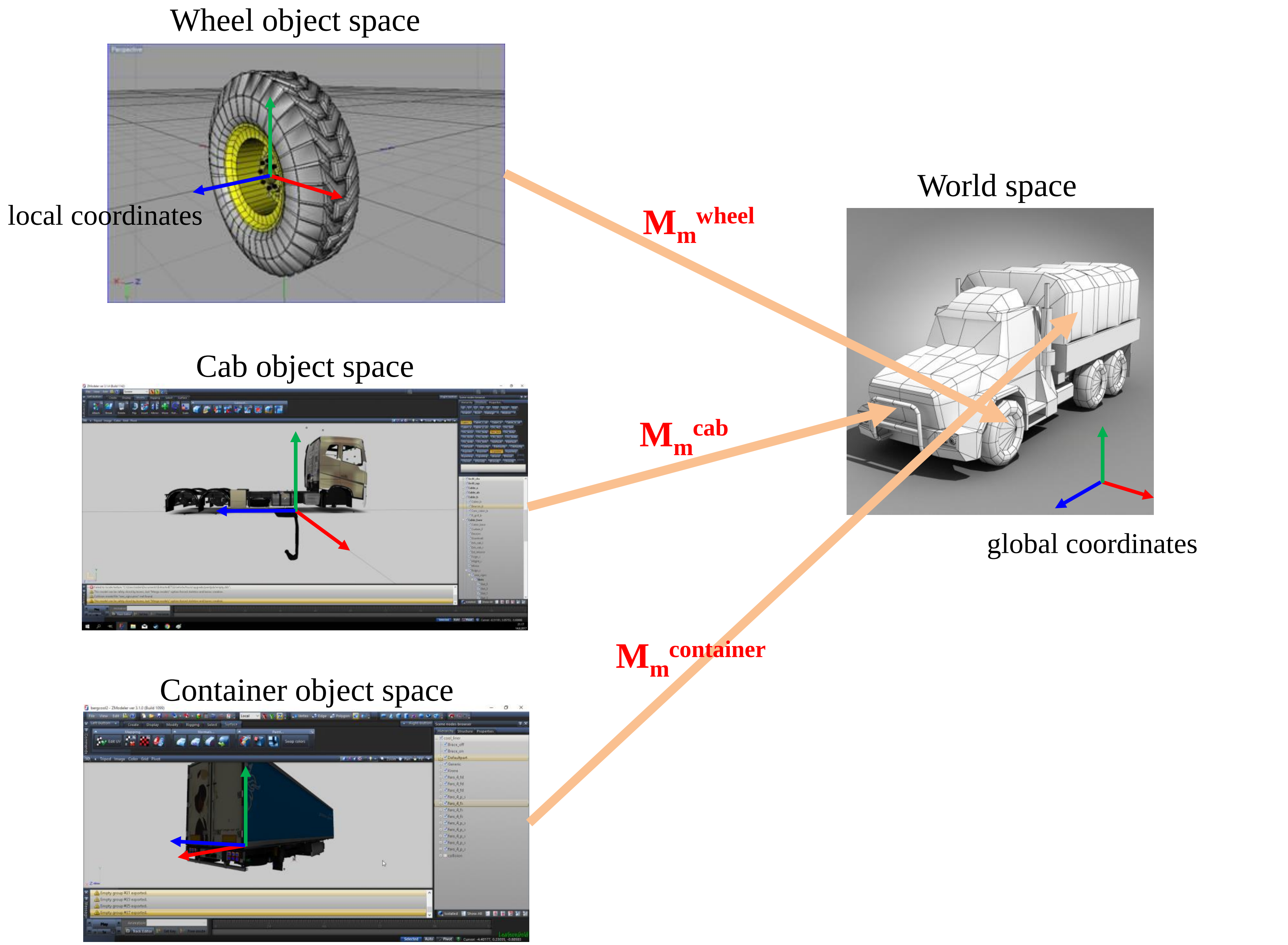


Translate, rotate, scale, …
(Affine transformation)

$\mathbf{M_m}$



World space



Wheel object space
local coordinates
Cab object space
Container object space
World space
global coordinates
M_m^{wheel}
M_m^{cab}
$M_m^{container}$

Next Time

- Lab in this week:
  - Lab assignment 5

- Next lecture:
  - 6 - Viewing, Projection

- Acknowledgement: Some materials come from the lecture slides of
  - Prof. Jinxiang Chai, Texas A&M Univ., http://faculty.cs.tamu.edu/jchai/csce441_2016spring/lectures.html
  - Prof. Jehee Lee, SNU, http://mrl.snu.ac.kr/courses/CourseGraphics/index_2017spring.html
  - Prof. Sung-eui Yoon, KAIST, https://sglab.kaist.ac.kr/~sungeui/CG/