Computer Graphics

5 - Affine Matrix, Rendering Pipeline

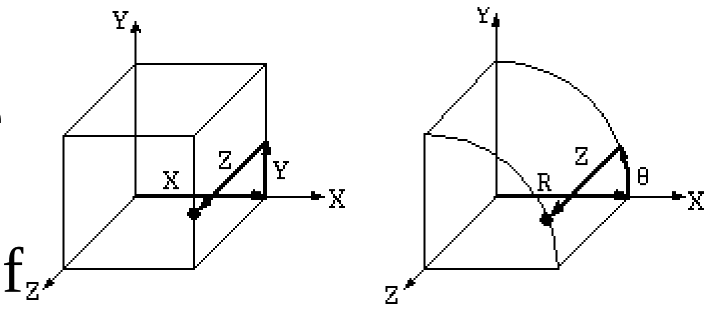
Yoonsang Lee
Spring 2020

Topics Covered

- Coordinate System & Reference Frame
- Meanings of an Affine Transformation Matrix
- Interpretation of a Series of Transformations
- Rendering Pipeline
  - Vertex Processing
    - Modeling transformation

Coordinate System & Reference Frame

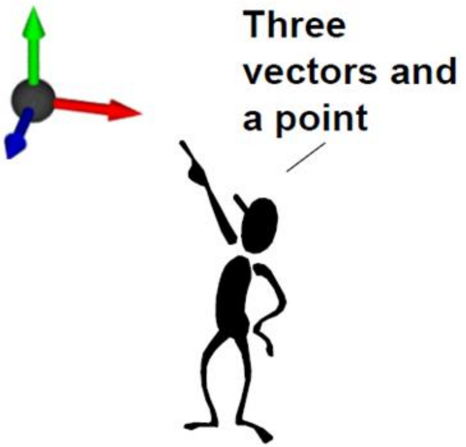
- Coordinate system
  - A system which uses one or more numbers, or coordinates, to uniquely determine the position of points.



Cartesian (X,Y,Z components) coordinate system 0 (C.S. 0)

Cylindrical (R,q,Z components) coordinate system 1 (C.S. 1)

- Reference frame
  - Abstract coordinate system + physical reference points (to uniquely fix the coordinate system).



Coordinate System & Reference Frame

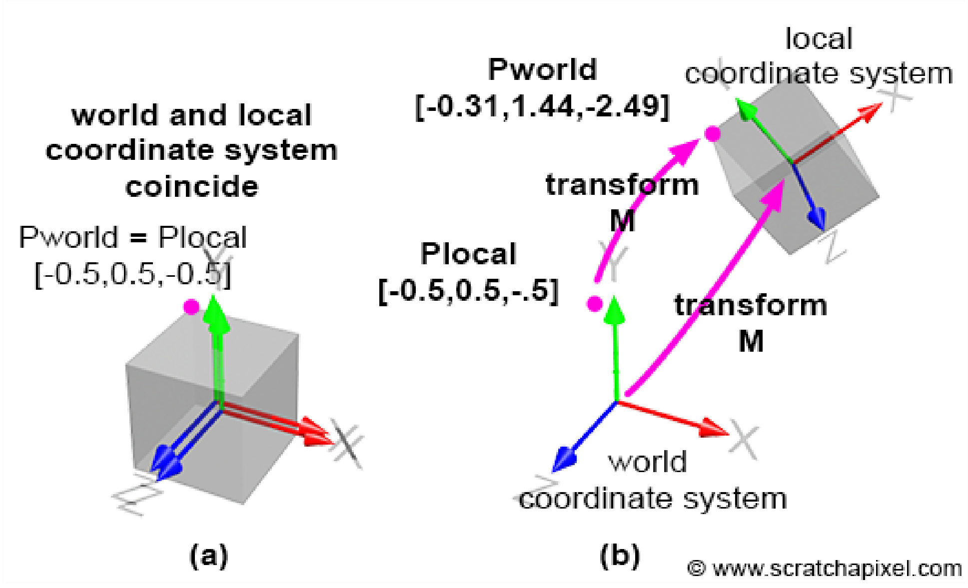
- Two terms are slightly different:
  - **Coordinate system** is a mathematical concept, about a choice of "language" used to describe observations.
  - **Reference frame** is a physical concept related to state of motion.
  - You can think the coordinate system determines the way one describes/observes the motion in each reference frame.

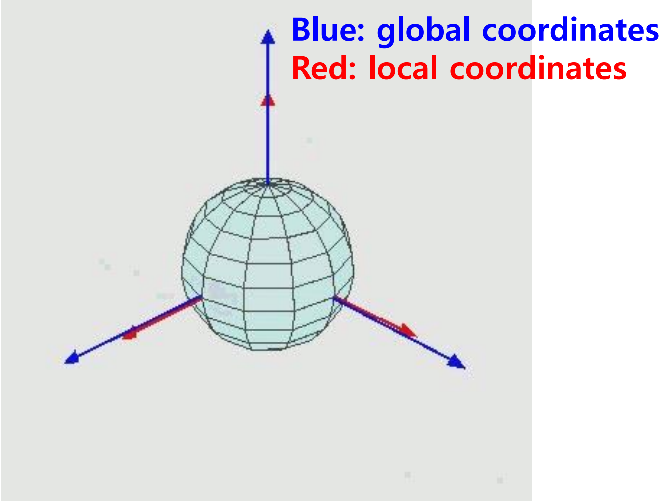
- But these two terms are often mixed.

Global & Local Coordinate System(or Frame)

- **global coordinate system** (or **global frame**)
  - A coordinate system(or frame) attached to the **world.**
  - A.k.a. **world** coordinate system, **fixed** coordinate system

- **local coordinate system** (or **local frame**)
  - A coordinate system(or frame) attached to a **moving object.**



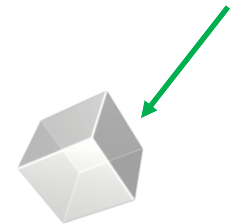


https://commons.wikimedia.org/wiki/File:Euler2a.gif

Meanings of an Affine Transformation Matrix

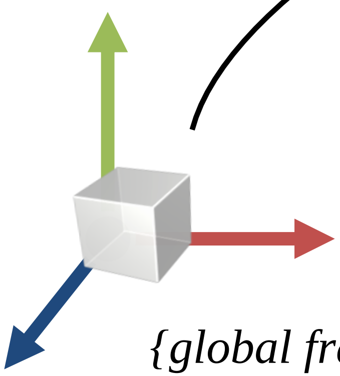
1) A 4x4 Affine Transformation Matrix transforms a Geometry w.r.t. Global Frame

Transformed geometry



Translate, rotate, scale, ...

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

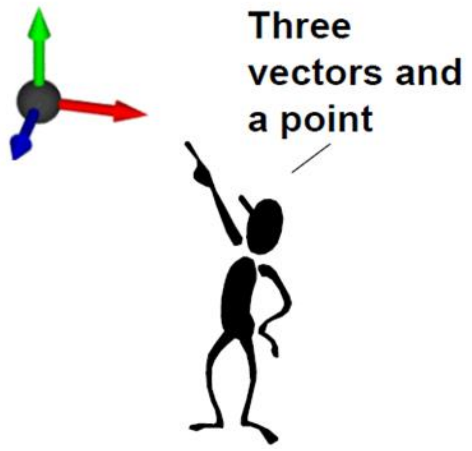


{global frame}

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

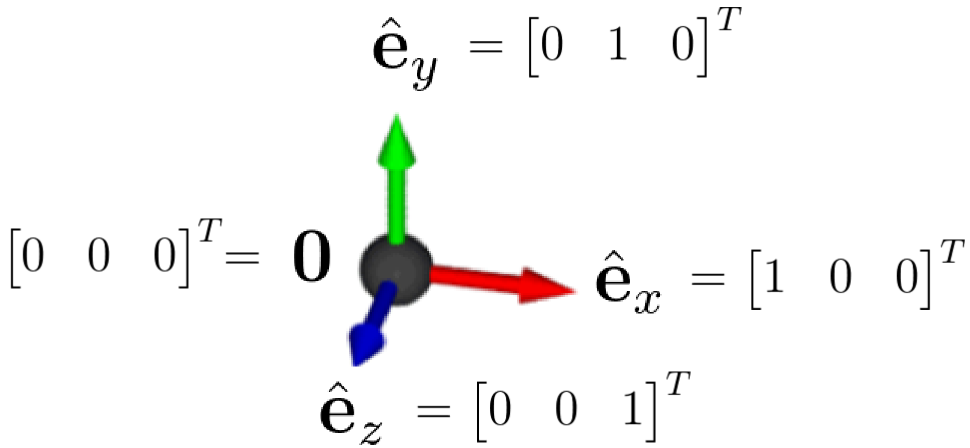
Review: Affine Frame

- 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*

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

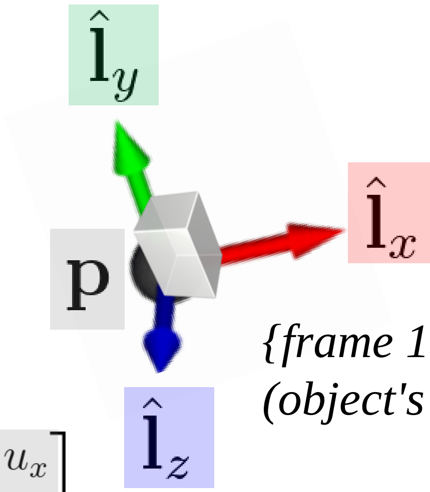
z axis *vector*

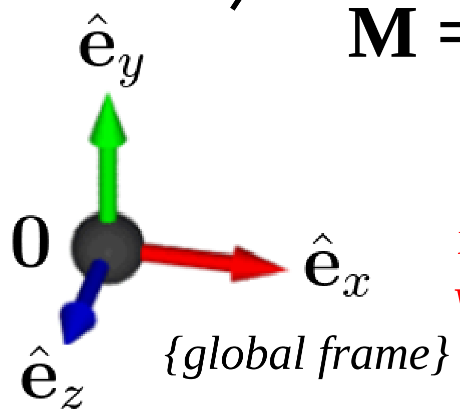
$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & u_x \\ m_{21} & m_{22} & m_{23} & u_y \\ m_{31} & m_{32} & m_{33} & u_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ \boxed{0} \end{bmatrix} = \begin{bmatrix} m_{13} \\ m_{23} \\ m_{33} \\ \boxed{0} \end{bmatrix}$$

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*

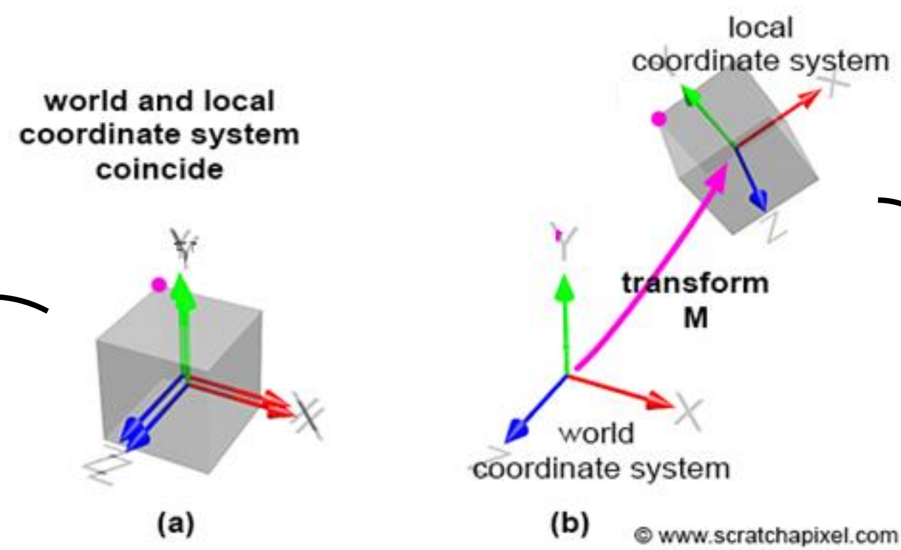
y axis *vector*

z axis *vector*

origin *point*

→ M is **the axis vectors and origin point of a new frame (represented in the global frame)**

Examples



This 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 frame ***represented in the global frame***

This 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*

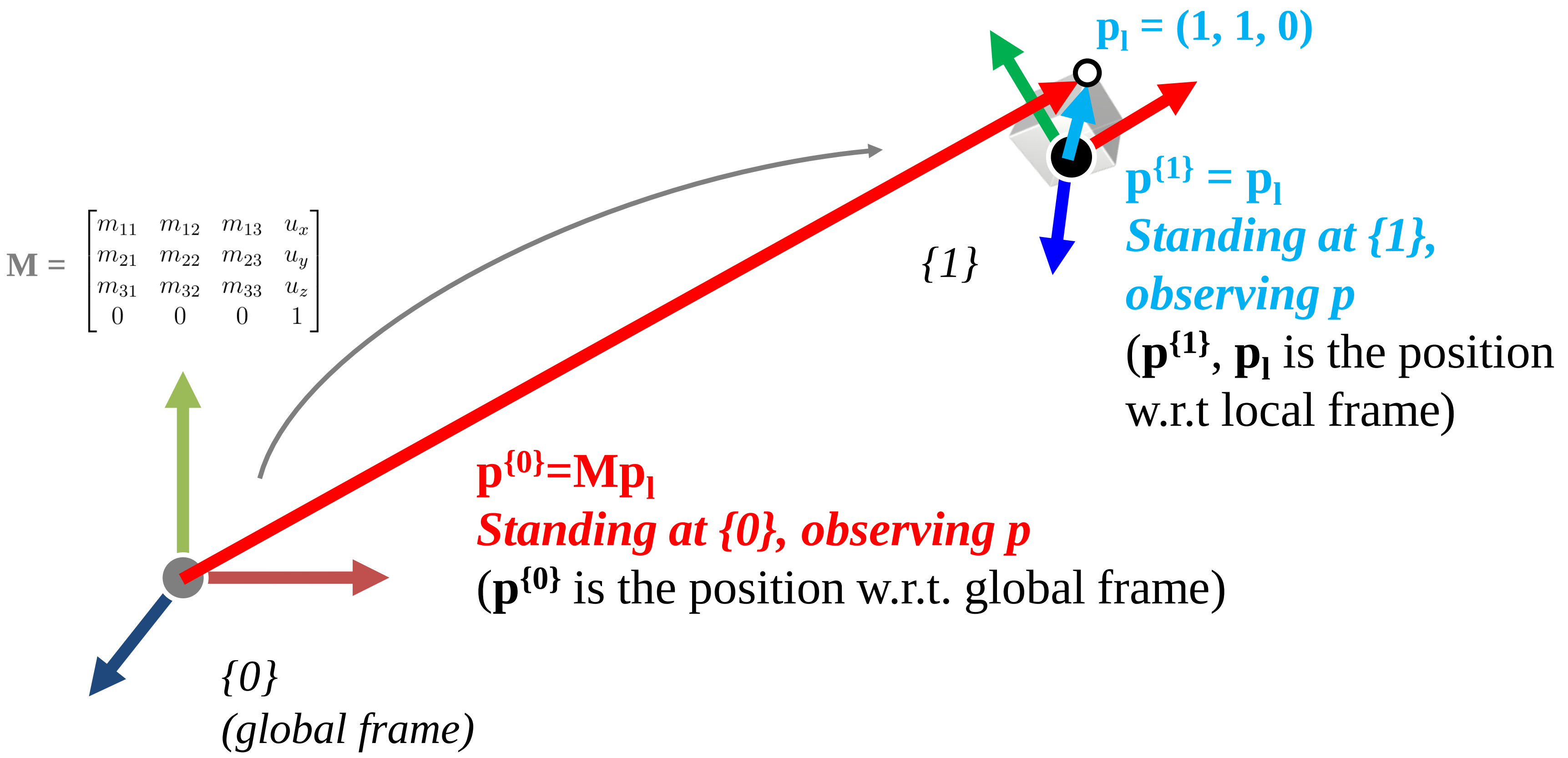
Quiz #1

- 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".

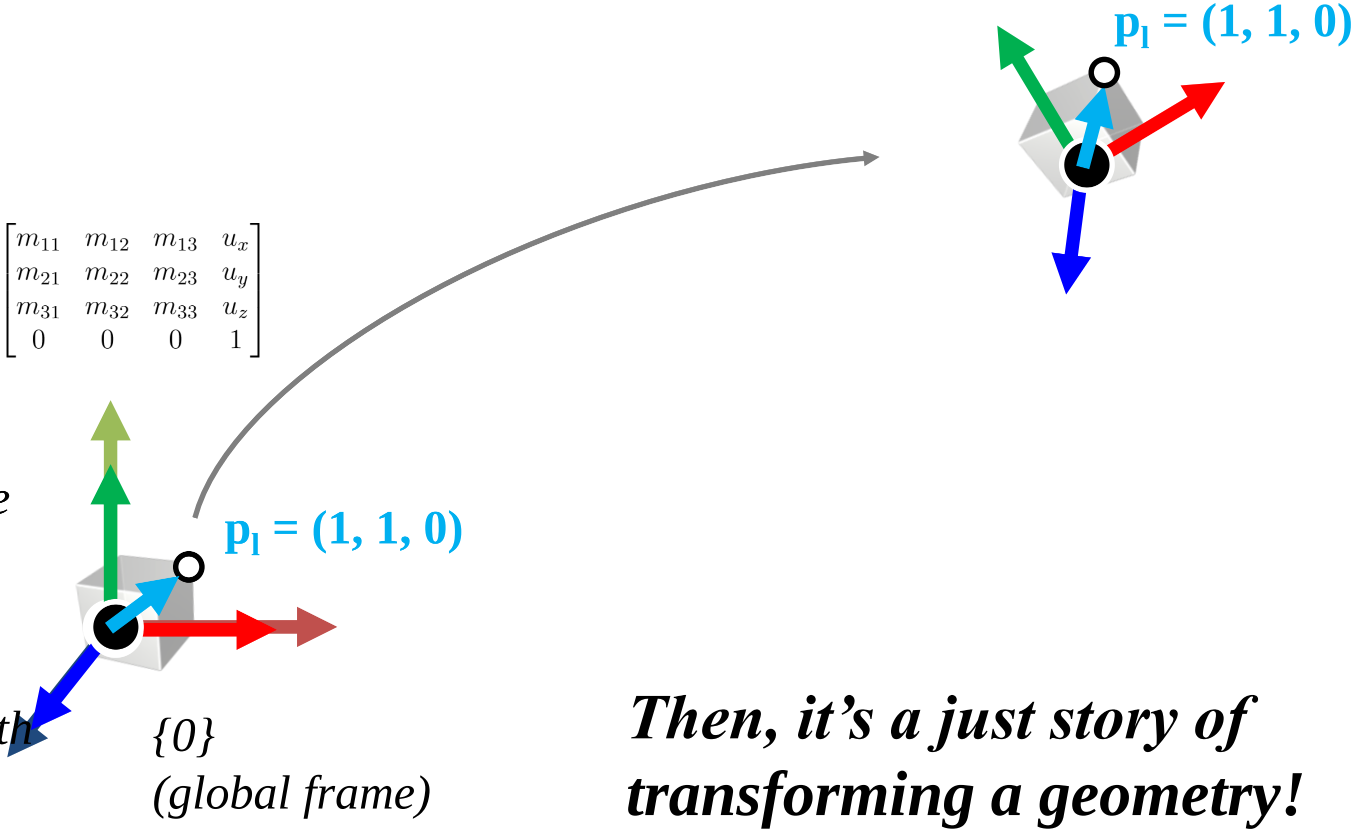
3) A 4x4 Affine Transformation Matrix transforms a Point Represented in an Affine Frame to (the same) Point (but) Represented in Global Frame



3) A 4x4 Affine Transformation Matrix transforms a Point Represented in an Affine Frame to (the same) Point (but) 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!

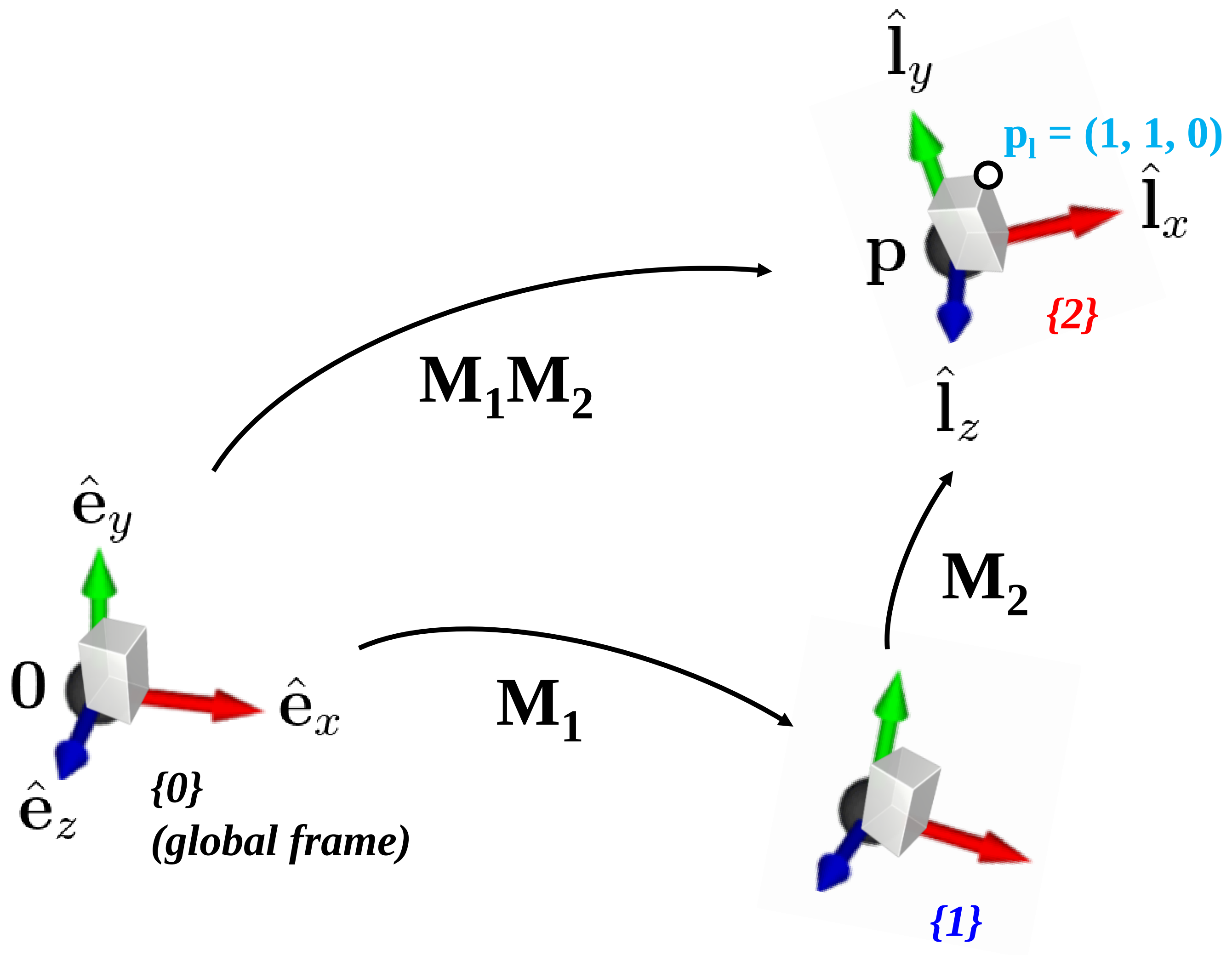
Quiz #2

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

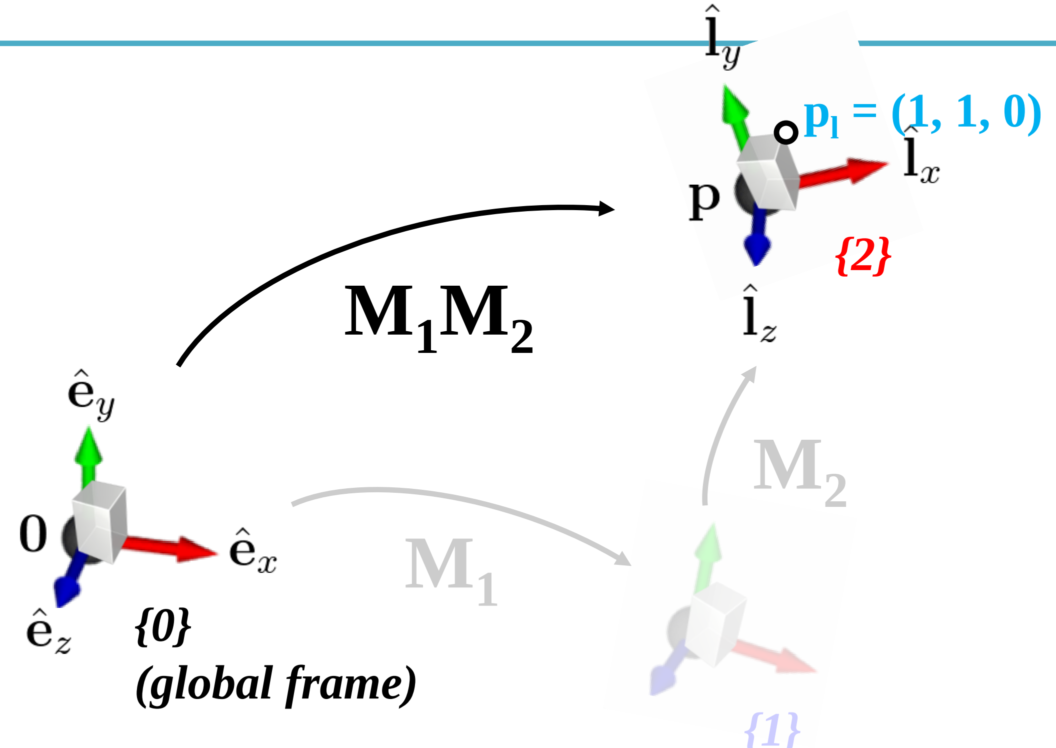
- Submit your answer in the following format:
  - **Student ID: Your answer**
  - **e.g. 2017123456: 4)**

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

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

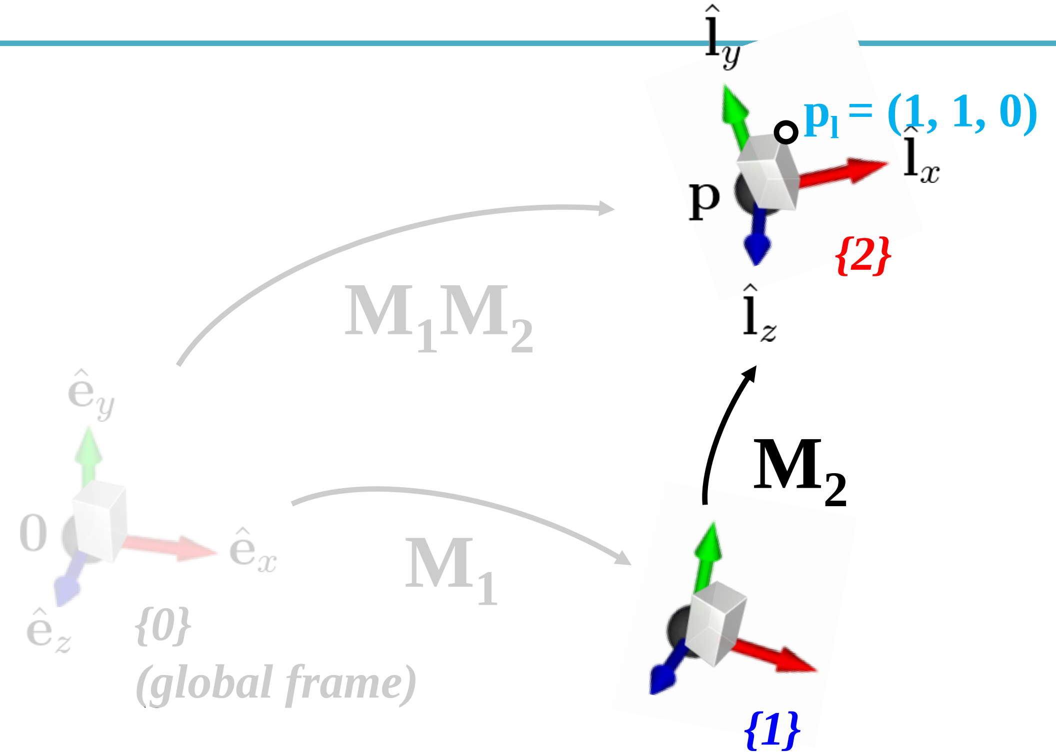


That is,



- 1) $\mathbf{M_1M_2}$ transforms a geometry (represented in ***{0}***) w.r.t. ***{0}***
  - $p^{\{2\}}=p_l$, $p^{\{1\}}=M_2p_l$, $p\{0\}=M_1M_2p_l$
- 2) $\mathbf{M_1M_2}$ defines an ***{2}*** w.r.t. ***{0}***
- 3) $\mathbf{M_1M_2}$ transforms a point represented in ***{2}*** to the same point but represented in ***{0}***

That is,



- 1) $\mathbf{M}_2$ transforms a geometry (represented in ***{1}***) w.r.t. ***{1}***
- 2) $\mathbf{M}_2$ defines an ***{2}*** w.r.t. ***{1}***
- 3) $\mathbf{M}_2$ transforms a point represented in ***{2}*** to the same point but represented in ***{1}***

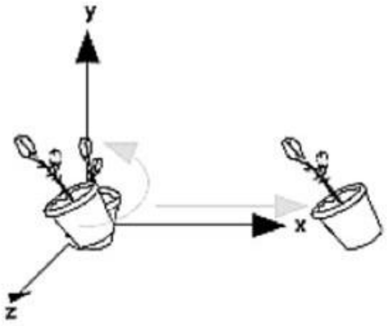
Interpretation of a Series of Transformations

Revisit: Order Matters!

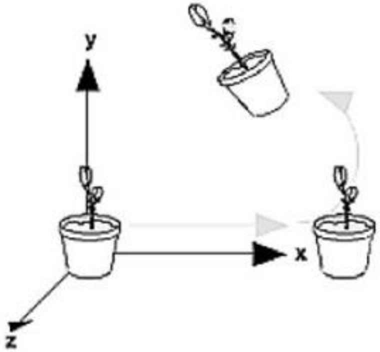
- If T and R are matrices representing affine transformations,

- **p'** = TR**p**
  - First apply transformation R to point **p**, then apply transformation T to transformed point R**p**

- **p'** = RT**p**
  - First apply transformation T to point **p**, then apply transformation R to transformed point T**p**



Rotate then Translate



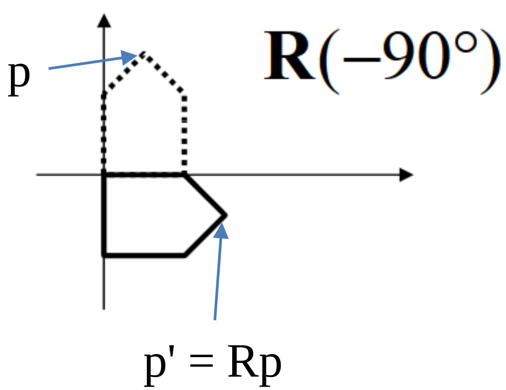
Translate then Rotate

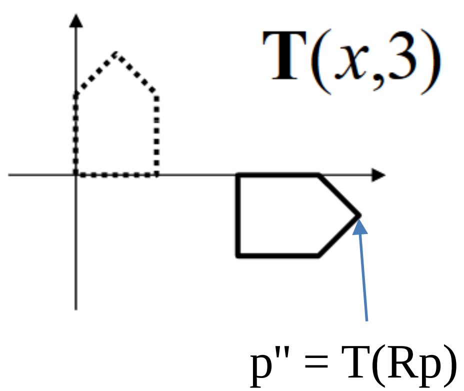
Interpretation of Composite Transformations #1

- An example transformation:

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

- This is how we've interpreted so far:
  - R-to-L: Transforms ***w.r.t. global frame***



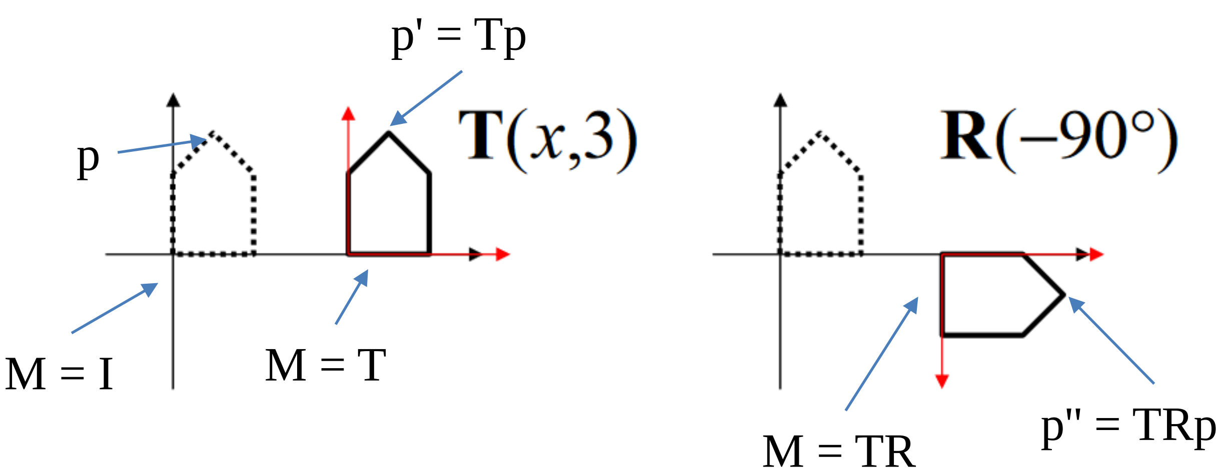


Interpretation of Composite Transformations #2

- An example transformation:

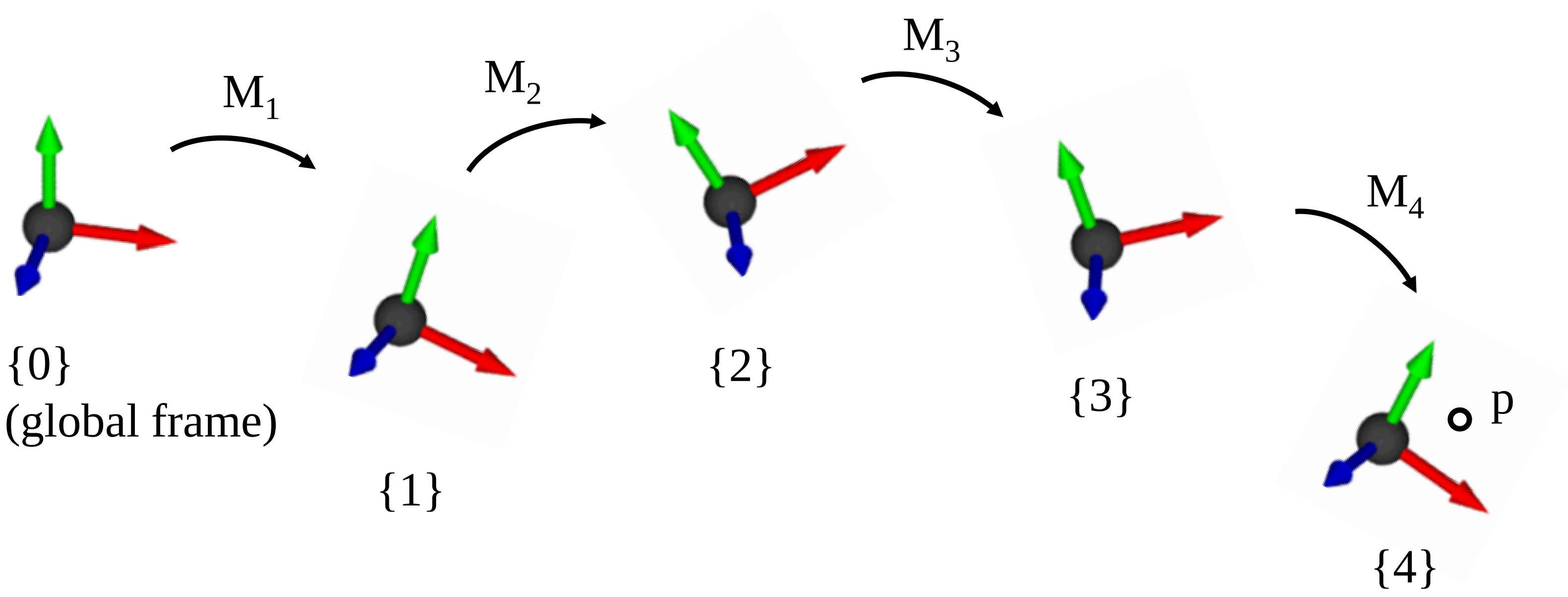
$$M = \mathbf{T}(x,3) \cdot \mathbf{R}(-90°)$$

- **Another way of interpretation:**
  - L-to-R: Transforms ***w.r.t. local frame***



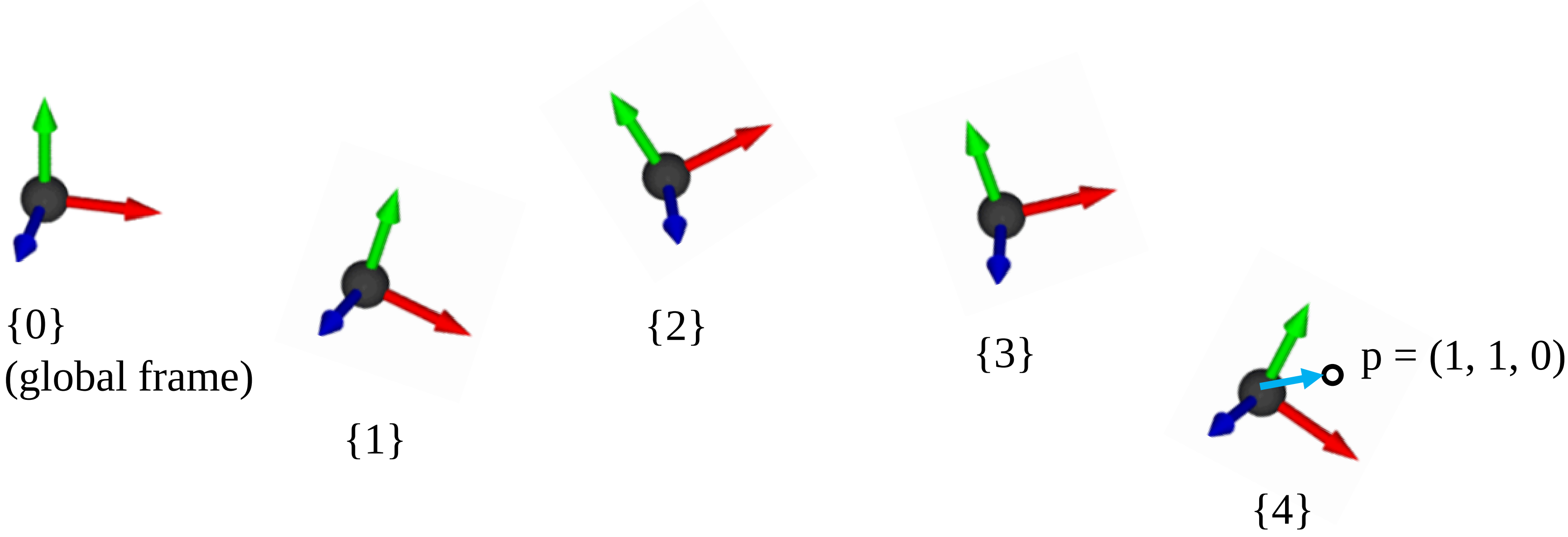
Interpretation of a Series of Transformations #1

- $p' = M_1 M_2 M_3 M_4 \, p$



Interpretation of a Series of Transformations #1

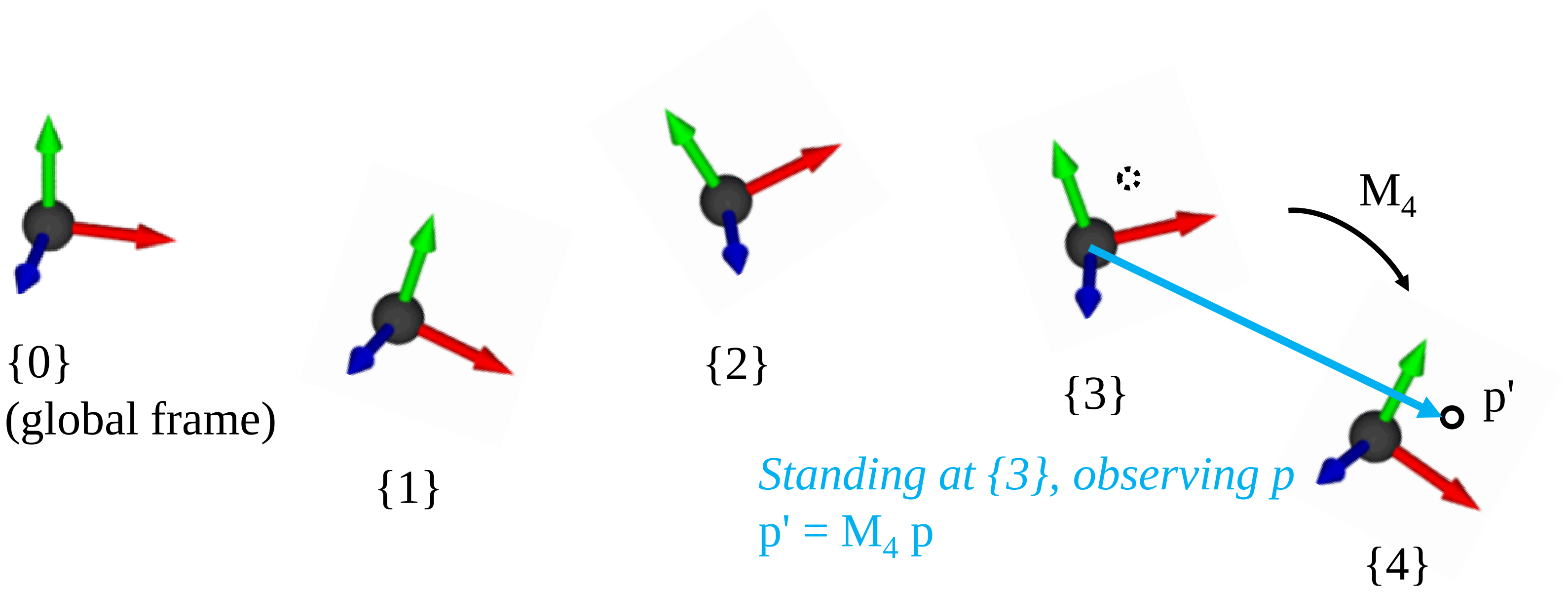
- $p' = M_1 M_2 M_3 M_4 \boxed{p}$



Standing at {4}, observing p

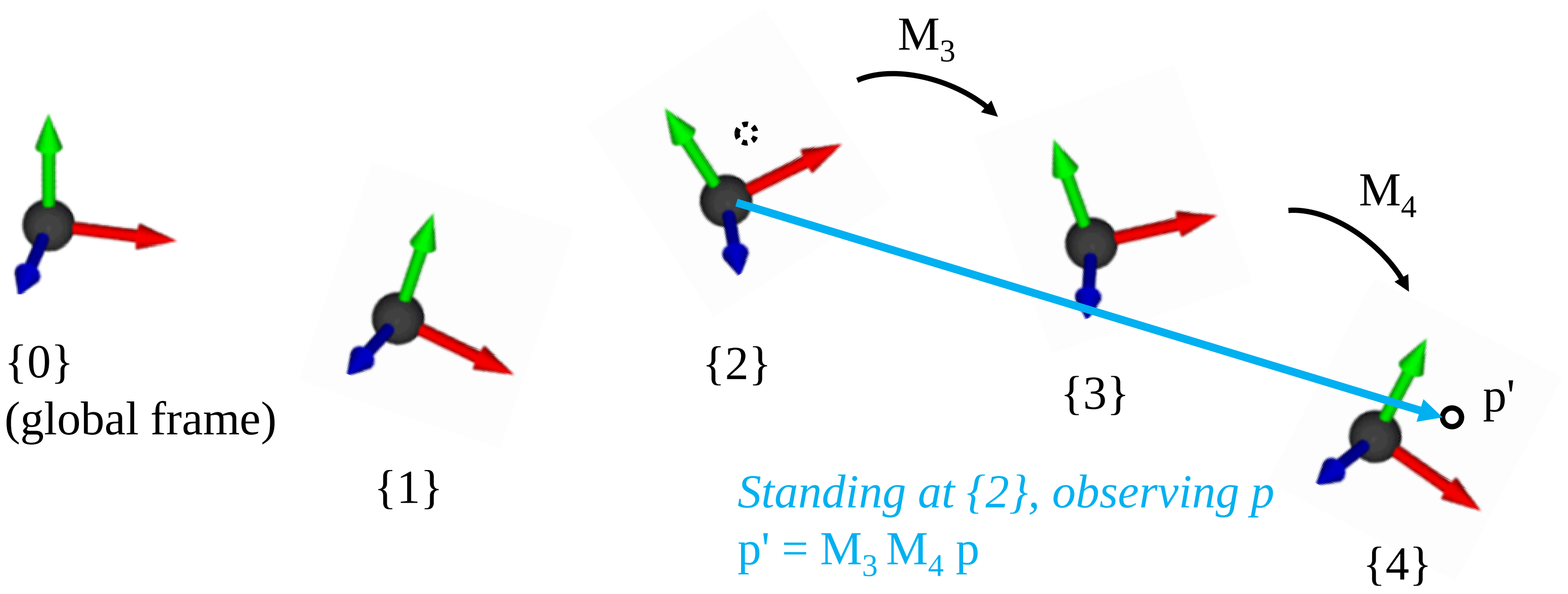
Interpretation of a Series of Transformations #1

- $p' = M_1 M_2 M_3 \boxed{M_4\ p}$



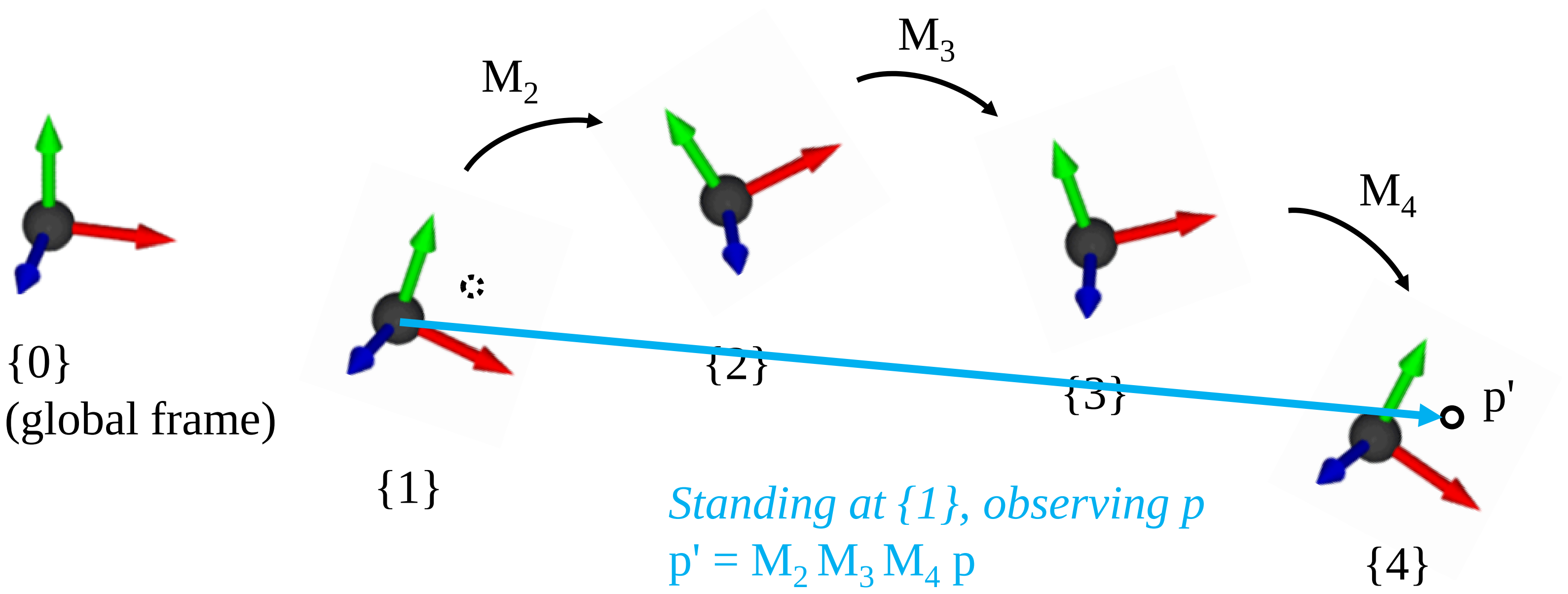
Interpretation of a Series of Transformations #1

- $p' = M_1 M_2 \boxed{M_3 M_4\ p}$



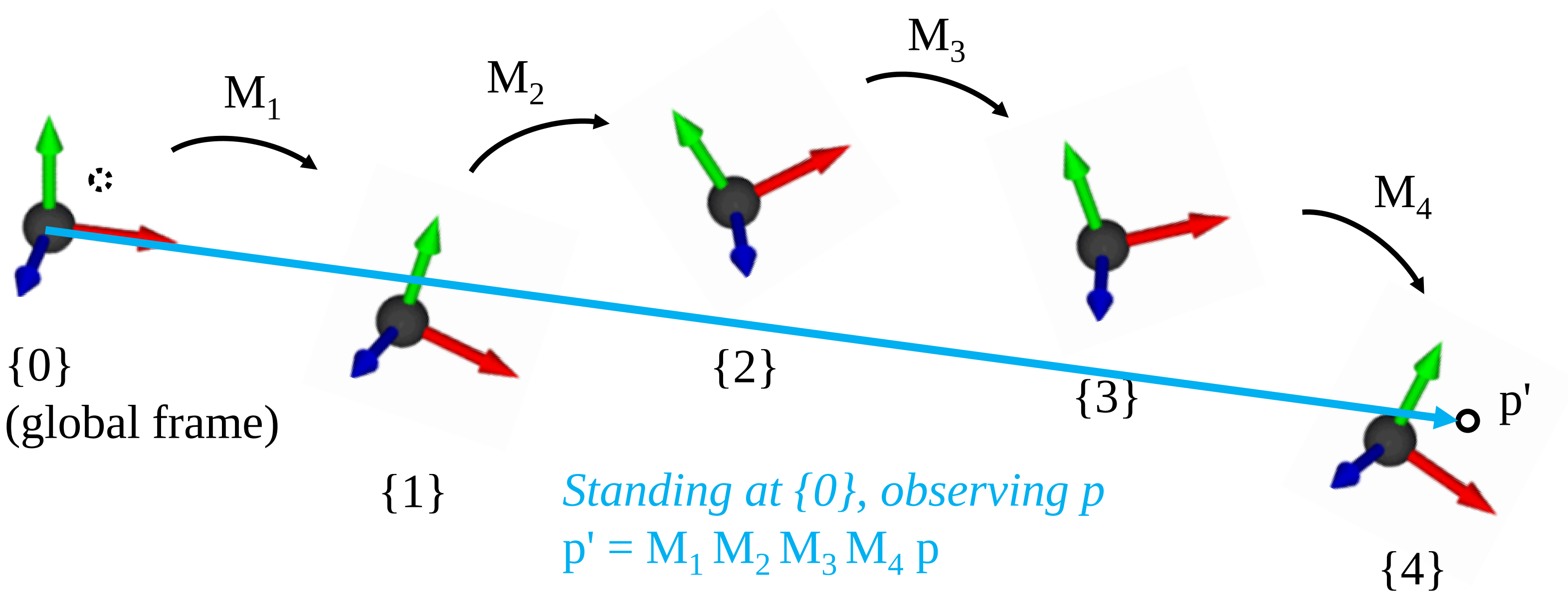
Interpretation of a Series of Transformations #1

- $p' = M_1 \boxed{M_2 M_3 M_4\ p}$



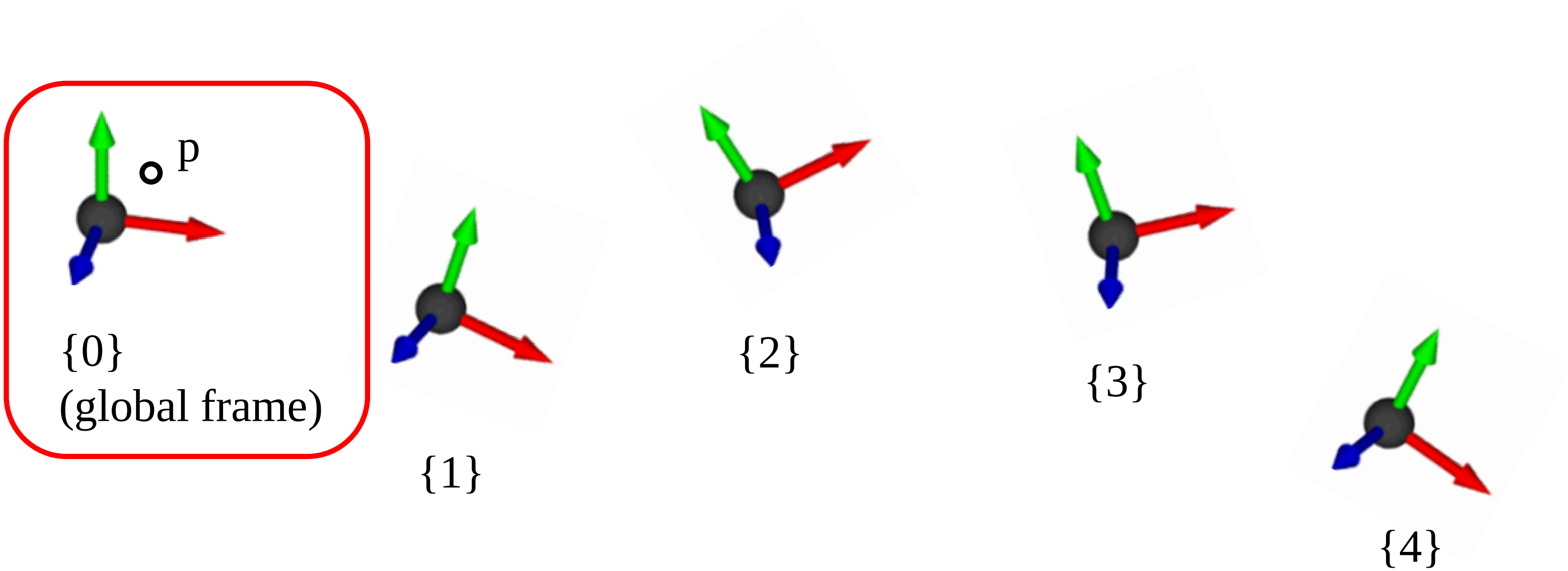
Interpretation of a Series of Transformations #1

- p' = $\boxed{M_1 M_2 M_3 M_4\ p}$



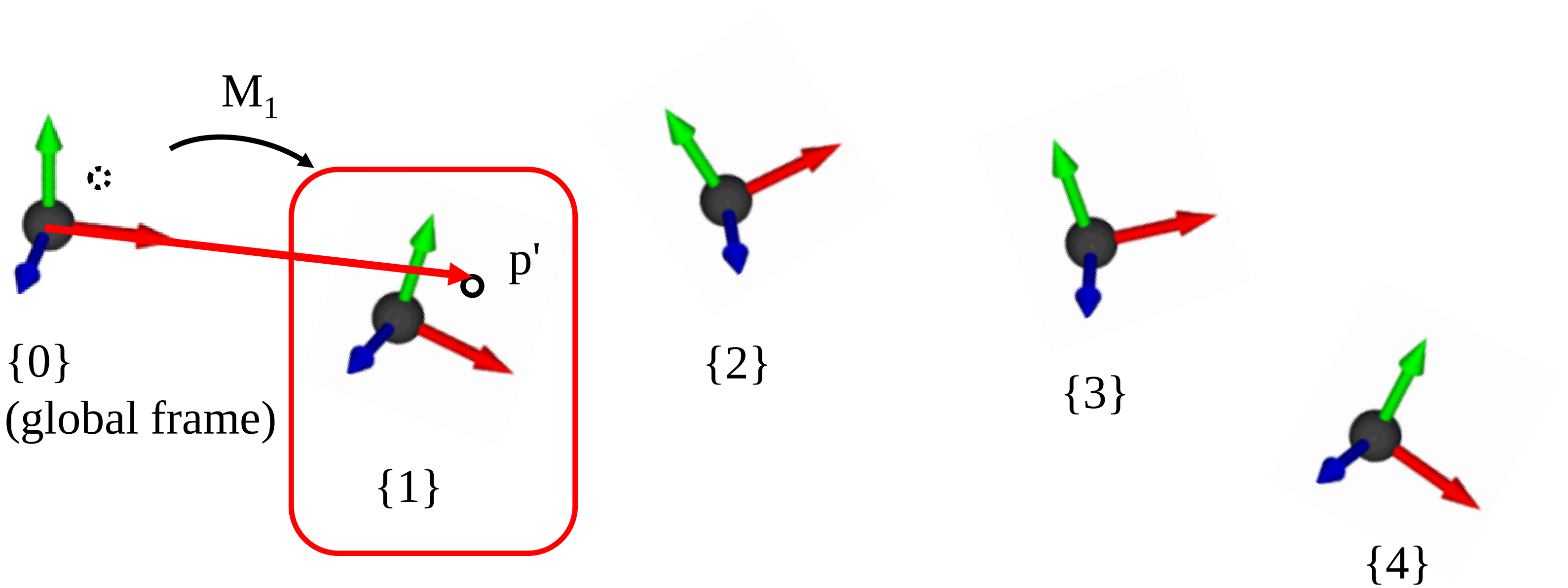
Interpretation of a Series of Transformations #2

- $p' = M_1 M_2 M_3 M_4 \, p$



Interpretation of a Series of Transformations #2

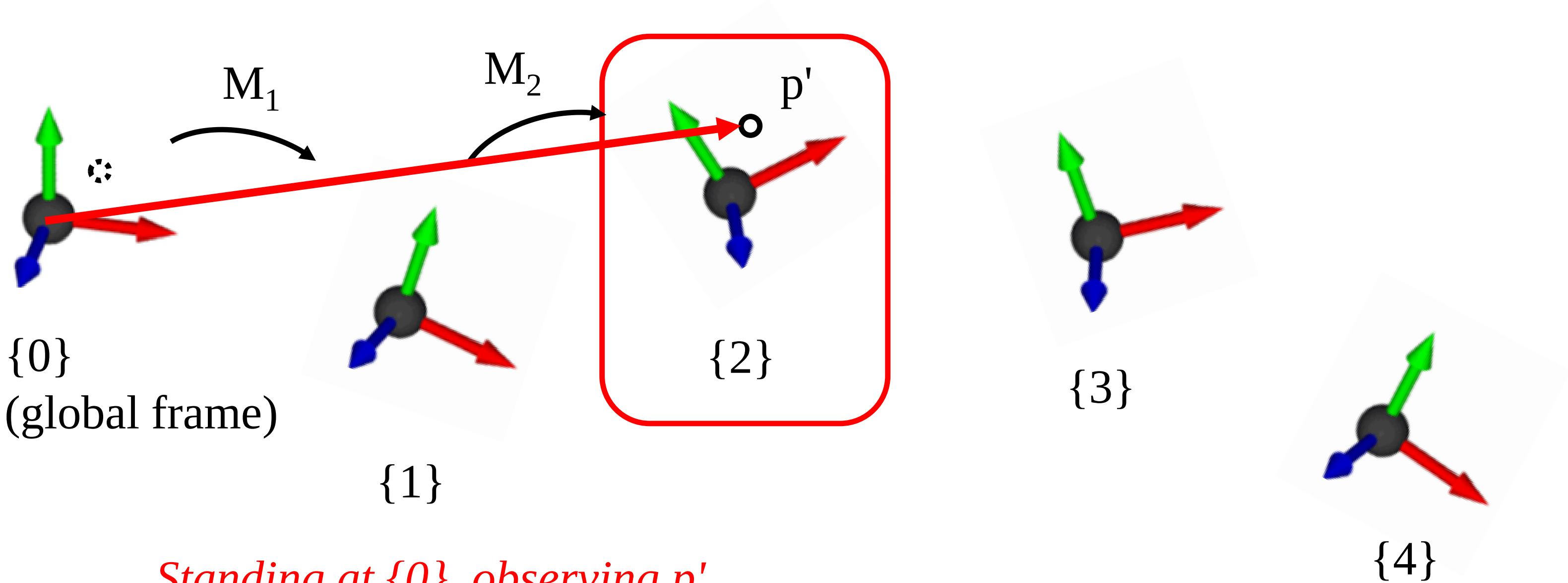
- p' = $\boxed{M_1}M_2M_3M_4$ p



Standing at {0}, observing p'
p' = M_1 p

Interpretation of a Series of Transformations #2

- p' = $\boxed{M_1 M_2} M_3 M_4$ p

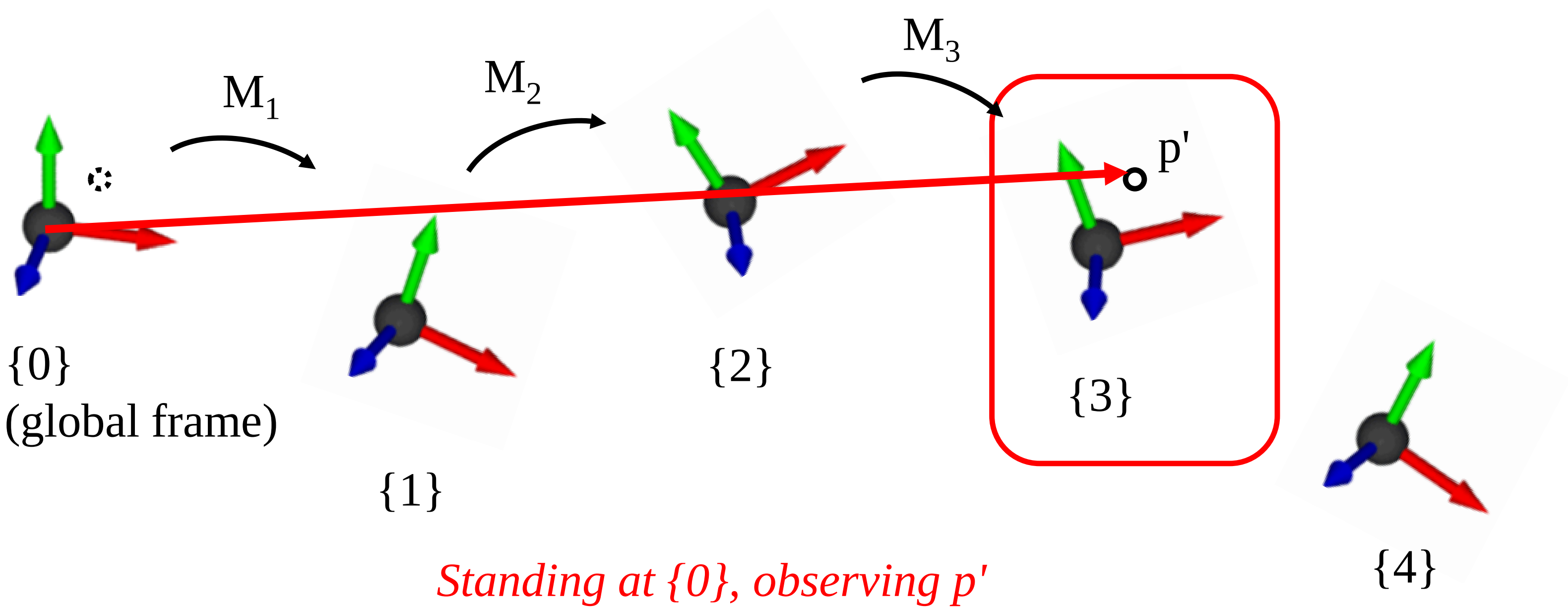


Standing at {0}, observing p'

p' = M_1 M_2 p

Interpretation of a Series of Transformations #2

- $p' = \boxed{M_1 M_2 M_3} M_4\ p$

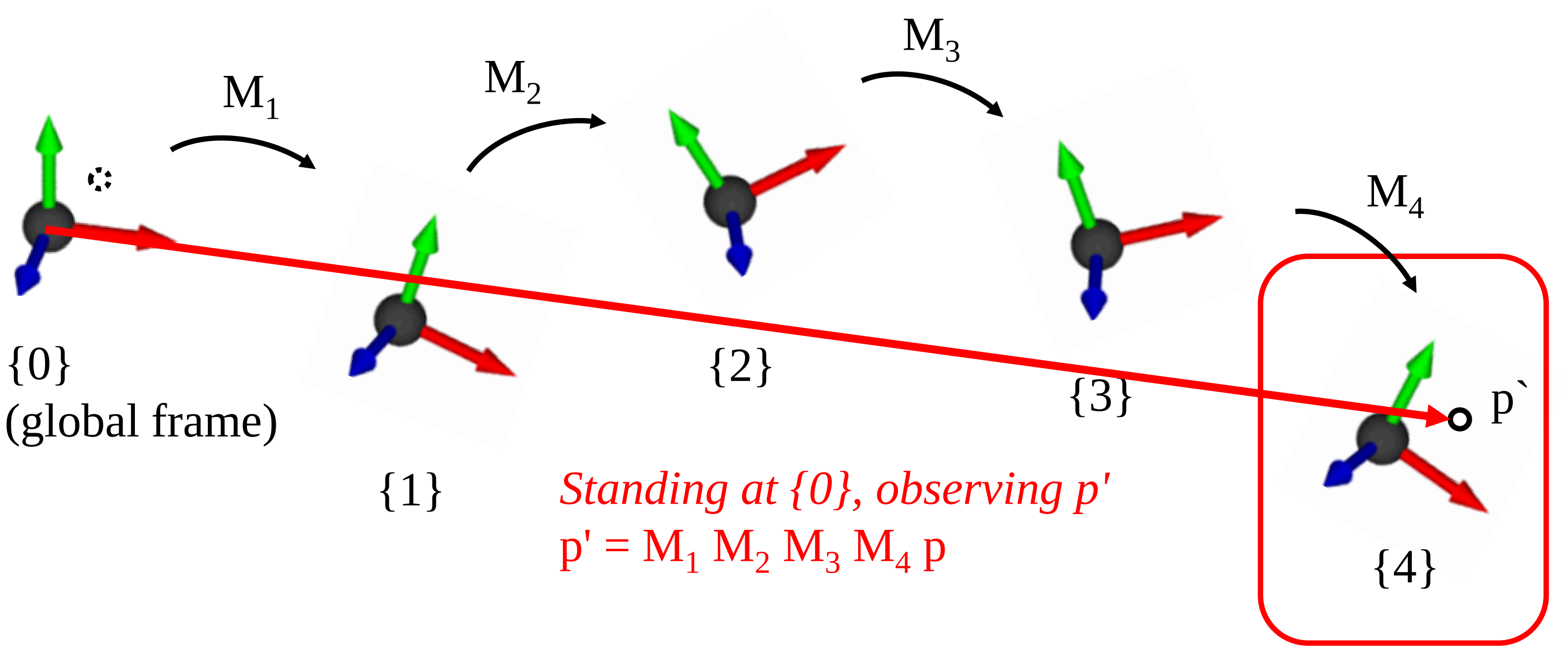


Standing at {0}, observing p'

$p' = M_1\ M_2\ M_3\ p$

Interpretation of a Series of Transformations #2

- p' = $\boxed{M_1 M_2 M_3 M_4 \text{ p}}$



Left & Right Multiplication

- Thinking it deeper, we can see:

- **p' = RTp (left-multiplication by R)**
  - (R-to-L) Apply T to a point p w.r.t. global frame.
  - **Apply R to a point Tp w.r.t. global frame.**

- **p' = TRp (right-multiplication by R)**
  - (L-to-R) Apply T to a point p w.r.t. local frame.
  - **Apply R to a point Tp w.r.t local frame.**

[Practice] Interpretation of Composite Transformations

- Just start from the previous lecture code "[Practice] OpenGL Trans. Functions".

- Differences are:

```
def drawFrame():
    glBegin(GL_LINES)
    glColor3ub(255, 0, 0)
    glVertex3fv(np.array([0.,0.,0.]))
    glVertex3fv(np.array([1.,0.,0.]))
    glColor3ub(0, 255, 0)
    glVertex3fv(np.array([0.,0.,0.]))
    glVertex3fv(np.array([0.,1.,0.]))
    glColor3ub(0, 0, 255)
    glVertex3fv(np.array([0.,0.,0]))
    glVertex3fv(np.array([0.,0.,1.]))
    glEnd()
```

[Practice] Interpretation of Composite Transformations

```
def render(camAng):
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
    glEnable(GL_DEPTH_TEST)
    glLoadIdentity()
    glOrtho(-1,1, -1,1, -1,1)
    gluLookAt(.1*np.sin(camAng),.1,.1*np.cos(camAng), 0,0,0, 0,1,0)

    # draw global frame
    drawFrame()

    # 1) p'=TRp
    glTranslatef(.4, .0, 0)
    drawFrame()     # frame defined by T
    glRotatef(60, 0, 0, 1)
    drawFrame()     # frame defined by TR

    # # 2) p'=RTp
    # glRotatef(60, 0, 0, 1)
    # drawFrame()   # frame defined by R
    # glTranslatef(.4, .0, 0)
    # drawFrame()   # frame defined by RT

    drawTriangle()
```

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”.

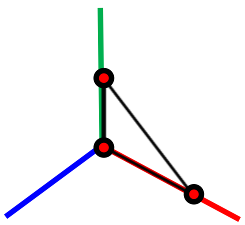
Rendering Pipeline

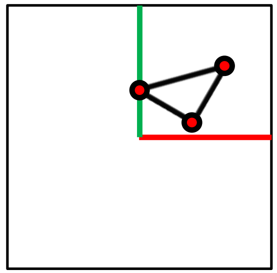
Rendering Pipeline

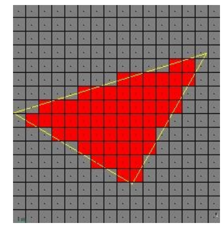
- A conceptual model that describes what steps a graphics system needs to perform to render a 3D scene to a 2D image.

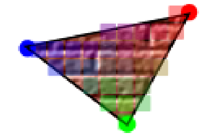
- Also known as **graphics pipeline**.

Rendering Pipeline









vertex processing

: performs a sequence of vertex transformations

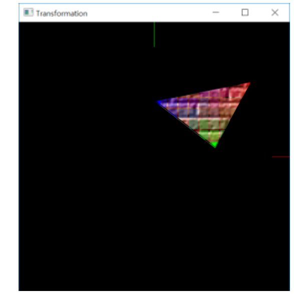
rasterization

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

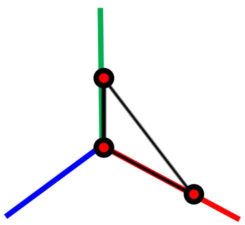
fragment processing

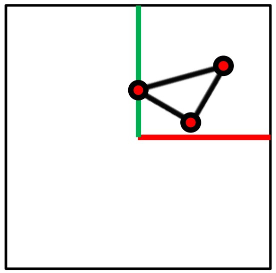
: determines color of each fragment with light & texture

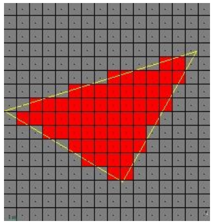
output merging

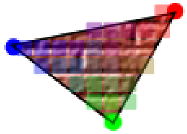


Rendering Pipeline

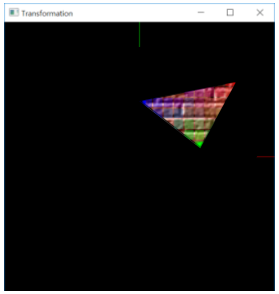








vertex processing → **rasterization** → **fragment processing** → output merging



: performs a sequence of **vertex transformations**

What we've been done so far

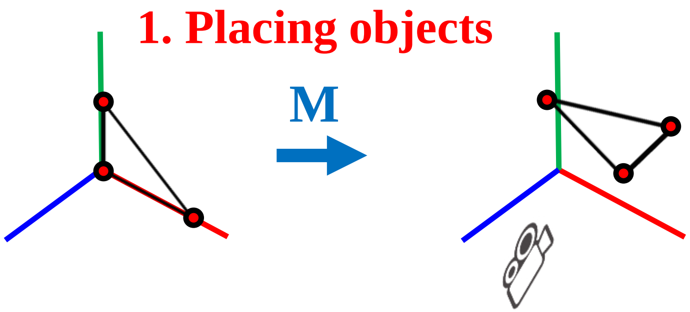
→ We'll see today & next lecture

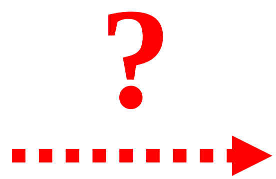
Vertex Processing

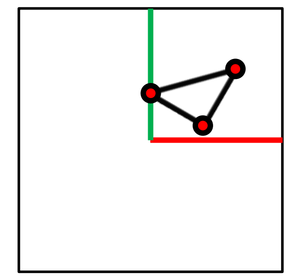
Set vertex positions

Transformed vertices

Vertex positions in 2D viewport







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

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**

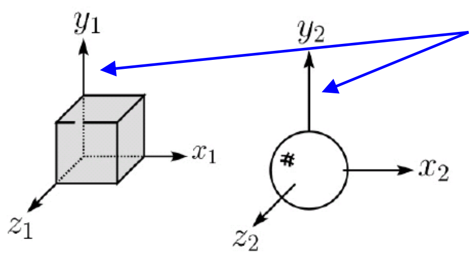
- 4. Displaying on a “cinema screen”

→ **Viewport transformation**

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

Vertex Processing (Transformation Pipeline)

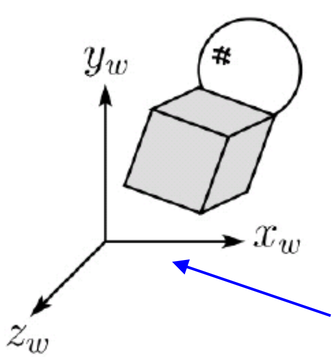
Object space



Local coordinates

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

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

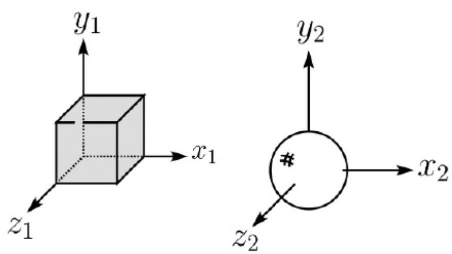


Global coordinates

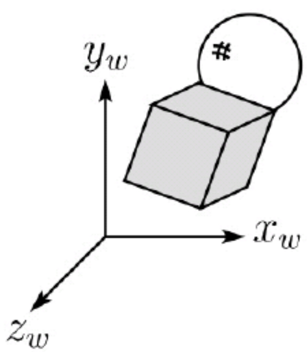
World space

Vertex Processing (Transformation Pipeline)

Object space



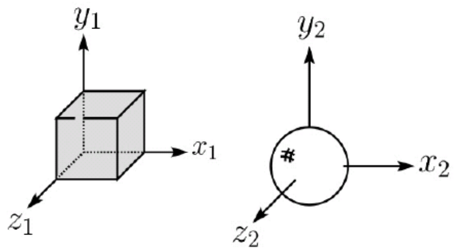
Modeling transformation



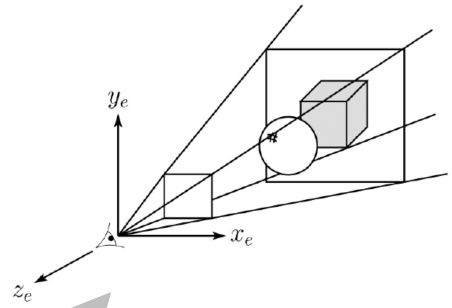
World space

Vertex Processing (Transformation Pipeline)

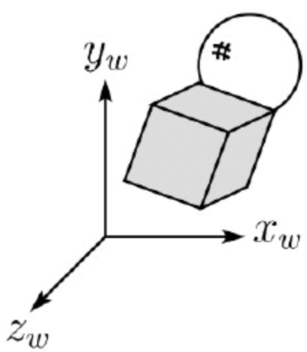
Object space



View space
(Camera space)

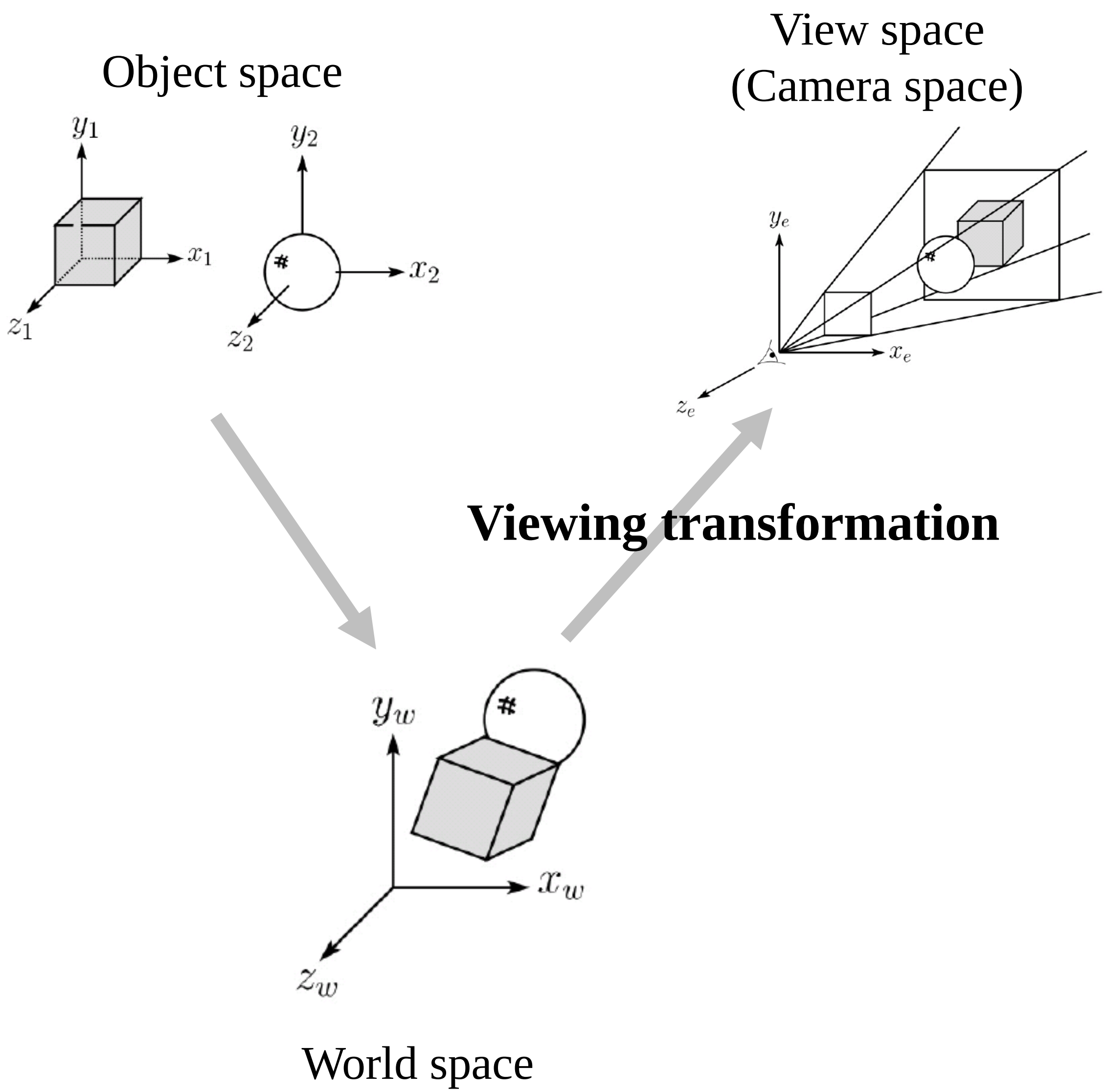


Placing the "camera"

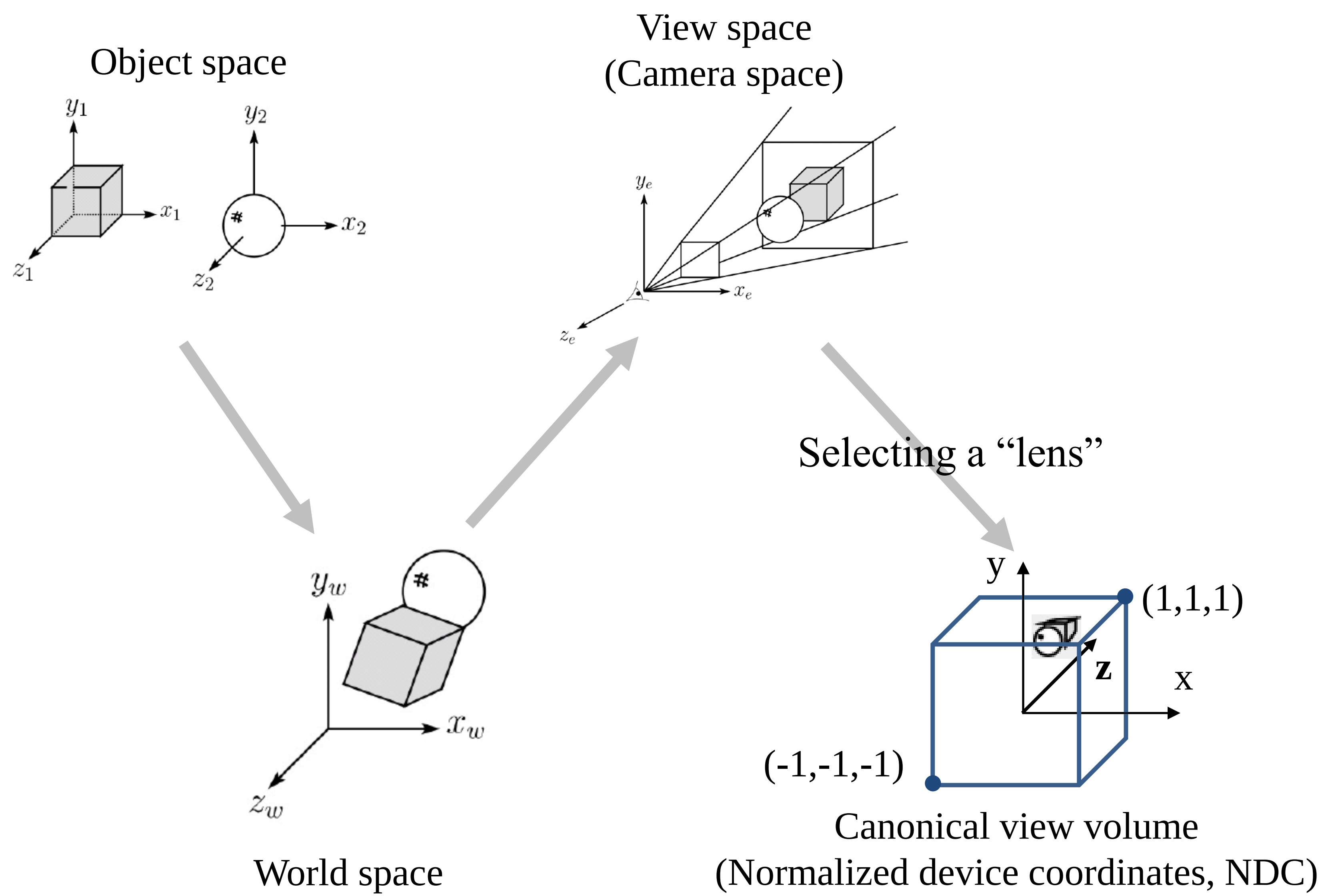


World space

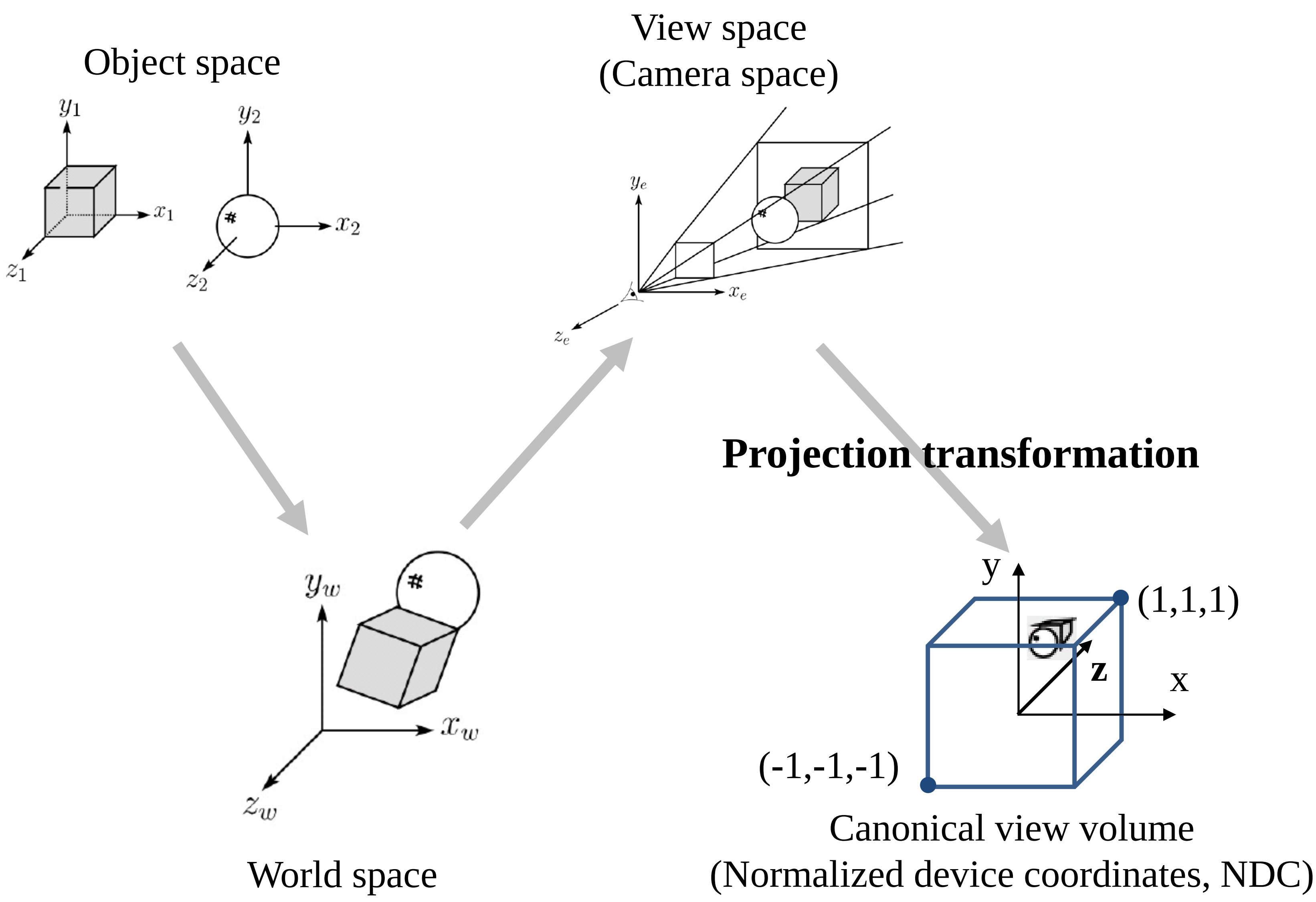
Vertex Processing (Transformation Pipeline)



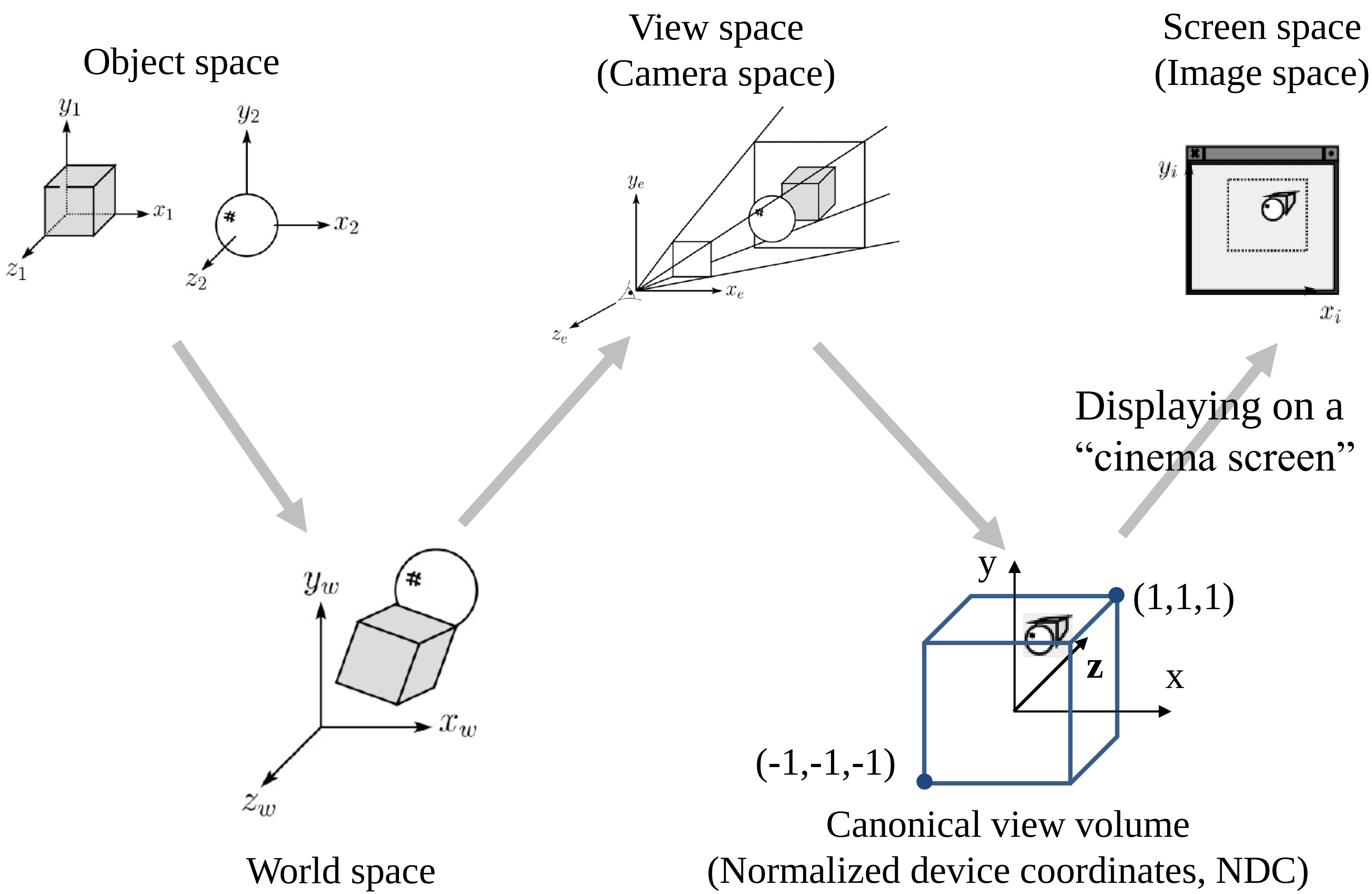
Vertex Processing (Transformation Pipeline)



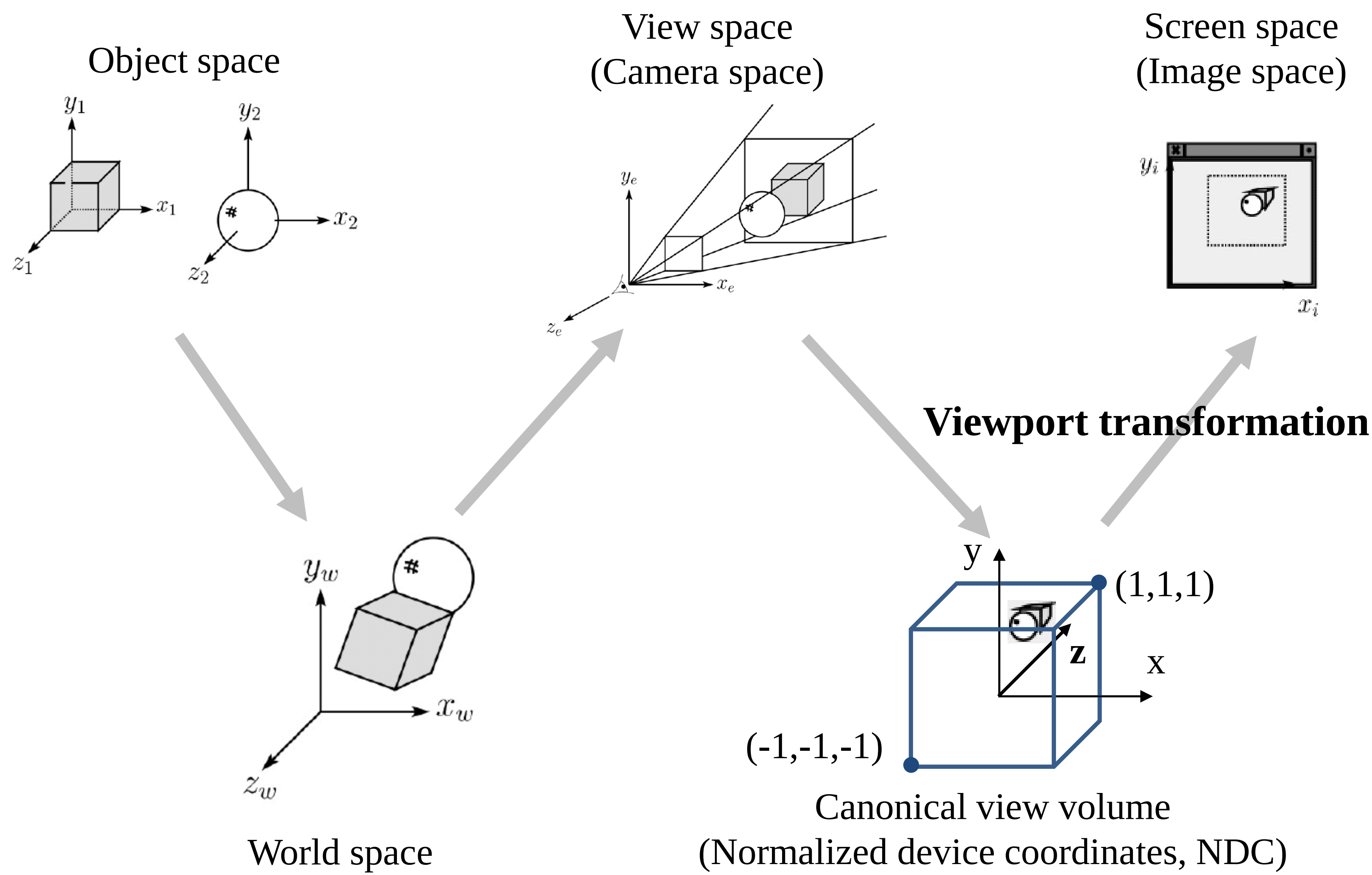
Vertex Processing (Transformation Pipeline)



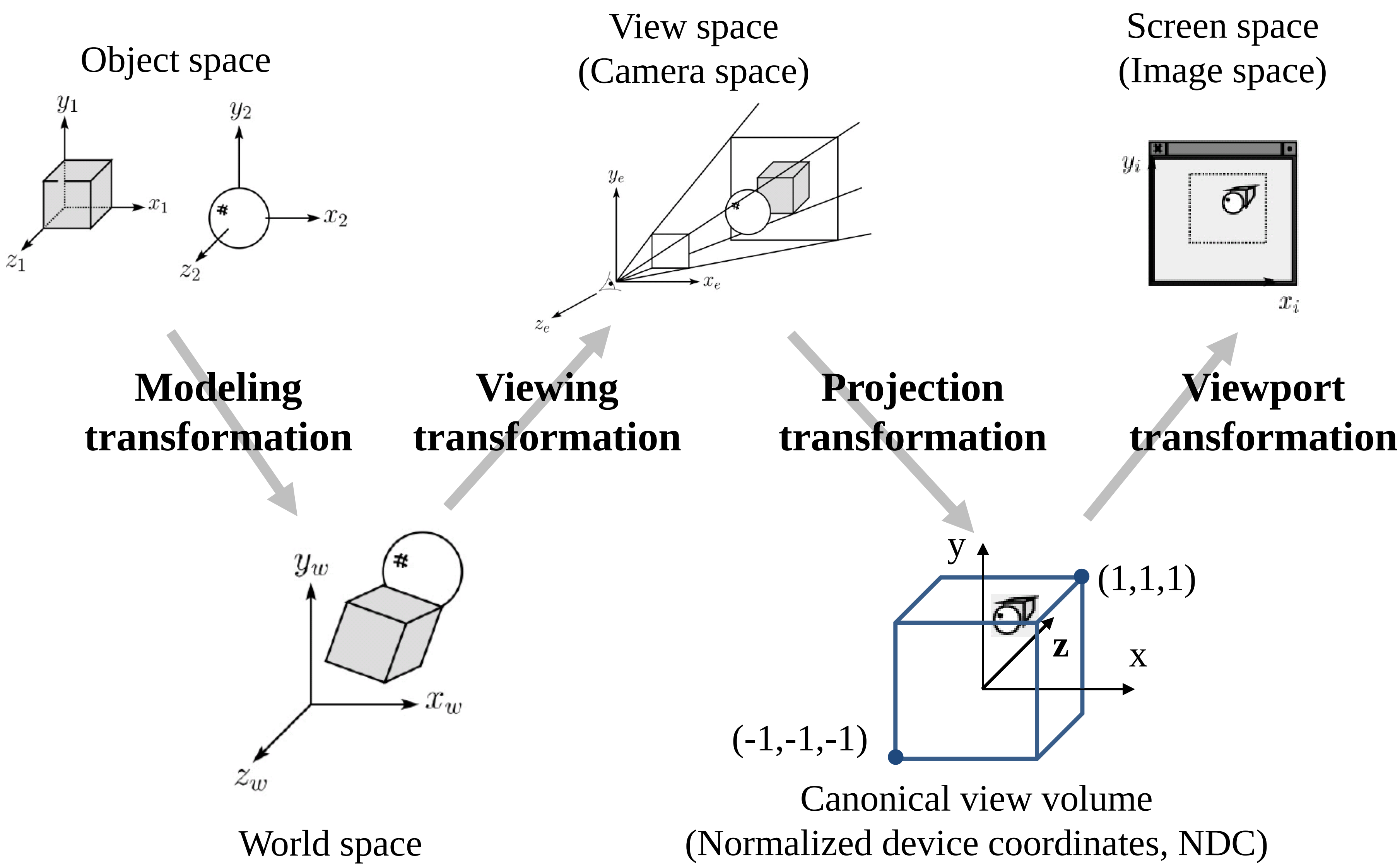
Vertex Processing (Transformation Pipeline)



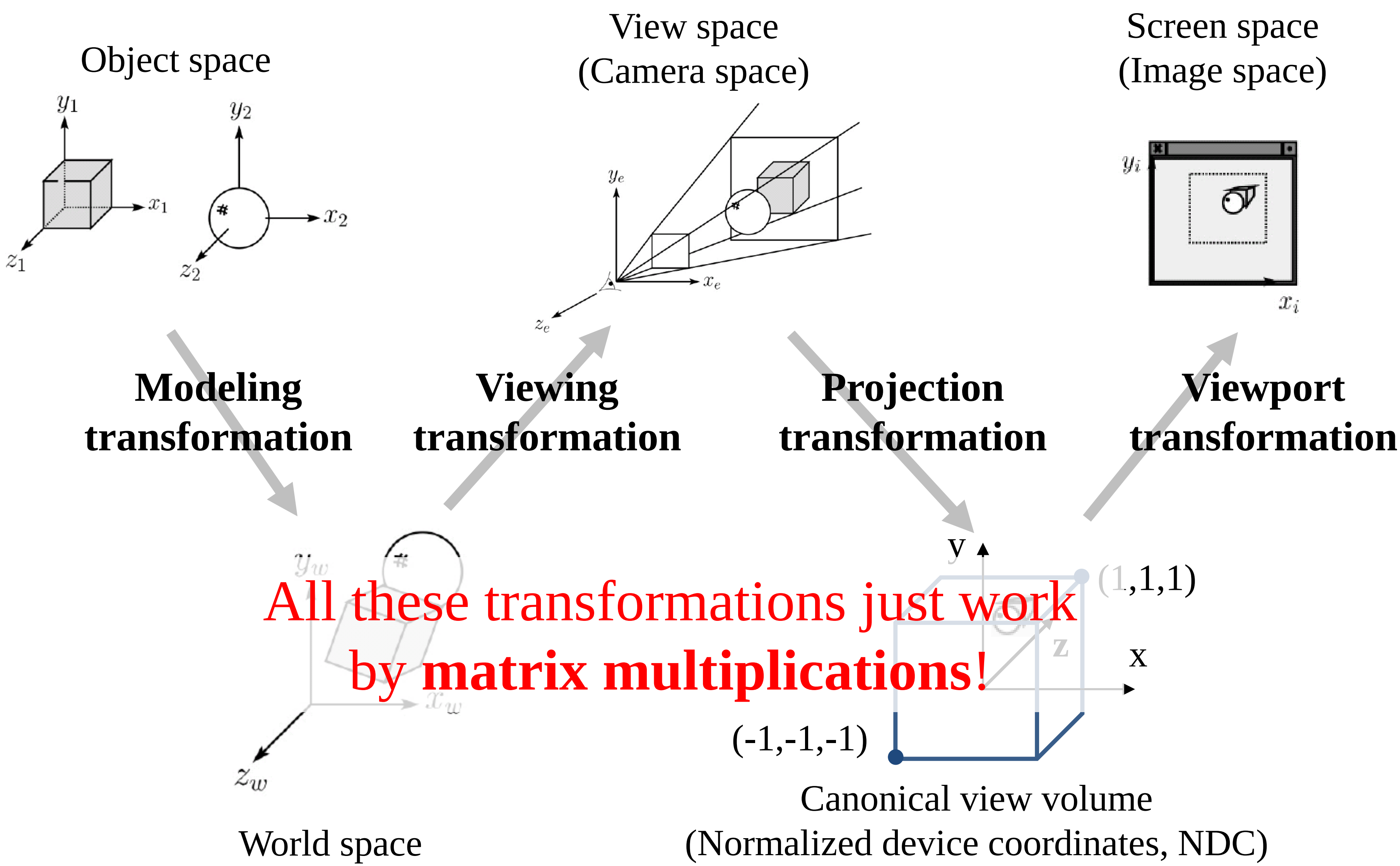
Vertex Processing (Transformation Pipeline)



Vertex Processing (Transformation Pipeline)

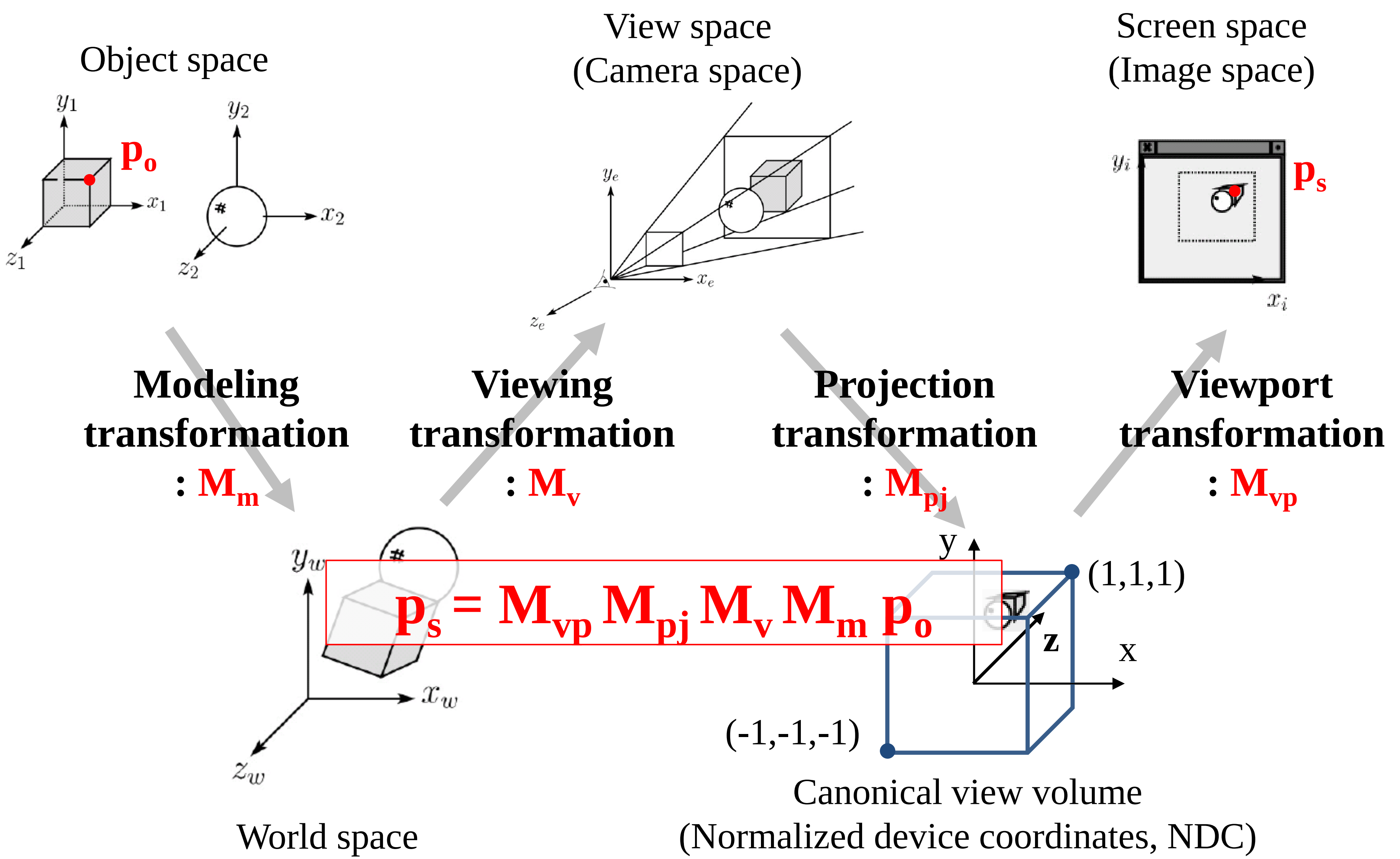


Vertex Processing (Transformation Pipeline)



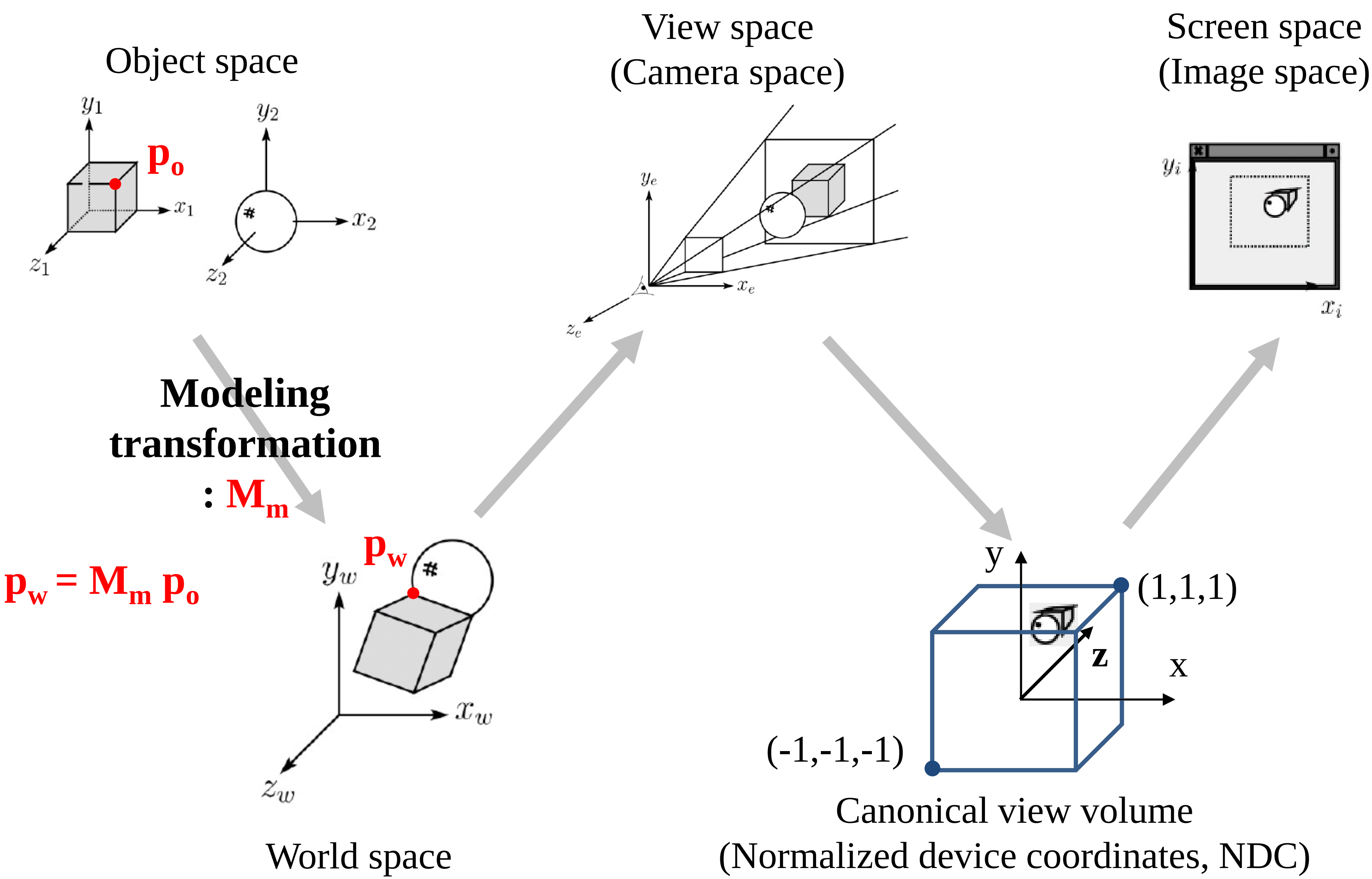
All these transformations just work by **matrix multiplications!**

Vertex Processing (Transformation Pipeline)



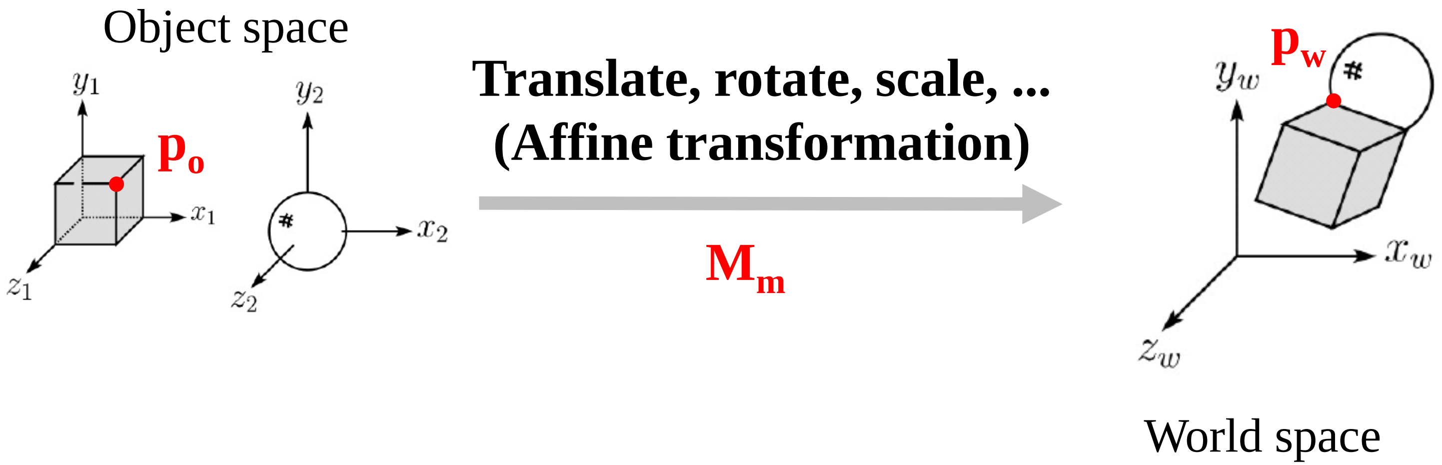
$$\mathbf{p_s = M_{vp}\, M_{pj}\, M_v\, M_m\, p_o}$$

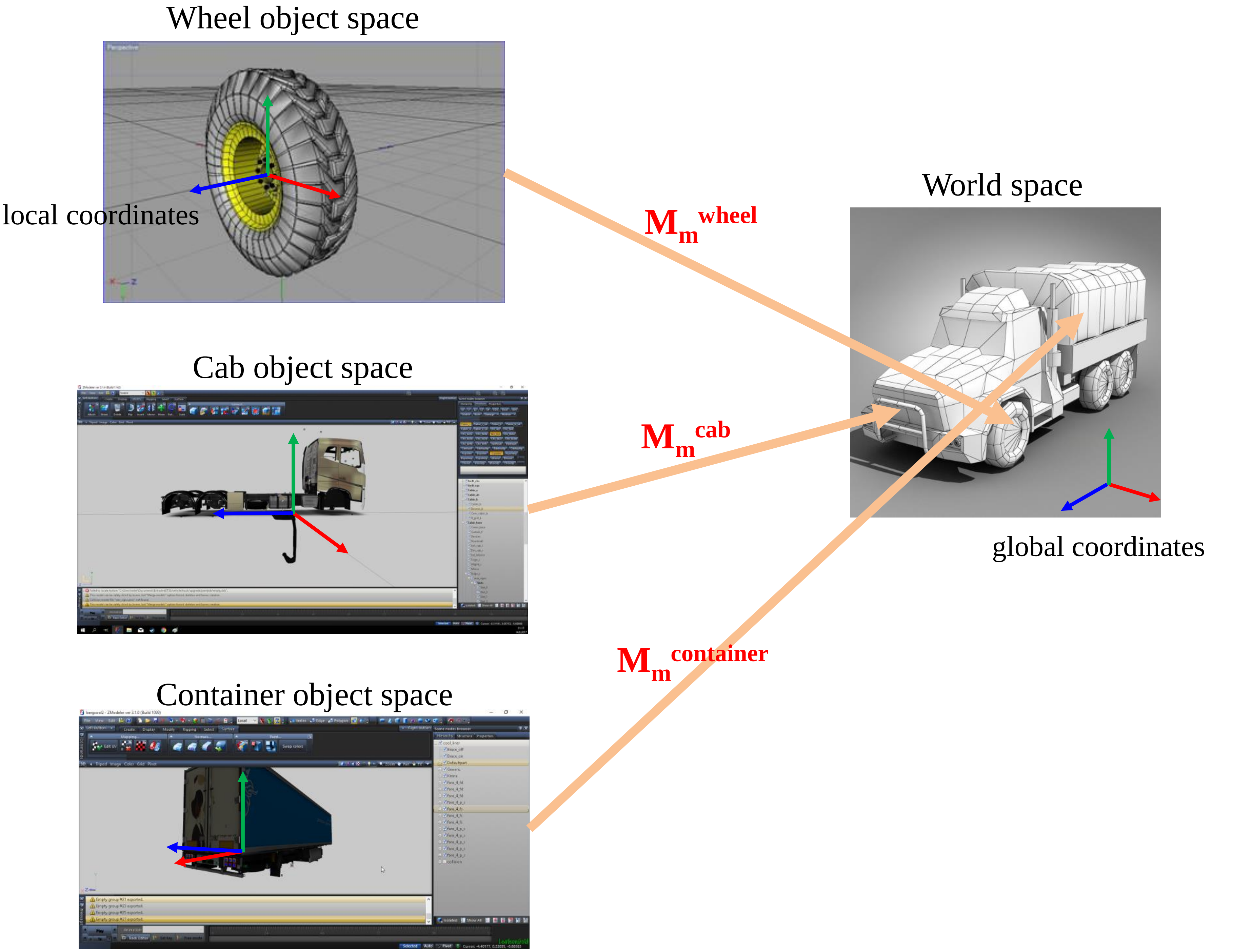
Modeling Transformation



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!)




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:
  - No lab this week, but **the assignment will be handed out** with extended due.

- 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