

# **Viewing Transformations**

## **CS425: Computer Graphics I**

**Fabio Miranda**

<https://fmiranda.me>

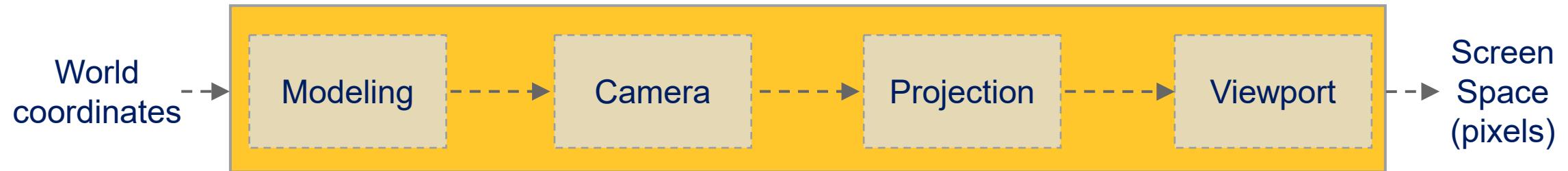
# Overview

---

- Coordinate systems
- Camera
- Viewing transformations
- Orthographic and perspective projections
- Hidden surface removal

# Viewing transformations

- Viewing transformation is the mapping of coordinates of points and lines from world coordinates into screen space pixels.



# Viewing transformations

---

- Previously, we saw how transformations can manipulate primitives (points, vectors) in space.
- Today, we will see how transformations can manipulate primitives to 2D screen coordinates (that will be later rasterized).

# Perspective projection

Distant objects appear smaller

Parallel lines converge at the horizon



# Early paintings

---



Lorsch Gospels (8<sup>th</sup> century)

# Perspective in art

---

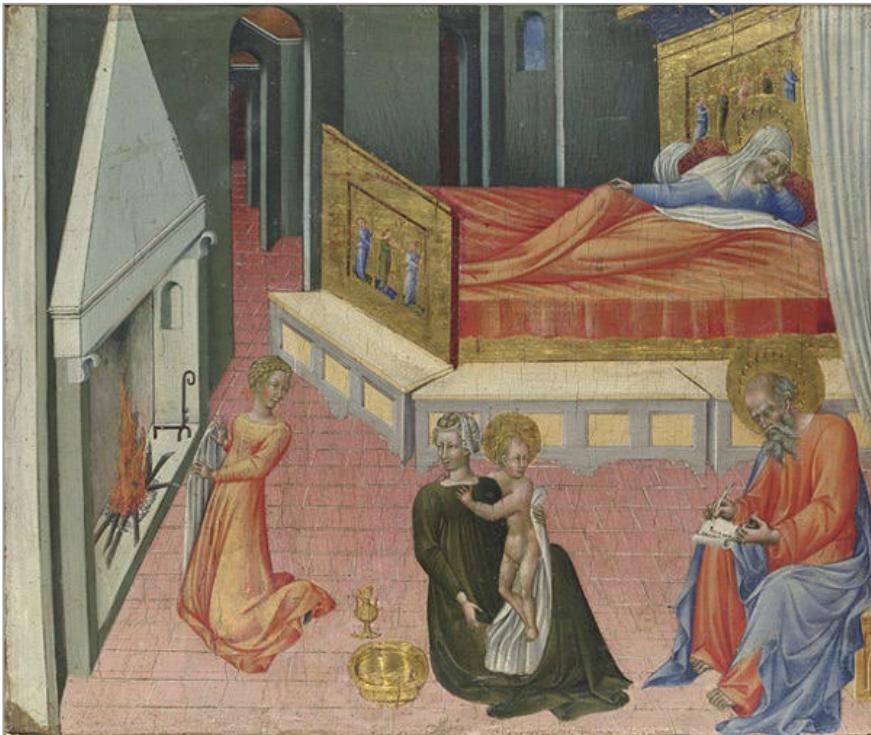


Giotto di Bondone (14<sup>th</sup> century)



Giotto di Bondone (14<sup>th</sup> century)

# Birth of perspective in art: One-point perspective



Giovanni di Paolo (15<sup>th</sup> century)



Piero della Francesca (15<sup>th</sup> century)

# Birth of perspective in art: One-point perspective



Perugino (15<sup>th</sup> century)

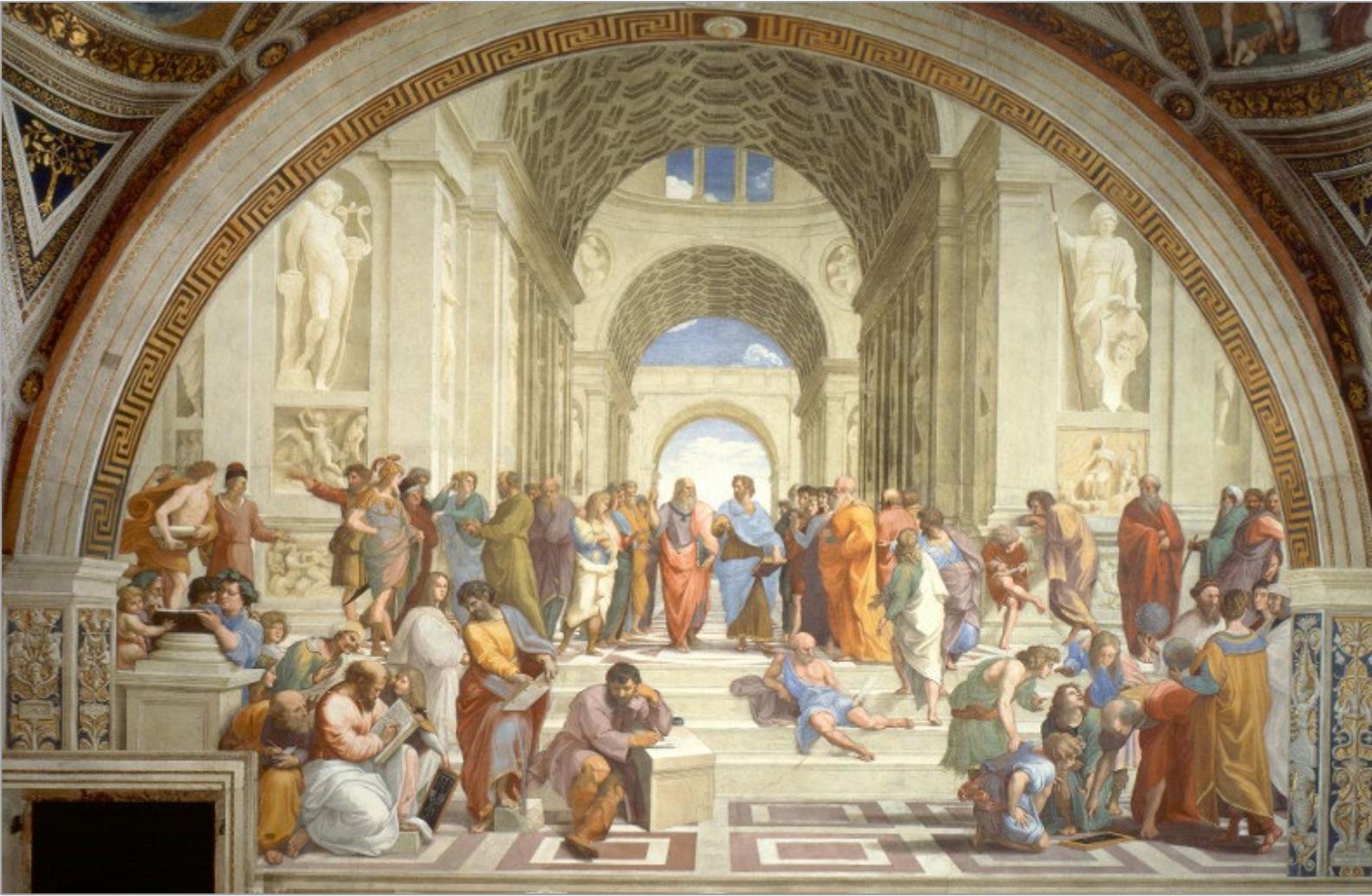
# Birth of perspective in art: One-point perspective



The Ideal City (15<sup>th</sup> century)

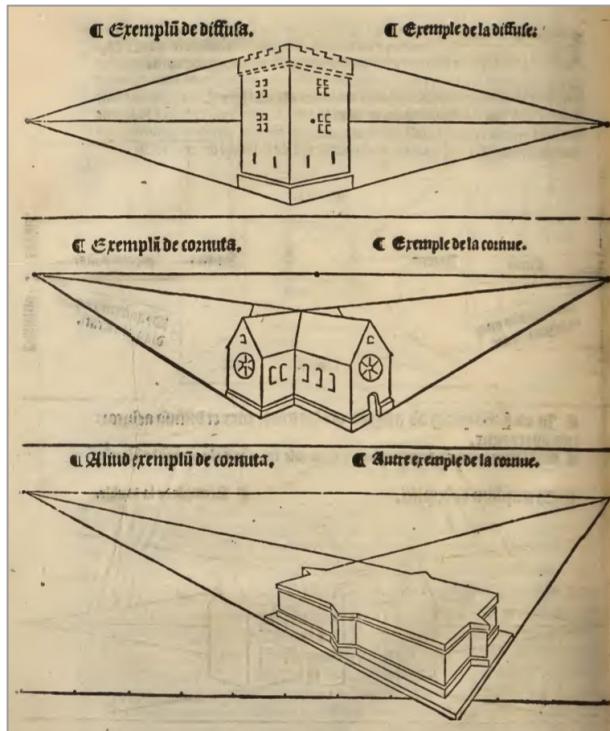
# Birth of perspective in art: One-point perspective

---



Rafael (16<sup>th</sup> century)

# Birth of perspective in art: Two-point perspective

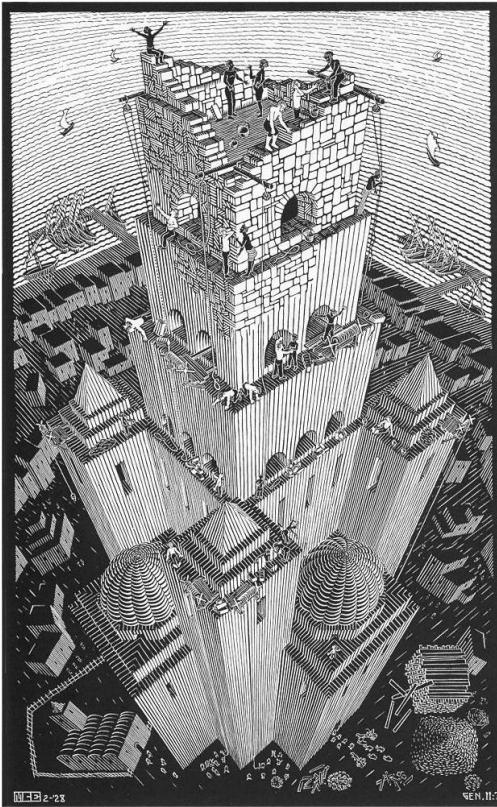


Jean Pélérin (16<sup>th</sup> century)



Gustave Caillebotte (19<sup>th</sup> century)

# Birth of perspective in art: Three-point perspective



M.C. Escher (1928)



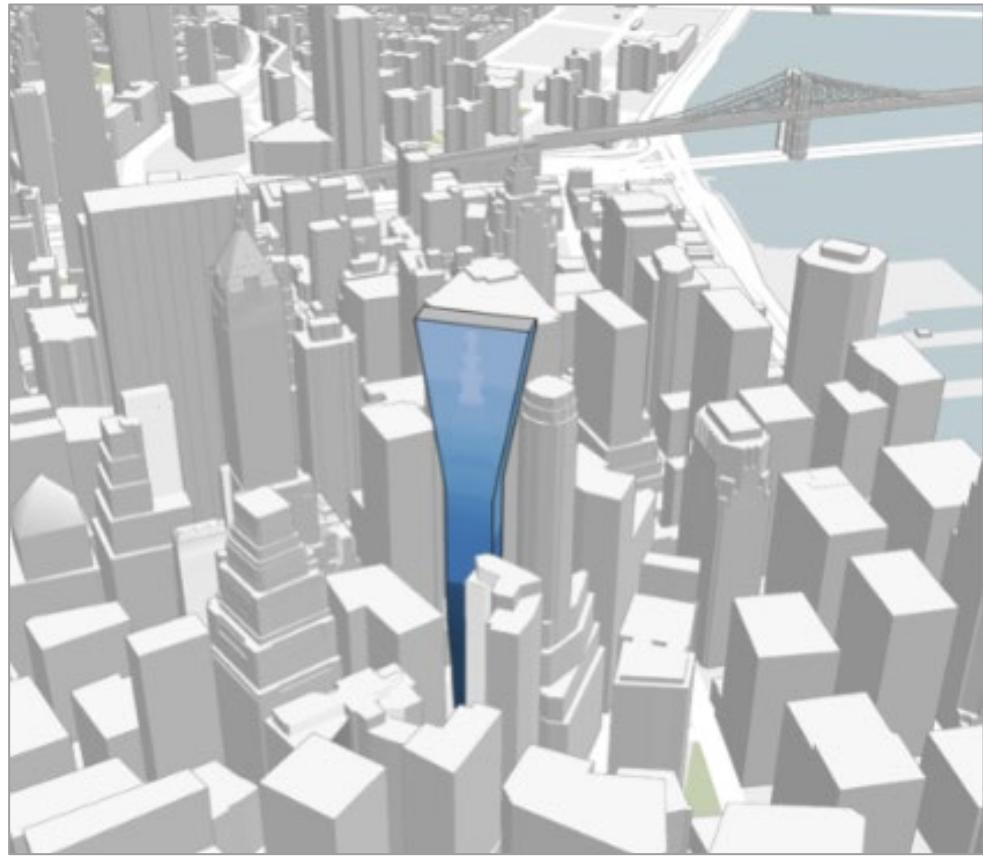
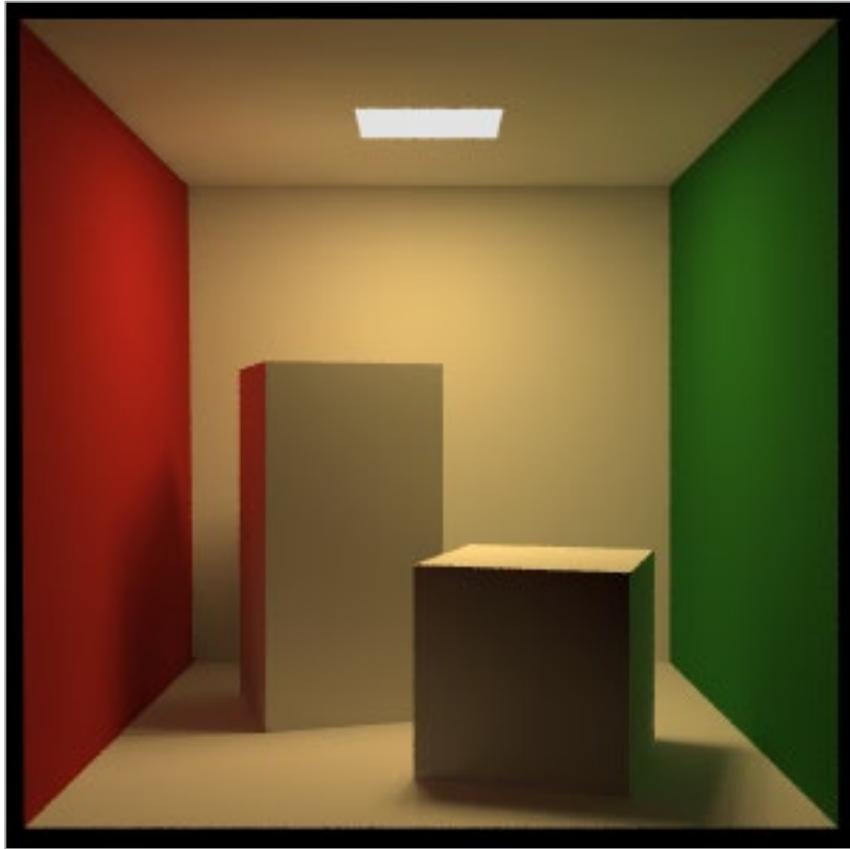
# Multi perspective



The Frozen City by  
Matthias A. K.  
Zimmermann (2006)

# Perspective in computer graphics

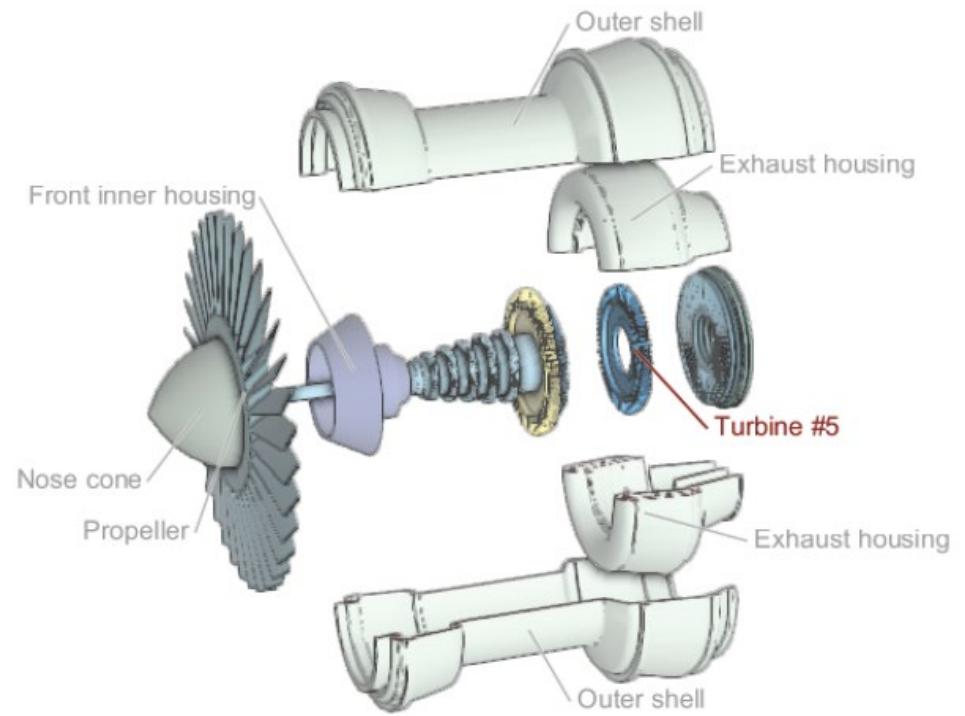
---



# Rejection of perspective in CG

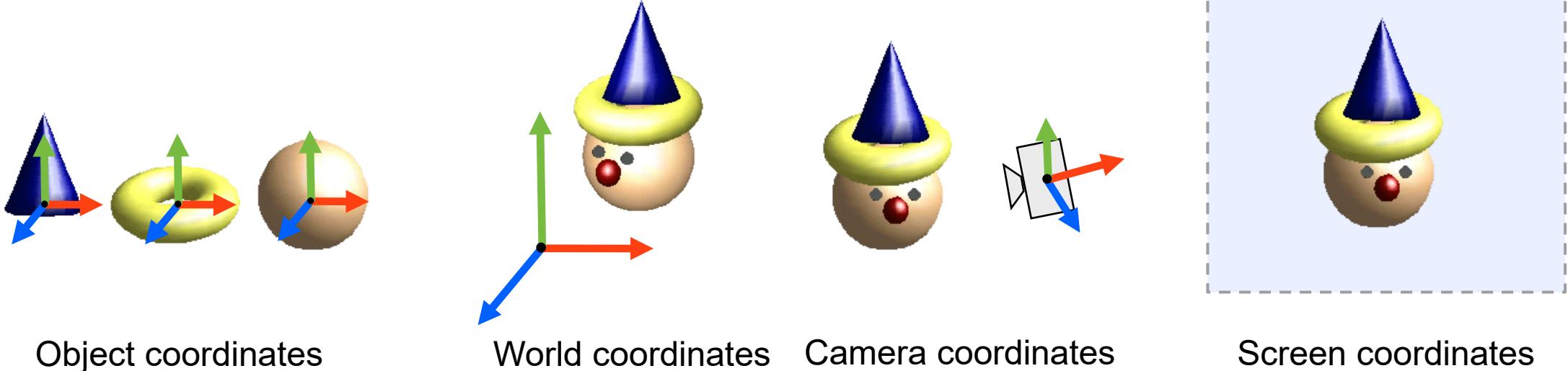


SimCity 2000



“Automated Generation of Interactive 3D  
Exploded Views”

# Coordinate spaces



Object coordinates

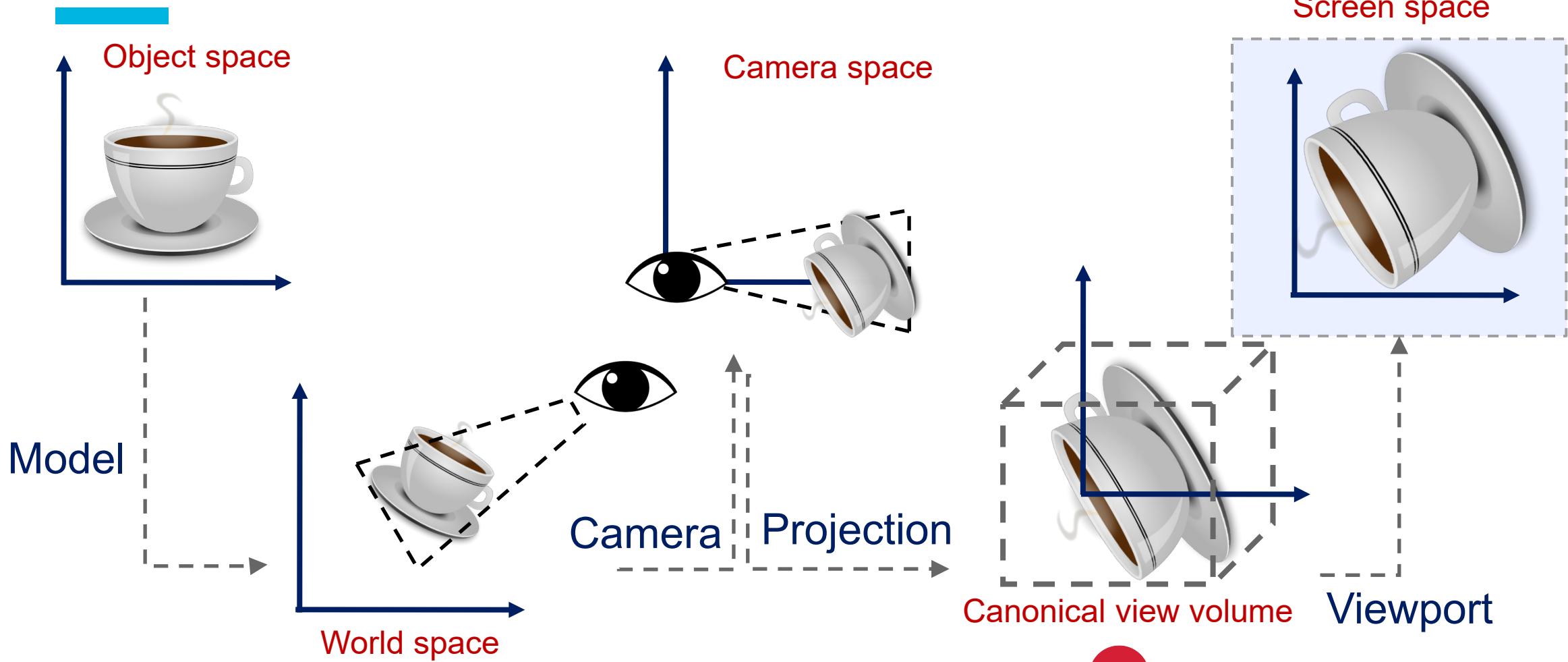
World coordinates

Camera coordinates

Screen coordinates

From: Mark Pauly

# Viewing transformation



# Camera transformation

---

- Construct the camera reference system:
  - The eye position  $e$ .
  - The forward direction  $d$ .
  - The view-up vector  $u$ .
- A view matrix transform all coordinates into view coordinates.

# Change of frame

- Given a vector  $\mathbf{v} = x\mathbf{i} + y\mathbf{j}$  in the standard frame, to take a vector  $\mathbf{v}' = x'\mathbf{b}_1 + y'\mathbf{b}_2$  to the standard, we solve the linear system:

$$x\mathbf{i} + y\mathbf{j} = x'\mathbf{b}_1 + y'\mathbf{b}_2$$

# Change of frame

- Given a vector  $\mathbf{v} = xi + yj$  in the standard frame, to take a vector  $\mathbf{v}' = x'b_1 + y'b_2$  to the standard, we solve the linear system:

$$xi + yj = x'b_1 + y'b_2$$

- In matrix form:

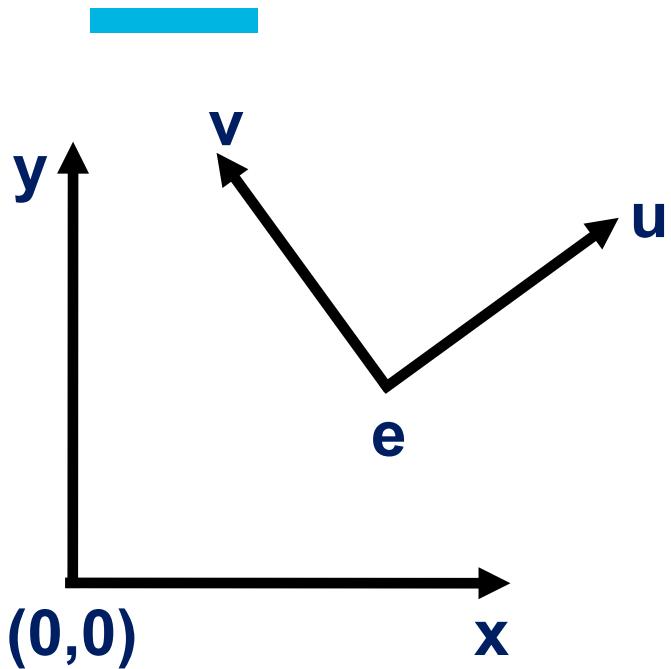
$$\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = (\mathbf{b}_1 \quad \mathbf{b}_2) \begin{pmatrix} x' \\ y' \end{pmatrix}$$

- Other way around, find inverse of the basis matrix

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}^{-1} \begin{pmatrix} x \\ y \end{pmatrix}$$

# Change of frame



$$\text{Rotation: } \mathbf{R} = \begin{pmatrix} \mathbf{u} & \mathbf{v} & \mathbf{w} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\text{Translate: } \mathbf{T} = \begin{pmatrix} 0 & 0 & 0 & \mathbf{e} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{M}_{e \rightarrow o} = \mathbf{TR}$$

$$\mathbf{M}_{o \rightarrow e} = (\mathbf{TR})^{-1} = \mathbf{R}^{-1}\mathbf{T}^{-1}$$

We know:

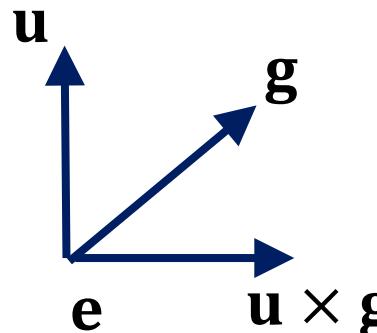
$$\mathbf{R}^{-1} = \mathbf{R}^T \text{ (Pure rotation is orthogonal)}$$

$$\mathbf{T}^{-1} = \begin{pmatrix} 0 & 0 & 0 & -\mathbf{e} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



# Camera transformation

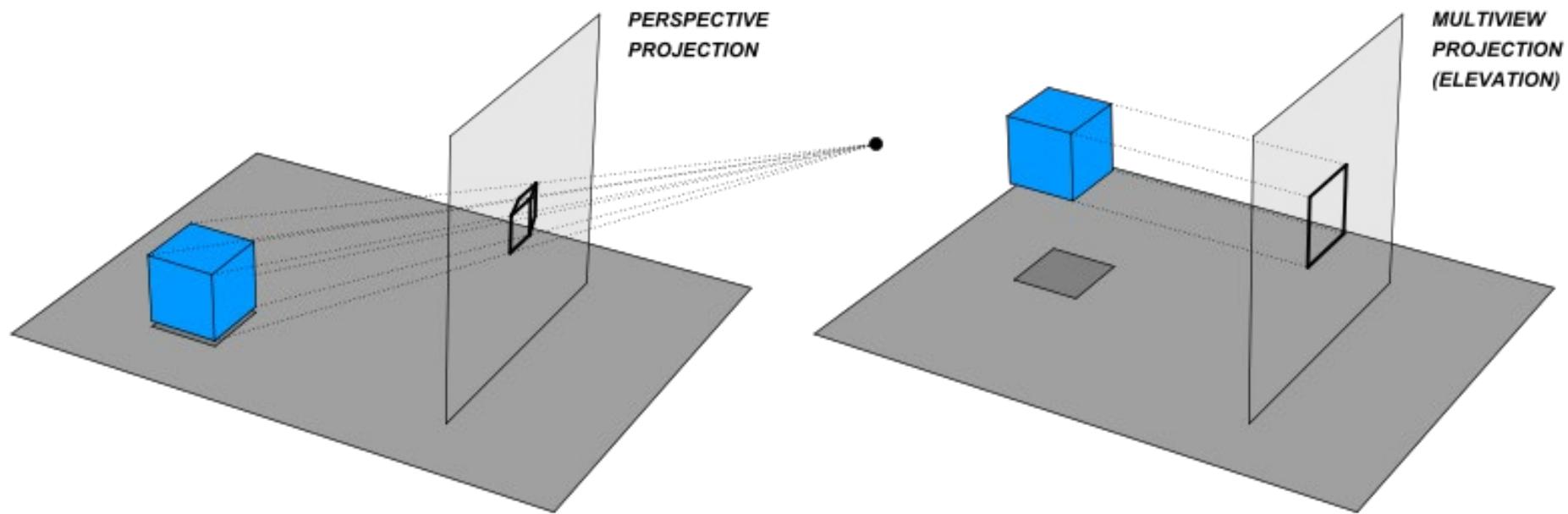
- Construct the camera reference system:
  - The eye position  $e$ .
  - The forward direction  $d$ .
  - The view-up vector  $u$ .
- A view matrix transform all coordinates into view coordinates.



$$\mathbf{M}_{camera2world} = \begin{pmatrix} \mathbf{u} \times \mathbf{g} & \mathbf{u} & \mathbf{d} & \mathbf{e} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

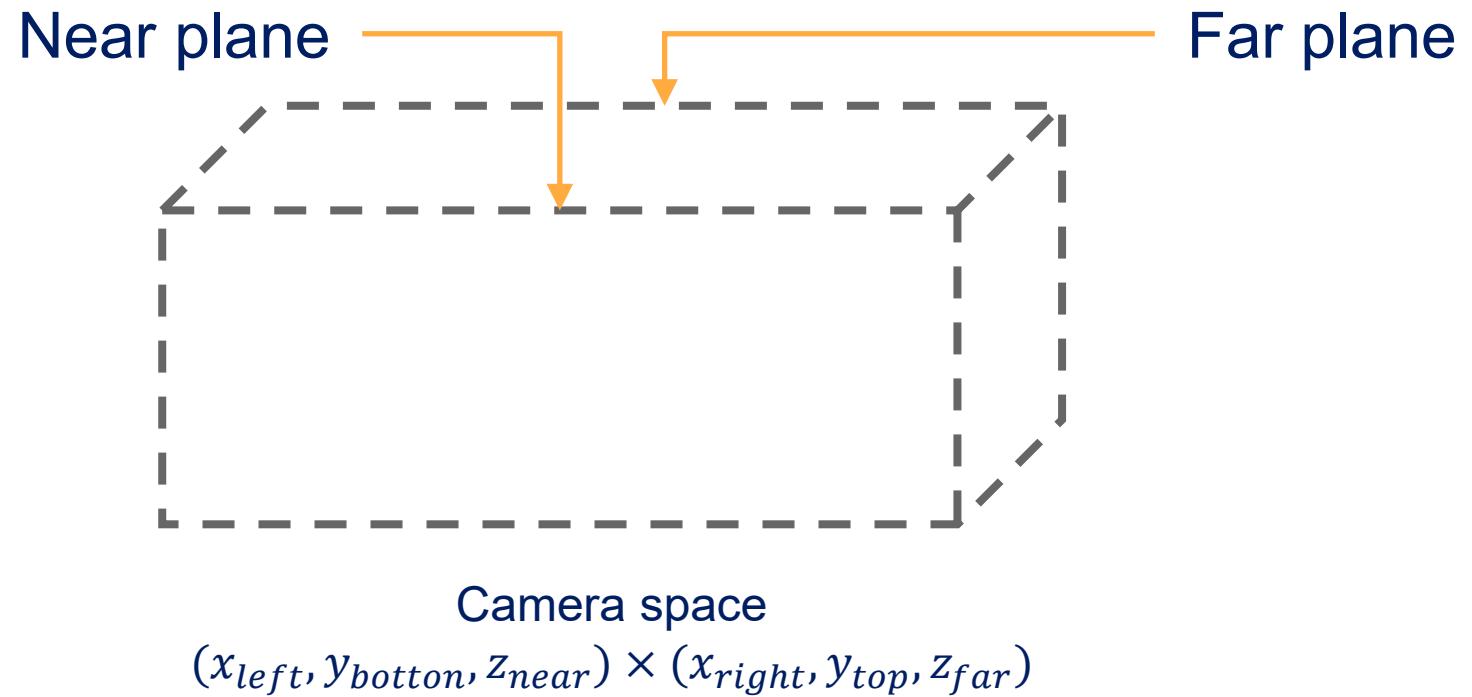
$$\mathbf{M}_{world2camera} = \begin{pmatrix} \mathbf{u} \times \mathbf{g} & \mathbf{u} & \mathbf{d} & \mathbf{e} \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1}$$

# Orthographic and perspective projections

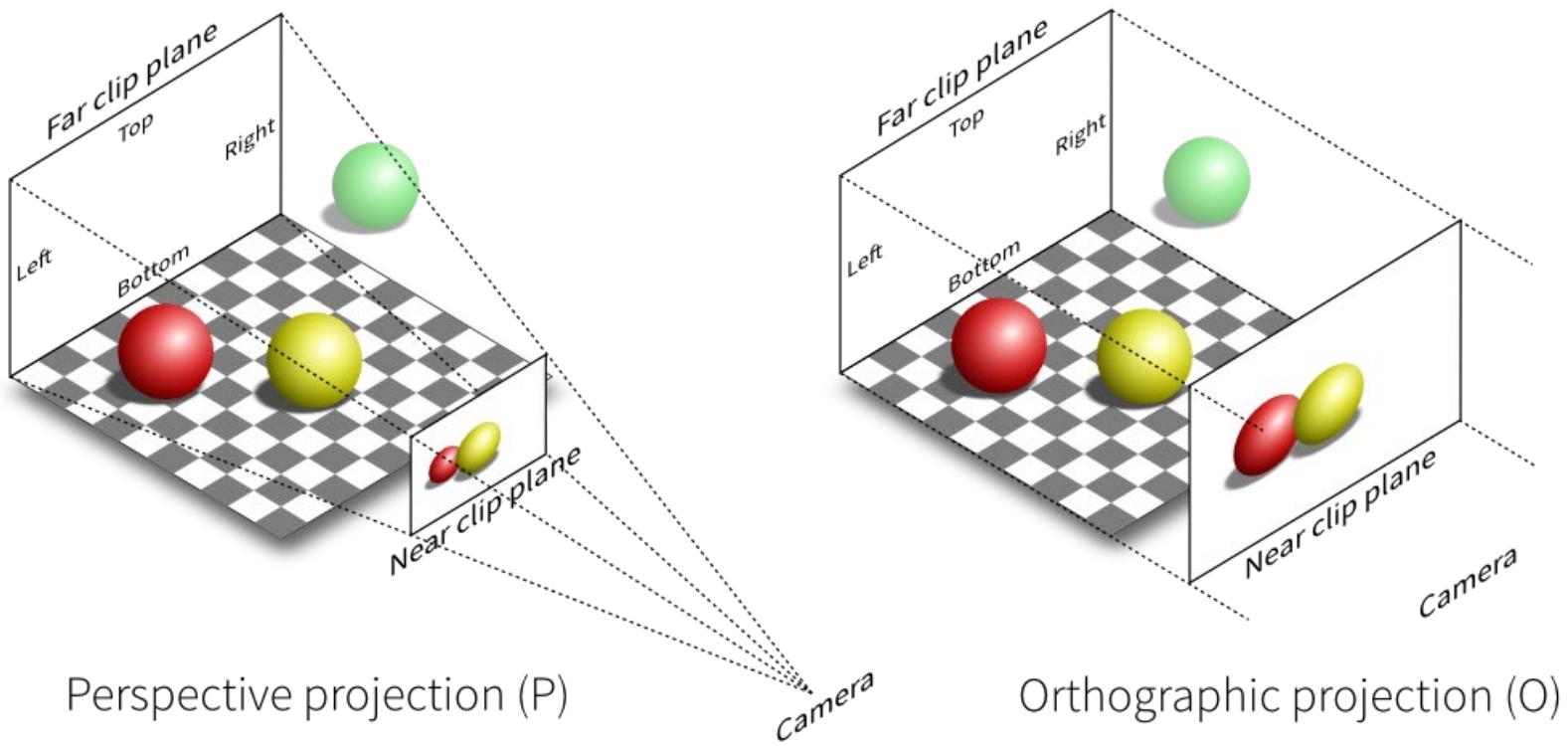


[https://en.wikipedia.org/wiki/File:Various\\_projections\\_of\\_cube\\_above\\_plane.svg](https://en.wikipedia.org/wiki/File:Various_projections_of_cube_above_plane.svg)

# View frustum



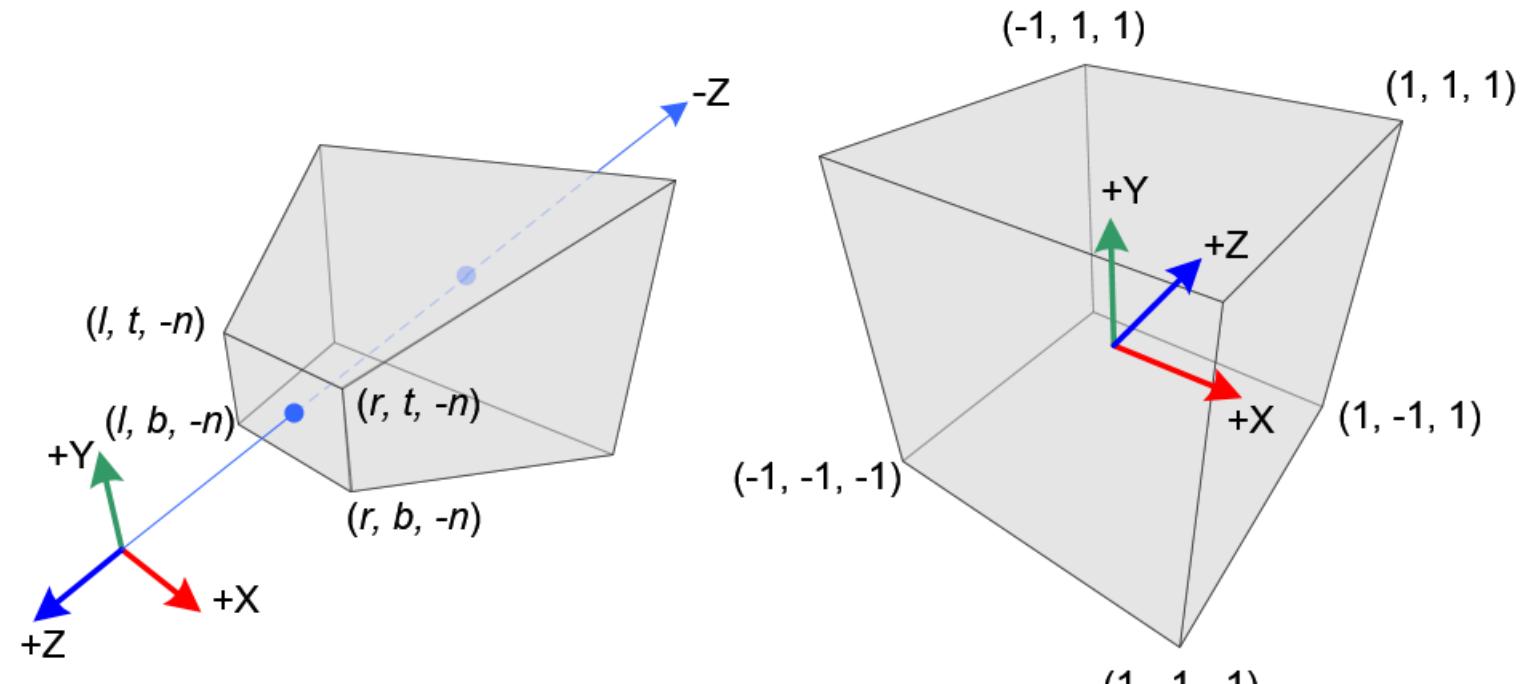
# View frustum



Perspective projection (P)

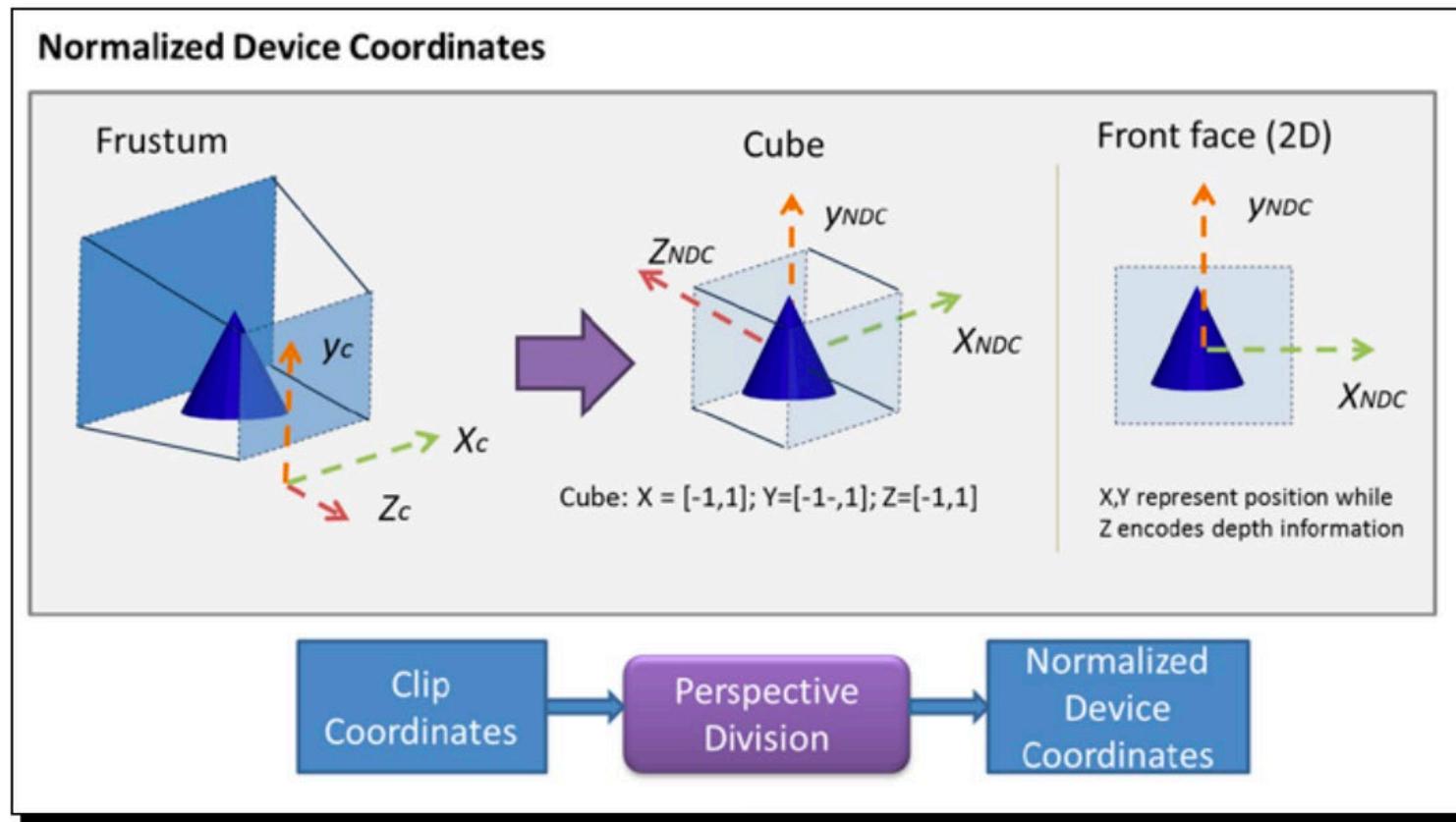
Orthographic projection (O)

# View frustum



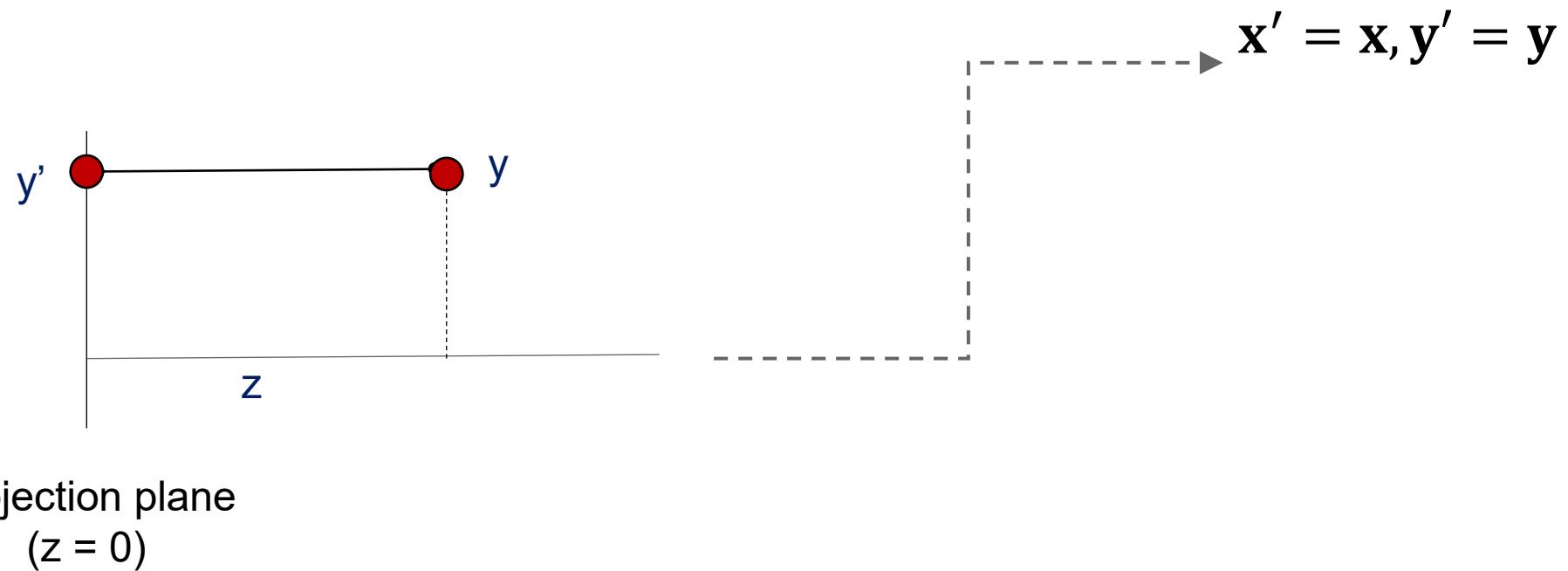
[songho.ca](http://songho.ca)

# View frustum

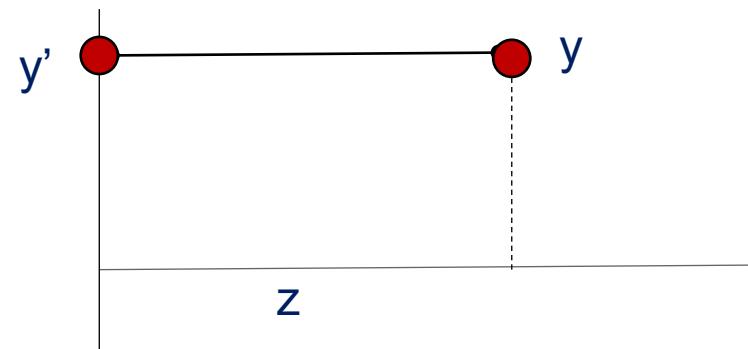


# Basic orthographic projection

---



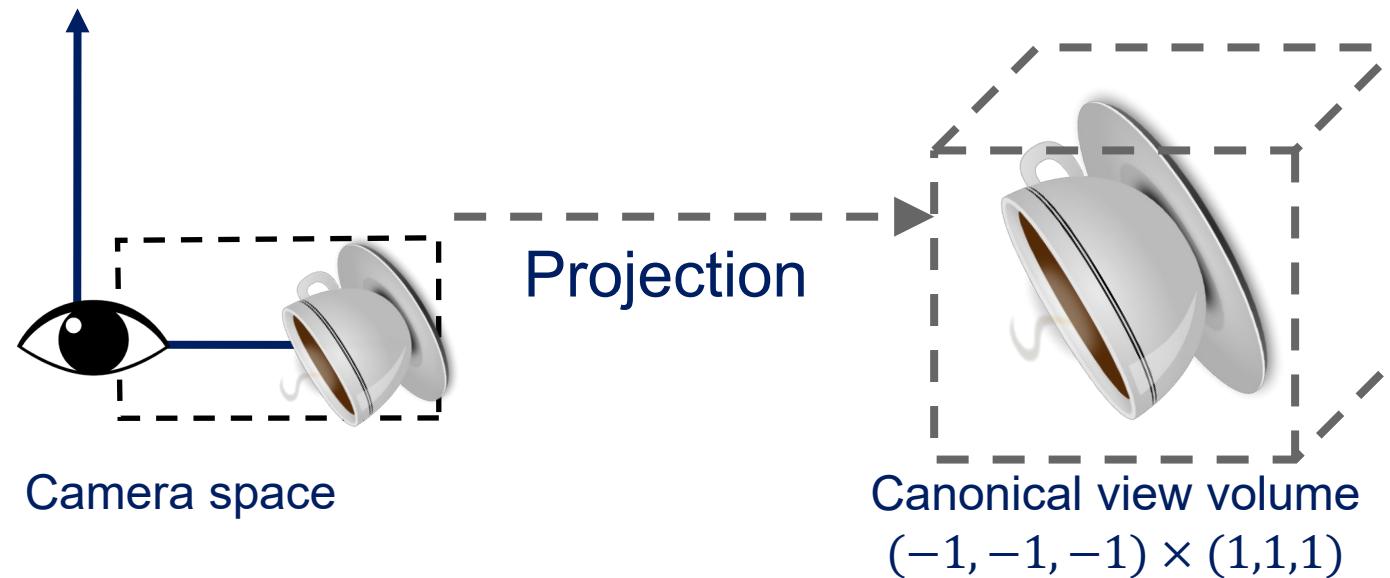
# Basic orthographic projection



Projection plane  
( $z = 0$ )

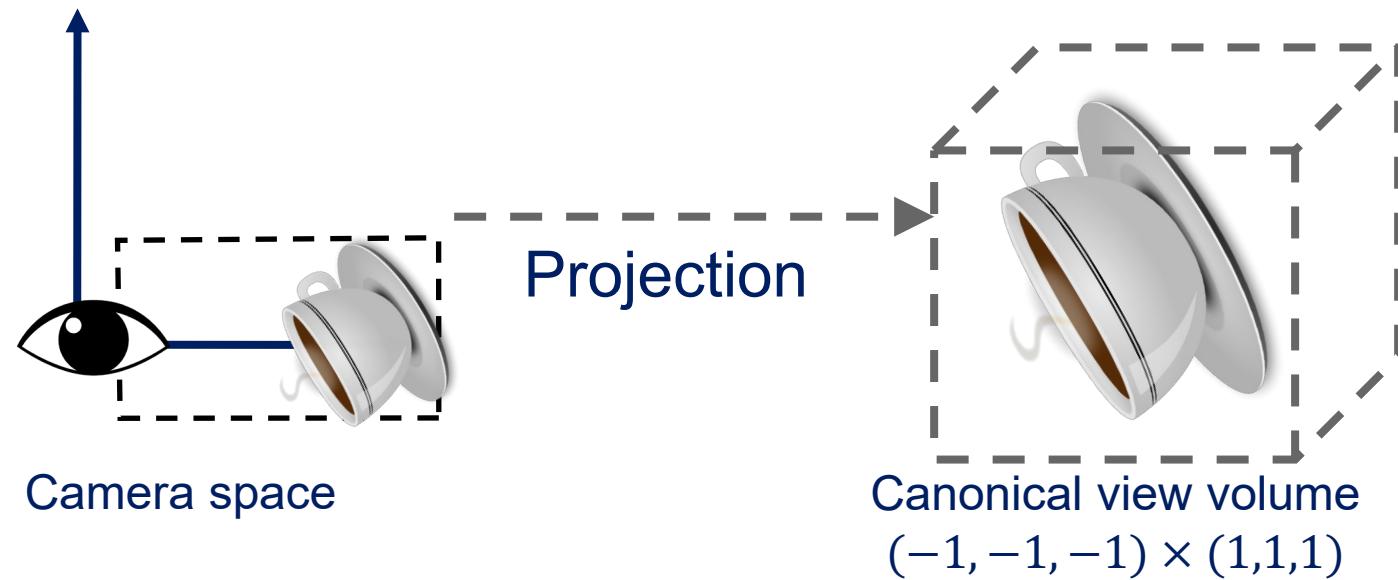
$$\begin{array}{c} \xrightarrow{\quad \text{---} \quad} \\ \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \end{array}$$
$$x' = x, y' = y$$

# Orthographic transformation



1. Translate to the origin.
2. Scale the volume to a 2-by-2 unit square:  $(-1, -1)$  to  $(+1, +1)$ .
3. Switch coordinate system.

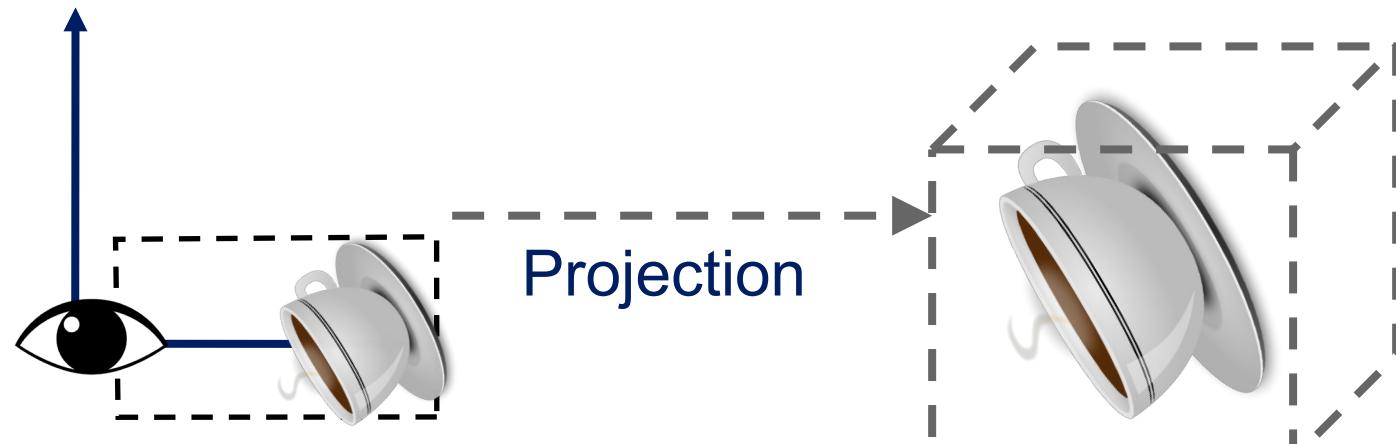
# Orthographic transformation



$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} =$$

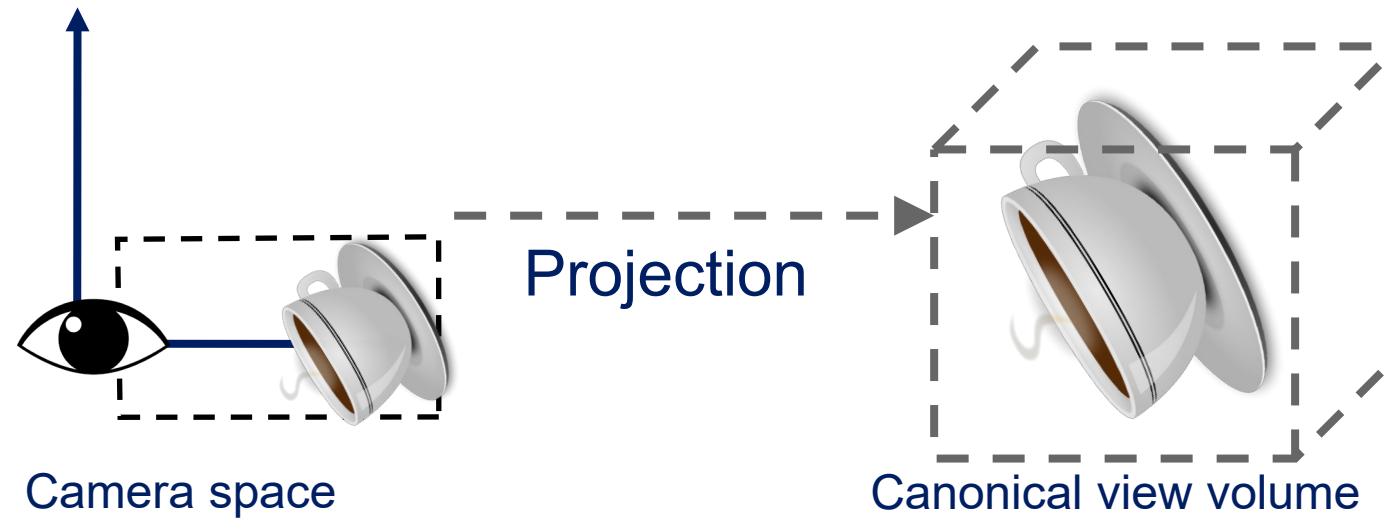
$$\begin{pmatrix} 1 & 0 & 0 & -x_{mid} \\ 0 & 1 & 0 & -y_{mid} \\ 0 & 0 & 1 & -z_{mid} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

# Orthographic transformation



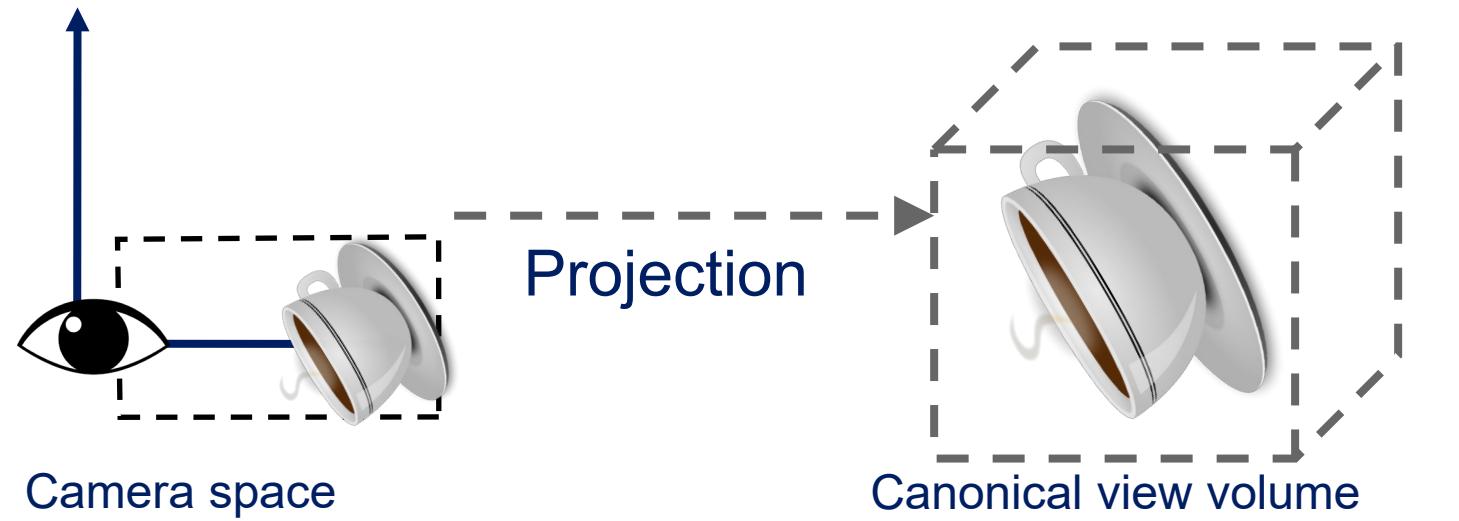
$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{2}{(right - left)} & 0 & 0 & 0 \\ 0 & \frac{2}{(top - bottom)} & 0 & 0 \\ 0 & 0 & \frac{2}{(far - near)} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -x_{mid} \\ 0 & 1 & 0 & -y_{mid} \\ 0 & 0 & 1 & -z_{mid} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

# Orthographic transformation



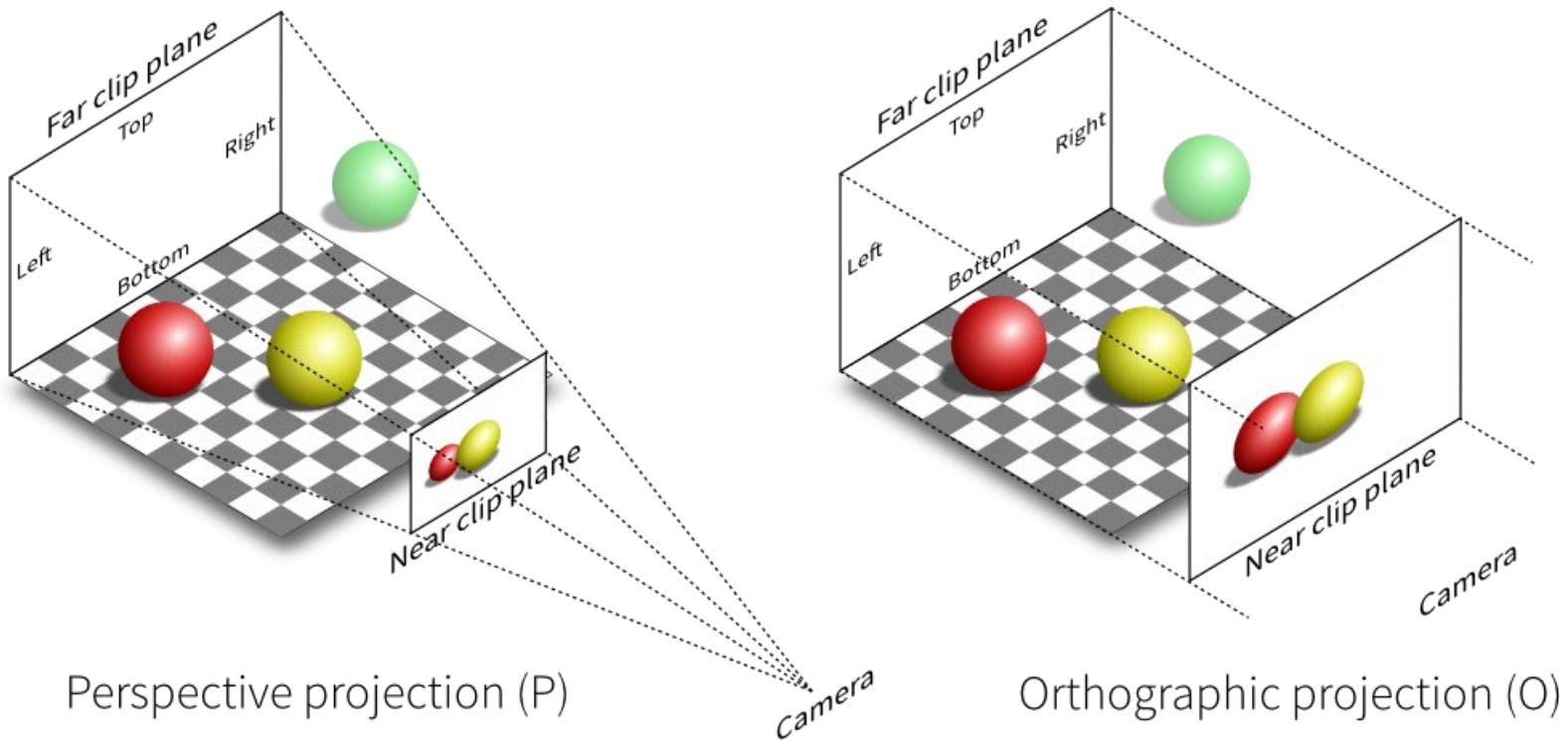
$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{2}{(right - left)} & 0 & 0 & 0 \\ 0 & \frac{2}{(top - bottom)} & 0 & 0 \\ 0 & 0 & \frac{2}{(far - near)} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -x_{mid} \\ 0 & 1 & 0 & -y_{mid} \\ 0 & 0 & 1 & -z_{mid} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

# Orthographic transformation



$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{2}{(right - left)} & 0 & 0 & \frac{-(right + left)}{(right - left)} \\ 0 & \frac{2}{(top - bottom)} & 0 & \frac{-(top + bottom)}{(top - bottom)} \\ 0 & 0 & \frac{-2}{(far - near)} & \frac{-(far + near)}{(far - near)} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

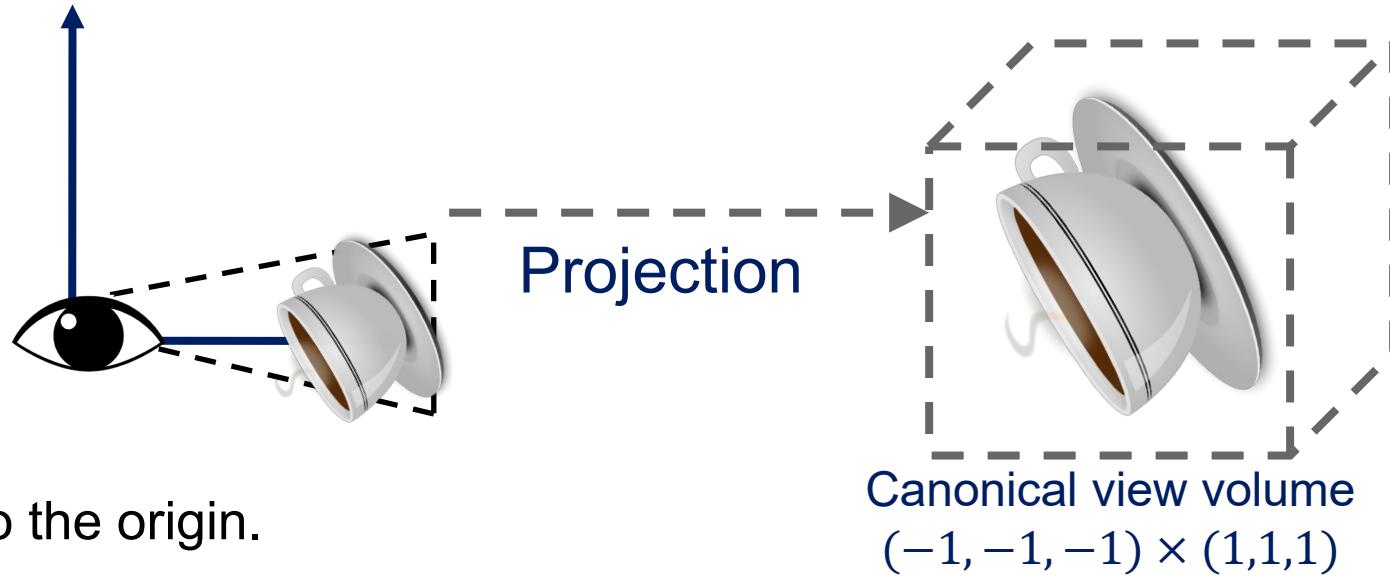
# View frustum



Perspective projection (P)

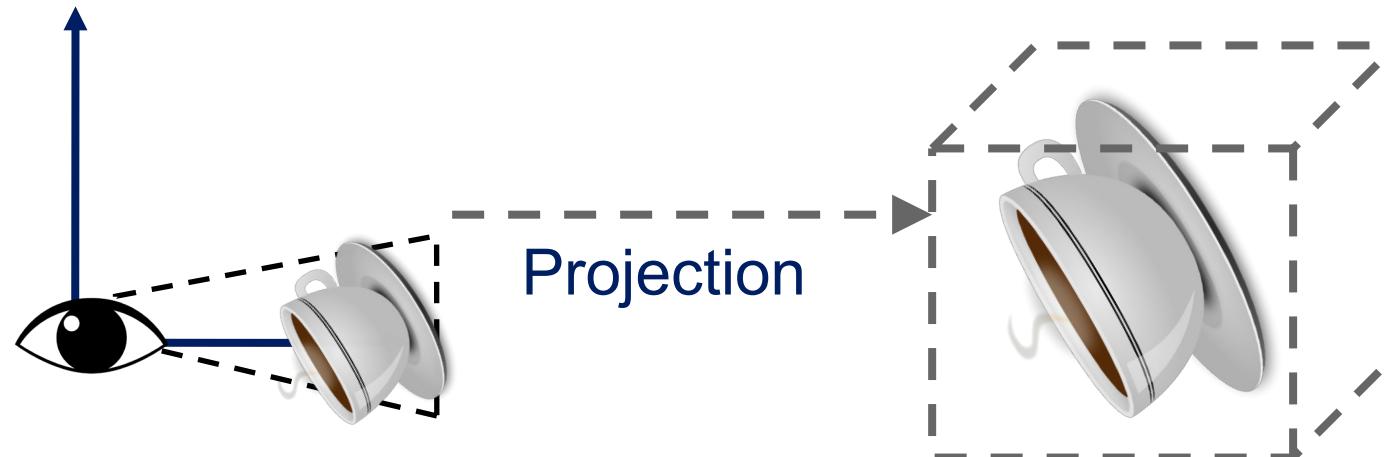
Orthographic projection (O)

# Perspective projection



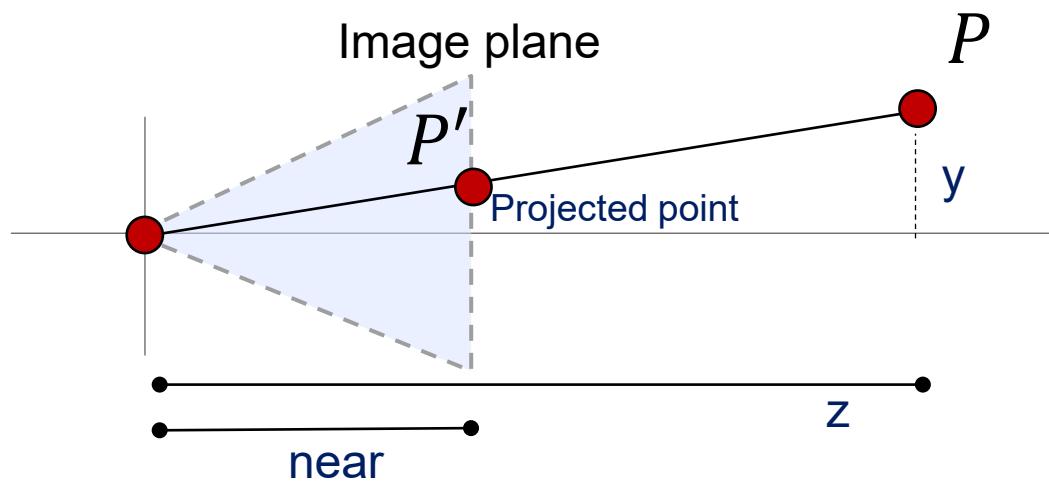
1. Translate to the origin.
2. Rotate to look along the negative z-axis.
3. Scale the volume to a 2-by-2 unit square:  $(-1, -1)$  to  $(+1, +1)$ .
4. Scale the volume to a 2-by-2 unit square:  $(-1, -1)$  to  $(+1, +1)$ .
5. Switch coordinate system.

# Perspective projection

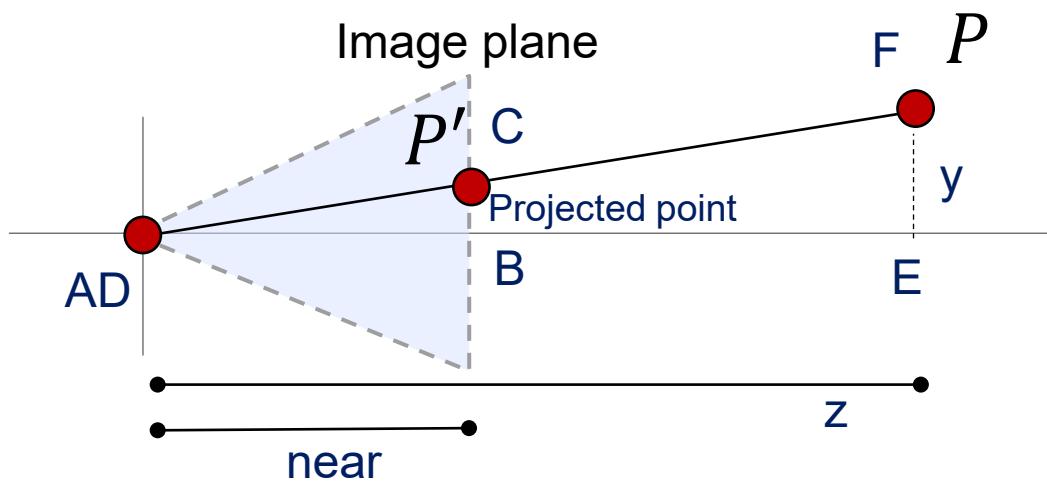


1. Translate to the origin.
2. Scale depth values into normalized range:  $(-1, +1)$ .
3. Perspective calculation.
4. Scale the volume to a 2-by-2 unit square:  $(-1, -1)$  to  $(+1, +1)$ .
5. Switch coordinate system.

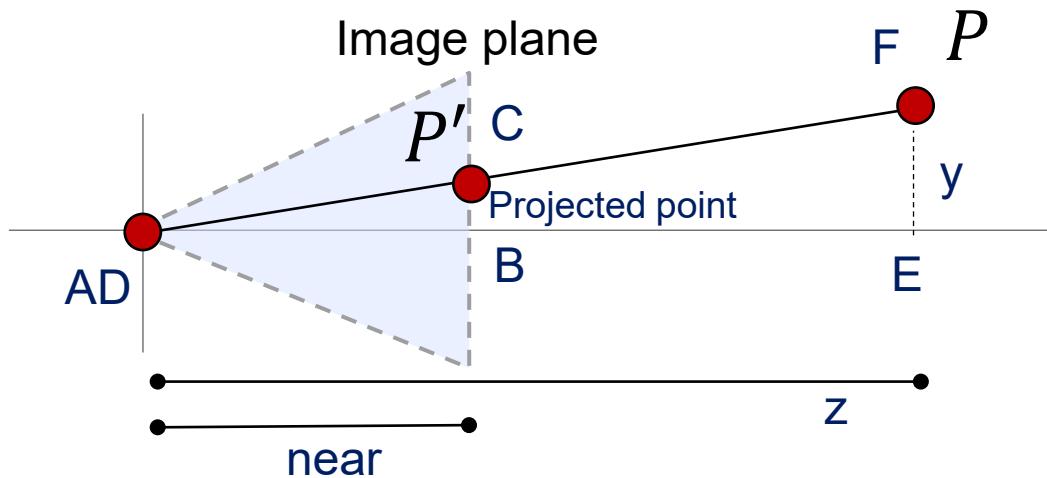
# Perspective calculation



# Perspective calculation



# Perspective calculation



Project point *P'*?

$$\frac{AB}{DE} = \frac{BC}{EF}$$

$$\frac{\text{near}}{-P_z} = \frac{P'_y}{P_y}$$

$$P'_y = \frac{\text{near} * P_y}{-P_z}$$

$$P'_x = \frac{\text{near} * P_x}{-P_z}$$

What is the matrix transformation?

# Perspective calculation

---

- What is the matrix transformation?
- 4-by-4 transformation matrix is a linear combination of terms:
  - We can calculate  $a * x + b * y + c * z + d$
  - But not  $\frac{a*x}{z} + \dots$

# Perspective calculation

---

- What is the matrix transformation?
- 4-by-4 transformation matrix is a linear combination of terms:
  - We can calculate  $a * x + b * y + c * z + d$
  - But not  $\frac{a*x}{z} + \dots$
- Solution: homogeneous coordinates.

$$(x, y, z, w) \rightarrow \left( \frac{x}{w}, \frac{y}{w}, \frac{z}{w}, 1 \right)$$

# Perspective calculation

- What is the matrix transformation?
- 4-by-4 transformation matrix is a linear combination of terms:
  - We can calculate  $a * x + b * y + c * z + d$
  - But not  $\frac{a*x}{z} + \dots$
- Solution: homogeneous coordinates.

$$(x, y, z, w) \rightarrow \left( \frac{x}{w}, \frac{y}{w}, \frac{z}{w}, 1 \right)$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} near & 0 & 0 & 0 \\ 0 & near & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

# Perspective calculation

---

- What is the matrix transformation?
- 4-by-4 transformation matrix is a linear combination of terms:
  - We can calculate  $a * x + b * y + c * z + d$
  - But not  $\frac{a*x}{z} + \dots$
- Solution: homogeneous coordinates.

$$(x, y, z, w) \rightarrow \left( \frac{x}{w}, \frac{y}{w}, \frac{z}{w}, 1 \right)$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} near & 0 & 0 & 0 \\ 0 & near & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

# Perspective calculation

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} near & 0 & 0 & 0 \\ 0 & near & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} x * near \\ y * near \\ z \\ -z \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{x * near}{-z} \\ \frac{y * near}{-z} \\ \frac{-z}{-z} \\ \frac{-z}{-z} \end{pmatrix}$$

Graphics pipeline expects perspective division in the w component

Exactly what we wanted to get  
(slide 33):

$$P'_y = \frac{near * P_y}{-P_z}$$

$$P'_x = \frac{near * P_x}{-P_z}$$

# Scaling depth values

---

- Scaling depth values into a normalized range.
- Objective: mapping between (-near,-far) to (-1,+1).
- Naïve solution:
$$\begin{aligned}-1 &= -\text{near} \\ +1 &= -\text{far}\end{aligned}$$
- However, float point numbers suffer from round-off errors (difference between 0.1234567 $\mathbf{8}$  and 0.1234567 $\mathbf{7}$  can have a visual impact.

# Scaling depth values

---

- Scaling depth values into a normalized range.
- Objective: **non-linear** mapping between (-near,-far) to (-1,+1).
- More precision for values close to the camera, less precision for values farther from the camera.
- Non-linear mapping with two constants:

$$\frac{c_1}{-z} + c_2$$

# Scaling depth values

---

- Scaling depth values into a normalized range.
- Non-linear mapping with two constants:

$$\frac{c_1}{-z} + c_2$$

- When  $z = -near$ , mapping should return  $-1$
- When  $z = -far$ , mapping should return  $+1$

$$-1 = \frac{c_1}{-(-near)} + c_2 \text{ and } +1 = \frac{c_1}{-(-far)} + c_2$$

# Scaling depth values

---

- Scaling depth values into a normalized range.
- Non-linear mapping with two constants:

$$\frac{c_1}{-z} + c_2$$

- When  $z = -near$ , mapping should return  $-1$
- When  $z = -far$ , mapping should return  $+1$

$$-1 = \frac{c_1}{-(-near)} + c_2 \text{ and } +1 = \frac{c_1}{-(-far)} + c_2$$

$$c_1 = 2 * far * \frac{near}{near - far}$$

$$c_2 = \frac{far + near}{far - near}$$

# Scaling depth values

- What is the matrix transformation?
- Same problem as before: 4-by-4 transformation matrix is a linear combination of terms:
  - We can calculate  $a * x + b * y + c * z + d$
  - But not  $\frac{a*x}{z} + \dots$
- Solution: homogeneous coordinates (again).

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -c_2 & c_1 \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

# Perspective projection

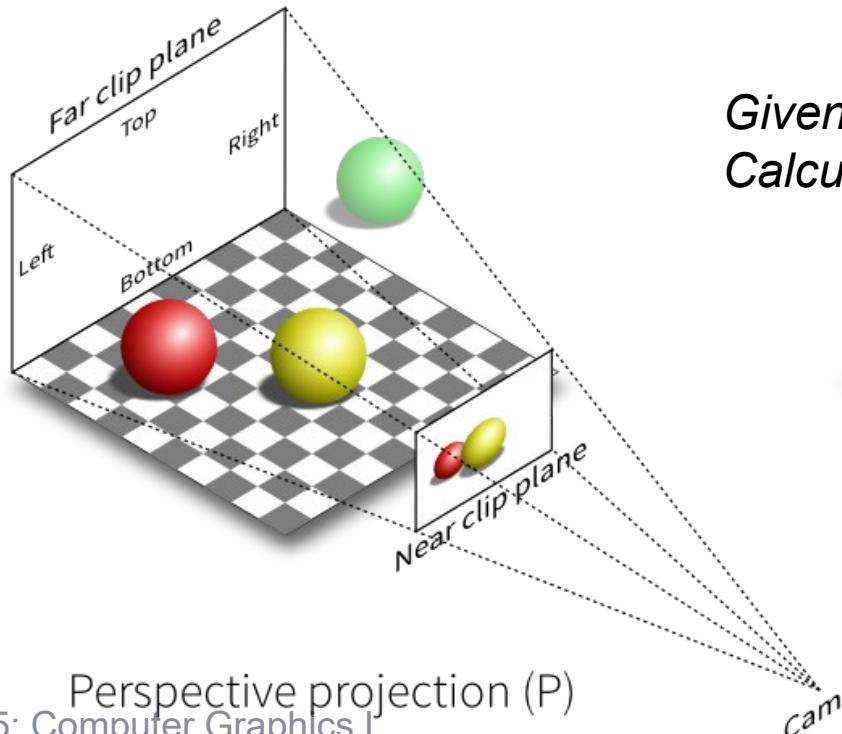
1. Translate to the origin.
2. Scale depth values into normalized range: (-1, +1). (and switch coordinate system)
3. Perspective calculation.
4. Scale the volume to a 2-by-2 unit square: (-1,-1) to (+1, +1).

Note: you only need to perform perspective calculation once.

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{2}{(right - left)} & 0 & 0 & 0 \\ 0 & \frac{2}{(top - bottom)} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} near & 0 & 0 & 0 \\ 0 & near & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -c_2 & c_1 \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -x_{mid} \\ 0 & 1 & 0 & -y_{mid} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

# Perspective projection

- How to create a perspective transformation matrix given a view of view (FOV) and aspect ratio width:height?



*Given: fov, aspect, near, far  
Calculate frustum properties (left, right, bottom, top, near, far):*

$$top = near * \tan\left(\frac{fov}{2}\right)$$

$$bottom = -top$$

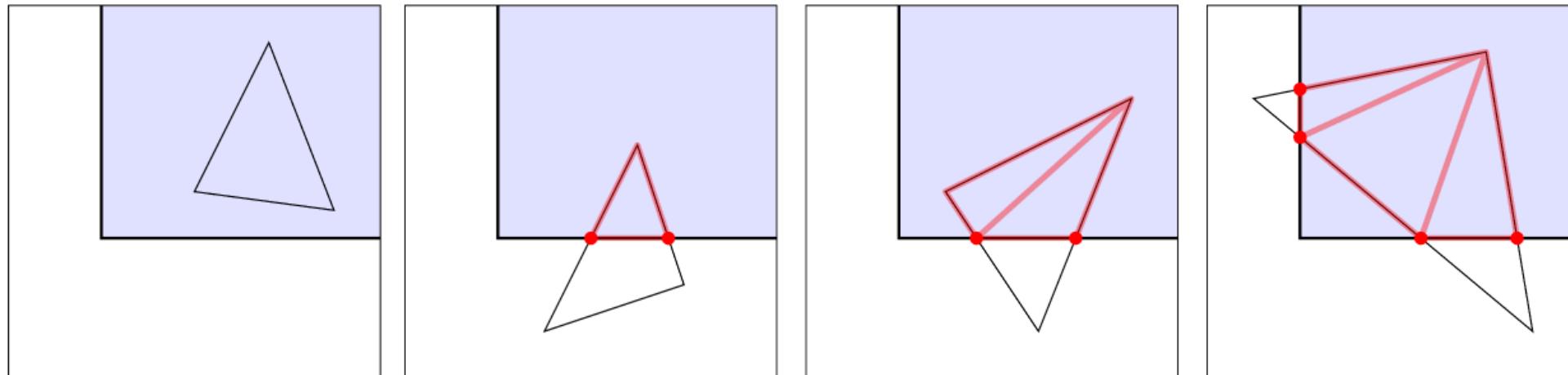
$$right = top * aspect$$

$$left = -right$$

# Canonical view volume

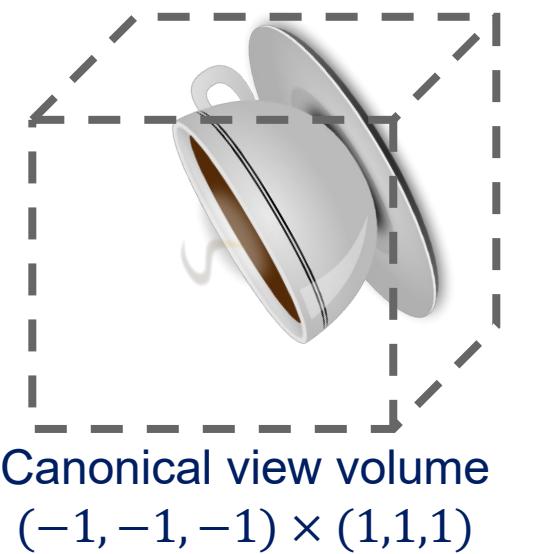
---

- Why?
  - Makes clipping much easier! GL can quickly discard geometry outside  $-1,1$ .

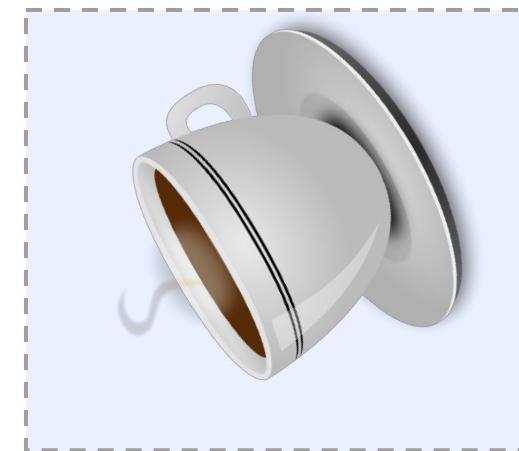


From: <https://paroj.github.io/gltut/>

# Viewport transformation



Viewport



$$\begin{pmatrix} x_{screen} \\ y_{screen} \\ z_{depth} \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{width}{2} & 0 & 0 & \frac{width}{2} \\ 0 & \frac{height}{2} & 0 & \frac{height}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

`gl.viewport(0, 0, width, height);`



COMPUTER SCIENCE

55

# Lab

---

- Draw a cube on screen with size  $(-1, -1, -1) \times (1, 1, 1)$ .
  - Apply a model transformation that scales it by 0.5, and tilts it slightly.
  - Apply a projection matrix (orthographic and perspective).