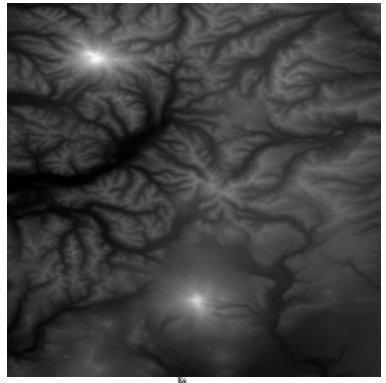
Viewing Transformations

CS425: Computer Graphics I

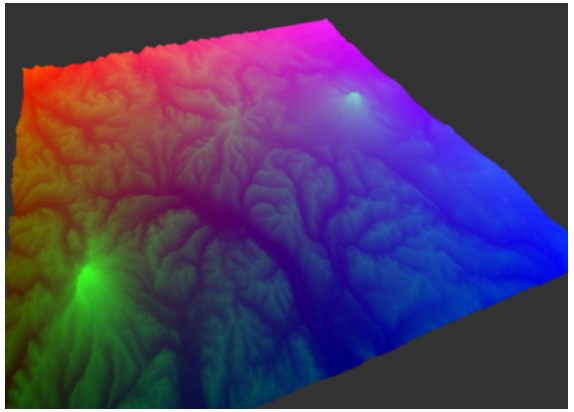
Khairi Reda

Assignment 2: out today

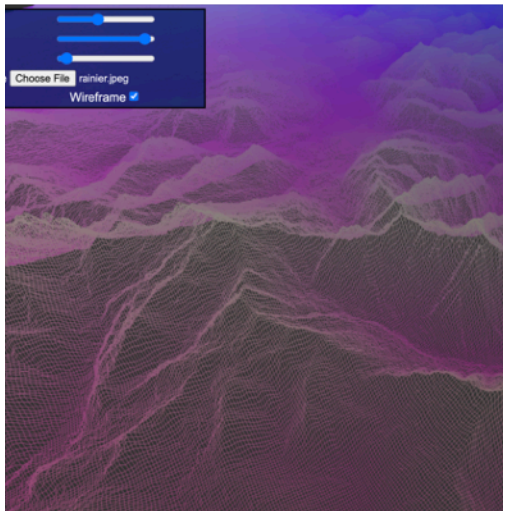
- Due **Wednesday October 8th (7:00pm)**
- Learning goals:
  - Triangle meshes from height maps
  - Viewing and projection transformations
  - No lighting yet—coming up next assignment



Source data:
Greyscale height field



Triangle mesh
RGB color
from XYZ



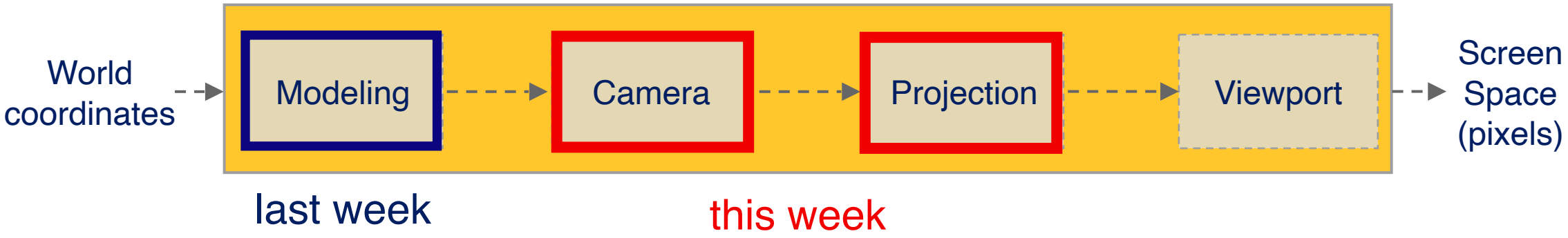
Transforms to
view/navigate

Overview

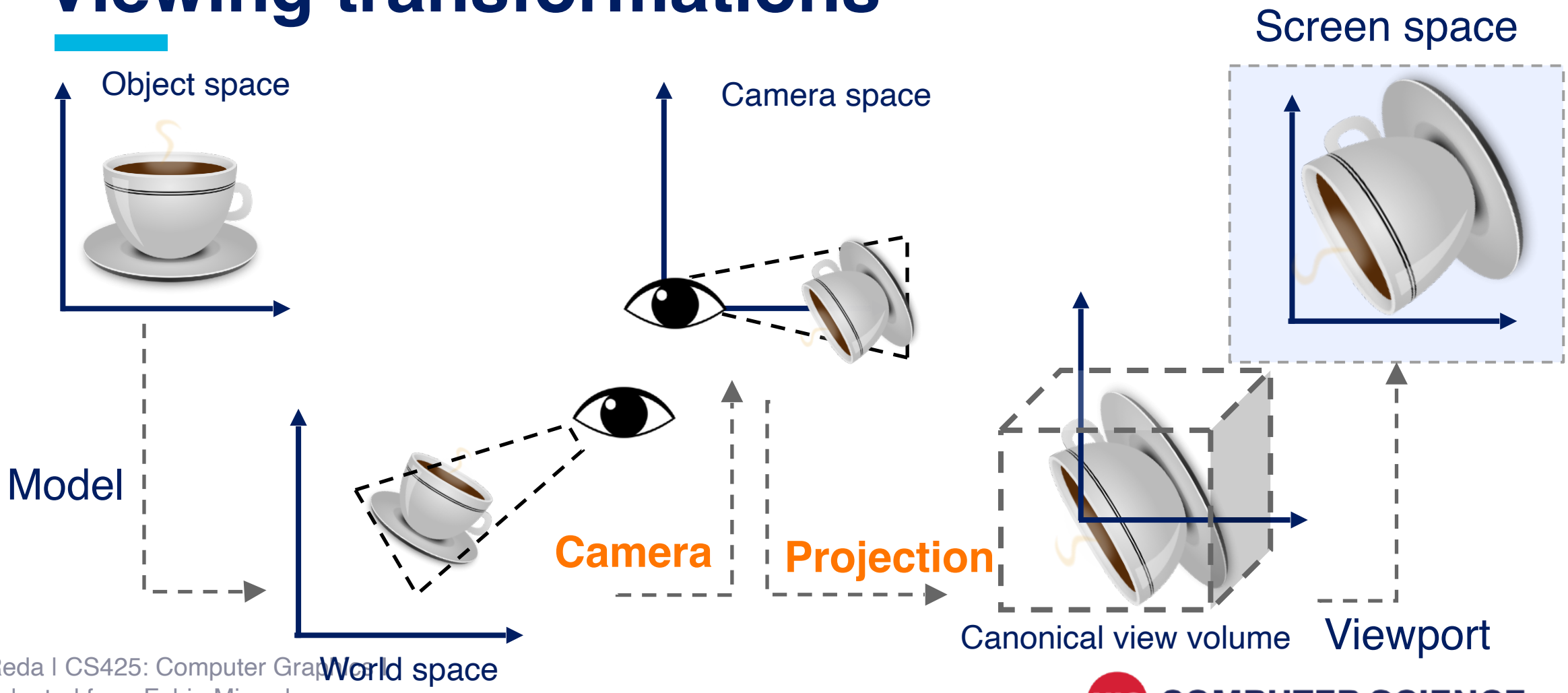
- Camera
- Viewing transformations
- Orthographic and perspective projections
- Hidden surface removal

Viewing transformations

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



Viewing transformations



Perspective projection

Distant objects appear smaller

Parallel lines converge at the horizon

Early paintings

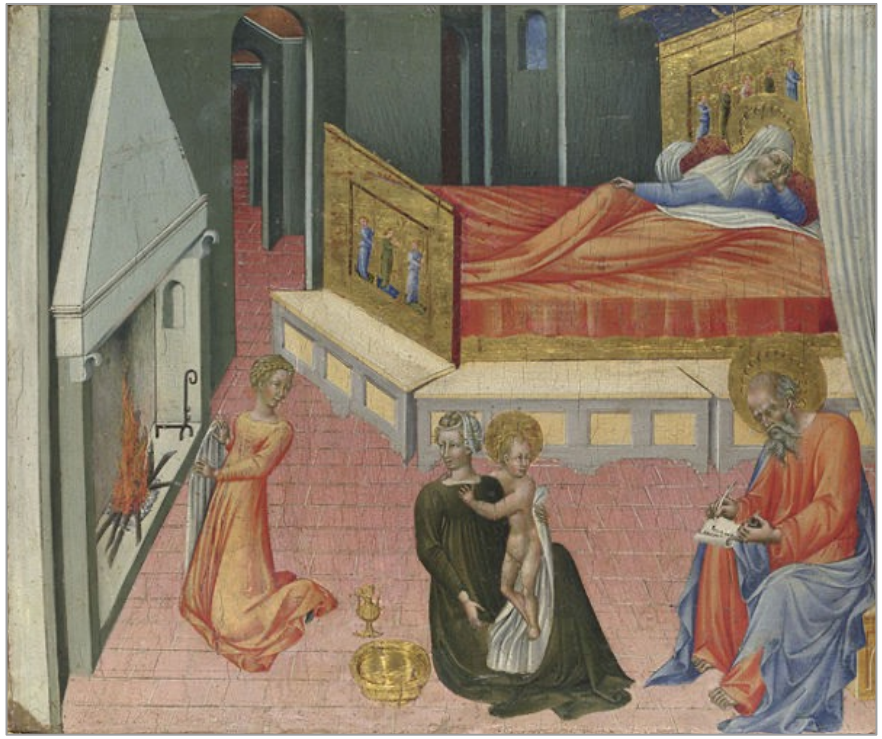
Lorsch Gospels (8th century)

Perspective in art

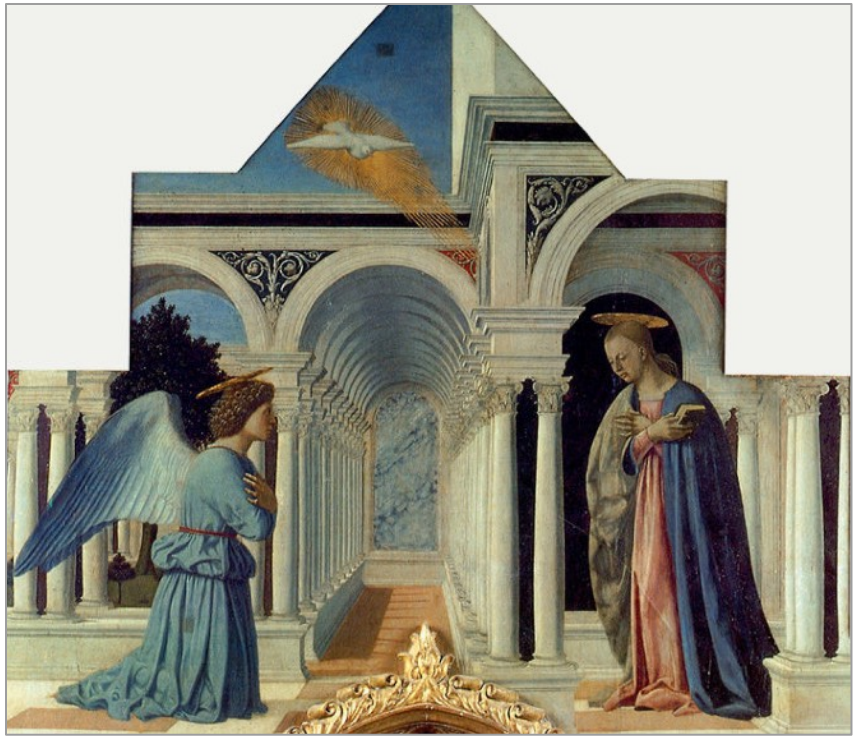
Giotto di Bondone (14th century)

Giotto di Bondone (14th century)

Birth of perspective in art: One-point perspective



Giovanni di Paolo (15th century)



Piero della Francesca (15th century)

Birth of perspective in art: One-point perspective

Perugino (15th century)

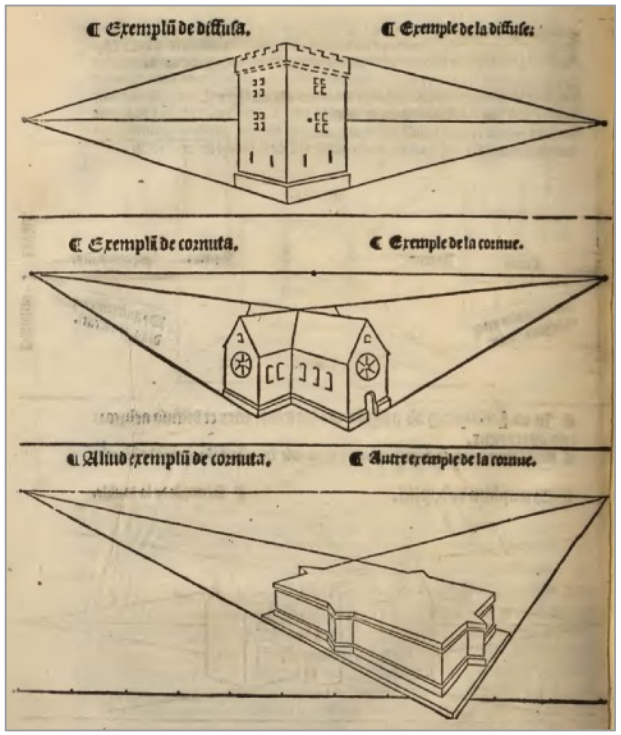
Birth of perspective in art: One-point perspective

The Ideal City (15th century)

Birth of perspective in art: One-point perspective

Rafael (16th century)

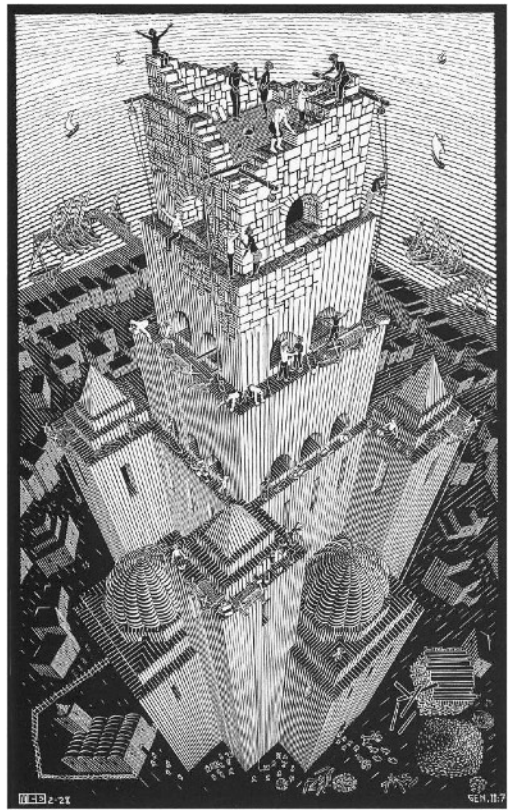
Birth of perspective in art: Two-point perspective



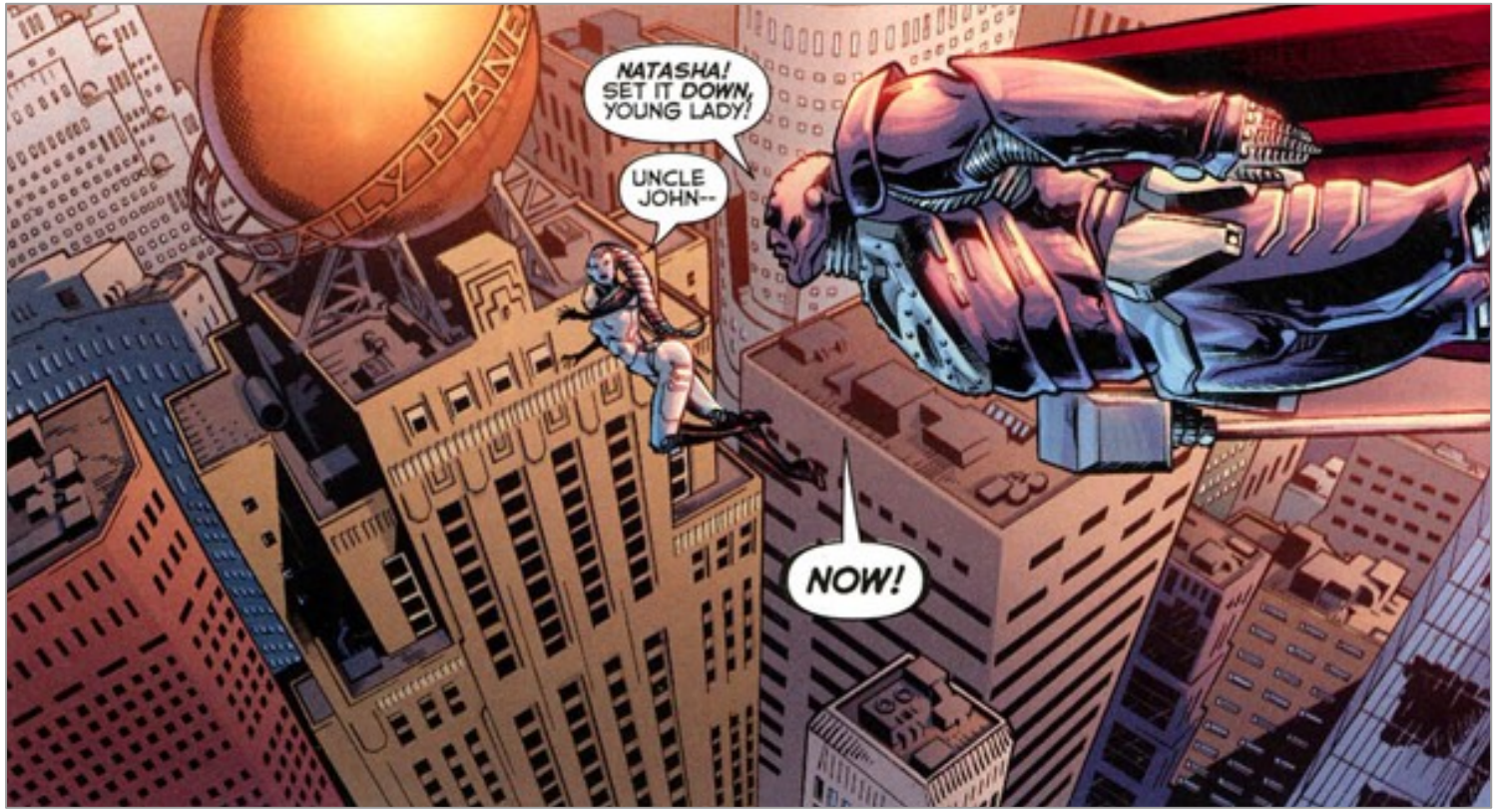
Jean Pélérin (16th century)

Gustave Caillebotte (19th century)

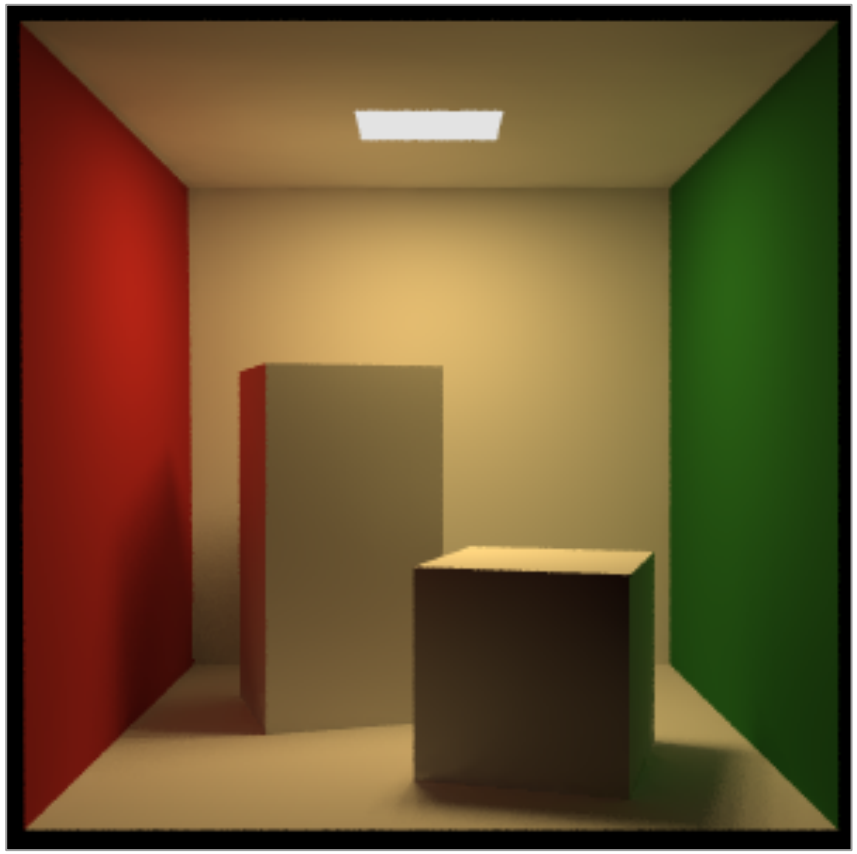
Birth of perspective in art: Three-point perspective

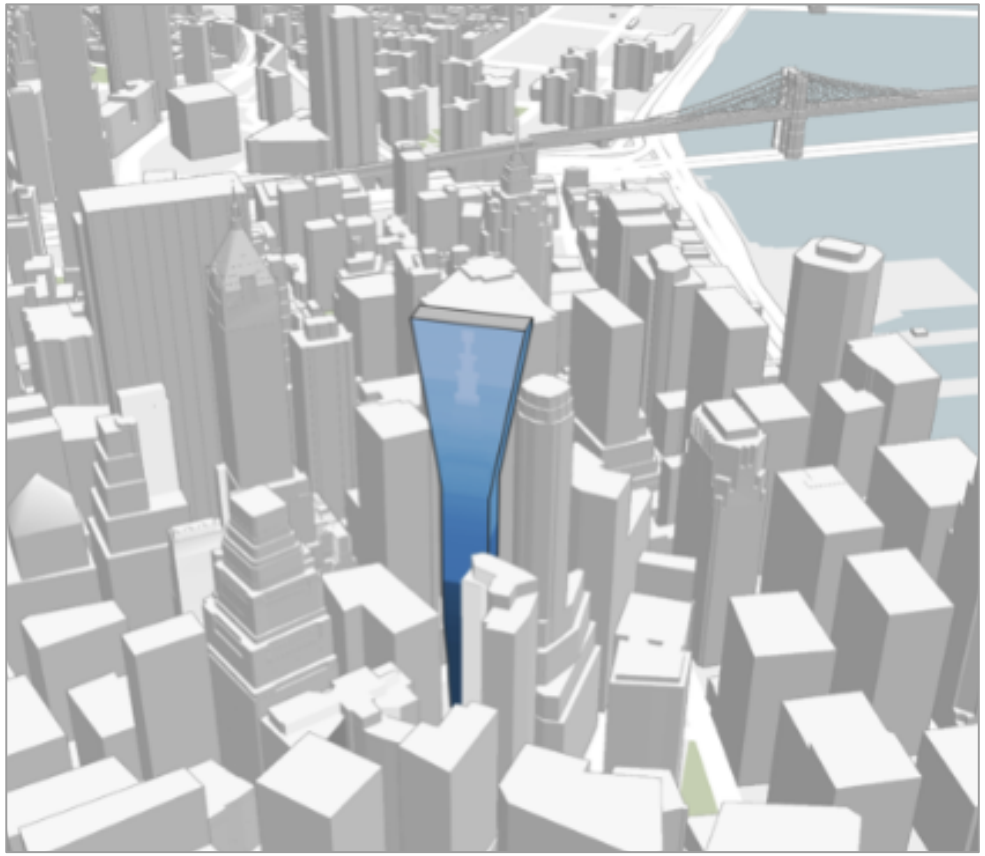


M.C. Escher (1928)



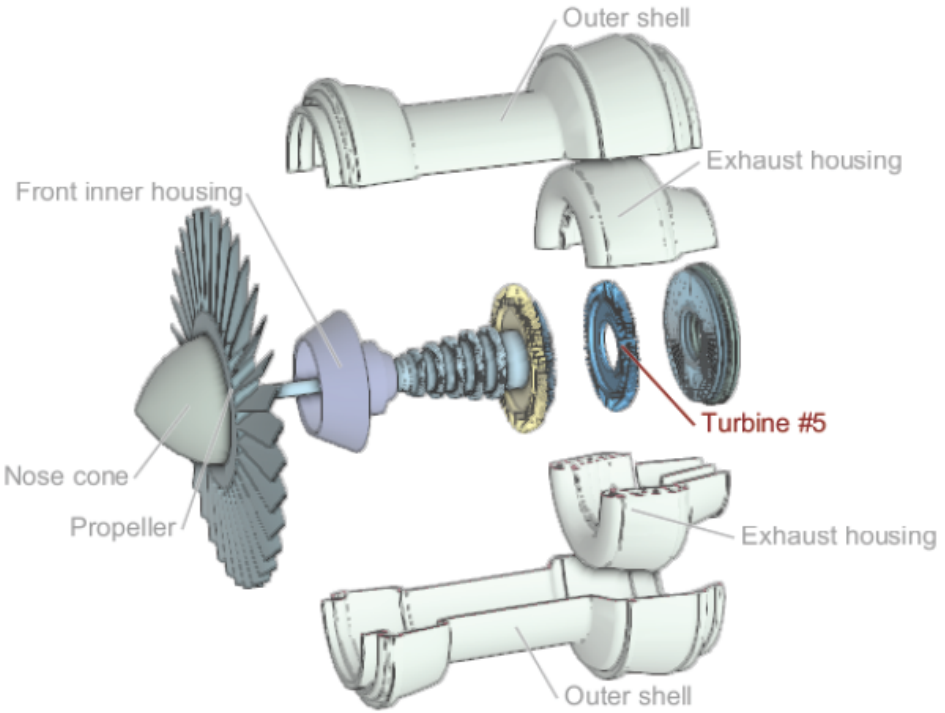
Perspective in computer graphics





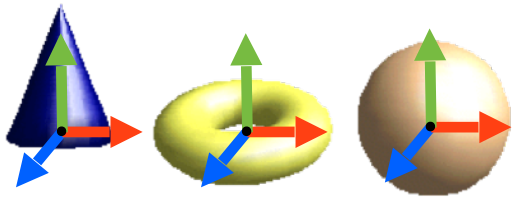
Rejection of perspective in CG

SimCity 2000

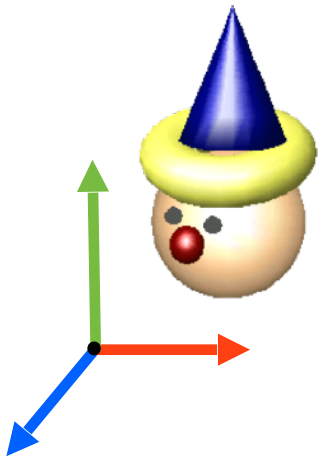


"Automated Generation of Interactive 3D Exploded Views"

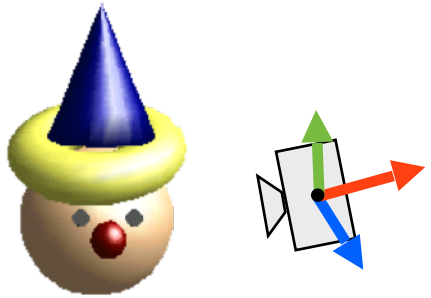
Coordinate spaces



Object coordinates



World coordinates

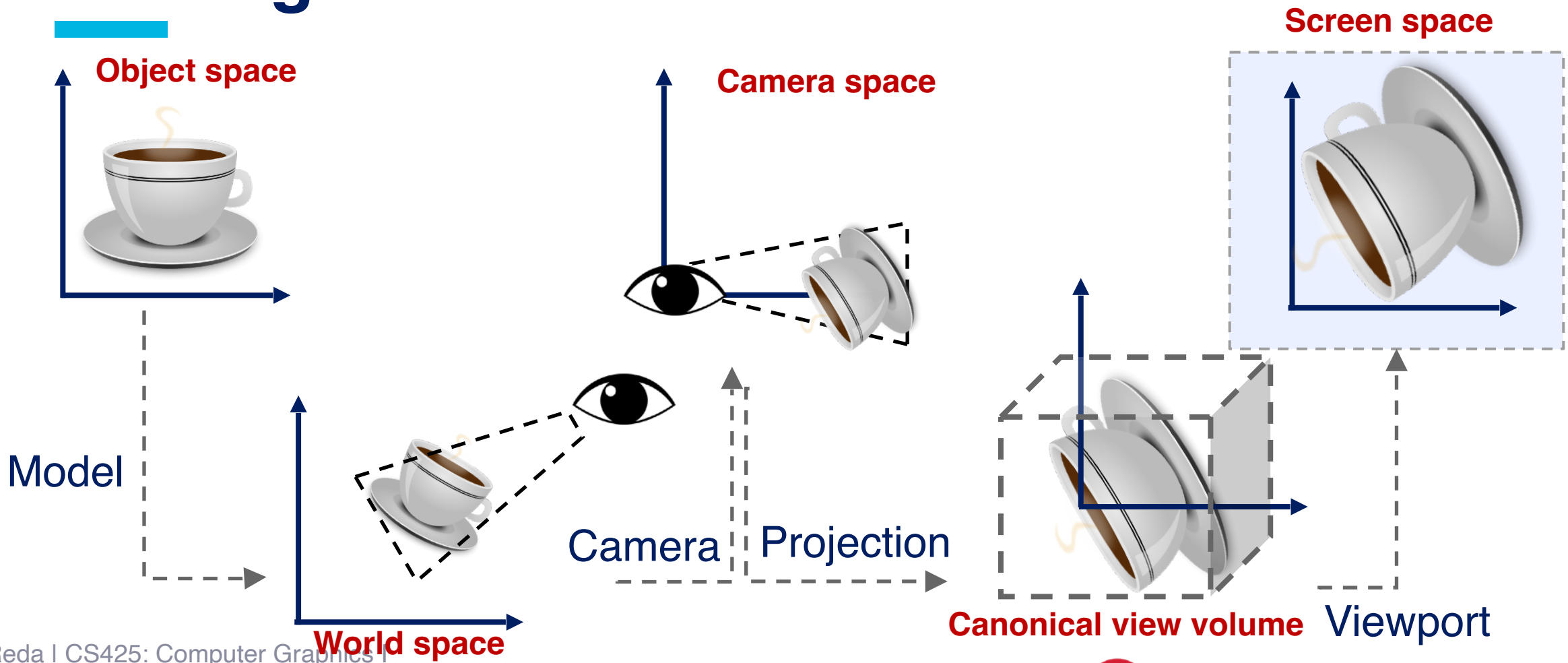


Camera coordinates

Screen coordinates

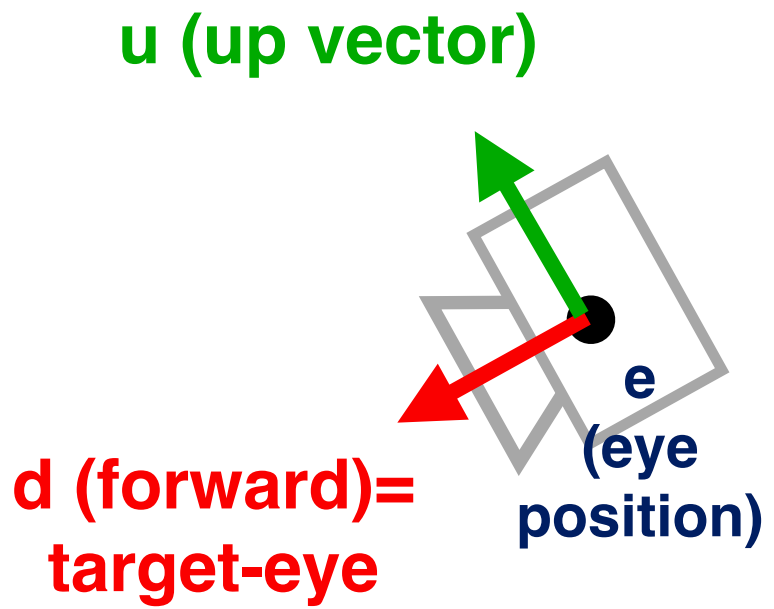
From: Mark Pauly

Viewing transformation

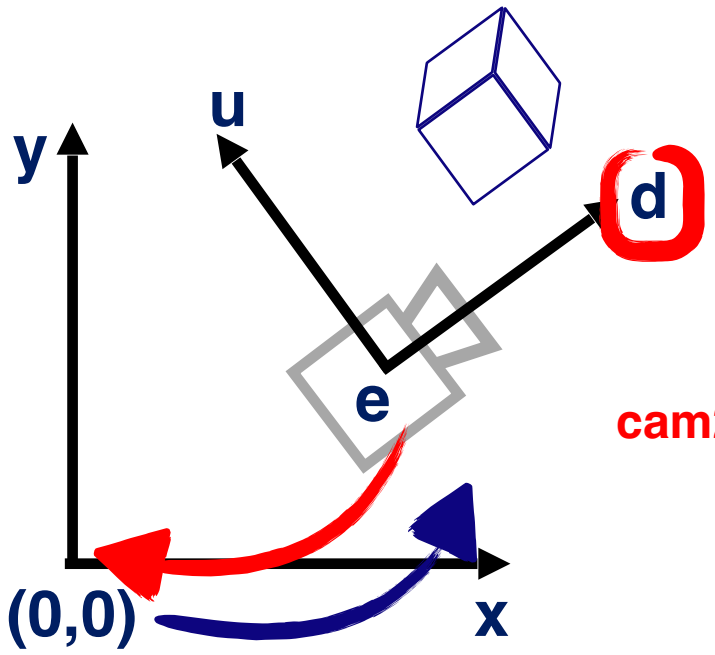


Camera transformation

- Construct the camera reference system:
  - The eye position $\mathbf{e}$.
  - The forward direction $\mathbf{d}$.
  - The view-up vector $\mathbf{u}$.
- We need a matrix that transforms all coordinates into camera coordinates.



Change of frame

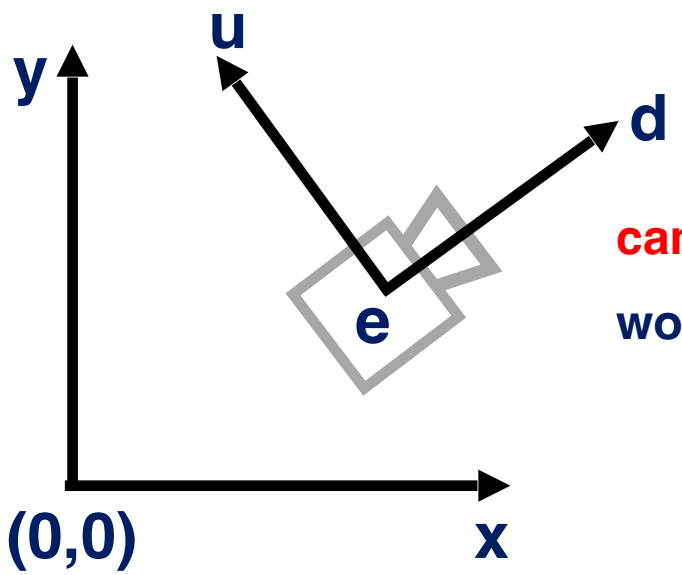


Rotation: $M = \begin{pmatrix} d_x & u_x & 0 \\ d_y & u_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Translate: $\mathbf{T} = \begin{pmatrix} 1 & 0 & e_x \\ 0 & 1 & e_y \\ 0 & 0 & 1 \end{pmatrix}$

cam2world $\mathbf{M}_{e \to o} = \mathbf{TR}$

Change of frame



Rotation: $M = \begin{pmatrix} d_x & u_x & 0 \\ d_y & u_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Translate: $\mathbf{T} = \begin{pmatrix} 1 & 0 & e_x \\ 0 & 1 & e_y \\ 0 & 0 & 1 \end{pmatrix}$

cam2world $\mathbf{M}_{e \to o} = \mathbf{TR}$

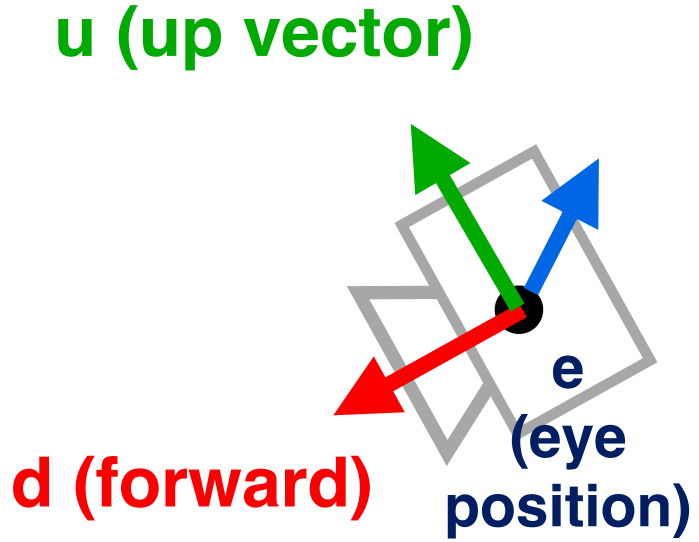
world2cam $\mathbf{M}_{o \to e} = (\mathbf{TR})^{-1} = \mathbf{R}^{-1}\mathbf{T}^{-1}$

We know:

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

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

Camera transformation (3D)



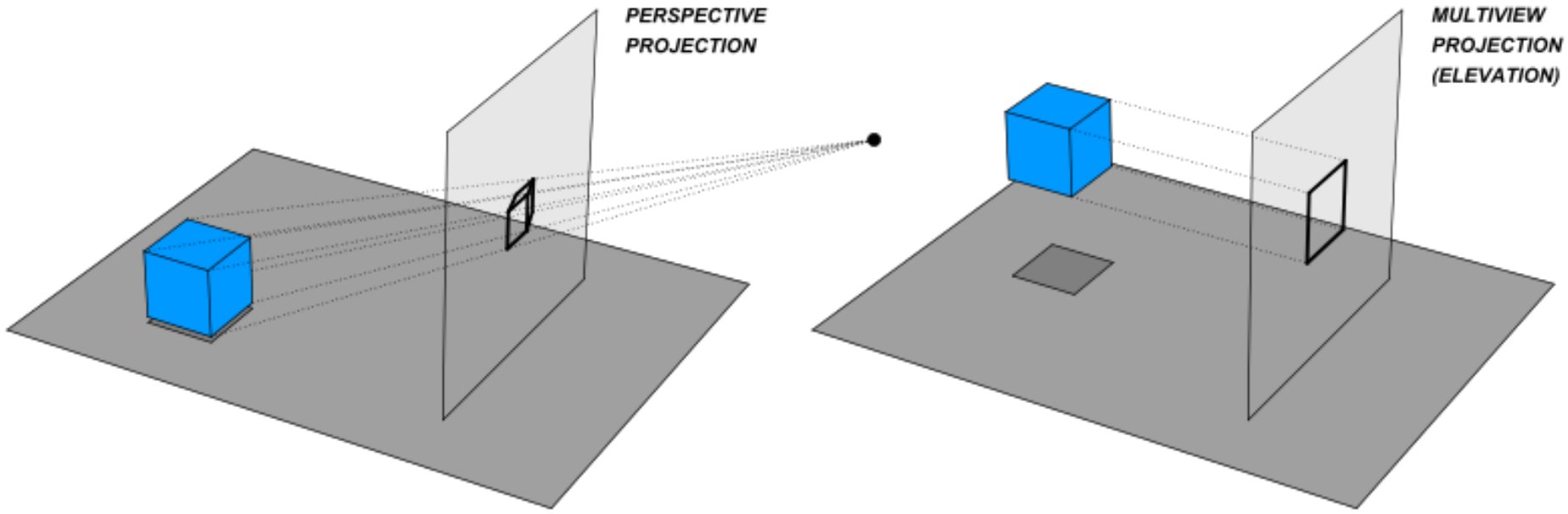
- Construct the camera reference system:
  - The eye position $\mathbf{e}$.
  - The forward direction $\mathbf{d}$.
  - The view-up vector $\mathbf{u}$.
  - Cross product of $\mathbf{u}$ and $\mathbf{d}$ to get the 3rd orthogonal vector
- A view matrix transform all coordinates into view coordinates.

$$\mathbf{M}_{camera2world} = \begin{pmatrix} \mathbf{u}\times\mathbf{d} & \mathbf{u} & \mathbf{d} & \mathbf{e} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$ (4x4 matrix)

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

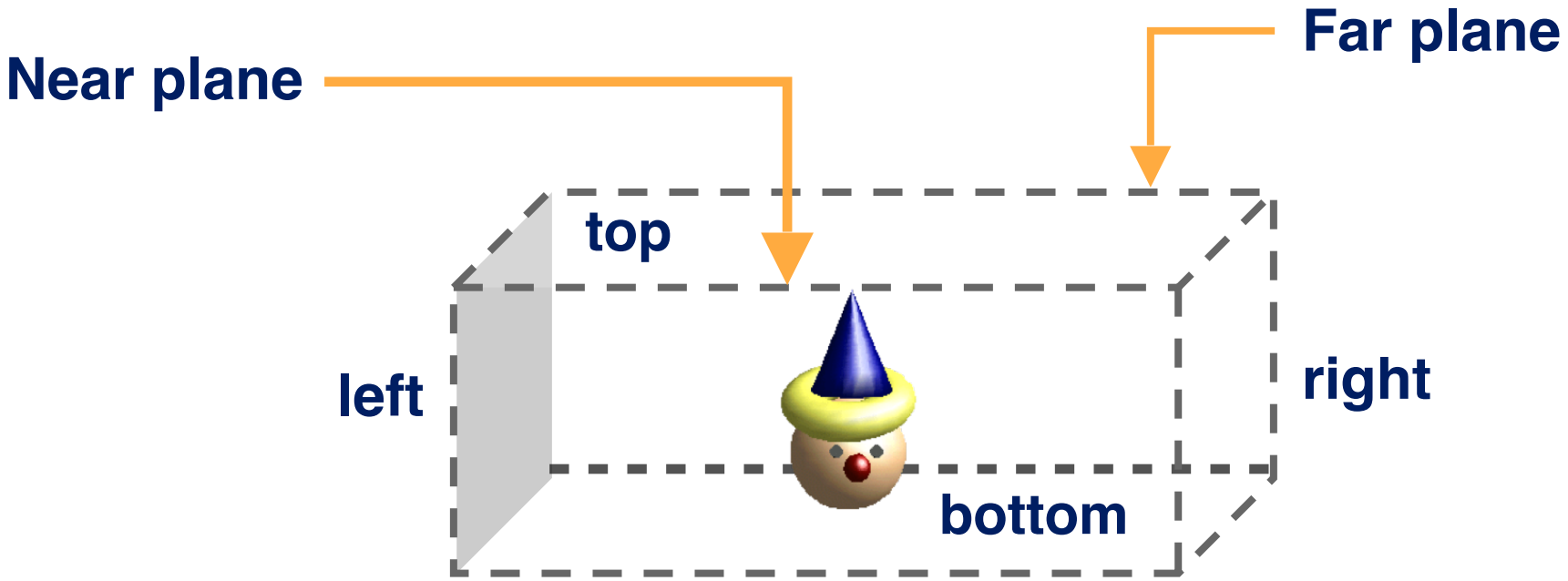
Projection

Orthographic and perspective projections



https://en.wikipedia.org/wiki/File:Various_projections_of_cube_above_plane.svg

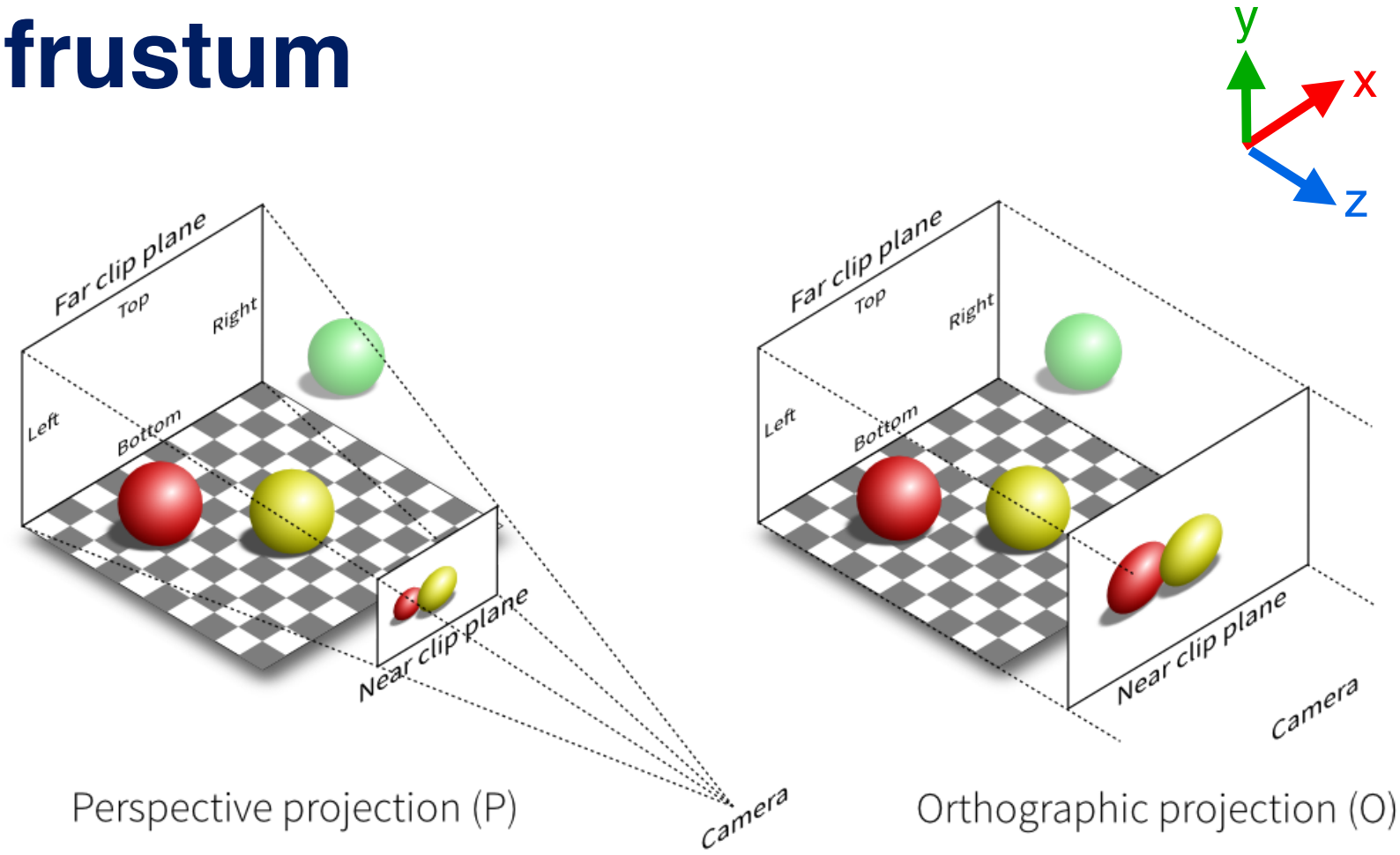
View frustum



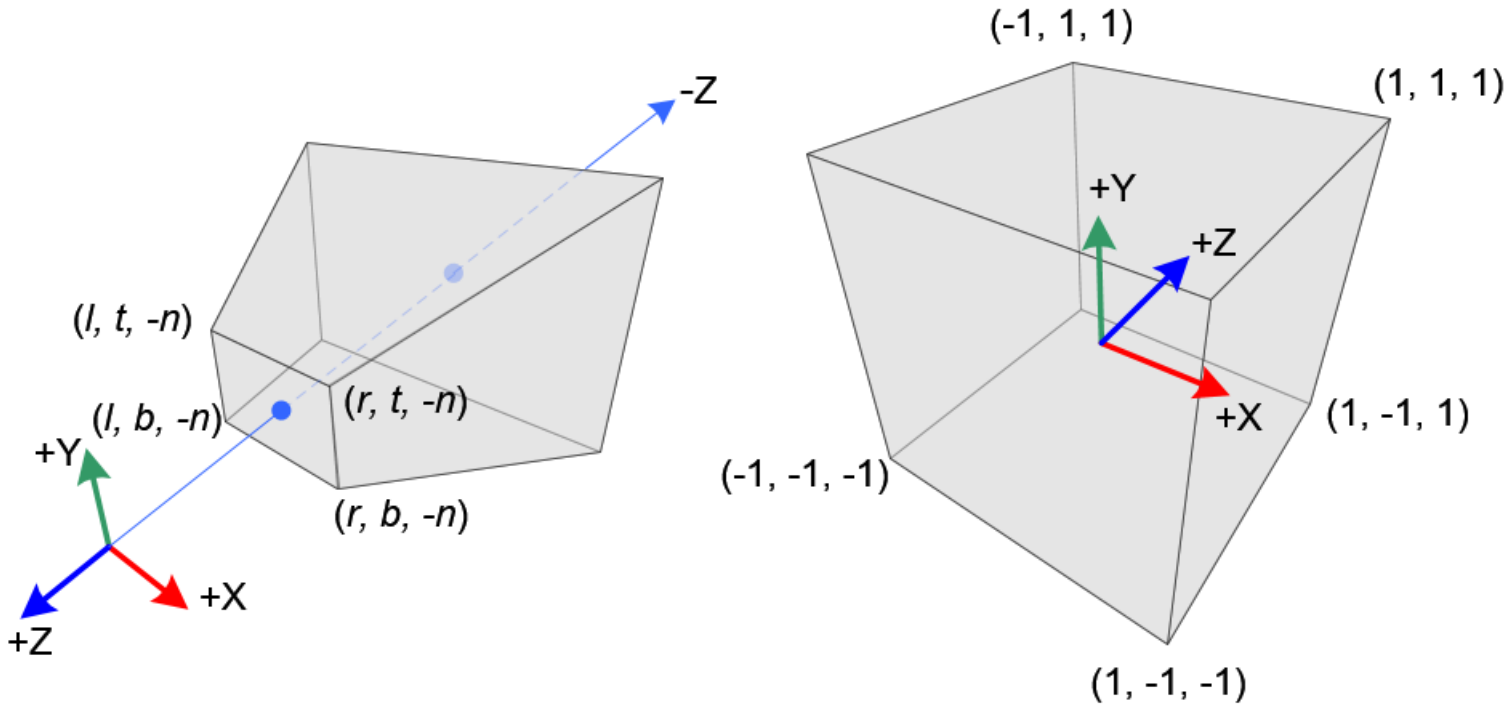
Camera space

$(x_{left}, y_{botton}, z_{near}) \times (x_{right}, y_{top}, z_{far})$

View frustum



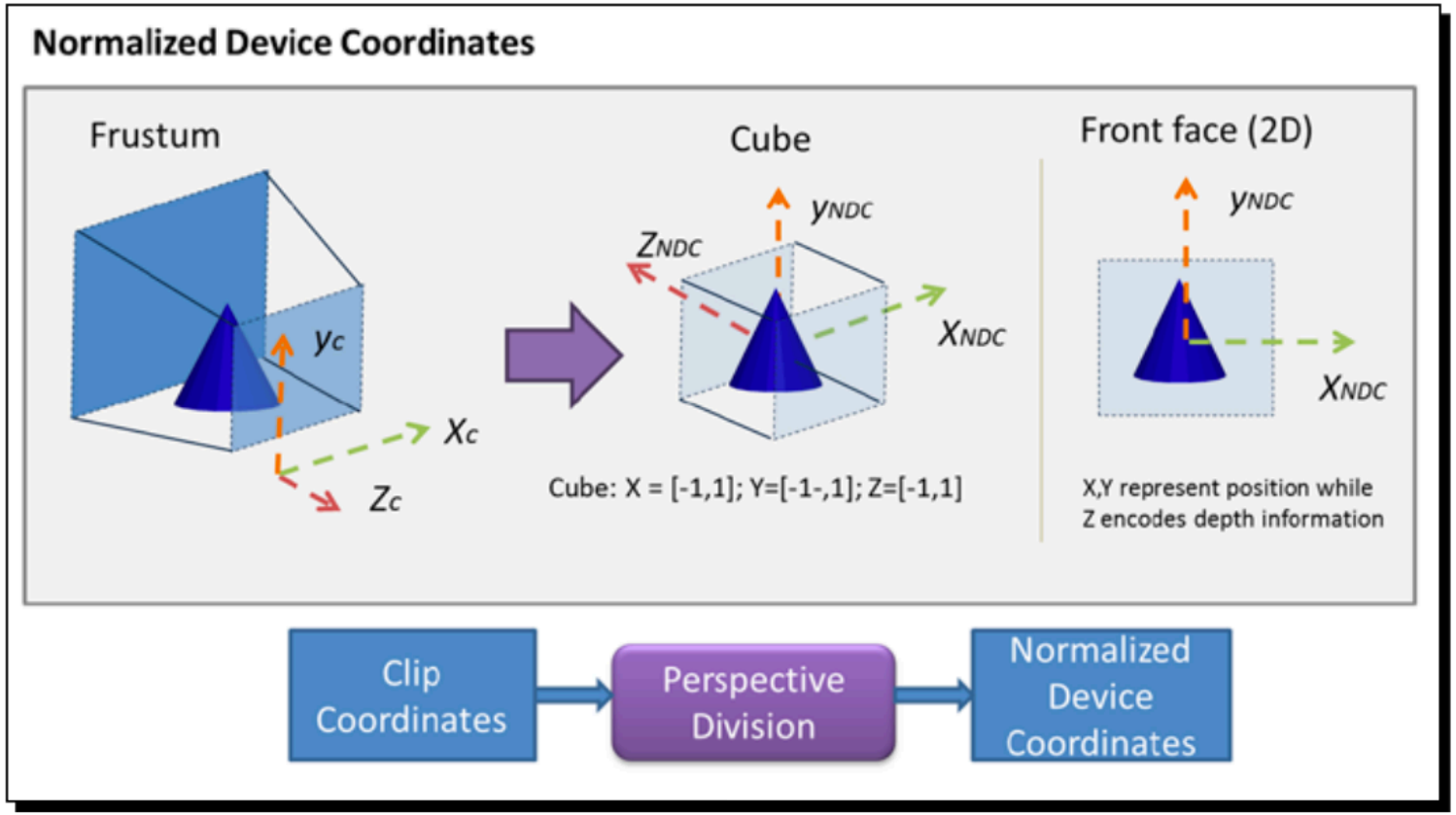
View frustum



Perspective **Orthographic**

songho.ca

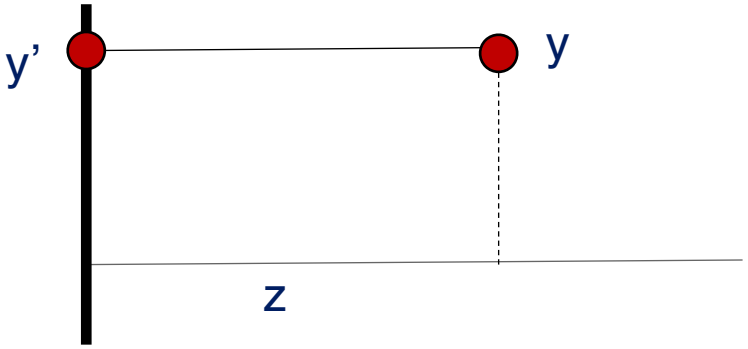
View frustum



Basic orthographic projection

Projection plane
(z = 0)

Basic orthographic projection

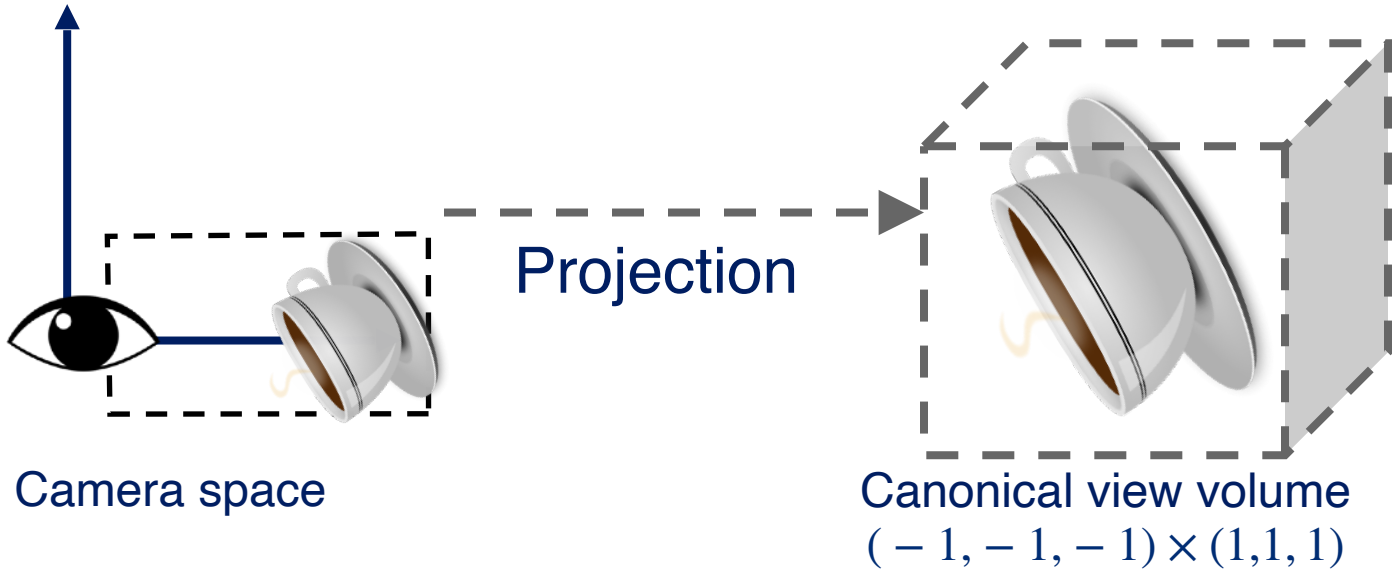


Projection plane
(z = 0)

$$\mathbf{x}' = \mathbf{x}, \mathbf{y}' = \mathbf{y}$$

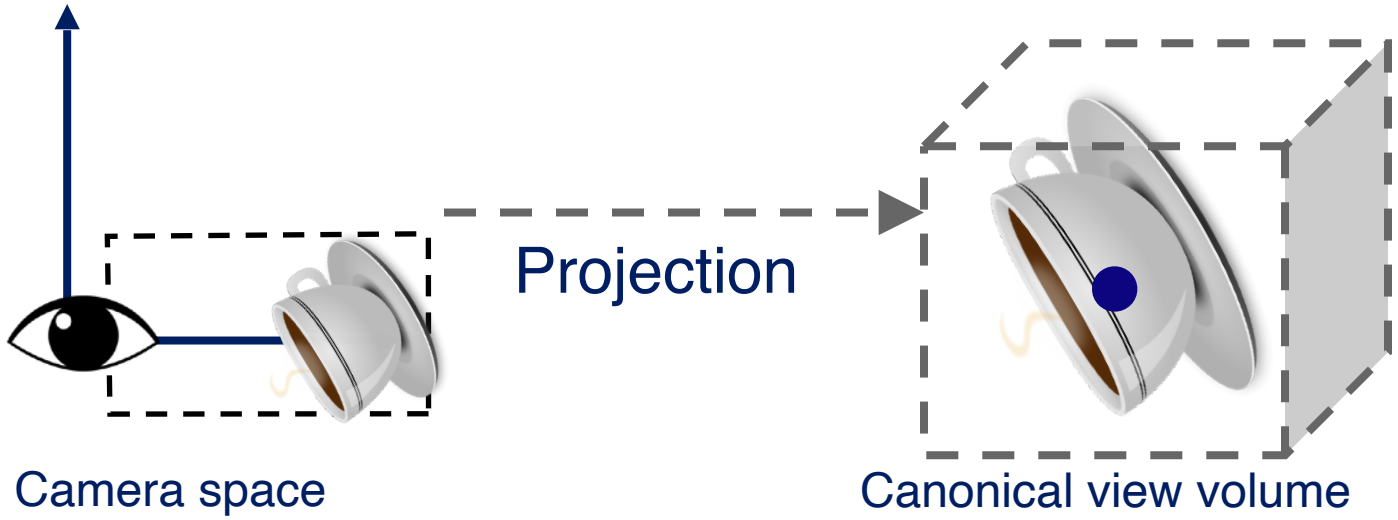
$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Orthographic transformation



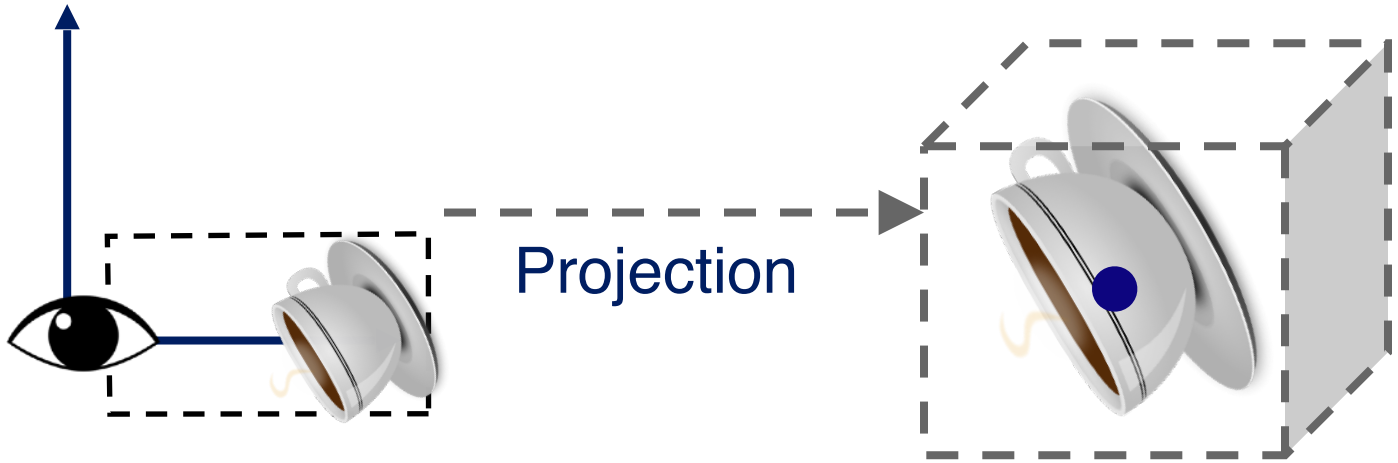
1. Translate so that center of the box lies at the origin.
2. Scale the volume to a 2-by-2 unit square: (-1,-1) to (+1, +1)
3. Flip the Z axis

Orthographic transformation



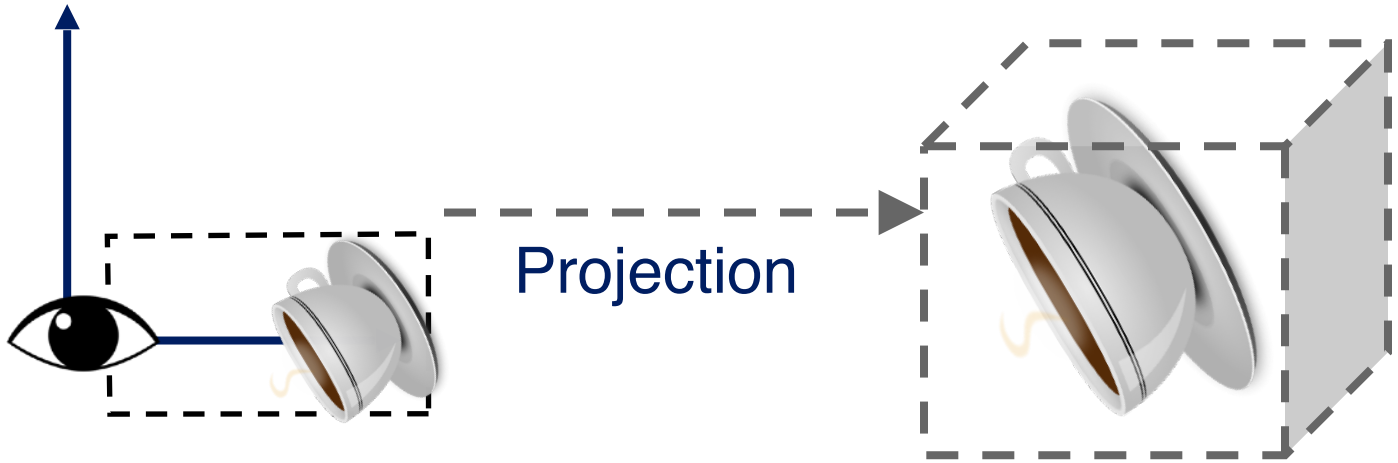
$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & -x_{mid} \\ 0 & 1 & 0 & -y_{mid} \\ 0 & 0 & 1 & -z_{mid} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Orthographic transformation



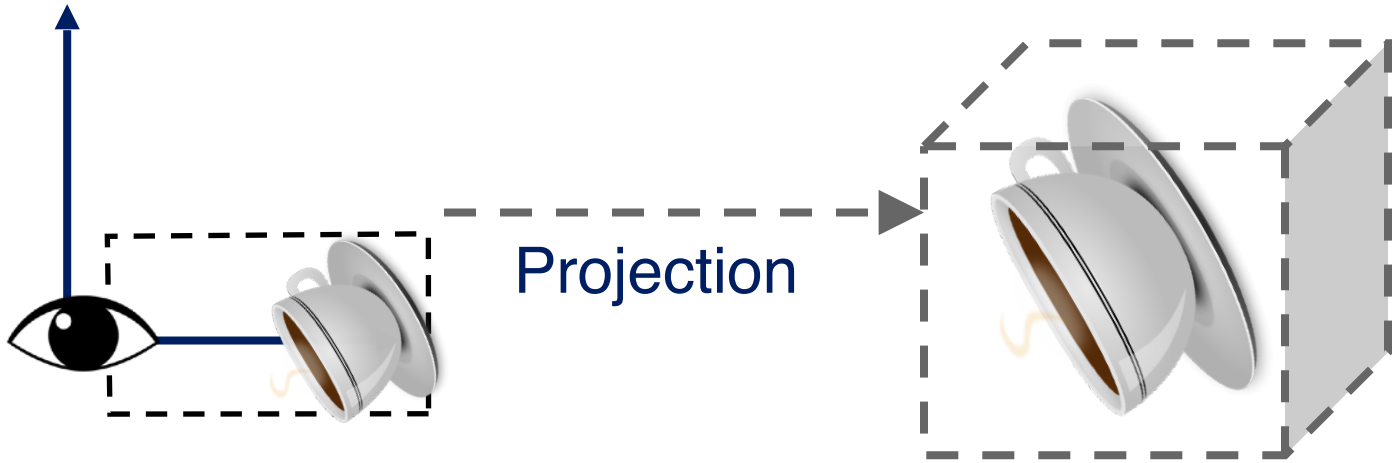
$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{2}{(right-left)} & 0 & 0 & 0 \\ 0 & \frac{2}{(top-bottom)} & 0 & 0 \\ 0 & 0 & \frac{2}{(far-near)} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -x_{mid} \\ 0 & 1 & 0 & -y_{mid} \\ 0 & 0 & 1 & -z_{mid} \\ 0 &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

Orthographic transformation



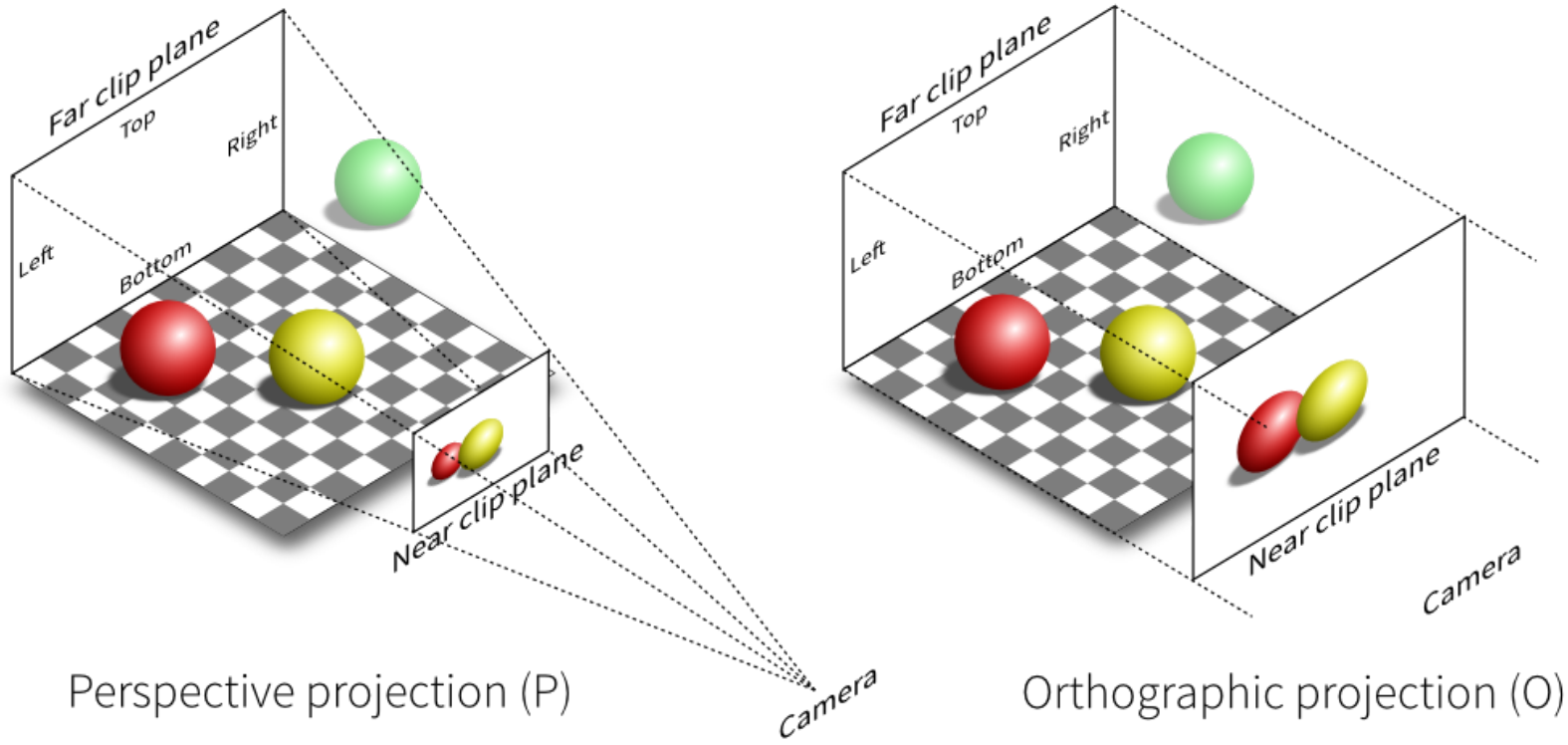
$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{2}{(right-left)} & 0 & 0 & 0 \\ 0 & \frac{2}{(top-bottom)} & 0 & 0 \\ 0 & 0 & \frac{2}{(far-near)} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}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

Orthographic transformation

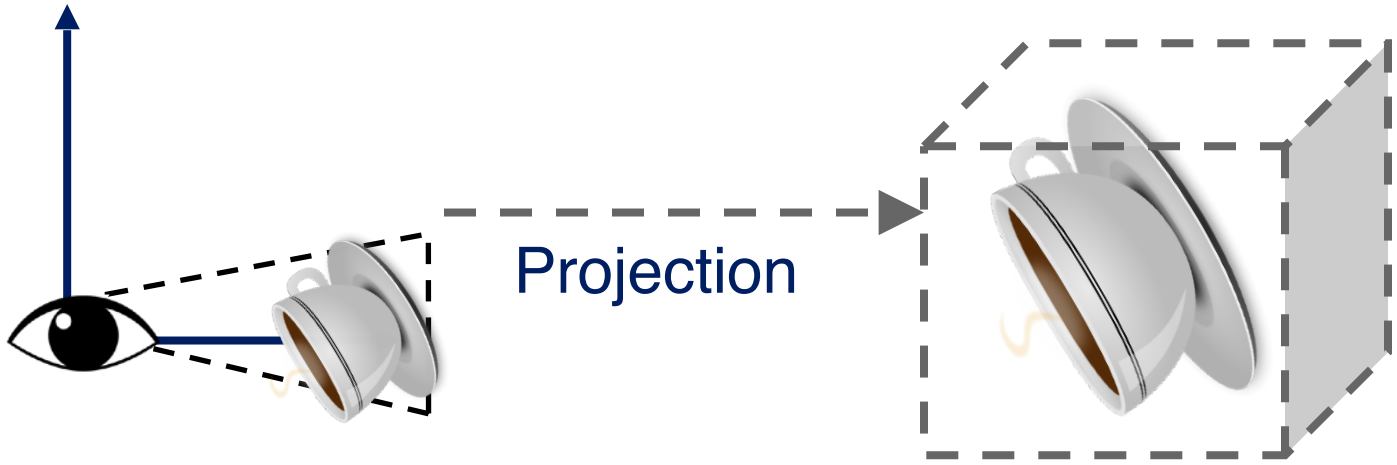


$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{2}{(right-left)} & 0 & 0 & \frac{-(right+left)}{(right-left)} \\ 0 & \frac{2}{(top-bottom)} & 0 & \frac{-(top+bottom)}{(top-bottom)} \\ 0 & 0 & \frac{-2}{(far-near)} & \frac{-(far+near)}{(far-near)} \\ 0 & 0 & 0 & 1 \end{pmatrix}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

View frustum



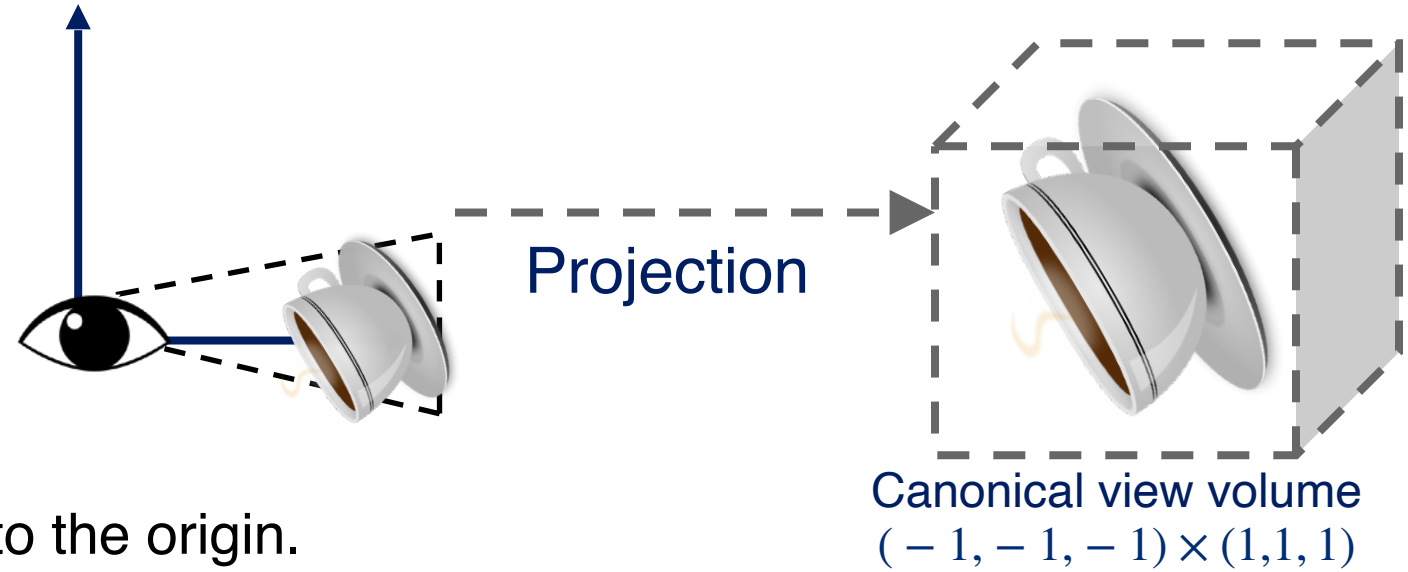
Perspective projection



1. Translate so that frustum center lies at the origin.

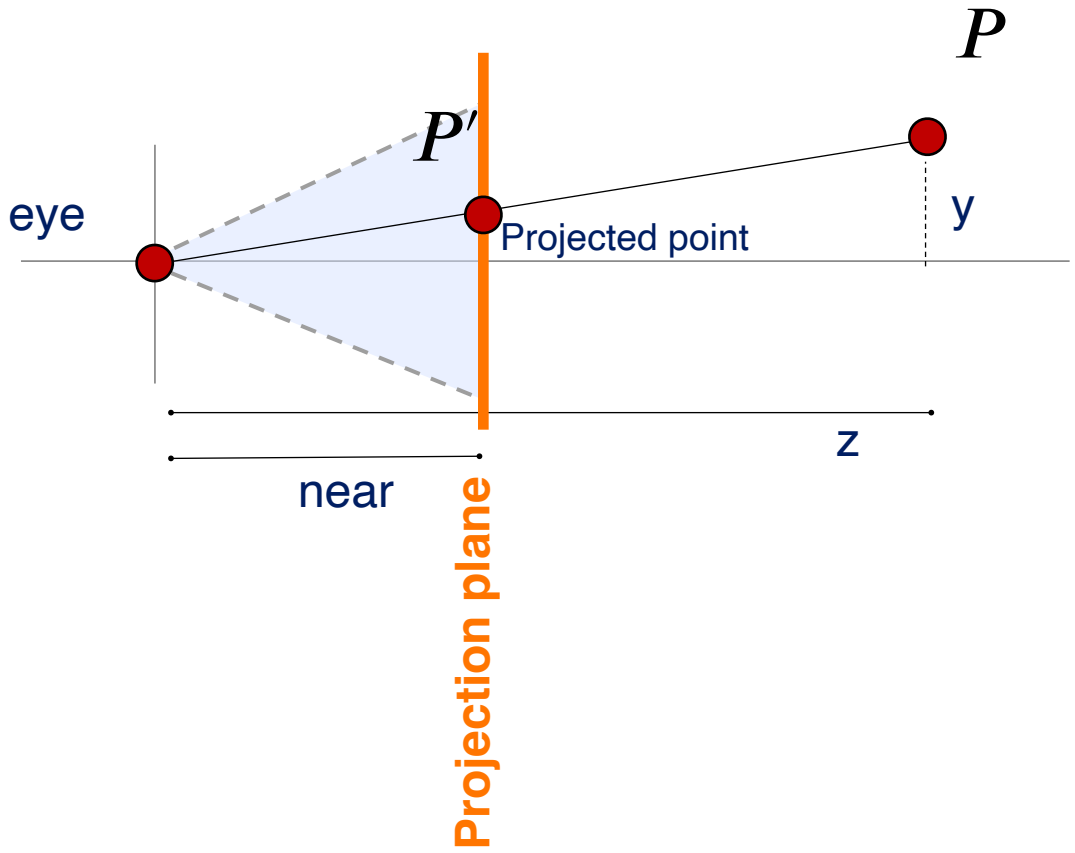
4. Scale the volume to a 2-by-2 unit square: (-1,-1) to (+1, +1).
5. Flip the z axis.

Perspective projection

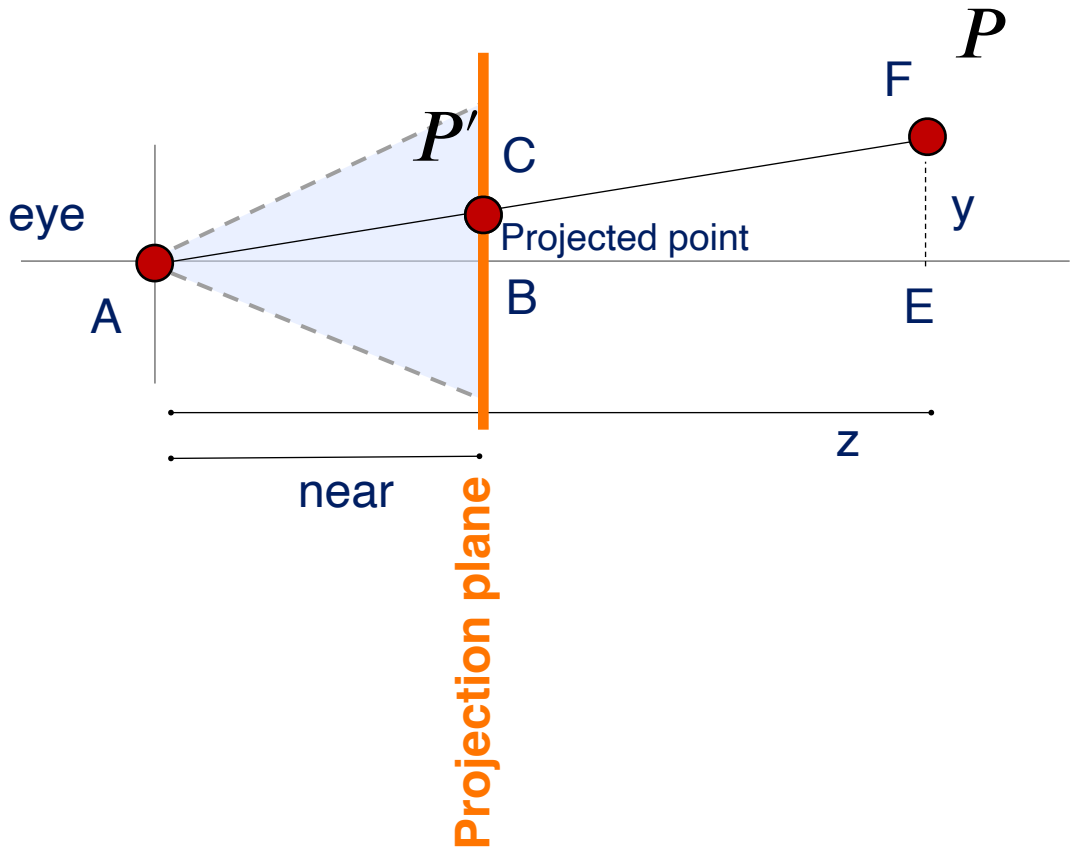


1. Translate to the origin.
2. Scale depth values into normalized range: (-1, +1).
3. **Perspective (foreshortening)** calculation.
4. Scale the volume to a 2-by-2 unit square: (-1,-1) to (+1, +1).
5. Flip the Z axis.

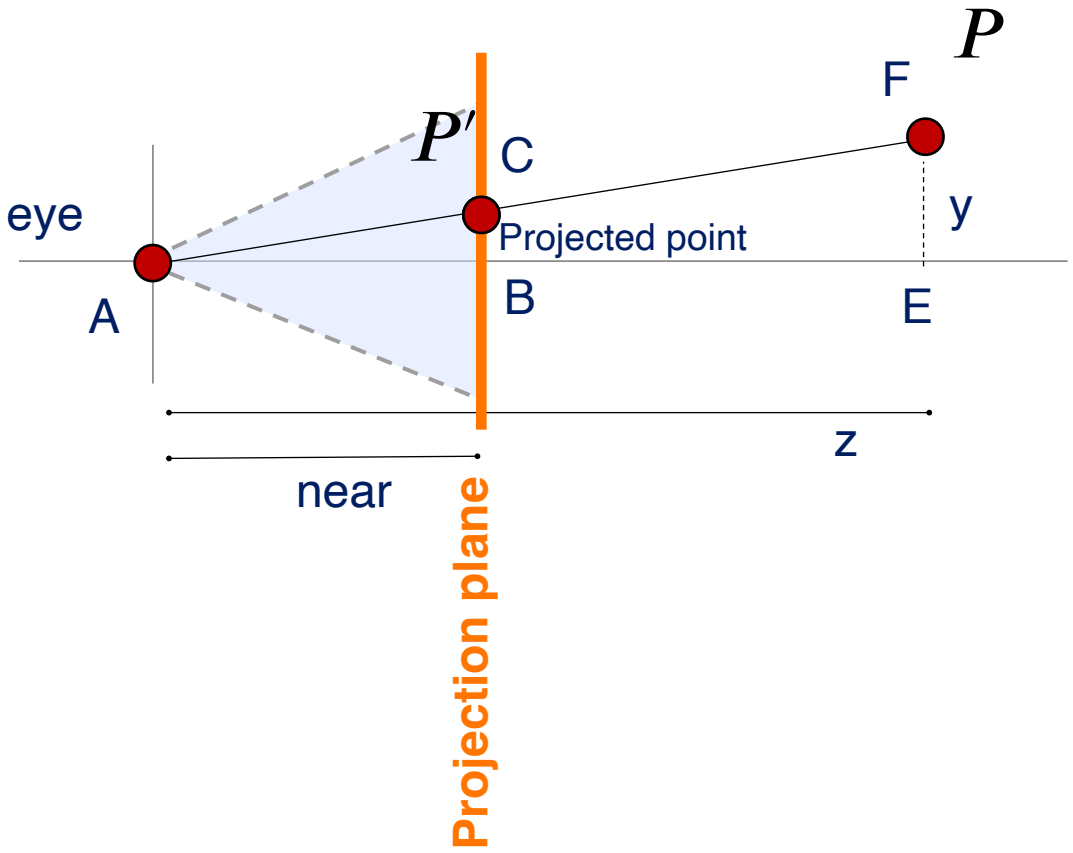
Perspective calculation



Perspective calculation



Perspective calculation



Project point P'?

$$\frac{AB}{AE} = \frac{BC}{EF}$$ (Triangles share the same angle at the eye)

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

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

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

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 $\frac{a * x}{z} + \ldots$

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 $\frac{a * x}{z} + \ldots$
- Solution: homogeneous coordinates.

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

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 $\frac{a * x}{z} + \ldots$
- Solution: homogeneous coordinates.

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

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

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 $\frac{a * x}{z} + \ldots$
- Solution: homogeneous coordinates.

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

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

Perspective calculation

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

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

Graphics pipeline expects perspective division in the w component

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

Exactly what we wanted to:

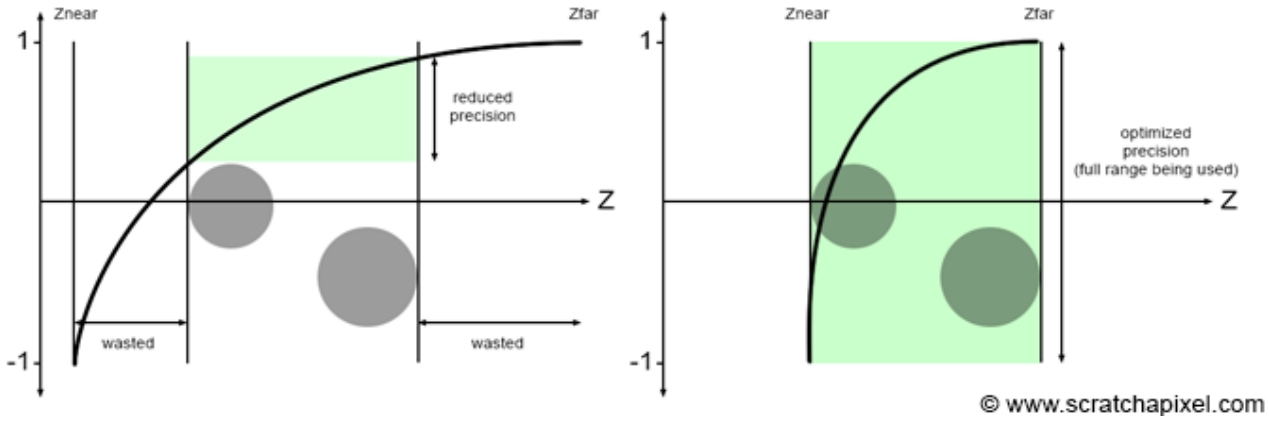
$$P'_y = \frac{near * P_y}{-P_z} \quad P'_x = \frac{near * P_x}{-P_z}$$

Scaling depth values

- Scaling depth values into a normalized range.
- Objective: mapping between (-near,-far) to (-1,+1).
- Naïve solution:

$$-1 = -near + 1 = -far$$

- However, float point numbers suffer from round-off errors (difference between 0.12345678 and 0.12345677 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 $\frac{a * x}{z} + \ldots$
- 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

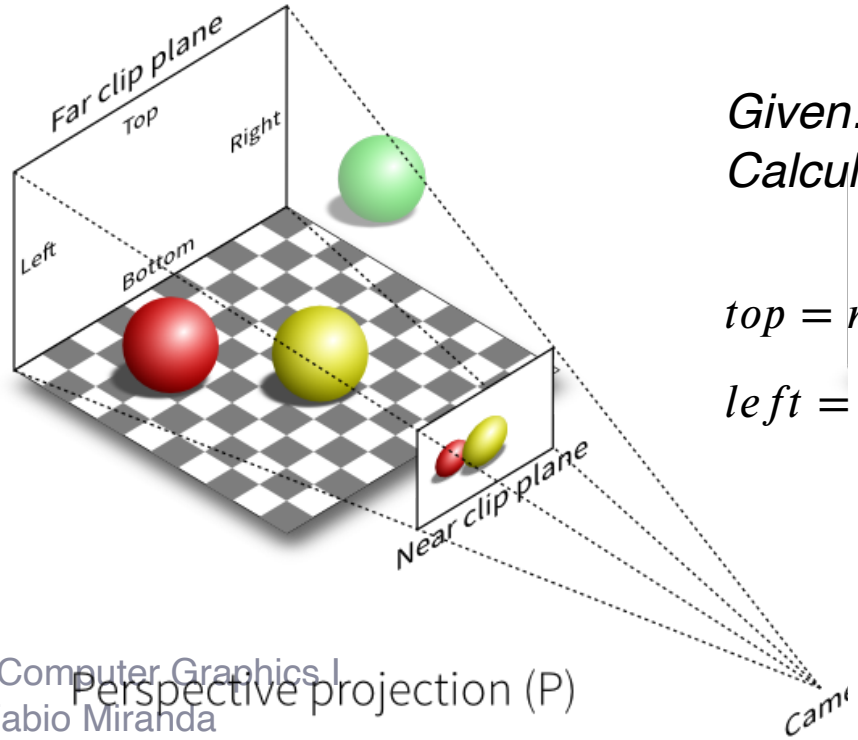
1. Translate to the origin.
2. Scale depth values into normalized range: (-1, +1). (and switch coordinate system)
3. Perspective calculation.
4. Scale the volume to a 2-by-2 unit square: (-1,-1) to (+1, +1).

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?



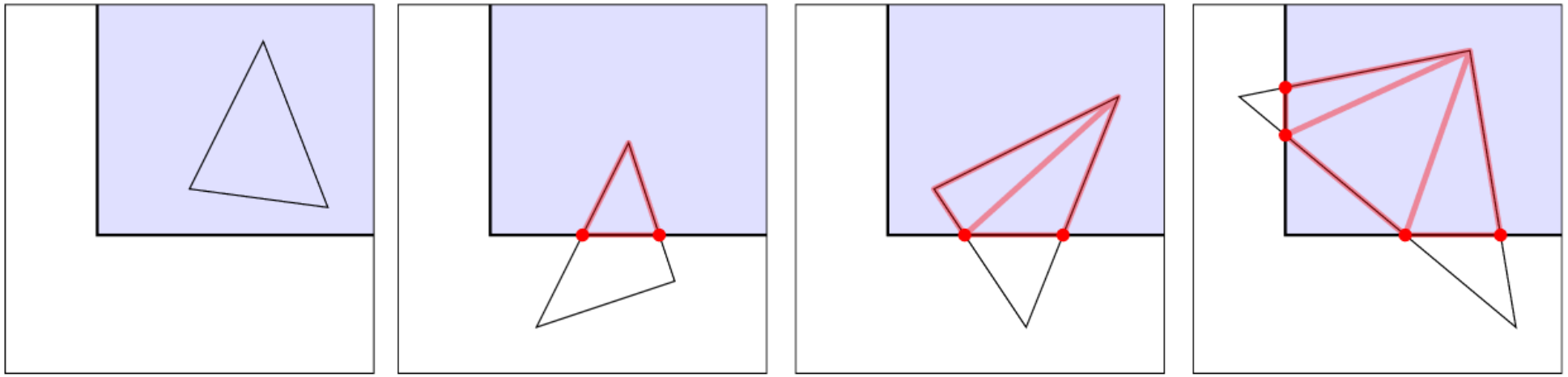
Given: fov, aspect, near, far
Calculate frustum properties (left, right, bottom, top, near, far):

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

$$left = -right$$

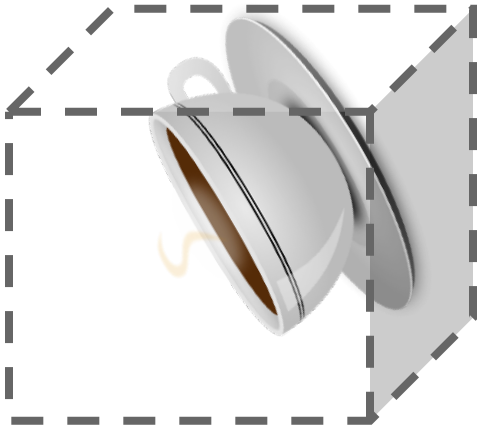
Canonical view volume

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



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

Viewport transformation



Canonical view volume

$(-1,-1,-1) \times (1,1,1)$

Viewport

Screen space

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

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

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).