CSC 4356
Interactive Computer Graphics
Lecture 8: 3D Viewing (Part 1)

Jinwei Ye

http://www.csc.lsu.edu/~jye/CSC4356/

Tue & Thu: 10:30 - 11:50am
218 Tureaud Hall

3D Scene → 2D Image

- We have learned how to build a 3D scene by modeling transformation
- How to map the 3D scene into 2D image?
  - Camera projection

What is a camera?

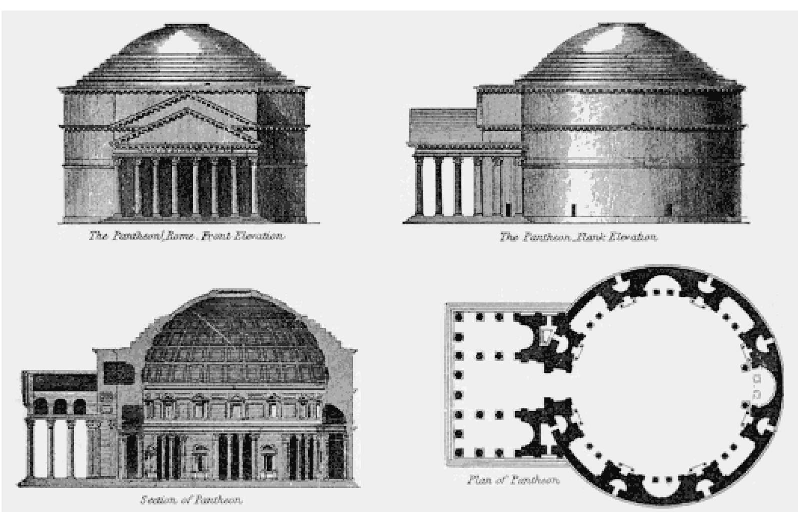
- In real world, camera is a light sensing device that collects light rays emitted from the scene to form 2D images
- In computer graphics, a camera projects 3D scene to 2D images
  - Projection matrix
  - Common methods:

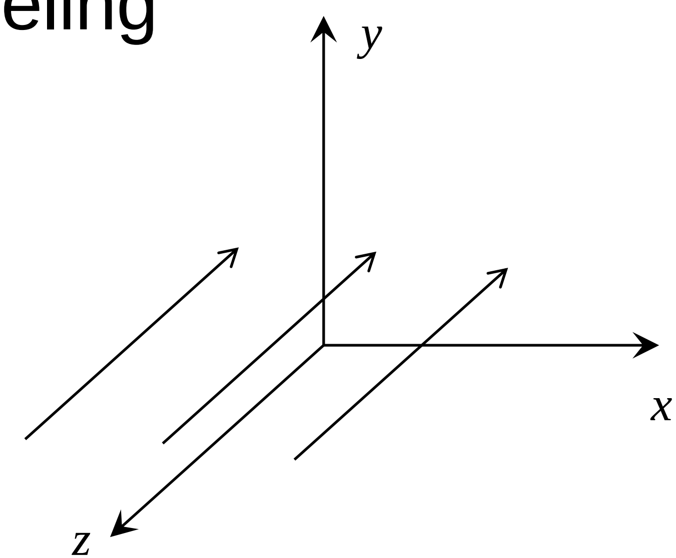
    Orthographic projection

    Perspective projection

Orthographic Projection

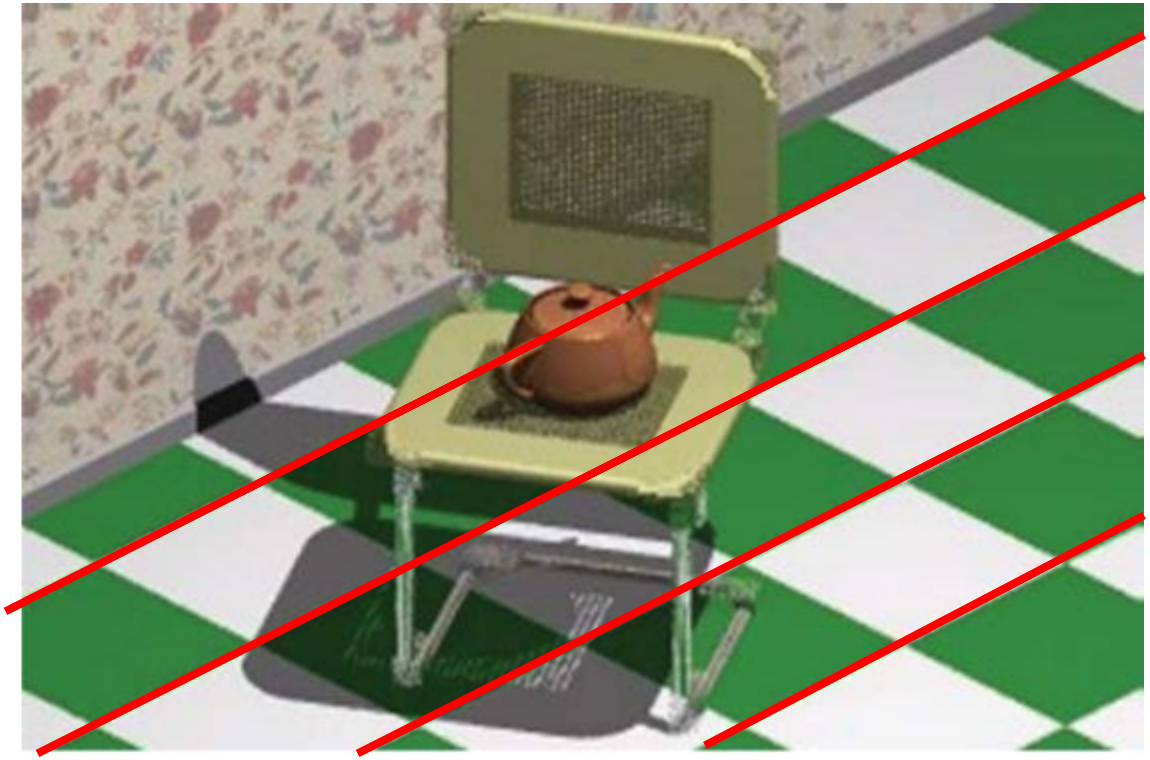
- Project every 3D points along lines parallel to the z-axis (in camera coordinate)
  - Simplest form of projection
  - Also called parallel projection
  - Commonly used for top, bottom, and side view in drafting and modeling





Orthographic Image

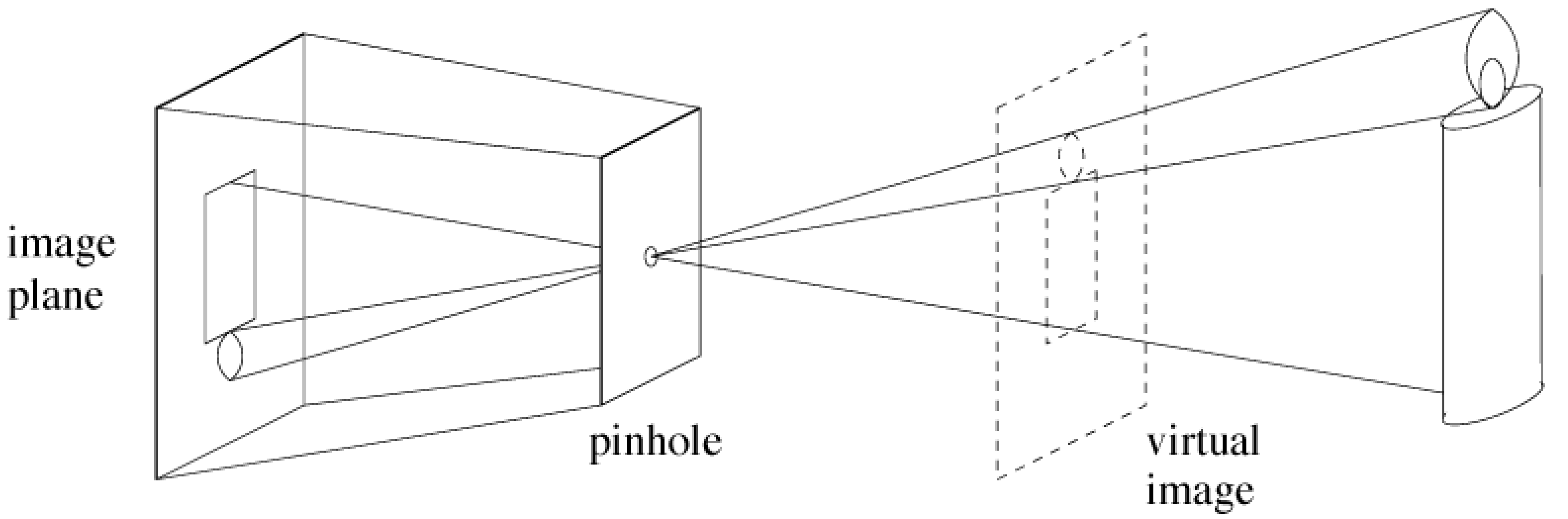
- Parallel lines remains parallel
- Appear unnatural due to lack of perspective foreshortening



Perspective Projection

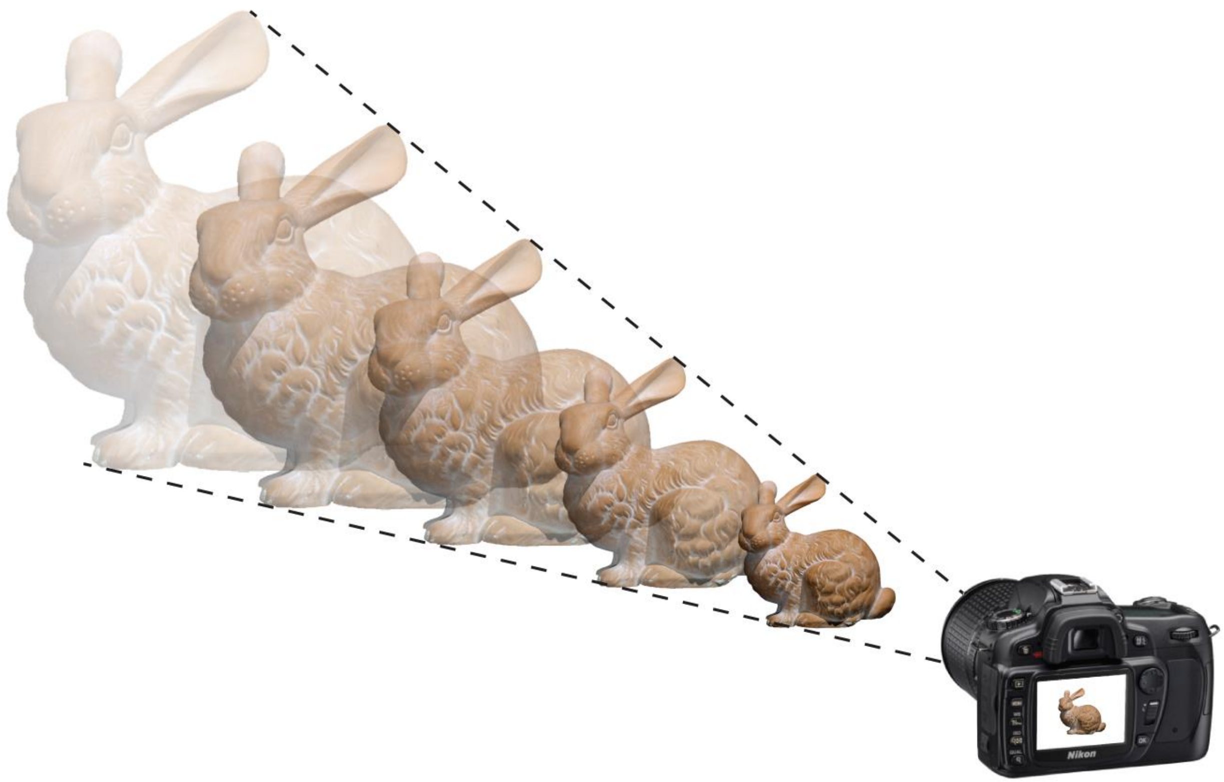
- Artists during the renaissance discovered the importance of perspective for making images appear realistic

Perspective Camera
(a.k.a. Pinhole Camera)



Perspective Image Properties

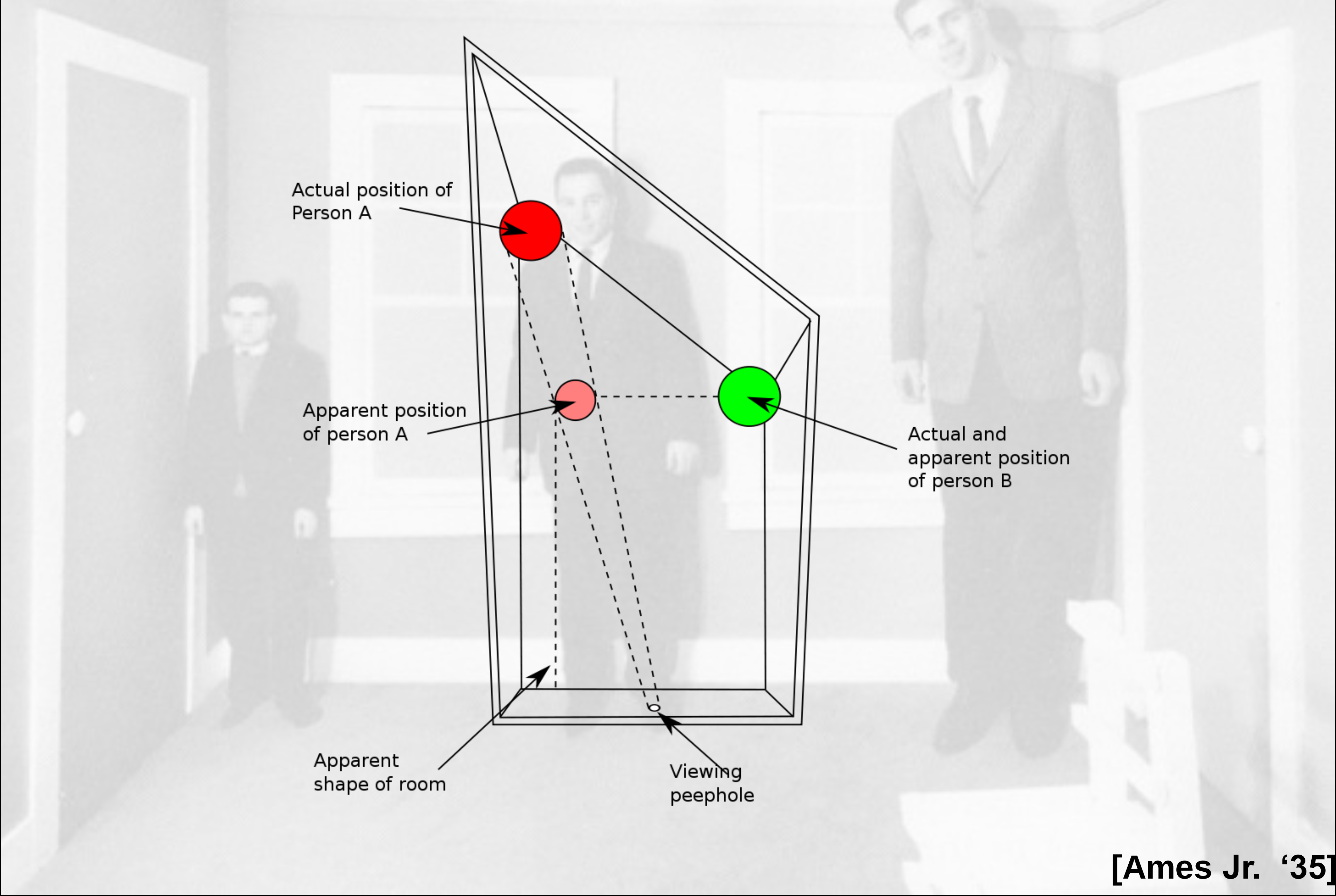
- Objects closer to the viewer appear larger
- Farther away objects appear smaller



Perspective Images

Images from Twitter and Flicker

Ames Room



[Ames Jr. ‘35]

Perspective Image Properties

- Parallel lines converge at a vanishing point

Images from Flicker

Perspective Images

- Distinguish a perspective image by vanishing points

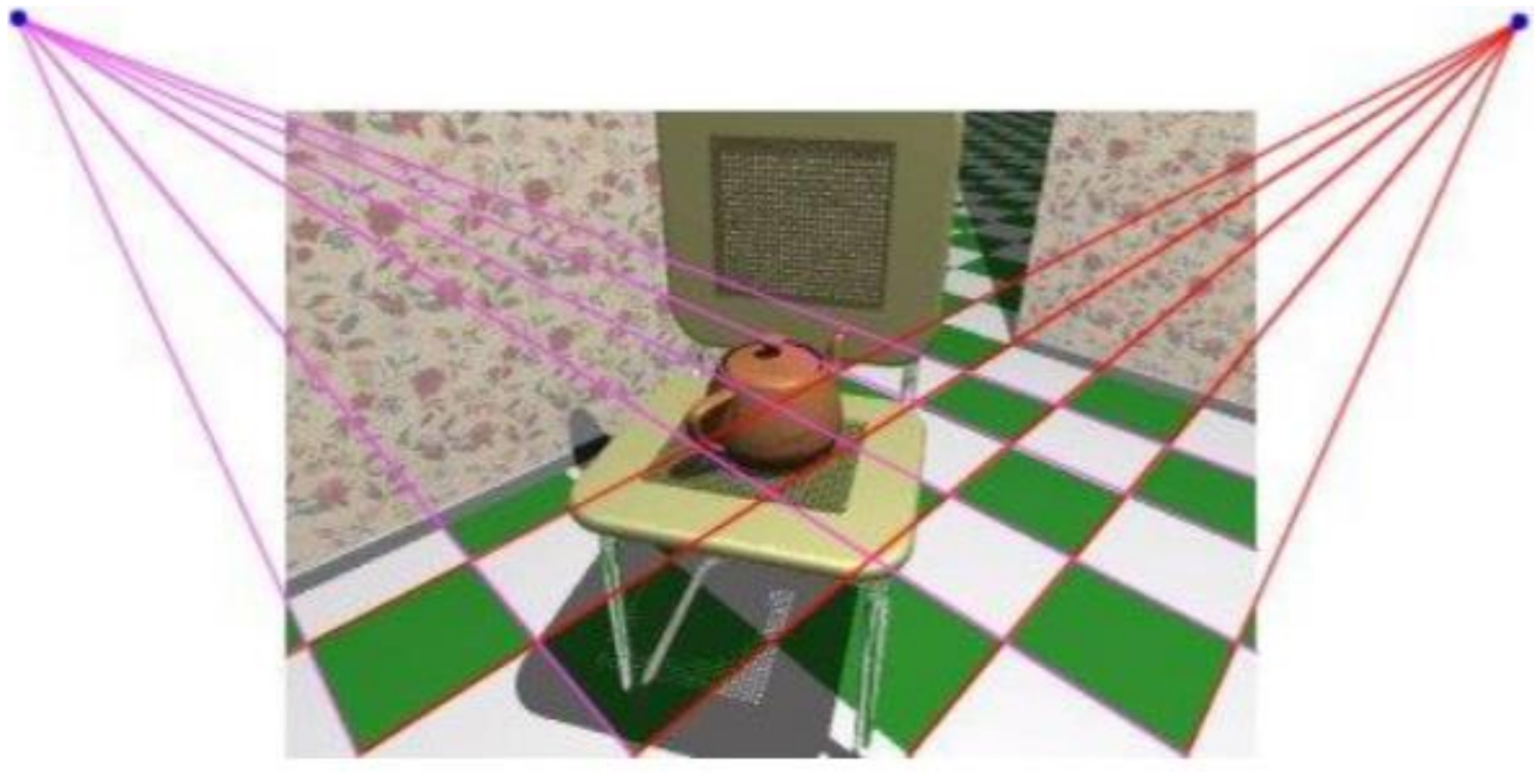
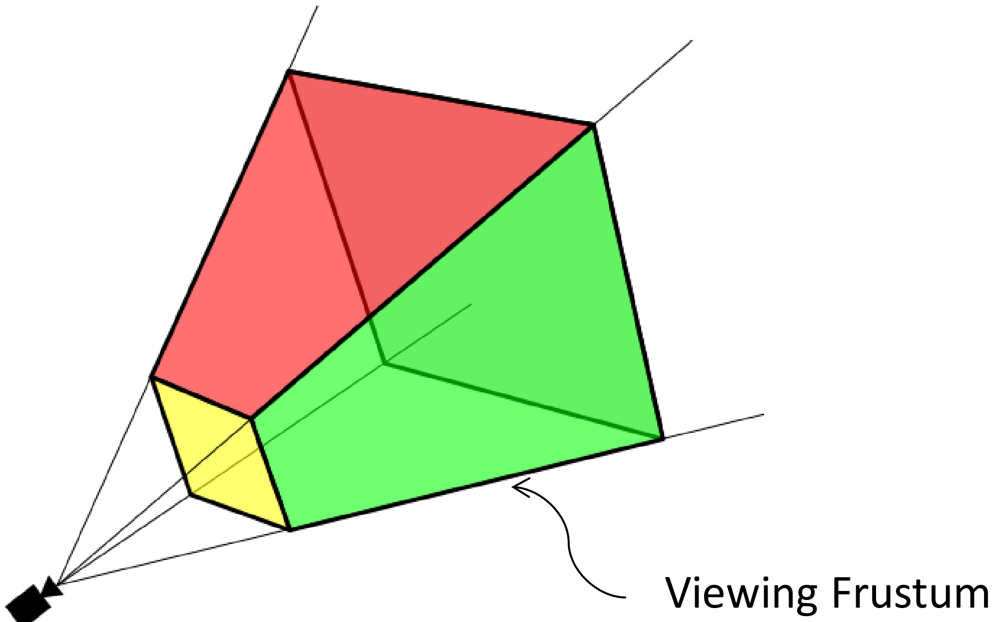


Image from Flicker

Stenop.Es Project

How to perform projection in OpenGL?

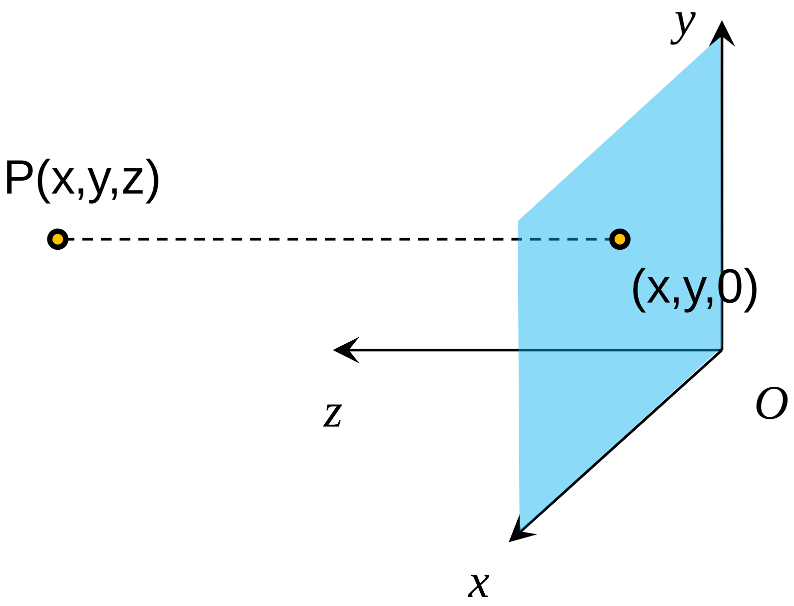
- Need to specify a viewing frustum
- Projection performed by multiplying scene point with a projection matrix
- Use Homogeneous coordinate



Orthographic Projection Matrix

- $[x,y,z] \rightarrow [x,y,0]$

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



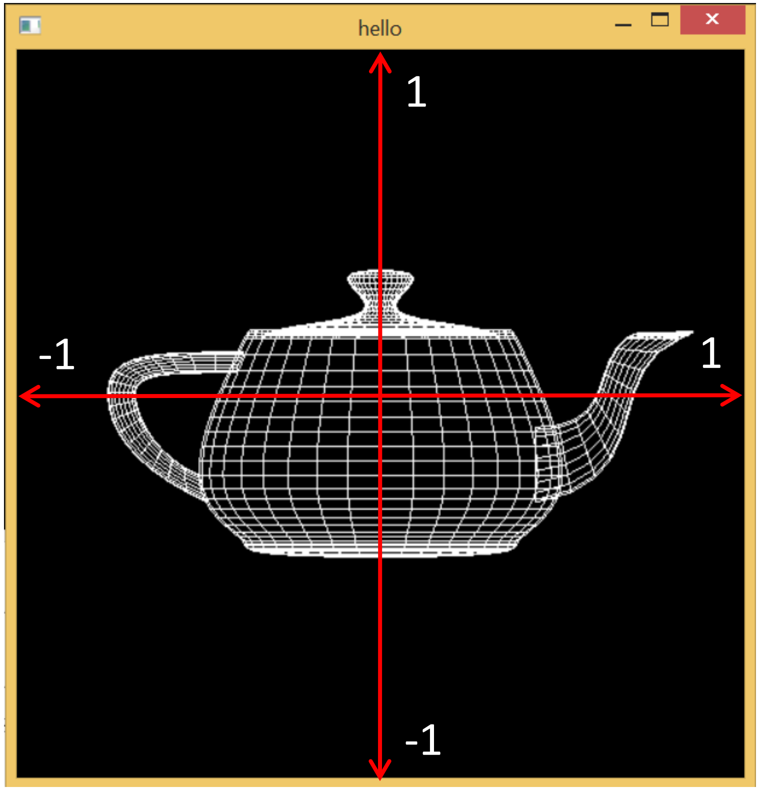
Orthographic Projection Matrix

- Orthographics projection matrix is simple
- Problem: the units of the transformed points are still the same as the model
- Need to map to normalized coordinate space

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

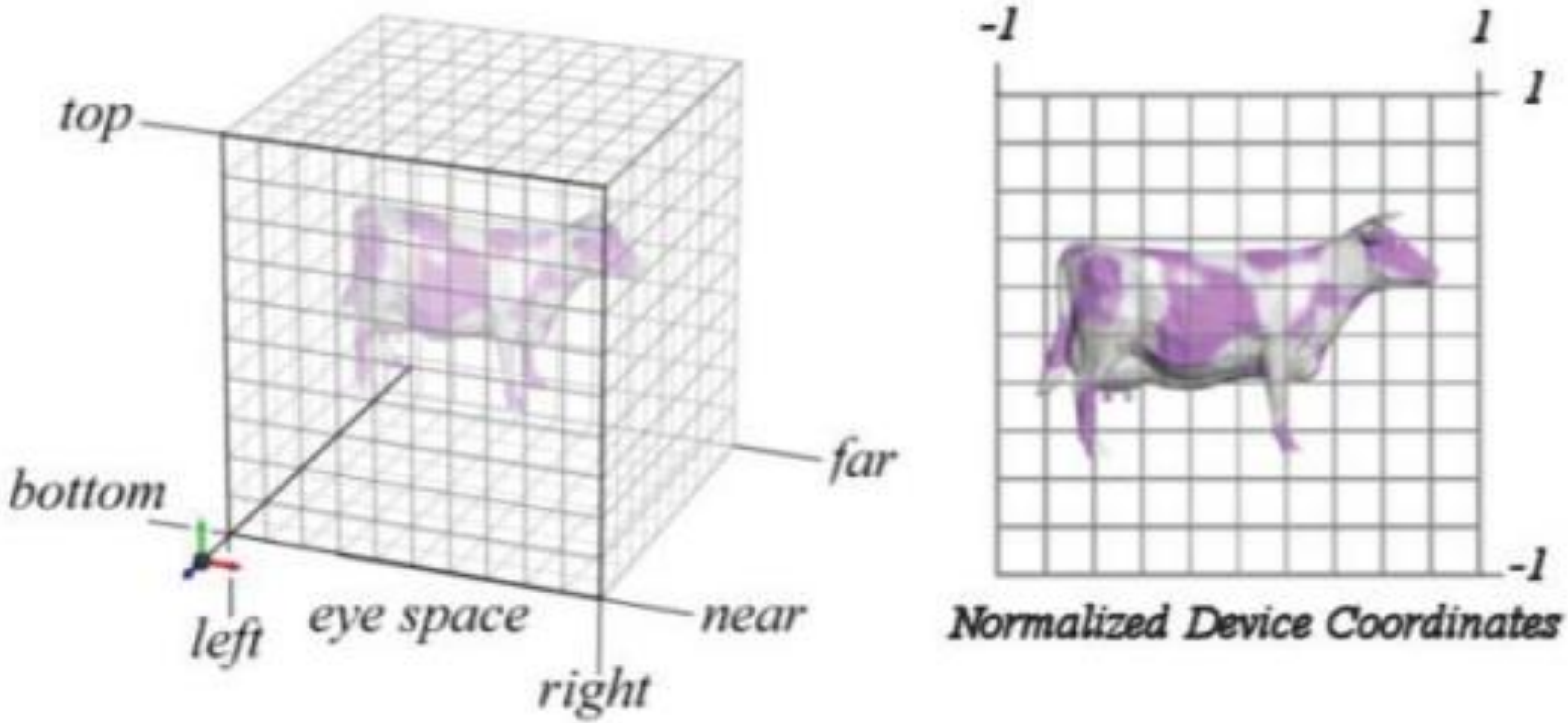
Normalized Device Coordinate (NDC)

- Normalized coordinate for display window
- Always ranging from -1 to 1 for x, y, and z



Mapping to NDC

- Translation & Scaling



Orthographic Projection in NDC

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

- Sanity check:

$$x = right \ \rightarrow x' = \frac{2right}{right - left} + \frac{-(right + left)}{right - left} = 1$$

$$x = left \ \rightarrow x' = \frac{2left}{right - left} + \frac{-(right + left)}{right - left} = -1$$

Orthographic Projection in OpenGL

- Projection Transformation happens *after* Modelview Transformation
  - MVP transformation: $\mathrm{v}' = \mathrm{PVMv}$
- Set matrix stack:

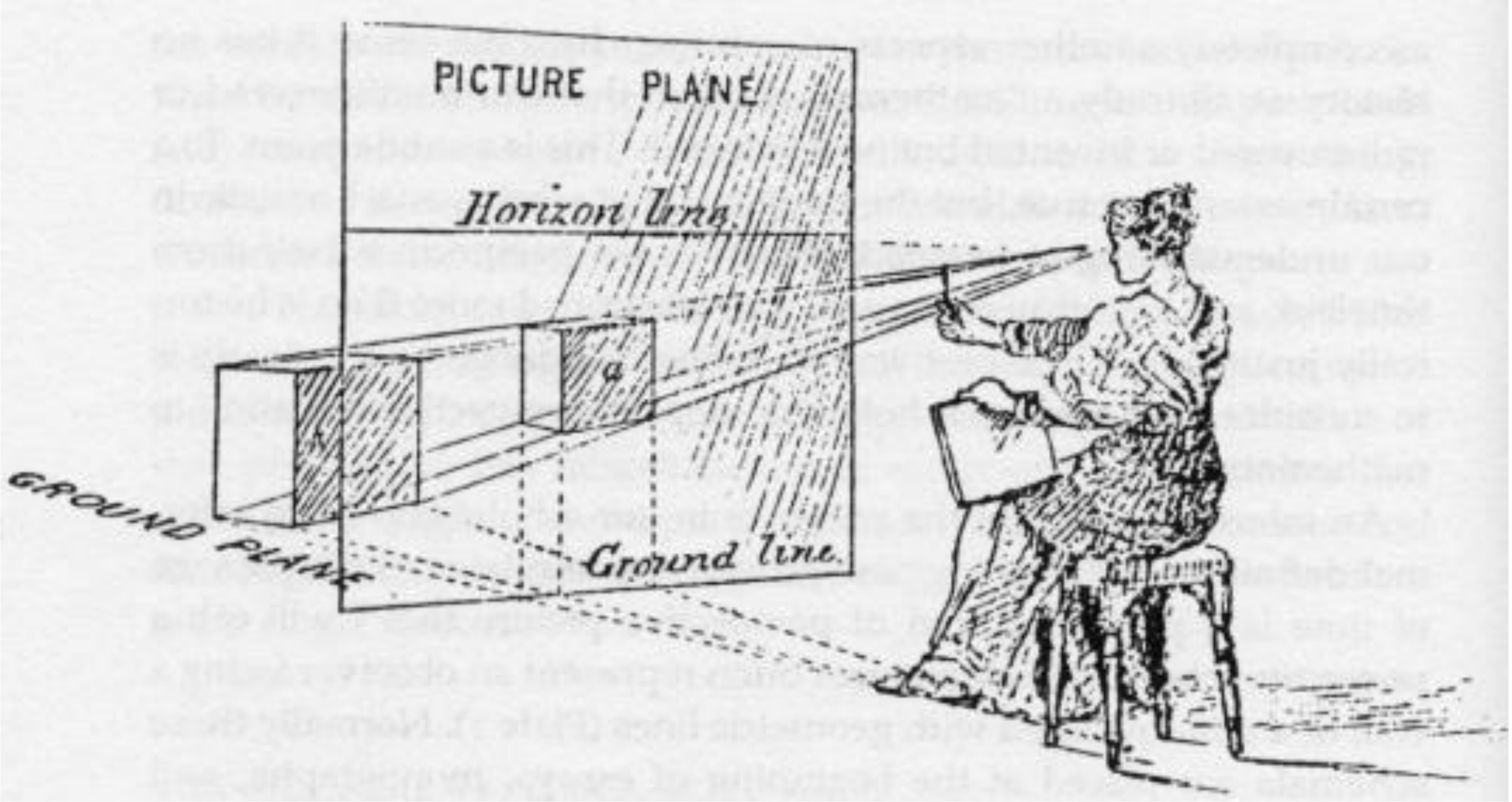
```
glMatrixMode(GL_PROJECTION);
```

- Orthographic projection matrix is constructed by

```
void glOrtho(double left, double right,
             double bottom, double top,
             double near, double far );
void glOrtho2D(double left, double right,
               double bottom, double top);
```

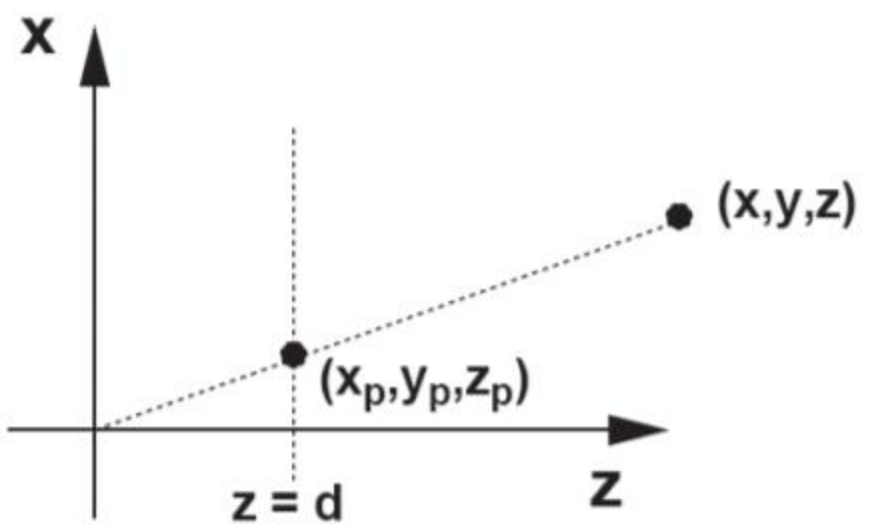
(assume near = -1, far =1)

Perspective Projection



Perspective Projection: Derivation

- Assume the pinhole (or center of projection) is the origin (0,0,0)
- Image plane at z = d



Perspective Projection: Derivation

- What are the coordinates of projected point?

$$\frac{x_p}{x} = \frac{d}{z}$$

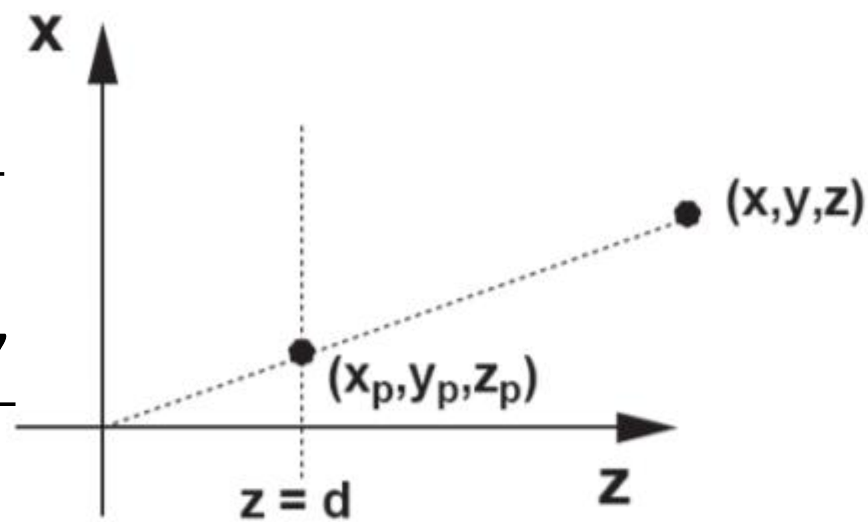
$$\frac{y_p}{y} = \frac{d}{z}$$

$$z_p = d$$

$$x_p = \frac{d \cdot x}{z}$$

$$y_p = \frac{d \cdot y}{z}$$

$$z_p = d$$



Perspective Projection Matrix

- How to express in form of matrix multiplication?

$$x_p = \frac{d \cdot x}{z} = \frac{x}{z/d}$$

$$y_p = \frac{d \cdot y}{z} = \frac{y}{z/d}$$

$$z_p = d$$

$$\begin{bmatrix} wx' \\ wy' \\ wz' \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Divide w to make the fourth element 1

Perspective Projection Matrix

- Why closer objects appear larger?

$$\begin{bmatrix} wx' \\ wy' \\ wz' \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$w = z / d$$

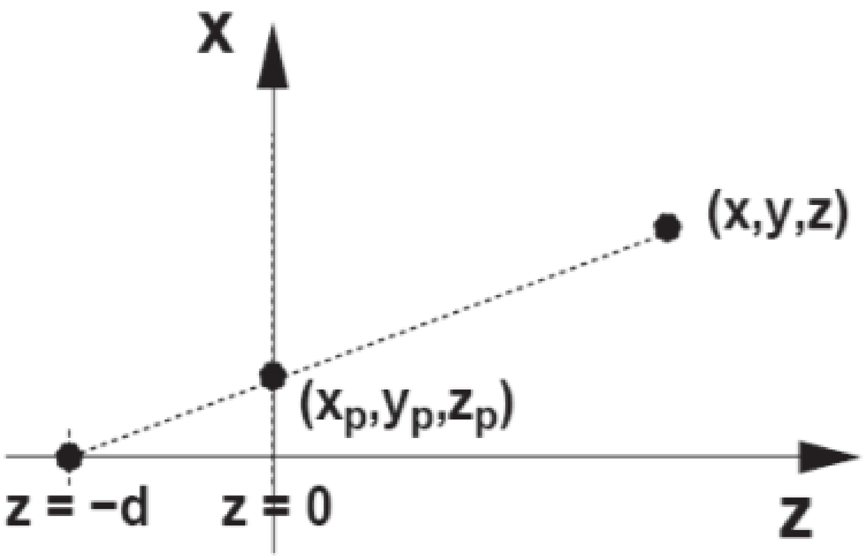
Another Perspective Projection

- CoP at (0,0,-d)
- Image plane at z = 0 (x-y plane)

$$x_p = \frac{d \cdot x}{d + z} = \frac{x}{z/d + 1}$$

$$y_p = \frac{d \cdot y}{d + z} = \frac{y}{z/d + 1}$$

$$z_p = 0$$



$$\begin{bmatrix} wx' \\ wy' \\ wz' \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/d & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

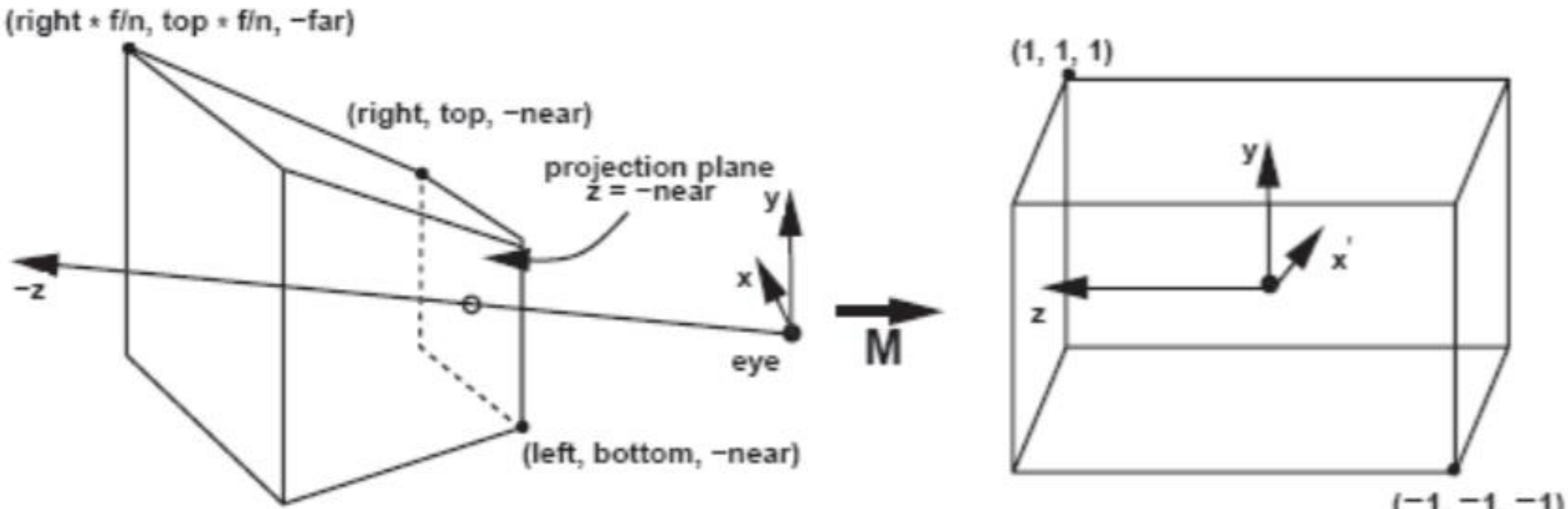
What happens if d goes to infinity?

Perspective Viewing Frustum

- Perspective viewing frustum looks like a rectangular pyramid

Mapping to NDC

- Scaling, Shear, & Translation



Perspective Projection in NDC

$$\begin{bmatrix} wx' \\ wy' \\ wz' \\ w \end{bmatrix} = \begin{bmatrix} \frac{2 \cdot near}{right - left} & 0 & \frac{-(right + left)}{right - left} & 0 \\ 0 & \frac{2 \cdot near}{top - bottom} & \frac{-(top + bottom)}{top - bottom} & 0 \\ 0 & 0 & \frac{far + near}{far - near} & \frac{-2 \cdot far \cdot near)}{far - near} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Sanity check:

$$\begin{gathered} x = right \\ z = near \end{gathered} \quad \rightarrow x' = \frac{\frac{2 \cdot near \cdot right}{right - left} + \frac{-(right + left) \cdot near}{right - left}}{near} = 1$$

$$\begin{gathered} x = right \cdot \frac{far}{near} \\ z = far \end{gathered} \quad \rightarrow x' = \frac{\frac{2 \cdot far \cdot right}{right - left} + \frac{-(right + left) \cdot far}{right - left}}{far} = 1$$

Perspective Projection in OpenGL

- Set matrix stack:

```
glMatrixMode(GL_PROJECTION);
```

- Perspective projection matrix is constructed by

```
void glFrustum(double left, double
right, double bottom, double top,
double near, double far);
```

or

```
void gluPerspective(double vertfov,
double aspect, double near, double far);
```

gluPerspective()

- Use vertical FOV and aspect ratio to specify the viewing frustum
- vert fov: $\theta = 2\arctan(0.5height/near)$