CS 543: Computer Graphics

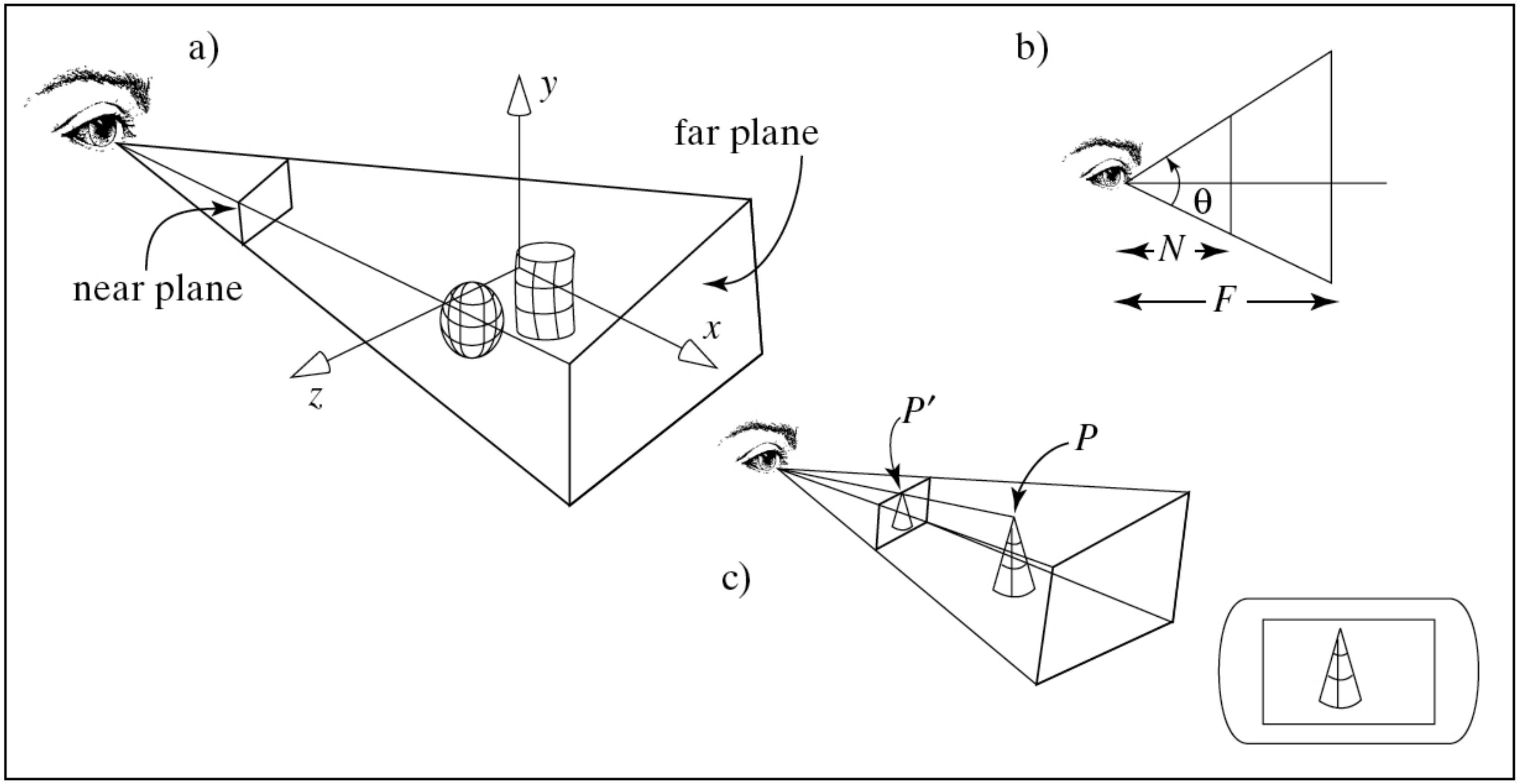
Projection

Robert W. Lindeman
Associate Professor
Interactive Media & Game Development
Department of Computer Science
Worcester Polytechnic Institute
gogo@wpi.edu

(with lots of help from Prof. Emmanuel Agu :-)

3D Viewing and View Volume

- Recall: 3D viewing set up

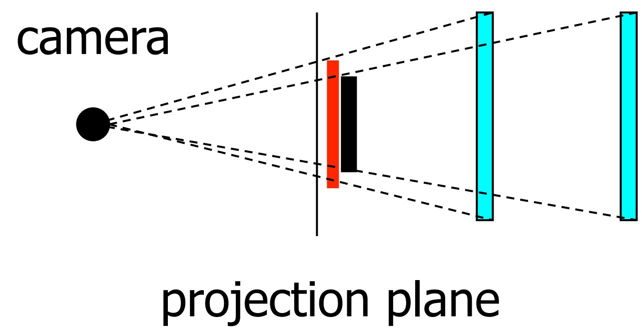


Projection Transformation

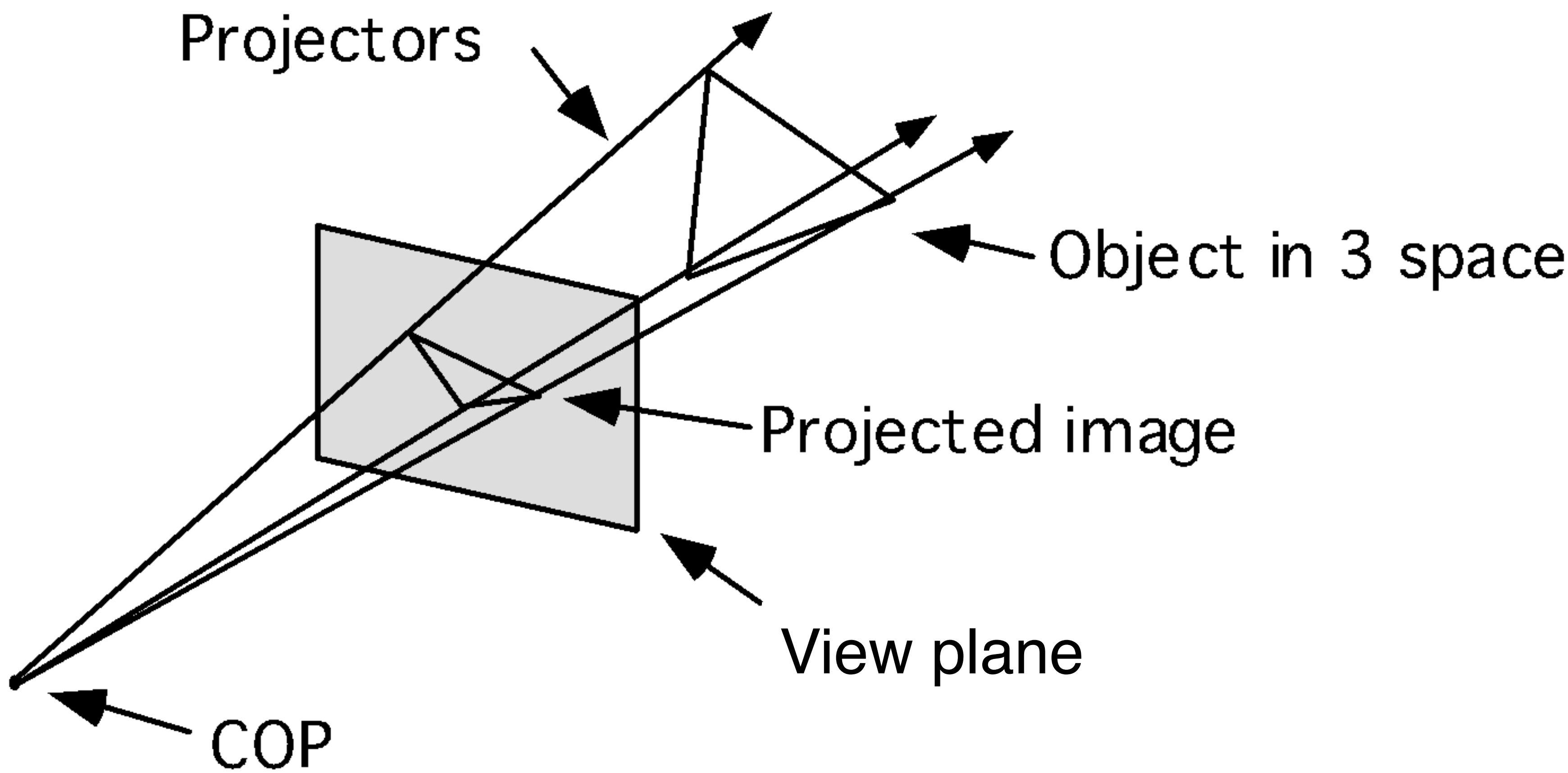
- View volume can have different shapes
  - Parallel, perspective, isometric
- Different types of projection
  - Parallel (orthographic), perspective, *etc.*
- Important to control
  - Projection type: perspective or orthographic, *etc.*
  - Field of view and image aspect ratio
  - Near and far clipping planes

Perspective Projection

- Similar to real world
- Characterized by *object foreshortening*
  - Objects appear larger if they are closer to camera
- Need to define
  - Center of projection (COP)
  - Projection (view) plane
- Projection
  - Connecting the object to the center of projection

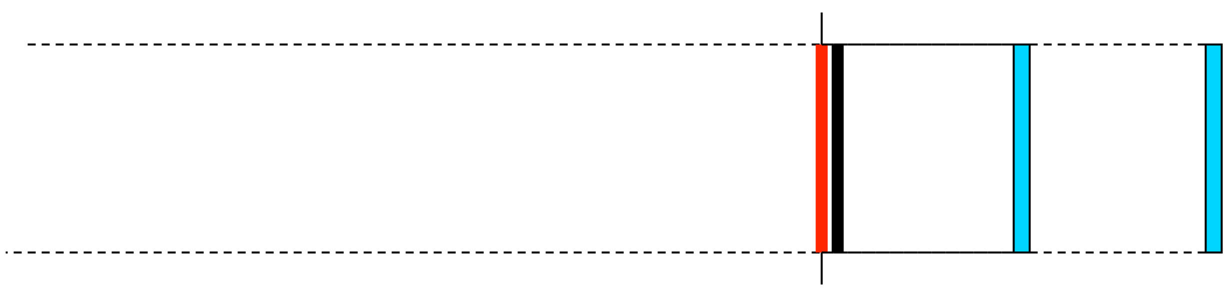


Why is it Called *Projection*?



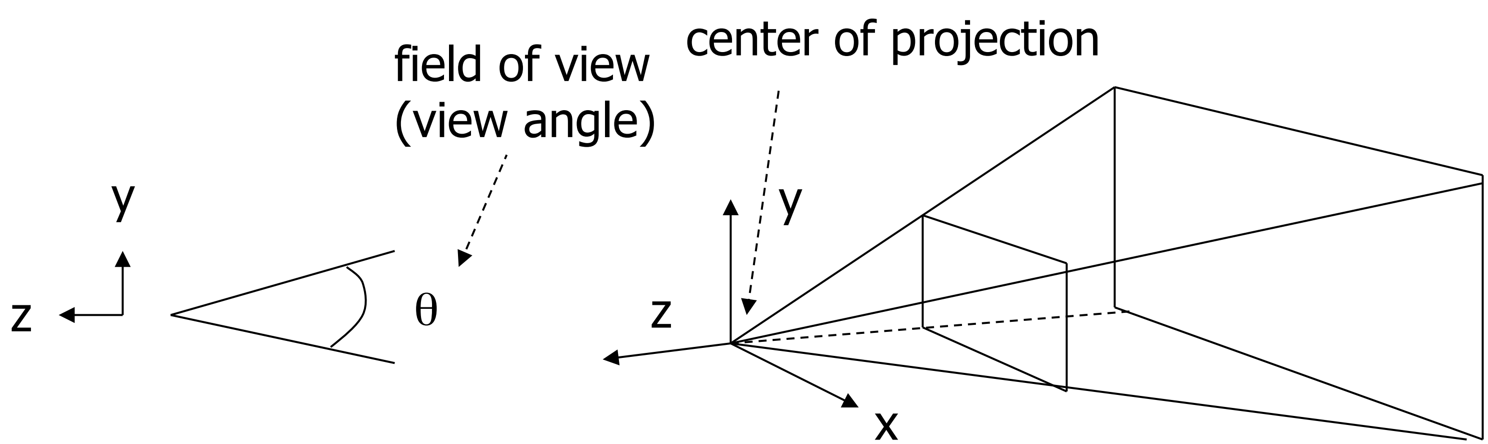
Orthographic (Parallel) Projection

- No foreshortening effect
  - Distance from camera does not matter
- The center of projection is at infinity
- Projection calculation
  - Just choose equal z coordinates



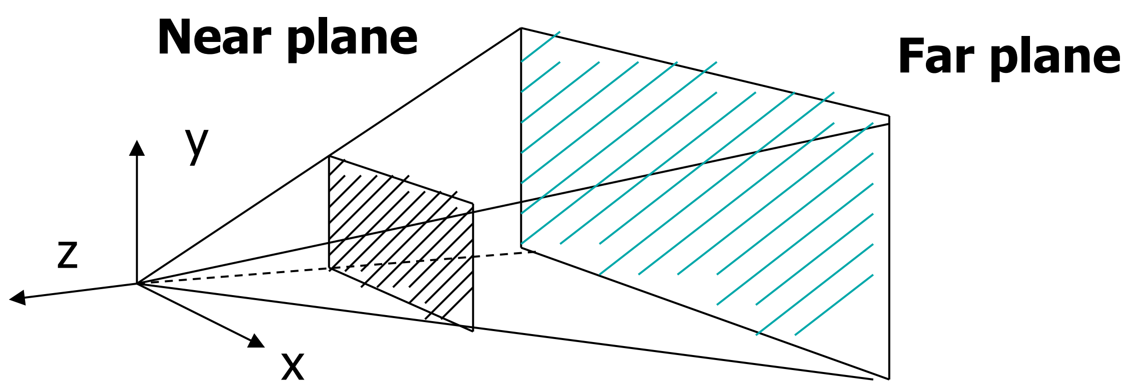
Field of View

- Determine how much of the world is taken into the picture
- Larger field of view = smaller object-projection size



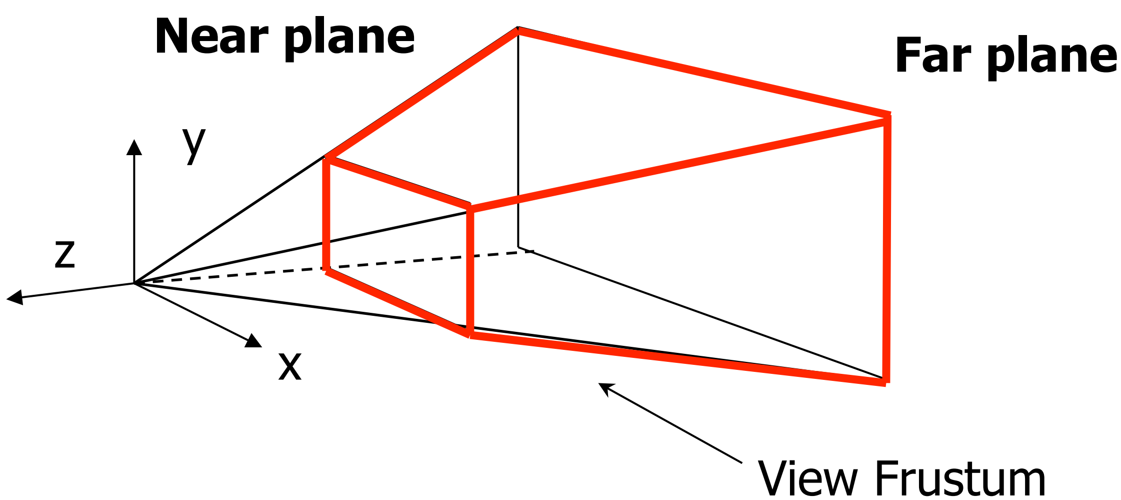
Near and Far Clipping Planes

- Only objects between near and far planes are drawn
- Near plane + far plane + field of view = **View Frustum**



View Frustum

- 3D counterpart of 2D-world clip window
- Objects outside the frustum are clipped



Projection Transformation

- In OpenGL
  - Set the matrix mode to **`GL_PROJECTION`**
  - For perspective projection, use

```
gluPerspective( fovy, aspect, near, far );
```

or

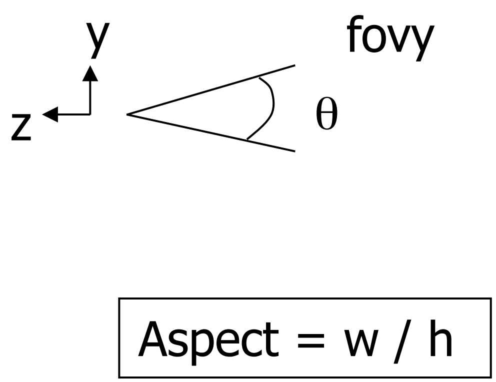
```
glFrustum( left, right, bottom, top,
           near, far );
```

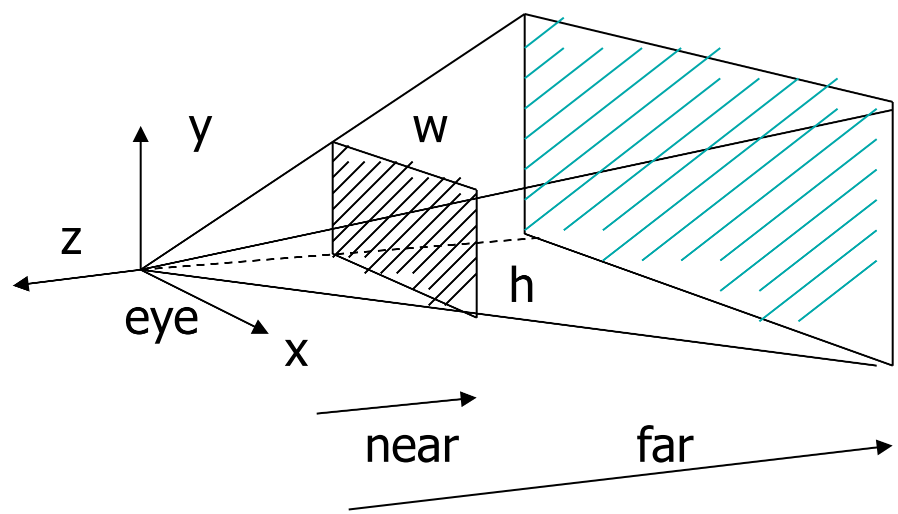
  - For orthographic projection, use

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

`gluPerspective( fovy, aspect, near, far )`

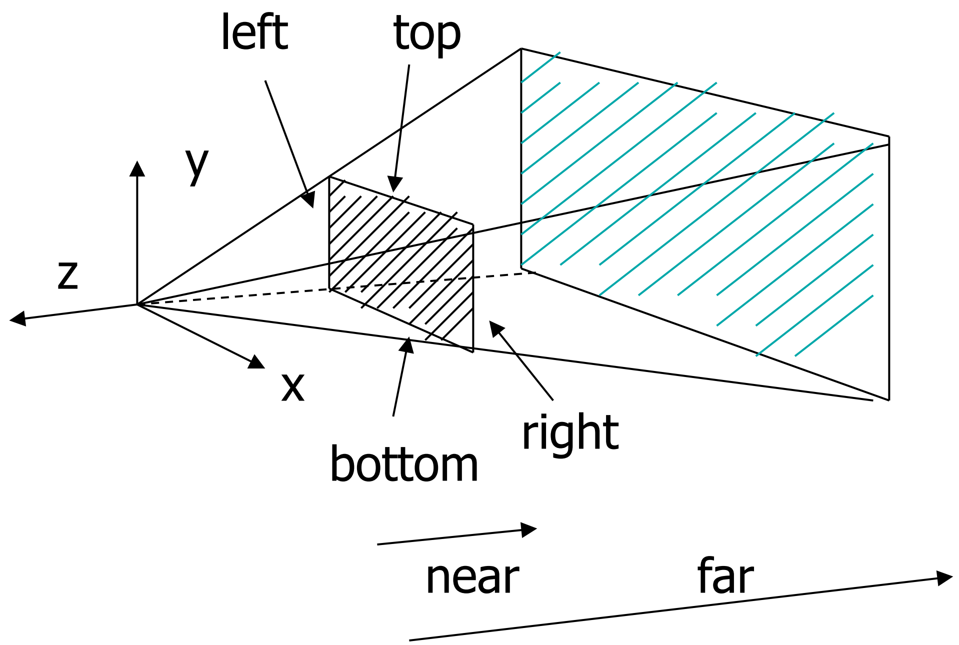
- Aspect ratio is used to calculate the window width





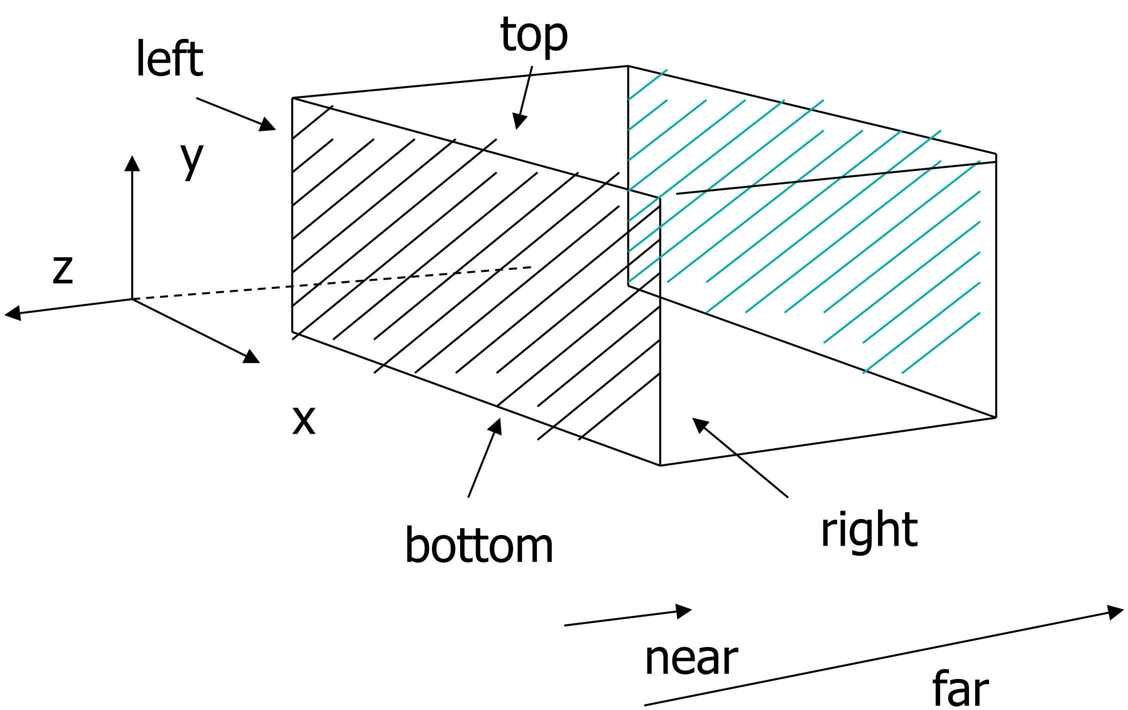
`glFrustum( left, right, bottom, top, near, far )`

- Can use this function in place of **`gluPerspective( )`**



```
glOrtho( left, right, bottom, top,
         near, far )
```

- For orthographic projection

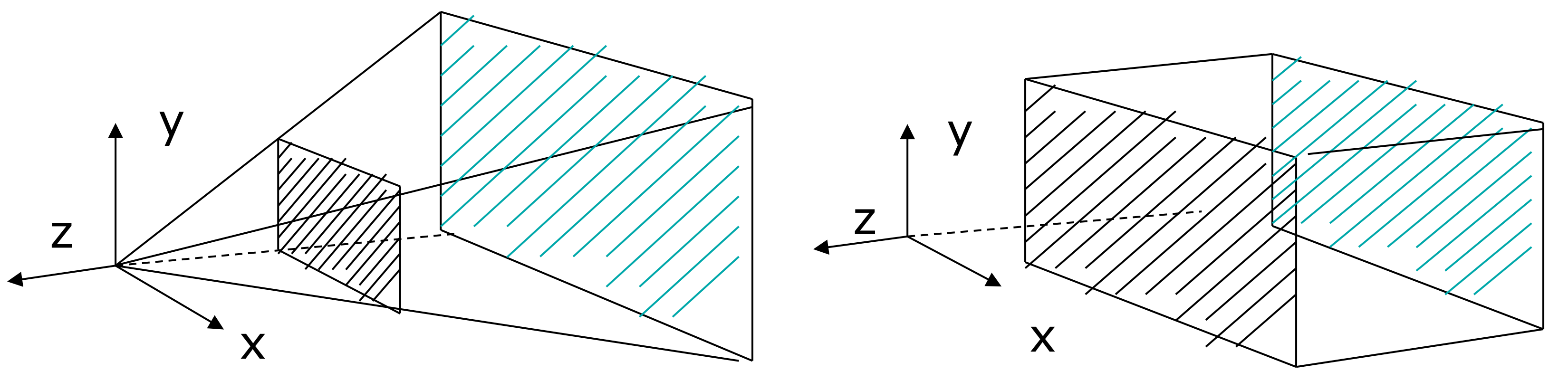


Example: Projection Transformation

```
void display( )  {
  glClear( GL_COLOR_BUFFER_BIT );
  glMatrixMode( GL_PROJECTION );
  glLoadIdentity( );
  gluPerspective( FovY, Aspect, Near, Far );
  glMatrixMode( GL_MODELVIEW );
  glLoadIdentity( );
  gluLookAt( 0, 0, 1, 0, 0, 0, 0, 1, 0 );
  myDisplay( );    // your display routine
}
```

Projection Transformation

- Projection
  - Map the object from 3D space to 2D screen

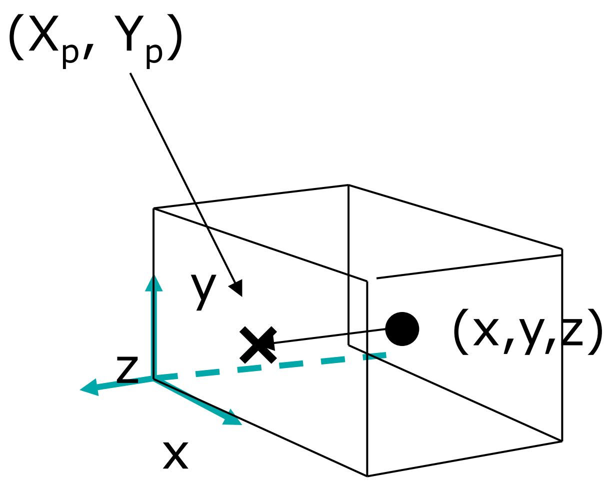


Perspective: **`gluPerspective( )`**

Parallel: **`glOrtho( )`**

Parallel Projection (The Math)

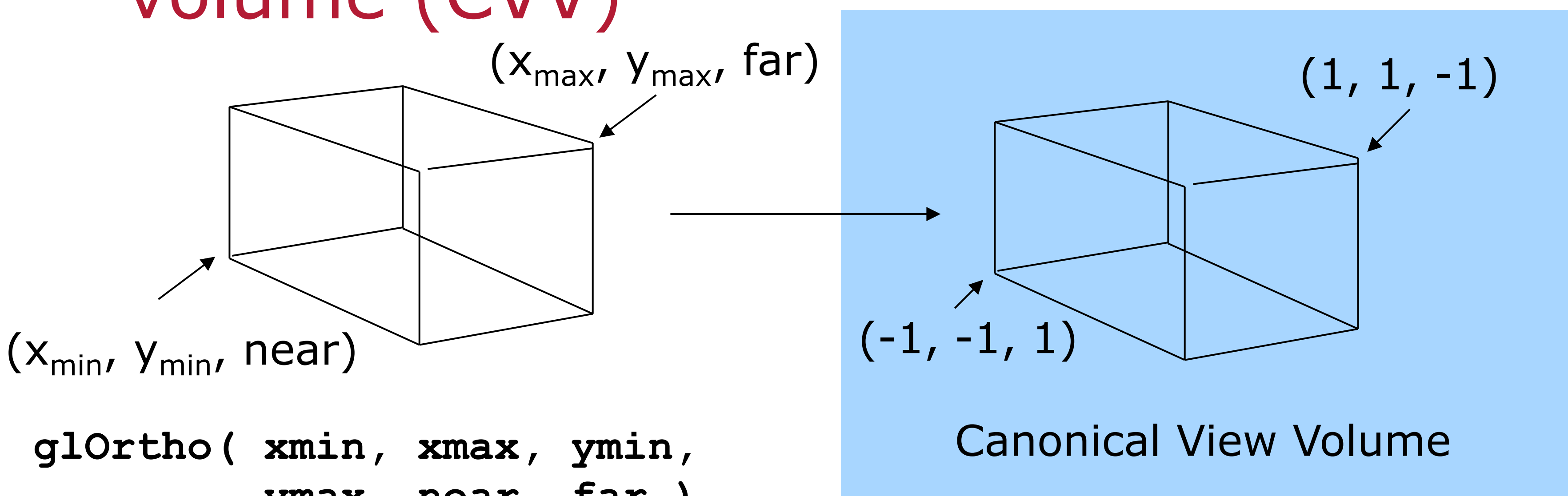
- After transforming the object to eye space, parallel projection is relatively easy: we could just set all Z to the same value
  - $X_p = x$
  - $Y_p = y$
  - $Z_p = -d$



- We actually want to remember Z
  – why?

Parallel Projection

- OpenGL maps (projects) everything in the visible volume into a canonical view volume (CVV)




```
glOrtho( xmin, xmax, ymin,
         ymax, near, far )
```

Projection: Need to build 4x4 matrix to do mapping from actual view volume to CVV

Parallel Projection: `glOrtho`

- Parallel projection can be broken down into two parts
  - Translation, which centers view volume at origin
  - Scaling, which reduces cuboid of arbitrary dimensions to canonical cube
    - Dimension 2, centered at origin

Parallel Projection: `glOrtho` (cont.)

- Translation sequence moves midpoint of view volume to coincide with origin
  - e.g., midpoint of x = $(x_{max} + x_{min})/2$
- Thus, translation factors are

  $-(x_{max}+x_{min})/2$, $-(y_{max}+y_{min})/2$, $-(far+near)/2$
- So, translation matrix M1:

$$\begin{pmatrix} 1 & 0 & 0 & -(x\max + x\min)/2 \\ 0 & 1 & 0 & -(y\max + y\min)/2 \\ 0 & 0 & 1 & -(z\max + z\min)/2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Parallel Projection: `glOrtho` (cont.)

- Scaling factor is ratio of cube dimension to Ortho view volume dimension
- Scaling factors

  $2/(x_{max}-x_{min})$, $2/(y_{max}-y_{min})$, $2/(z_{max}-z_{min})$
- So, scaling matrix M2:

$$\begin{pmatrix} \frac{2}{x\max - x\min} & 0 & 0 & 0 \\ 0 & \frac{2}{y\max - y\min} & 0 & 0 \\ 0 & 0 & \frac{2}{z\max - z\min} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Parallel Projection: `glOrtho()` (cont.)

- Concatenating M1xM2, we get transform matrix used by **`glOrtho`**

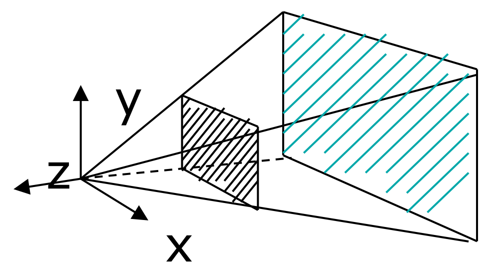
$$\begin{pmatrix} \frac{2}{x\max - x\min} & 0 & 0 & 0 \\ 0 & \frac{2}{y\max - y\min} & 0 & 0 \\ 0 & 0 & \frac{2}{z\max - z\min} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & -(x\max + x\min)/2 \\ 0 & 1 & 0 & -(y\max + y\min)/2 \\ 0 & 0 & 1 & -(z\max + z\min)/2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

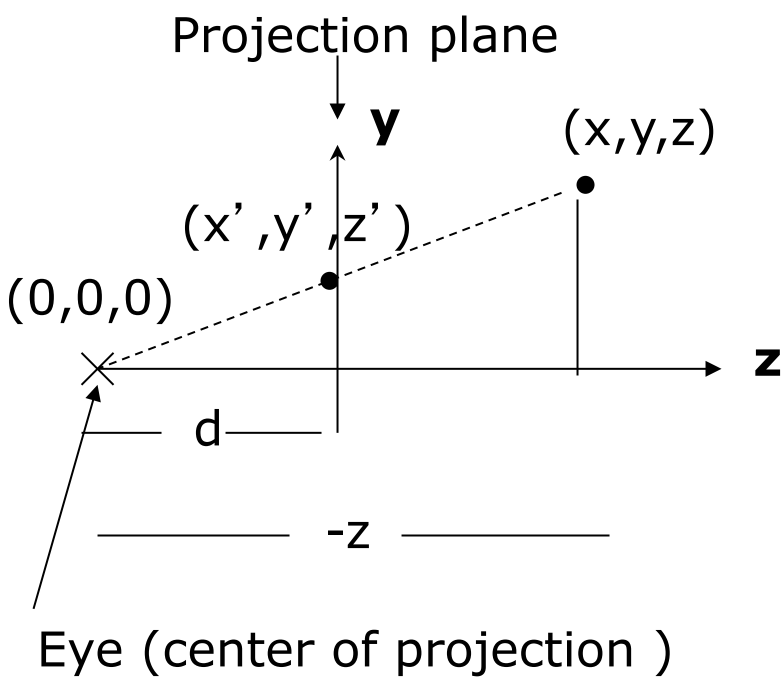
$$M2 \times M1 = \begin{pmatrix} 2/(x\max - x\min) & 0 & 0 & -(x\max + x\min)/(x\max - x\min) \\ 0 & 2/(y\max - y\min) & 0 & -(y\max + y\min)/(y\max - y\min) \\ 0 & 0 & 2/(z\max - z\min) & -(z\max + z\min)/(z\max - z\min) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Refer to: Hill, 7.6.2

Perspective Projection: Classical

- Side view





Based on similar triangles:

$$\frac{y}{y'} = \frac{-z}{d}$$

$$\Rightarrow \quad y' = y * \frac{d}{-z}$$

Perspective Projection: Classical (cont.)

- So (x*, y*), the projection of point, (x, y, z) onto the near plane N, is given as

$$\left(x^*, y^*\right) = \left(N\frac{P_x}{-P_z}, N\frac{P_y}{-P_z}\right)$$

  - Similar triangles
- Numerical example

Q: Where on the viewplane does P = (1, 0.5, -1.5) lie for a near plane at N = 1?

(x*, y*) = (1 x 1/1.5, 1 x 0.5/1.5) = (0.666, 0.333)

Pseudo Depth Checking

- Classical perspective projection drops z coordinates
- But we ***need*** z to find closest object (depth testing)
- Keeping actual distance of P from eye is cumbersome and slow

$$distance = \sqrt{\left(P_x^{\,2} + P_y^{\,2} + P_z^{\,2}\right)}$$

- Introduce ***pseudodepth***: all we need is a measure of which objects are further if two points project to the same (x, y)

$$\left(x^*, y^*, z^*\right) = \left(N\frac{P_x}{-P_z}, N\frac{P_y}{-P_z}, \frac{aP_z + b}{-P_z}\right)$$

- Choose a, b so that pseudodepth varies from –1 to 1 (canonical cube)

Pseudo Depth Checking (cont.)

- Solving:

$$z^* = \frac{aP_z + b}{-P_z}$$

- For two conditions, $z^* = -1$ when $P_z = -N$ and $z^* = 1$ when $P_z = -F$, we can set up two simultaneous equations
- Solving for *a* and *b*, we get

$$a = \frac{-(F+N)}{F-N} \qquad b = \frac{-2FN}{F-N}$$

Homogenous Coordinates

- Would like to express projection as 4x4 transform matrix
- Previously, homogeneous coordinates for the point P = (P_x, P_y, P_z) was (P_x, P_y, P_z, 1)
- Introduce arbitrary scaling factor, w, so that P = (wP_x, wP_y, wP_z, w) (Note: w is non-zero)
- For example, the point P = (2, 4, 6) can be expressed as
  - (2, 4, 6, 1)
  - or (4, 8, 12, 2) where w=2
  - or (6, 12, 18, 3) where w = 3
- So, to convert from homogeneous back to ordinary coordinates, divide all four terms by last component and discard 4th term

Perspective Projection

- Same for x, so we have

  x' = x * d / -z
  y' = y * d / -z
  z' = -d

- Put in a matrix form

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \left(1/_{-d}\right) & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ w \end{pmatrix} \Rightarrow \begin{pmatrix} -d\left(x/_{z}\right) \\ -d\left(y/_{z}\right) \\ -d \\ 1 \end{pmatrix}$$

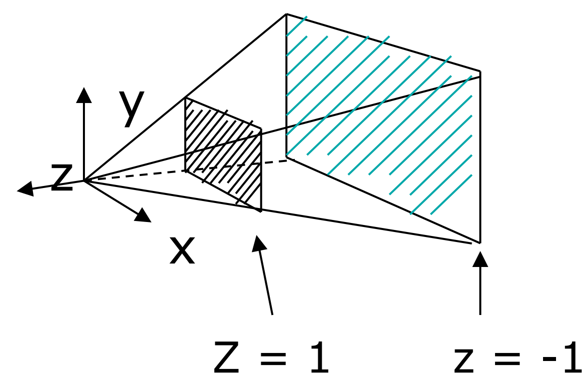
OpenGL assumes d = 1, *i.e.*, the image plane is at z = -1

Perspective Projection (cont.)

- We are not done yet!

- Need to modify the projection matrix to include a and b

$$\begin{vmatrix} x' \\ y' \\ z' \\ w \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & (1/-d) & 0 \end{vmatrix} \begin{vmatrix} x \\ y \\ z \\ 1 \end{vmatrix}$$



- We have already solved *a* and *b*

Perspective Projection (cont.)

- Not done yet! OpenGL also normalizes the x and y ranges of the view frustum to [-1, 1] (translate and scale)
- So, as in ortho, to arrive at final projection matrix
  - We translate by
    - –(xmax + xmin)/2 in x
    - -(ymax + ymin)/2 in y
  - And scale by
    - 2/(xmax – xmin) in x
    - 2/(ymax – ymin) in y

Perspective Projection (cont.)

- Final projection matrix

```
glFrustum( xmin, xmax, ymin, ymax, N, F )
```

  - N = near plane, F = far plane

$$\begin{pmatrix} \frac{2N}{x\max - x\min} & 0 & \frac{x\max + x\min}{x\max - x\min} & 0 \\ 0 & \frac{2N}{y\max - y\min} & \frac{y\max + y\min}{y\max - y\min} & 0 \\ 0 & 0 & \frac{-(F+N)}{F-N} & \frac{-2FN}{F-N} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Perspective Projection (cont.)

- After perspective projection, viewing frustum is also projected into a canonical view volume (like in parallel projection)