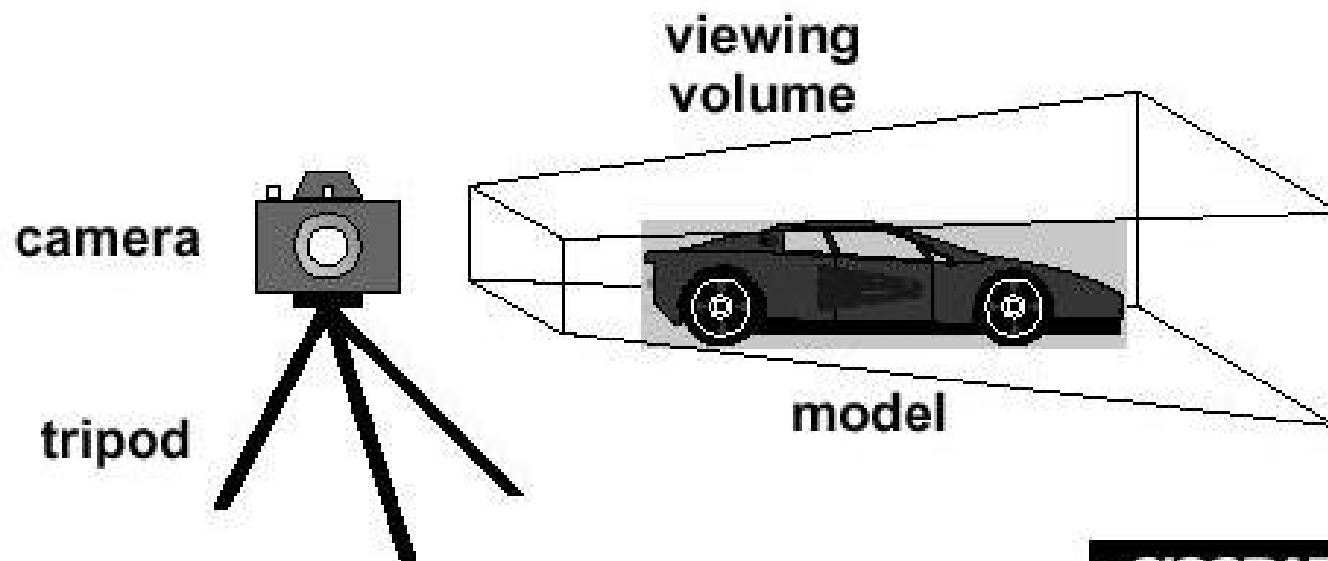


CS 4731/543: Computer Graphics
Lecture 5 (Part I): Projection

Emmanuel Agu

3D Viewing and View Volume

- Recall: 3D viewing set up

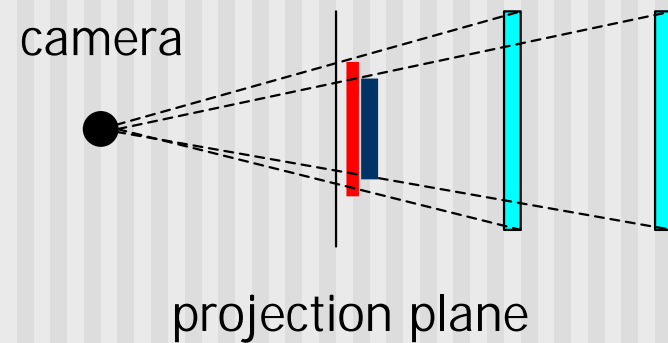


Projection Transformation

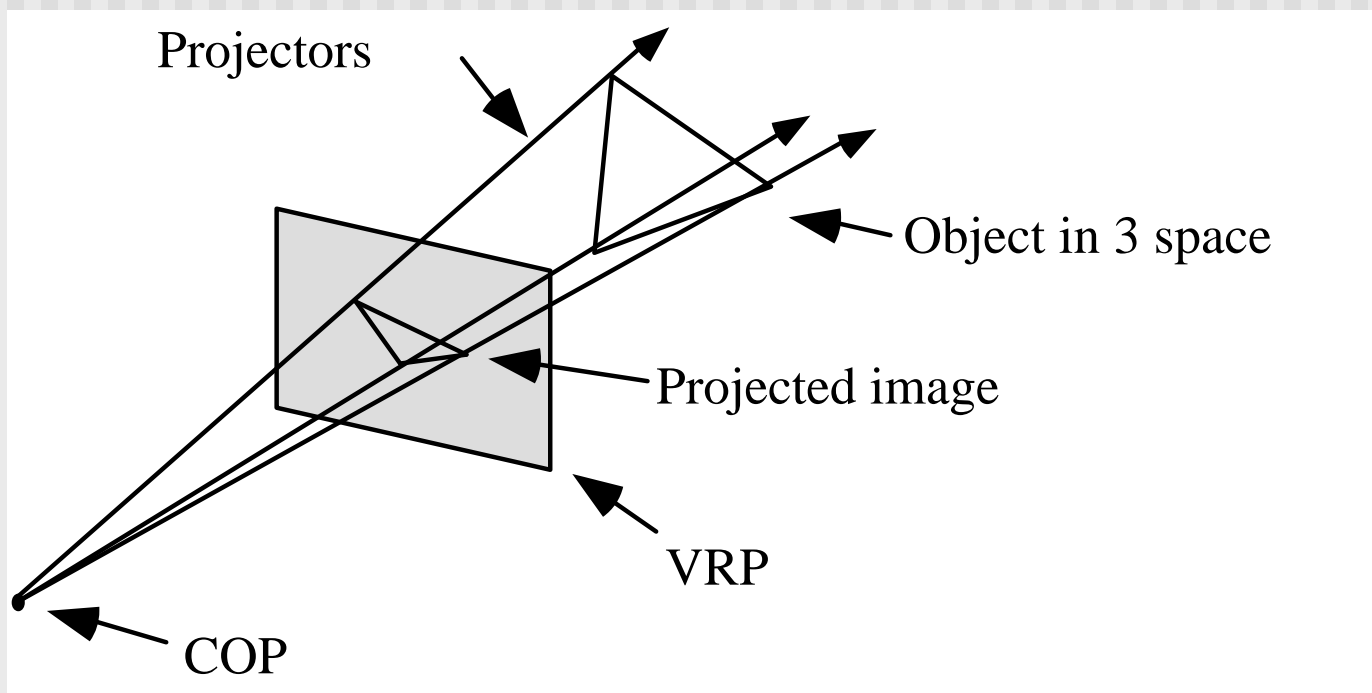
- View volume can have different shapes (different looks)
- Different types of projection: parallel, perspective, orthographic, etc
- Important to control
 - Projection type: perspective or orthographic, etc.
 - Field of view and image aspect ratio
 - Near and far clipping planes

Perspective Projection

- Similar to real world
- Characterized by **object foreshortening**
- Objects appear larger if they are closer to camera
- Need:
 - Projection center
 - Projection plane
- Projection: Connecting the object to the projection center

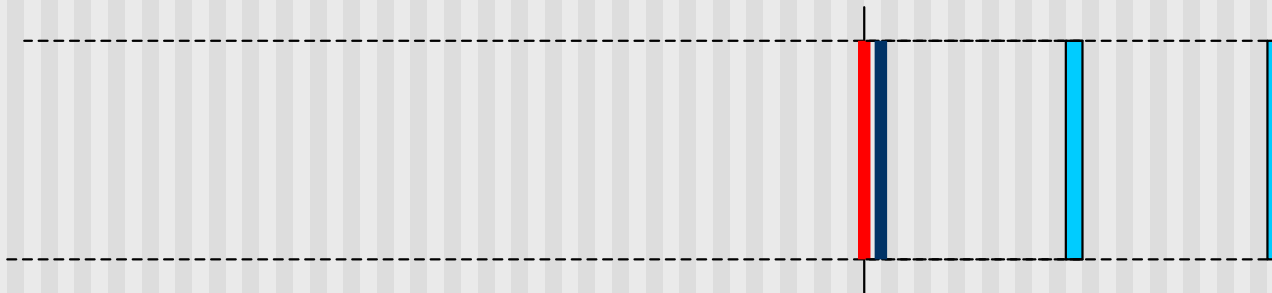


Projection?



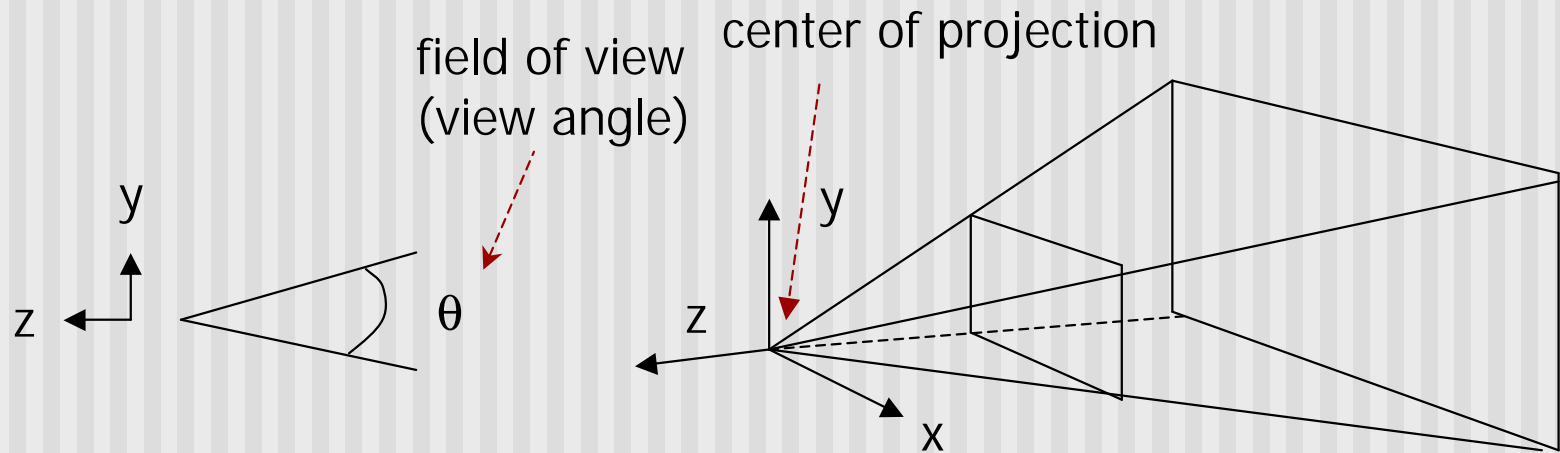
Orthographic Projection

- No foreshortening effect – distance from camera does not matter
- The projection center is at infinite
- Projection calculation – just drop z coordinates



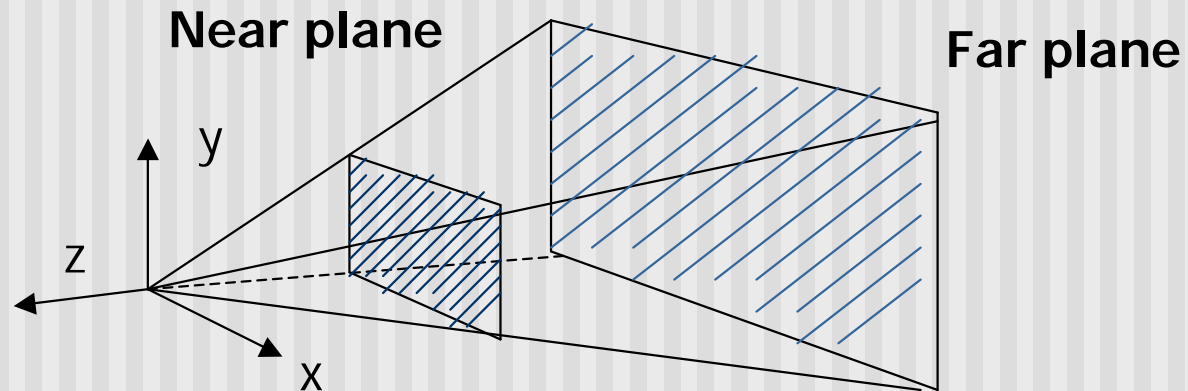
Field of View

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



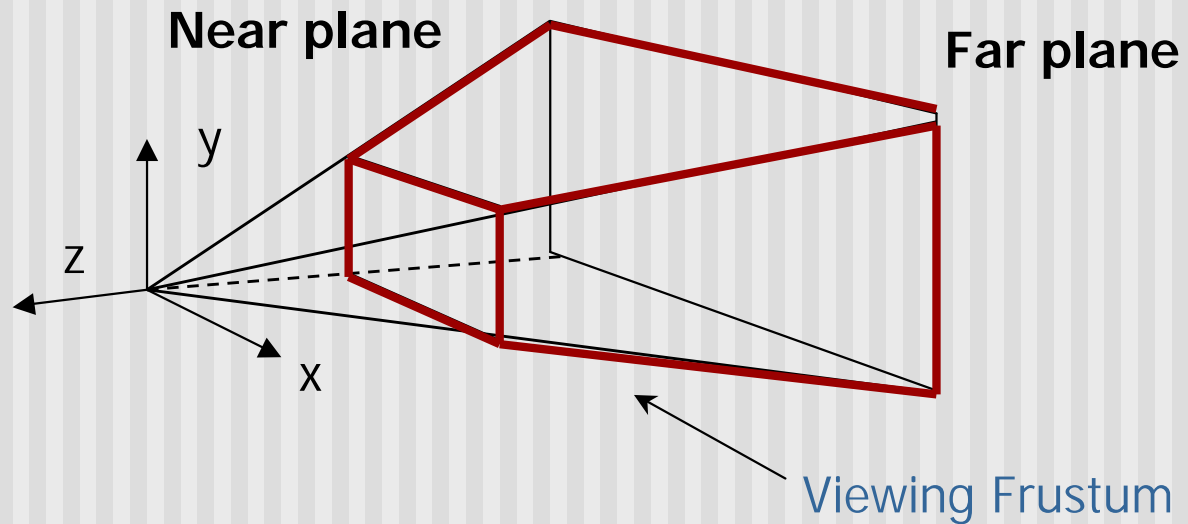
Near and Far Clipping Planes

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



Viewing Frustum

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

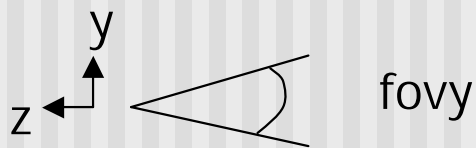


Projection Transformation

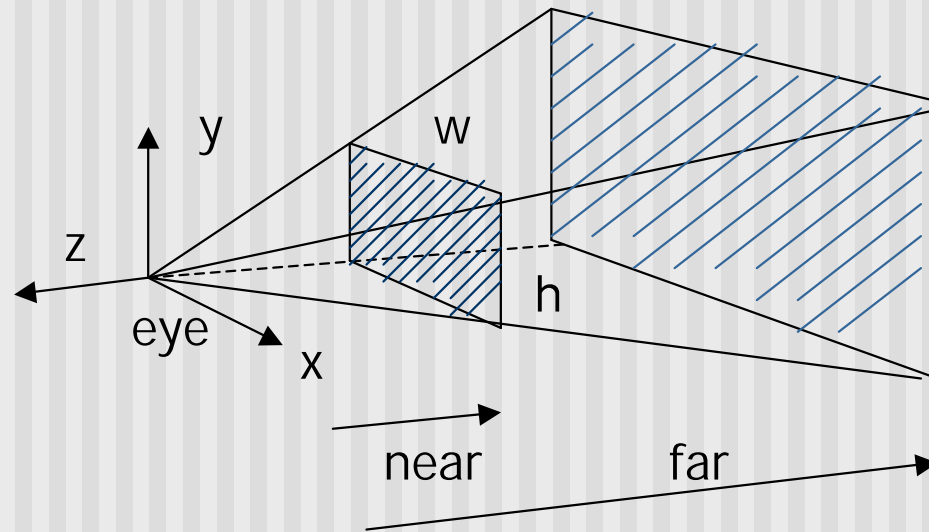
- In OpenGL:
 - Set the matrix mode to `GL_PROJECTION`
 - Perspective projection: use
 - `gluPerspective(fovy, aspect, near, far)` **or**
 - `glFrustum(left, right, bottom, top, near, far)`
 - Orthographic:
 - `glOrtho(left, right, bottom, top, near, far)`

gluPerspective(fovy, aspect, near, far)

- Aspect ratio is used to calculate the window width

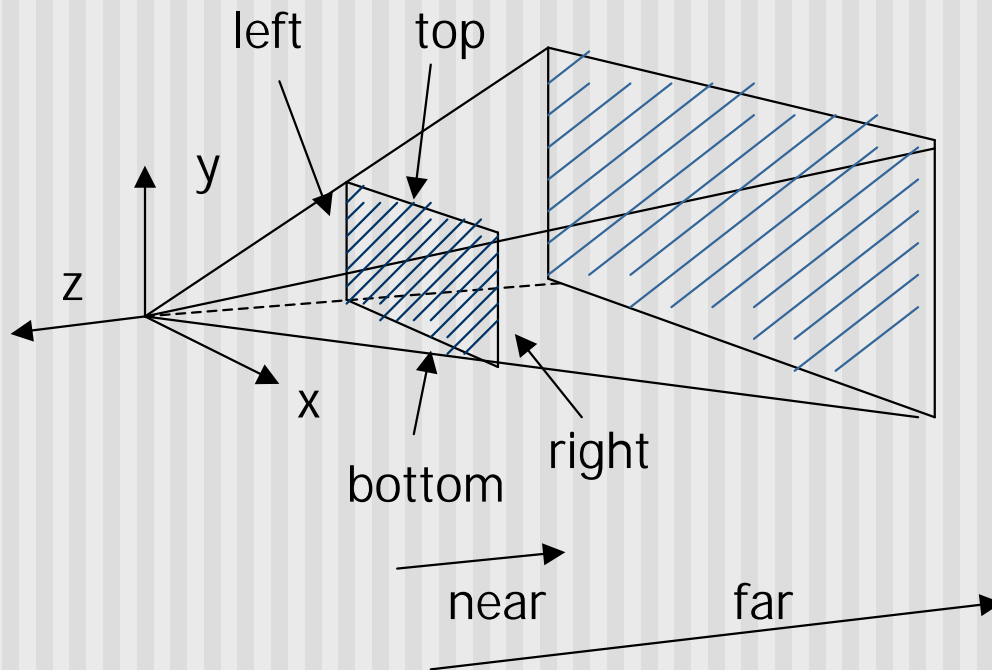


$$\text{Aspect} = w / h$$



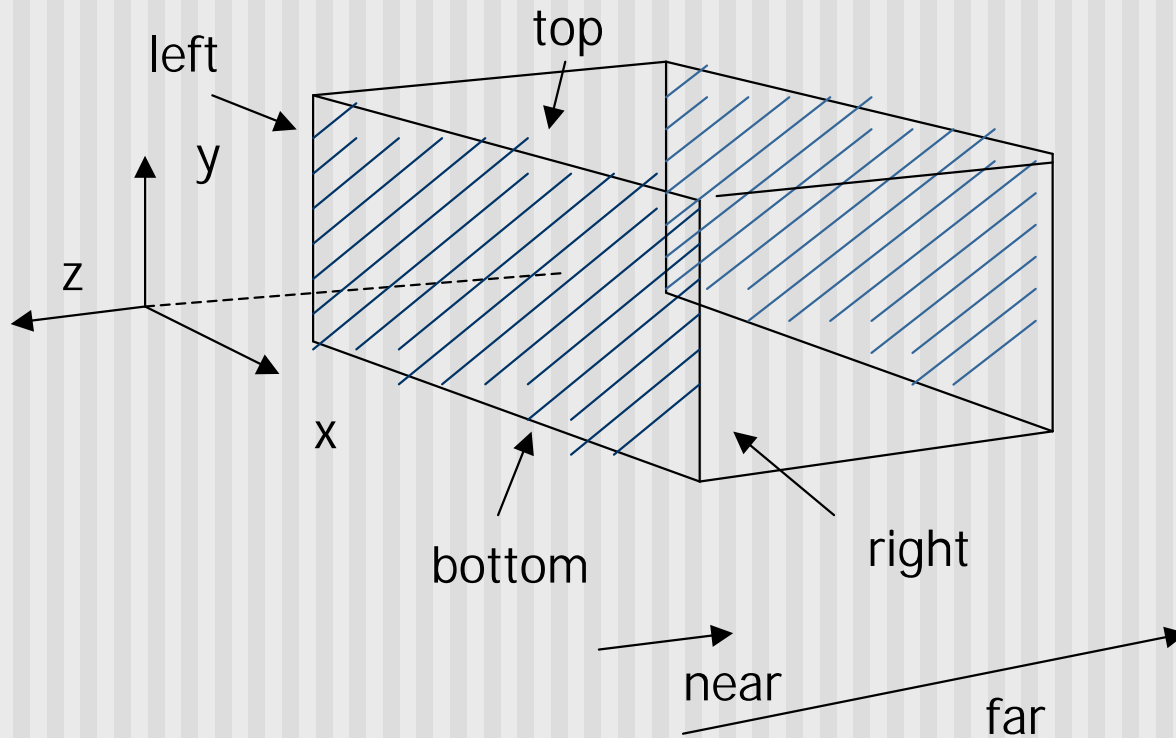
`glFrustum(left, right, bottom, top, near, far)`

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



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

- For orthographic projection



Example: Projection Transformation

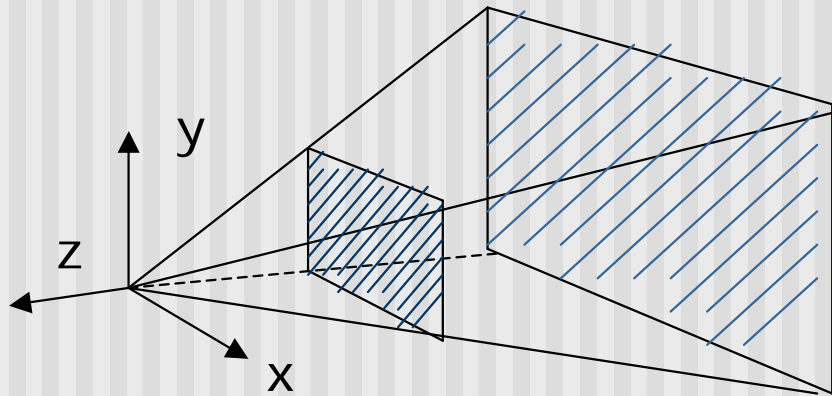
```
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(fovy, aspect, near, far);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(0,0,1,0,0,0,0,1,0);
    display_all();    // your display routine
}
```

Demo

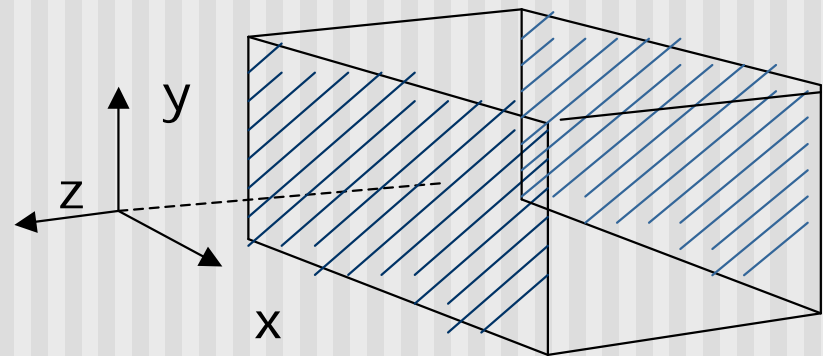
- Nate Robbins demo on projection

Projection Transformation

- Projection – map the object from 3D space to 2D screen



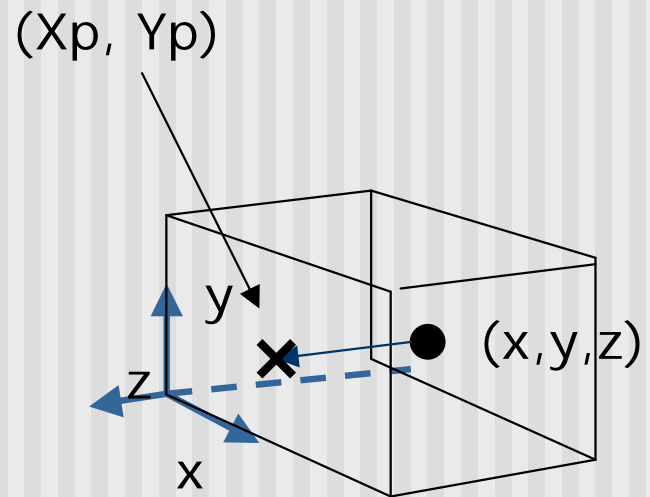
Perspective: **gluPerspective()**



Parallel: **glOrtho()**

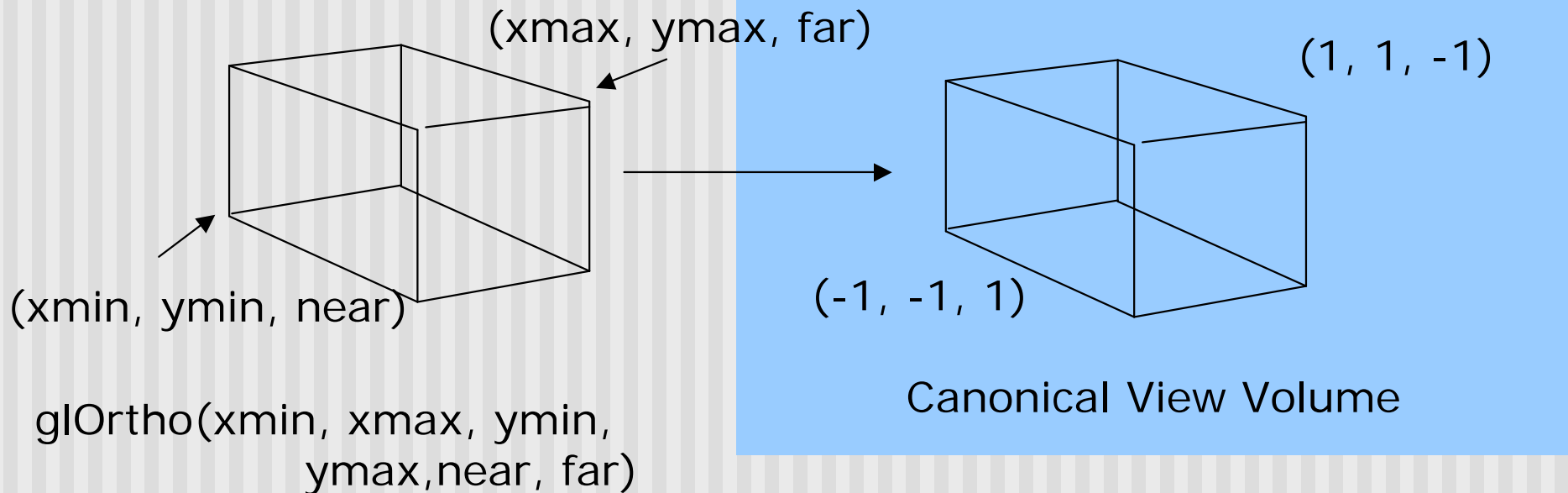
Parallel Projection

- After transforming the object to the eye space, parallel projection is relatively easy – we could just drop the Z
 - $X_p = x$
 - $Y_p = y$
 - $Z_p = -d$
- We actually want to keep Z
 - why?



Parallel Projection

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



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

- Translation sequence moves midpoint of view volume to coincide with origin:
- E.g. midpoint of $x = (x_{\max} + x_{\min})/2$
- Thus translation factors:
- $(x_{\max} + x_{\min})/2$, - $(y_{\max} + y_{\min})/2$, - $(z_{\max} + z_{\min})/2$
- And 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

- Scaling factor is ratio of cube dimension to Ortho view volume dimension
- Scaling factors:
 $2/(x_{\max}-x_{\min}), 2/(y_{\max}-y_{\min}), 2/(z_{\max}-z_{\min})$
- 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

Concatenating $M1 \times M2$, we get transform matrix used by glOrtho

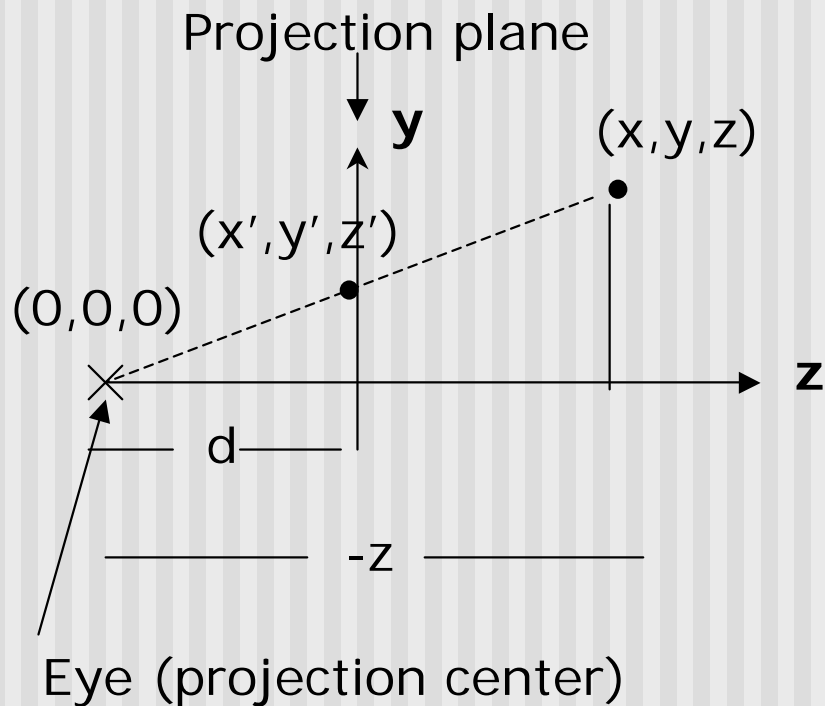
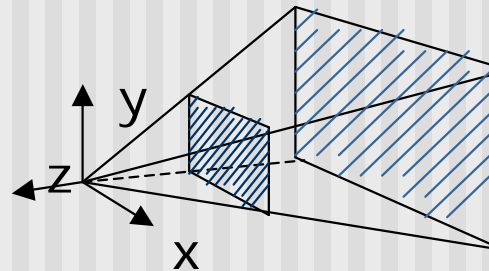
$$\begin{pmatrix} \frac{2}{x \max - x \min} & 0 & 0 & 0 \\ 0 & \frac{2}{y \max - y \min} & 0 & 0 \\ 0 & 0 & \frac{2}{z \max - z \min} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & -(x \max + x \min) / 2 \\ 0 & 1 & 0 & -(y \max + y \min) / 2 \\ 0 & 0 & 1 & -(z \max + z \min) /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

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

Refer: Hill, 7.6.2

Perspective Projection: Classical

- Side view:



Based on similar triangle:

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

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

Perspective Projection: Classical

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

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

- Numerical example:

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

- $(x^*, y^*) = (1 \times 1/1.5, 1 \times 0.5/1.5) = (0.666, 0.333)$

Pseudodepth

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

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

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

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

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

Pseudodepth

- 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:

$$a = \frac{-(F + N)}{F - N}$$

$$b = \frac{-2FN}{F - N}$$

Homogenous Coordinates

- Would like to express projection as 4x4 transform matrix
- Previously, homogeneous coordinates of 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:

$$x' = x \times d / -z$$

$$y' = y \times d / -z$$

$$z' = -d$$

- Put in a matrix form:

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

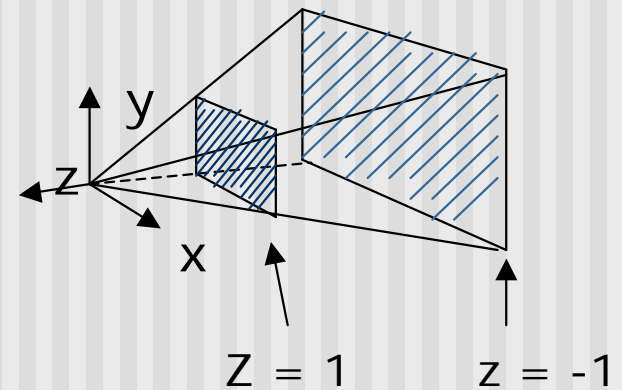
OpenGL assumes $d = 1$, i.e. the image plane is at $z = -1$

Perspective Projection

- We are not done yet.
- Need to modify the projection matrix to include a and b

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

We have already solved a and b



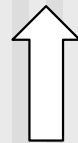
Perspective Projection

- Not done yet. OpenGL also normalizes the x and y ranges of the viewing frustum to $[-1, 1]$ (translate and scale)
- So, as in ortho to arrive at final projection matrix
- we translate by
 - $-(x_{\max} + x_{\min})/2$ in x
 - $-(y_{\max} + y_{\min})/2$ in y
- Scale by:
 - $2/(x_{\max} - x_{\min})$ in x
 - $2/(y_{\max} - y_{\min})$ in y

Perspective Projection

- Final Projection Matrix:

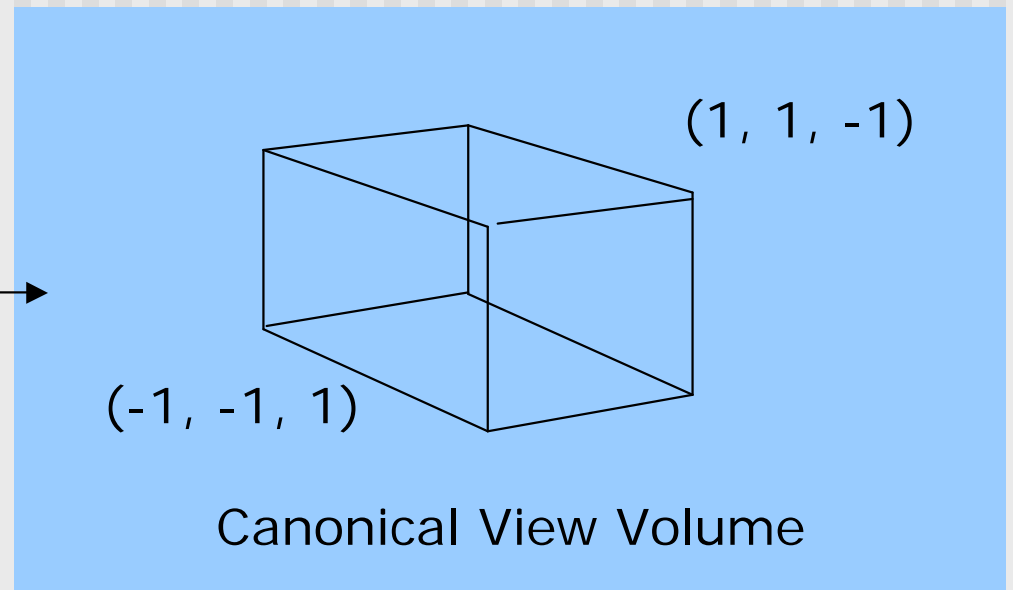
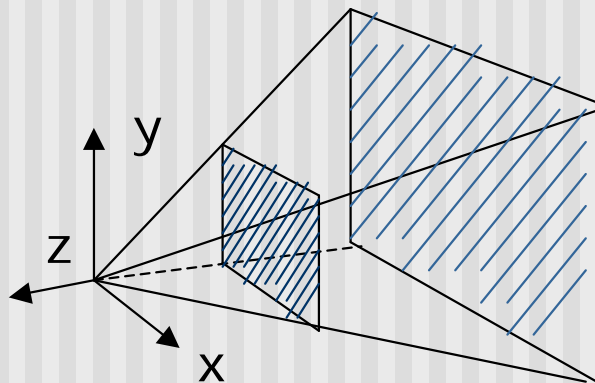
$$\begin{pmatrix} \frac{2N}{x_{\max} - x_{\min}} & 0 & \frac{x_{\max} + x_{\min}}{x_{\max} - x_{\min}} & 0 \\ 0 & \frac{2N}{y_{\max} - y_{\min}} & \frac{y_{\max} + y_{\min}}{y_{\max} - y_{\min}} & 0 \\ 0 & 0 & \frac{-(F + N)}{F - N} & \frac{-2FN}{F - N} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$



`glFrustum(xmin, xmax, ymin, ymax, N, F)` N = near plane, F = far plane

Perspective Projection

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



References

- Hill, chapter 7