

15-462 Computer Graphics I  
Lecture 5

## Viewing and Projection

Shear Transformation  
Camera Positioning  
Simple Parallel Projections  
Simple Perspective Projections  
[Angel, Ch. 5.2-5.4]  
[Red's Dream, Pixar, 1987]

January 30, 2003  
Frank Pfenning  
Carnegie Mellon University

<http://www.cs.cmu.edu/~fp/courses/graphics/>

## Transformation Matrices in OpenGL

- Transformation matrices in OpenGL are vectors of 16 values (column-major matrices)
- In `glLoadMatrixf(GLfloat *m);`

$m = \{m_1, m_2, \dots, m_{16}\}$  represents

$$M = \begin{bmatrix} m_1 & m_5 & m_9 & m_{13} \\ m_2 & m_6 & m_{10} & m_{14} \\ m_3 & m_7 & m_{11} & m_{15} \\ m_4 & m_8 & m_{12} & m_{16} \end{bmatrix}$$

- Some books transpose all matrices!

## Pondering Transformations

- Derive transformation given some parameters
  - Choose parameters carefully
  - Consider geometric intuition, basic trigonometry
- Compose transformation from others
  - Use translations to and from origin
- Test if matrix describes some transformation
  - Determine action on basis vectors
- Meaning of dot product and cross product

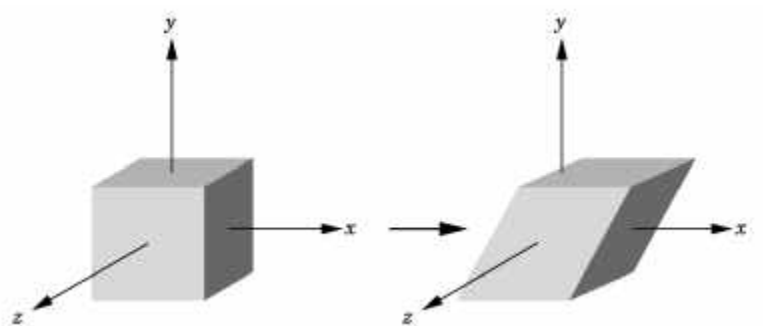
01/30/2003

15-462 Graphics I

3

## Shear Transformations

- x-shear scales x proportional to y
- Leaves y and z values fixed



01/30/2003

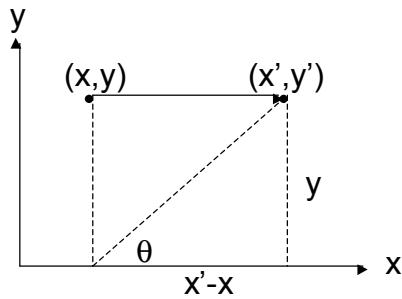
15-462 Graphics I

4

## Specification via Angle

- $\cot(\theta) = (x'-x)/y$
- $x' = x + y \cot(\theta)$
- $y' = y$
- $z' = z$

$$H_x(\theta) = \begin{bmatrix} 1 & \cot(\theta) & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



01/30/2003

15-462 Graphics I

5

## Specification via Ratios

- Shear in both x and z direction
- Leave y fixed
- Slope  $\alpha$  for x-shear,  $\gamma$  for z-shear

• Solve

$$H_{xz}(\alpha, \gamma) \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + \alpha y \\ y \\ z + \gamma y \\ 1 \end{bmatrix}$$

- Yields

$$H_{xz}(\alpha, \gamma) = \begin{bmatrix} 1 & \alpha & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \gamma & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

01/30/2003

15-462 Graphics I

6

## Composing Transformations

- Every affine transformation is a composition of rotations, scalings, and translations
- How do we compose these to form an x-shear?
- Exercise!

01/30/2003

15-462 Graphics I

7

## Thinking in Frames

- Action on frame determines affine transfn.
- Frame given by basis vectors and origin
- xz-shear: preserve basis vectors  $u_x$  and  $u_z$

$$M \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad M \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

- Move  $u_y = [0 \ 1 \ 0 \ 0]^T$   
to  $u'_y = [\alpha \ 1 \ \gamma \ 0]^T$

$$M \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \alpha \\ 1 \\ \gamma \\ 0 \end{bmatrix}$$

01/30/2003

15-462 Graphics I

8

## Preservation of Origin

- Preserve origin  $P_0$

$$M \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

- Results comprise columns of the transfn. matrix

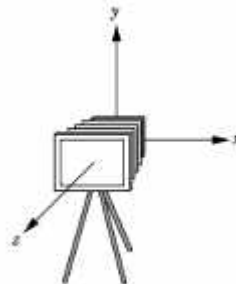
$$H_{xz}(\alpha, \gamma) = \begin{bmatrix} 1 & \alpha & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \gamma & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

## Outline

- Shear Transformation
- Camera Positioning
- Simple Parallel Projections
- Simple Perspective Projections

## Camera in Modeling Coordinates

- Camera position is identified with a frame
- Either move and rotate the objects
- Or move and rotate the camera
- Initially, pointing in negative z-direction
- Initially, camera at origin



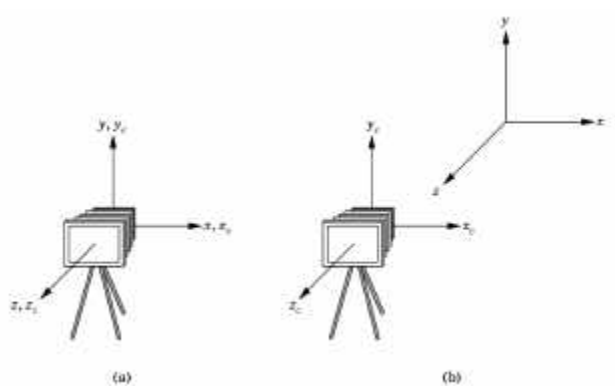
01/30/2003

15-462 Graphics I

11

## Moving Camera and World Frame

- Move world frame relative to camera frame
- `glTranslatef(0.0, 0.0, -d);` moves world frame



01/30/2003

15-462 Graphics I

12

## Order of Viewing Transformations

- Think of moving the world frame
- Viewing transfn. is inverse of object transfn.
- Order opposite to object transformations

```
glMatrixMode(GL_MODELVIEW);  
glLoadIdentity();  
glTranslatef(0.0, 0.0, -d); /*T*/  
glRotatef(-90.0, 0.0, 1.0, 0.0); /*R*/
```



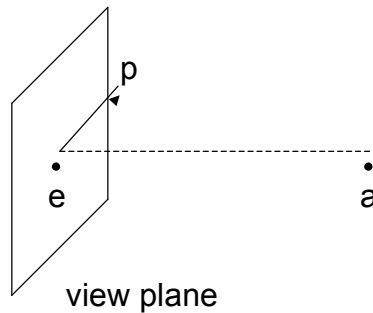
01/30/2003

15-462 Graphics I

13

## The Look-At Function

- Convenient way to position camera
- $\text{gluLookAt}(e_x, e_y, e_z, a_x, a_y, a_z, p_x, p_y, p_z);$
- $e$  = eye point
- $a$  = at point
- $p$  = up vector



01/30/2003

15-462 Graphics I

14

## Implementing the Look-At Function

- (1) Transform world frame to camera frame
- Compose a rotation  $R$  with translation  $T$
- $W = T R$
- (2) Invert  $W$  to obtain viewing transformation  $V$
- $V = W^{-1} = (T R)^{-1} = R^{-1} T^{-1}$
- Derive  $R$ , then  $T$ , then  $R^{-1} T^{-1}$

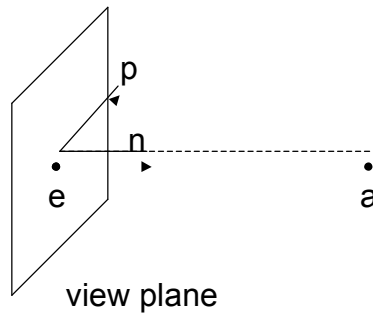
01/30/2003

15-462 Graphics I

15

## World Frame to Camera Frame I

- Camera points in negative  $z$  direction
- $n = (a - e) / |a - e|$  is unit normal to view plane
- $R$  maps  $[0 \ 0 \ -1 \ 0]^T$  to  $[n_x \ n_y \ n_z \ 0]^T$



01/30/2003

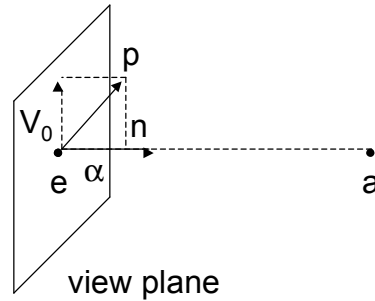
15-462 Graphics I

16



## World Frame to Camera Frame II

- $R$  maps  $y$  to projection of  $p$  onto view plane
- $\alpha = (p \cdot n) / |n| = p \cdot n$
- $v_0 = p - \alpha n$
- $v = v_0 / |v_0|$



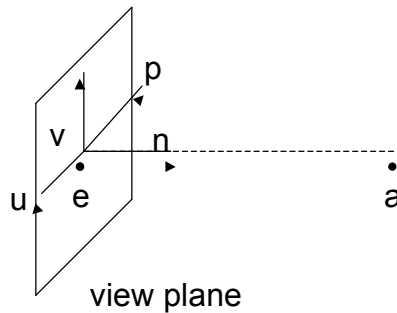
01/30/2003

15-462 Graphics I

17

## World Frame to Camera Frame III

- $x$  is orthogonal to  $n$  and  $v$  in view plane
- $u = n \times v$
- $(u, v, -n)$  is right-handed



01/30/2003

15-462 Graphics I

18

## Summary of Rotation

- `gluLookAt(ex, ey, ez, ax, ay, az, px, py, pz);`
- $n = (a - e) / |a - e|$
- $v = (p - (p \cdot n) n) / |p - (p \cdot n) n|$
- $u = n \times v$

$$R = \begin{bmatrix} u_x & v_x & -n_x & 0 \\ u_y & v_y & -n_y & 0 \\ u_z & v_z & -n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

01/30/2003

15-462 Graphics I

19

## World Frame to Camera Frame IV

- Translation of origin to  $e = [e_x \ e_y \ e_z \ 1]^T$

$$T = \begin{bmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

01/30/2003

15-462 Graphics I

20

## Camera Frame to World Frame

- $V = W^{-1} = (T R)^{-1} = R^{-1} T^{-1}$
- R is rotation, so  $R^{-1} = R^T$

$$R^{-1} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ -n_x & -n_y & -n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- T is translation, so  $T^{-1}$  negates displacement

$$T^{-1} = \begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

01/30/2003

15-462 Graphics I

21

## Putting it Together

- Calculate  $V = R^{-1} T^{-1}$

$$V = \begin{bmatrix} u_x & u_y & u_z & -u_x e_x - u_y e_y - u_z e_z \\ v_x & v_y & v_z & -v_x e_x - v_y e_y - v_z e_z \\ -n_x & -n_y & -n_z & n_x e_x + n_y e_y + n_z e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- This is different from book [Angel, Ch. 5.2.2]
- There, u, v, n are right-handed (here: u, v, -n)

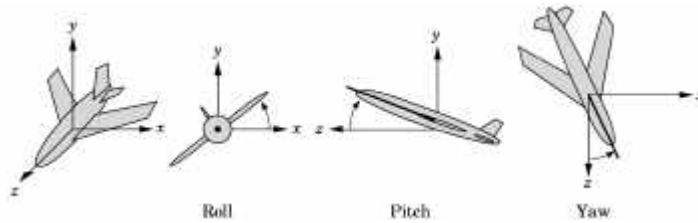
01/30/2003

15-462 Graphics I

22

## Other Viewing Functions

- Roll (about z), pitch (about x), yaw (about y)



- Assignment 2 poses related problem

01/30/2003

15-462 Graphics I

23

## Outline

- Shear Transformation
- Camera Positioning
- Simple Parallel Projections
- Simple Perspective Projections

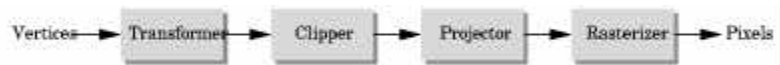
01/30/2003

15-462 Graphics I

24

## Projection Matrices

- Recall geometric pipeline



- Projection takes 3D to 2D
- Projections are not invertible
- Projections also described by matrix
- Homogenous coordinates crucial
- Parallel and perspective projections

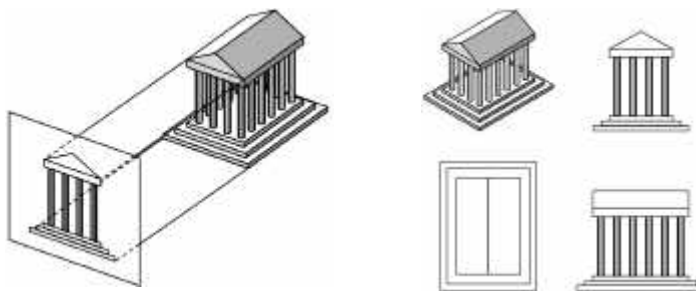
01/30/2003

15-462 Graphics I

25

## Orthographic Projections

- Parallel projection
- Projectors perpendicular to projection plane
- Simple, but not realistic
- Used in blueprints (multiview projections)



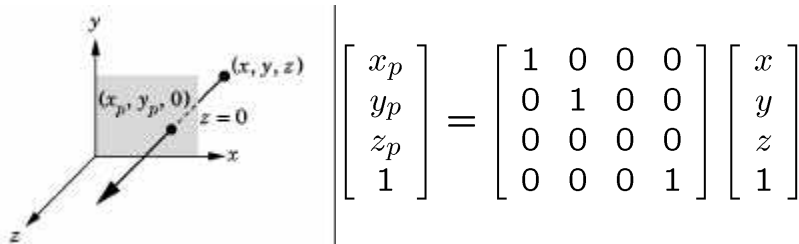
01/30/2003

15-462 Graphics I

26

## Orthographic Projection Matrix

- Project onto  $z = 0$
- $x_p = x, y_p = y, z_p = 0$
- In homogenous coordinates



01/30/2003

15-462 Graphics I

27

## Perspective

- Perspective characterized by foreshortening
- More distant objects appear smaller
- Parallel lines appear to converge
- Rudimentary perspective in cave drawings



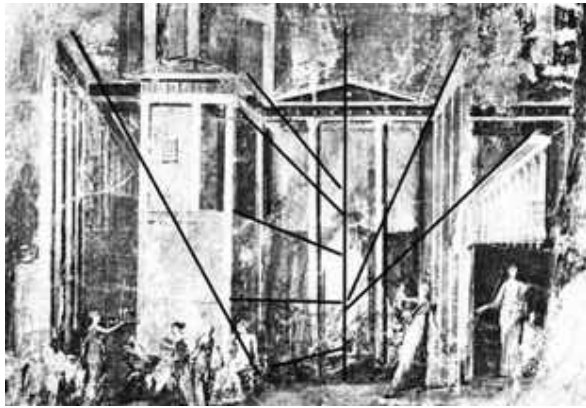
01/30/2003

15-462 Graphics I

28

## Discovery of Perspective

- Foundation in geometry (Euclid)



Mural from  
Pompeii

01/30/2003

15-462 Graphics I

29

## Middle Ages

- Art in the service of religion
- Perspective abandoned or forgotten



Ottonian manuscript,  
ca. 1000

01/30/2003

15-462 Graphics I

30

# Renaissance

- Rediscovery, systematic study of perspective



Filippo Brunelleschi  
Florence, 1415

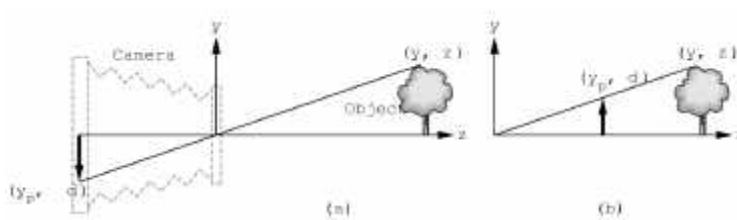
01/30/2003

15-462 Graphics I

31

# Perspective Viewing Mathematically

- More on history of perspective (icscis)  
<http://www.cyberus.ca/~icscis/icscis.htm>



- $y/z = y_p/d$  so  $y_p = y/(z/d)$
- Note this is non-linear!

01/30/2003

15-462 Graphics I

32



## Exploiting the 4<sup>th</sup> Dimension

- Perspective projection is not affine:

$$\mathbf{M} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{x}{z/d} \\ \frac{y}{z/d} \\ d \\ 1 \end{bmatrix} \text{ has no solution for } \mathbf{M}$$

- Idea: represent point  $[x \ y \ z \ 1]^T$  by line in 4D

$$\mathbf{p} = w \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \text{ for arbitrary } w \neq 0$$

01/30/2003

15-462 Graphics I

33

## Perspective Projection Matrix

- Represent multiple of point

$$(z/d) \begin{bmatrix} \frac{x}{z/d} \\ \frac{y}{z/d} \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix}$$

- Solve

$$\mathbf{M} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix} \text{ with } \mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$$

01/30/2003

15-462 Graphics I

34

## Perspective Division

- Normalize  $[x \ y \ z \ w]^T$  to  $[(x/w) \ (y/w) \ (z/w) \ 1]^T$
- Perform perspective division after projection



- Projection in OpenGL is more complex

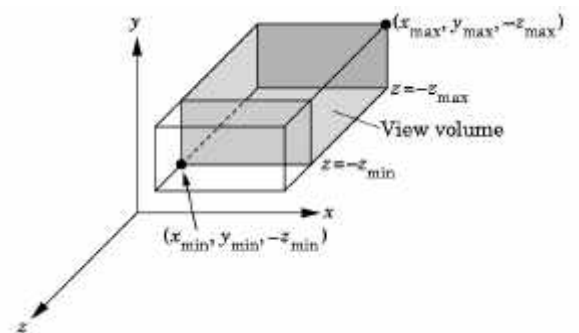
01/30/2003

15-462 Graphics I

35

## Parallel Viewing in OpenGL

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



$z_{\min} = \text{near}, z_{\max} = \text{far}$

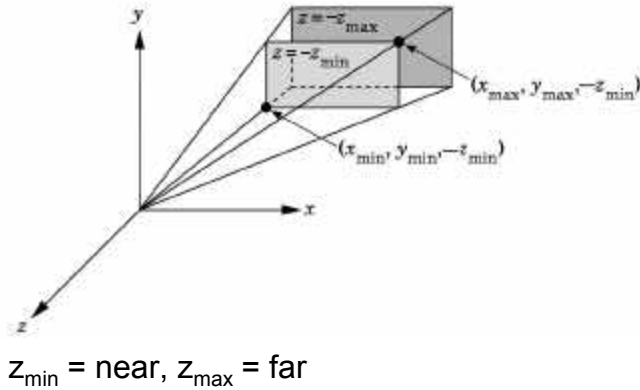
01/30/2003

15-462 Graphics I

36

## Perspective Viewing in OpenGL

- Two interfaces: glFrustum and gluPerspective
- glFrustum(xmin, xmax, ymin, ymax, near, far);



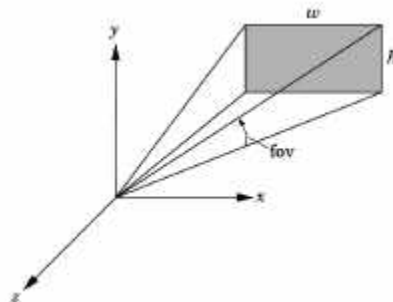
01/30/2003

15-462 Graphics I

37

## Field of View Interface

- gluPerspective(fovy, aspect, near, far);
- near and far as before
- Fovy specifies field of view as height (y) angle



01/30/2003

15-462 Graphics I

38

## Matrices for Projections in OpenGL

- Next lecture:
  - Use shear for predistortion
  - Use projections for “fake” shadows
  - Other kinds of projections

01/30/2003

15-462 Graphics I

39

## Announcements

- Assignment 1 due Thursday midnight (100 pts)
- Late policy
  - Up to 3 days any time, no penalty
  - No other late hand-in permitted
- Assignment 2 out Thursday (1 week, 50 pts)
- Extra credit policy
  - Up to 20% of assignment value
  - Recorded separately
  - Weighed for “borderline” cases
- Remember: no collaboration on assignments!

01/30/2003

15-462 Graphics I

40

## Looking Ahead

- Lighting and shading
- Video: *Red's Dream*, John Lasseter, Pixar, 1987  
<http://www.pixar.com/shorts/rd/index.html>