
HOMWORK

CSE 872 FALL 2009

1

COORDINATE SYSTEMS

Object coordinates

World coordinates

Camera(eye, image) coordinates

Perspective Coordinates

(Normalized device coordinates)

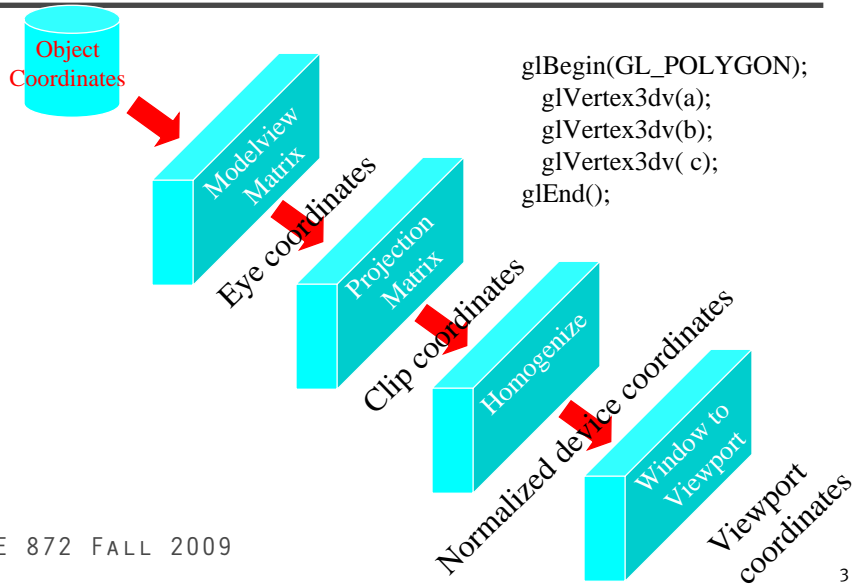
Screen Coordinates

(Window coordinates)

CSE 872 FALL 2009

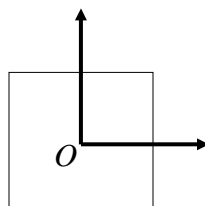
2

WITHIN OPENGL



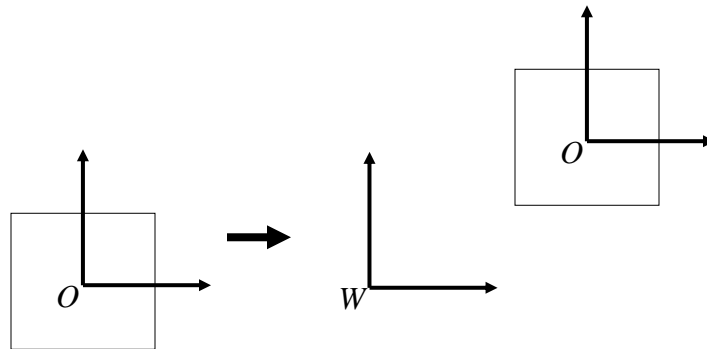
OBJECT COORDINATES

Convenient place to model the object



WORLD COORDINATES

Common coordinates for the scene

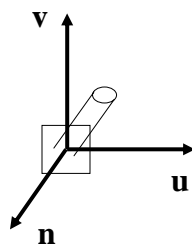


CSE 872 FALL 2009

5

CAMERA COORDINATES

Coordinate system with the camera
in a convenient pose



$$X_{iw} = \begin{bmatrix} u_x & u_y & u_z & -\mathbf{r} \cdot \mathbf{u} \\ v_x & v_y & v_z & -\mathbf{r} \cdot \mathbf{v} \\ n_x & n_y & n_z & -\mathbf{r} \cdot \mathbf{n} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

CSE 872 FALL 2009

6

CAMERA COORDINATES

$$n = \text{lookat} - \text{camera}$$

Normalize

$$v = \text{worldup} - (\text{worldup} \cdot n)n$$

Normalize

$$u = v \times n$$

NORMALIZED DEVICE COORDINATES

Device independent coordinates

Visible coordinate usually range
from: (d is focal length)

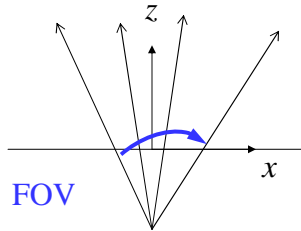
$$-1 \leq x \leq 1$$

$$-1 \leq y \leq 1$$

$$0 \leq z \leq 1$$

PERSPECTIVE PROJECTION

Taking the camera coordinates to NDC



$$\frac{1}{d} = \tan\left(\frac{FOV}{2}\right)$$

$$X_{pi} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \\ 0 & 0 & \frac{1}{d} & 1 \end{pmatrix}$$

WINDOW COORDINATES

Adjusting the NDC to fit the window

(x_0, y_0) is the lower left of the window

[In hw3, only screen coordinates are in RHC]

$$\begin{aligned} x_w &= (x_{nd} + 1) \left(\frac{\text{width}}{2} \right) + x_0 \\ y_w &= (-y_{nd} + 1) \left(\frac{\text{height}}{2} \right) + y_0 \end{aligned} \quad X_{sp} = \begin{pmatrix} w/2 & 0 & 0 & w/2 \\ 0 & -h/2 & 0 & h/2 \\ 0 & 0 & Z_{max} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

APIs

GzNewRender

- setup Xsp and anything only done once
- init default camera

GzBeginRender

- compute Xiw
- projection xform Xpi from camera definition
- init Ximage - put Xsp at base of stack, push on Xpi and Xiw

LIGHTING AND SHADING

HW4: Phong illumination

$$C = K_s \sum [I_e (R \cdot E)^s] + K_d \sum [I_e (N \cdot L)] + K_a I_a$$

■ Simplifications

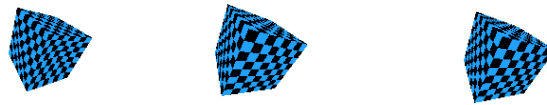
- L is constant for directional light
- E is view direction (0, 0, -1)
- N transformed by X_norm
- $R = 2(N \cdot L)N - L$

SHADING ISSUES

- Deal with backward faces
- Watch for color overflow
- Color interpolation
 - Gouraud shading
- Normal interpolation
 - Phong shading
- Xnorm--- Xsp and Xpi == Id
 - Rotation only

SCREEN-SPACE INTERPOLATION CORRECTION

- `glHint(gl_perspective_correction_hint,...)`



Correction every 16 pixel

[Mikael Kalms]

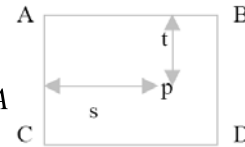
- Interpolation in screen space
 - Pixels are linearly interpolated
 - Parameters should not...(division is costly tho)
 - u, v texture coords, colors, normals, ...

$$P_s = P / \left(\frac{Z_s}{Z_{max} - Z_s} + 1 \right) \quad P = P_s \left(\frac{Z_s}{Z_{max} - Z_s} + 1 \right)$$

TEXTURE MAPPING (HW5)

■ Bilinear interpolation

$$p = stD + (1-s)tC + s(1-t)B + (1-s)(1-t)A$$



■ Modulate color by texture

$$C = K_{texture}(K_s \sum [I_e(R \cdot E)^s] + K_d \sum [I_e(N \cdot L)] + K_a I_a)$$