Coordinate Systems

Object coordinates
World coordinates
Camera(eye, image) coordinates
Perspective Coordinates
   (Normalized device coordinates)
Screen Coordinates
   (Window coordinates)
**Withi OpenGL**

Object Coordinates

Convenient place to model the object

```
glBegin(GL_POLYGON);
glVertex3dv(a);
glVertex3dv(b);
glVertex3dv(c);
glEnd();
```
**World Coordinates**

Common coordinates for the scene

![Diagram of world coordinates]

**Camera Coordinates**

Coordinate system with the camera in a convenient pose

\[
\begin{bmatrix}
X_{iw} = \\
ux & uy & uz & -\mathbf{r} \cdot \mathbf{u} \\
vx & vy & vz & -\mathbf{r} \cdot \mathbf{v} \\
nx & ny & nz & -\mathbf{r} \cdot \mathbf{n} \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
**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 hw2, only screen coordinates are in RHC]

\[
x_w = (x_{nd} + 1)\left(\frac{\text{width}}{2}\right) + x_0
\]

\[
y_w = (-y_{nd} + 1)\left(\frac{\text{height}}{2}\right) + y_c
\]

\[
X_{sp} = \begin{pmatrix}
w/2 & 0 & 0 & w/2 \\
0 & -h/2 & 0 & h/2 \\
0 & 0 & Z_{max} & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]
APIs

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

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

Lighting and Shading

Step 2: 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
- Interpolation in screen space
  - Pixels are linearly interpolated
  - Parameters should not... (division is costly tho)
  - u, v texture coords, colors, normals, ...

\[
P_S = P/(\frac{Z_S}{Z_{max} - Z_S} + 1)\quad P = P_S(\frac{Z_S}{Z_{max} - Z_S} + 1)
\]
Texture Mapping (Step 3)

- Bilinear interpolation

\[ p = stD + (1-s)tC + s(1-t)B + (1-s)(1-t)A \]

- Modulate color by texture

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