Chapter 5
Viewing/Perspective Transformations

### Computer Graphics

Rendering Pipeline

- **Scene graph Object geometry**
- **Modeling Transform**
- **Viewing Transform**
- **Projection Transform**

- **Result**: All vertices of scene in shared 3D world coordinate system

- **Specify view point (change of coordinate system)**
- **Project from 3D to 2D (introduce perspective)**

Copyright 2011, A. Sheffer, UBC
Projective Rendering Pipeline

From World to View Coordinates: W2V
- translate eye to origin
- rotate view vector (lookat - eye) to w axis
- rotate around w to bring up into vw-plane

Basic Viewing
- Starting spot - OpenGL
  - camera at world origin
  - y axis is up
  - looking down negative z axis
- To position - coordinate frame change
- Intuitive description
  - eye point, gaze/lookat direction, up vector

Deriving W2V Transformation

OpenGL Viewing Transformation

Copyright 2011, A. Sheffer, UBC
**World vs. Camera Coordinates**

\[
\begin{align*}
\mathbf{a} &= (1,1)_W \\
\mathbf{b} &= (1,1)_C_1 = (5,3)_W \\
\mathbf{c} &= (1,1)_C_2 = (1,3)_C_1 = (5,5)_W
\end{align*}
\]

**Clipping: View Volumes**
- Specifies field-of-view, used for clipping
- Restricts domain of \( z \) stored for visibility test

**Projective Rendering Pipeline**
- \( \text{glVertex3f(x,y,z)} \)
- \( \text{glTranslatef(x,y,z)} \)
- \( \text{gluLookAt(...)} \)
- \( \text{glFrustum(...)} \)
- \( \text{glutInitWindowSize(x,y)} \)

**Understanding Z**
- Why near and far plane?
  - Near plane:
    - Avoid singularity for perspective projection (division by zero, or very small numbers)
  - Far plane:
    - Store depth in fixed-point representation (integer), thus have to have fixed range of values (0...1)
    - Avoid/reduce numerical precision artifacts for distant objects

**Projection Transformations**
- In practice “ignore” \( z \) axis – use \( x \) and \( y \) coordinates for screen coordinates

---

Copyright 2011, A. Sheffer, UBC
Orthographic Derivation

- scale, translate, reflect for new coord sys

\[ y' = a \cdot y + b \]
\[ y = \text{top} \rightarrow y' = 1 \]
\[ y = \text{bot} \rightarrow y' = -1 \]

VCS: (1,1,1)

NDCS: (-1,-1,-1)

Orthographic OpenGL

```c
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(left,right,bot,top,near,far);
```

NDC to Viewport Transformation

- generate pixel coordinates
- map x, y from range -1...1 (NDC) to pixel coordinates on the display
- involves 2D scaling and translation

OpenGL: lower left
most window systems: upper left
often have to flip your y coordinates
when interpreting mouse position

Origin Location

- yet more possibly confusing conventions
- OpenGL: lower left
- most window systems: upper left
- often have to flip your y coordinates
- when interpreting mouse position

Orthographic OpenGL

```c
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(left,right,bot,top,near,far);
```

Perspective Projection

- Viewing is from point at finite distance (origin)
- View volume is a frustum not a box
- Conversion to device coordinates
  - Warp view frustum to box

Copyright 2011, A. Sheffer, UBC
Perspective Derivation

VCS
y=top
x=left
z=near

y=bottom
z=far
x=right

NDCS
(1,1,1)
(-1,-1,-1)

Projective Transformations

OpenGL Convention

Camera coordinates

Clipping Coordinates

Frustum

Perspective OpenGL

glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(left,right,bot,top,near,far);
or

glPerspective(fovy,aspect,near,far);
- symmetric version

Copyright 2011, A. Sheffer, UBC
### Another Transformations Quiz

What does each transformation preserve?

<table>
<thead>
<tr>
<th>Transformation</th>
<th>Lines</th>
<th>Parallel Lines</th>
<th>Distance</th>
<th>Angles</th>
<th>Normals</th>
<th>Convexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scaling</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rotation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Translation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shear</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perspective</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>