Viewing in 3D – Part II

Foley & Van Dam, Chapter 6

Viewing Window

- After objects were projected onto the viewing plane, an image is taken from a **Viewing Window**
- A viewing window can be placed anywhere on the view plane
- In general, the view window is aligned with the viewing coordinates and is defined by its extreme points: \((l,b)\) and \((r,t)\)

Viewing Volume

- Given the specification of the viewing window, we can set up a **Viewing Volume**
- Only objects inside the viewing volume will appear in the display, the rest are clipped

Defining the Viewing Volume

- Definition for Orthographic Projection:
  ```
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity(); /* camera "looks" down */
  glOrtho(left, right, bottom, top, near, far);
  ```
  - In all definitions, near and far are distances (always positive). Near = 50 means that the near plane intersects the z-axis at z = -50
Defining the Viewing Volume

• Definition for general Perspective Projection:

```cpp
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(left, right, bottom, top, near, far);
```

Defining the Viewing Volume

• Definition for a Standard Perspective Projection:

```cpp
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
aspectRatio = w/h;
gluPerspective(viewAngle, aspectRatio, near, far);
```

Position of the Viewing Plane

• Parallel projection: viewing-plane positioning does not affect the projected image
• Perspective projection: viewing-plane positioning does affect the projected image

Canonical Viewing Volume

• In order to determine the objects that are seen in the viewing window we have to clip objects against six planes forming the view volume
• Clipping against arbitrary 3D planes requires considerable computation
• For fast clipping we transform the viewing volume into a canonical viewing volume against which clipping is easy to apply

Canonical Volume: Parallel Projection

• Depth-preserving shear

<table>
<thead>
<tr>
<th>( x' )</th>
<th>( y' )</th>
<th>( z' )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>( y )</td>
<td>( z )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( x' )</th>
<th>( y' )</th>
<th>( z' )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>( y )</td>
<td>( z )</td>
</tr>
</tbody>
</table>

Translation

<table>
<thead>
<tr>
<th>( x' )</th>
<th>( y' )</th>
<th>( z' )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>( y )</td>
<td>( z )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( x' )</th>
<th>( y' )</th>
<th>( z' )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>( y )</td>
<td>( z )</td>
</tr>
</tbody>
</table>
**Canonical Volume: Parallel Projection**

- Scaling

\[
\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
\]

**Canonical Volume: Perspective Projection**

- Scaling

\[
\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
\]

- Depth-preserving shear

\[
\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & f & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
\]

**Viewport Transformation**

\[
\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{2}{width} & 0 & 0 & 0 \\ 0 & \frac{2}{height} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}
\]

\[\text{glViewport}(x_0, y_0, \text{width}, \text{height})\]

* You can have multiple viewports on your window *