7. 3D Viewing

Projection:
why is projection necessary?
7. 3D Viewing

Projection:
why is projection necessary?

Because the display surface is 2D
7.1 Projections

Perspective projection
7.1 Projections

Orthographic (parallel) projection
Perspective projection

Vanishing point
- Perspective projections of parallel lines not parallel to the projection plane will converge to a vanishing point

Principal vanishing point
- vanishing point of parallel lines that are parallel to one of the three principal axes
Orthographic projection

Three orthographic projections:
Orthographic projection

Isometric projection of unit cube along direction \((1, -1, -1)\):
7.2 Mathematics of Projections

(projections can be defined by 4x4 matrices)

Perspective projection: (not Affine, irreversible)

\[ x_p = \frac{x}{-z/d} \]
\[ y_p = \frac{y}{-z/d} \]
\[ z_p = -d \]

- Projection plane is normal to the z axis
7.2 Mathematics of Projections

(projections can be defined by 4x4 matrices)

\[
\begin{bmatrix} x_p, y_p, z_p, 1 \end{bmatrix}^t = \begin{bmatrix} \frac{x}{-z/d}, \frac{y}{-z/d}, -d, 1 \end{bmatrix}^t
\]

\[
= \begin{bmatrix} x, y, z, -z/d \end{bmatrix}^t
\]

\[
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = M_{per} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
\]
7.2 Mathematics of Projections

(projections can be defined by 4x4 matrices)

Orthographic (parallel) projection:
7.2 Mathematics of Projections

(projections can be defined by 4x4 matrices)
7.2 Mathematics of Projections

(projections can be defined by 4x4 matrices)

Perspective projection: (COP is not at the origin)

\[
\begin{align*}
    x_p &= \frac{x}{1 - z/d} \\
    y_p &= \frac{y}{1 - z/d} \\
    z_p &= 0
\end{align*}
\]
7.2 Mathematics of Projections
(projections can be defined by 4x4 matrices)

\[
\begin{bmatrix}
x_p, y_p, z_p, 1
\end{bmatrix}^t = \left[ \frac{x}{1 - z/d}, \frac{y}{1 - z/d}, 0, 1 \right]^t \\
= \left[ 0, 0, d, 1 \right]^t + \left[ \frac{x}{1 - z/d}, \frac{y}{1 - z/d}, -d, 1 \right]^t
\]

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & d \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & -1/d & 0
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z - d \\
1
\end{bmatrix}
\]
7.2 Mathematics of Projections

(projections can be defined by 4x4 matrices)

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & d \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & -d \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]

\[= M_t M_{per} M_t \]

\[
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]
7.2 Mathematics of Projections

(projections can be defined by 4x4 matrices)

Where are the vanishing points?

• Parallel lines after perspective projection are still parallel lines if they are also parallel to the projection plane.  Why?

• Parallel lines after perspective projection are no longer parallel lines if they are not parallel to the projection plane.  Why?
• Parallel lines after perspective projection are still parallel lines if they are also parallel to the projection plane. Why?

If \( L(t) = A + c \cdot t \) is a line

\[
A = (A_x, A_y, A_z) ; \quad c = (c_x, c_y, c_z)
\]

if the view point (eye) is at the origin and the projection plane is perpendicular to the \( z \) axis at \(-d\), then the perspective projection of \( L(t) \) is:

\[
L_p(t) = ( d \frac{A_x + c_x t}{-A_z - c_z t} , \quad d \frac{A_y + c_y t}{-A_z - c_z t}) \quad (\ast)
\]

If \( L(t) \) is parallel to the projection plane \((c_z = 0)\) then

\[
L_p(t) = \frac{-d}{A_z} \left( A_x + c_x t , \quad A_y + c_y t \right)
\]
Parallel lines after perspective projection are no longer parallel lines if they are not parallel to the projection plane.

If \( L(t) \) is not parallel to the projection plane \( (c_z \neq 0) \) then from (*) we have that

\[
L_p(t) \rightarrow -d \left( \frac{c_x}{c_z}, \frac{c_y}{c_z} \right) \quad \text{when} \quad t \rightarrow \infty
\]

Hence, any line with the same direction vector would converge to this (vanishing) point

\[-d \left( \frac{c_x}{c_z}, \frac{c_y}{c_z} \right).\]
7.2 Mathematics of Projections

(projections can be defined by 4x4 matrices)

Principal vanishing point: vanishing point generated by lines parallel to one of the principal axes (at most three PVPs).

Two-point perspective projection is popular
7.2 Mathematics of Projections

(projections can be defined by 4x4 matrices)

How to find vanishing points?

Construct a line parallel to $AB$ that passes thru the view point (eye). The intersection of this line with the projection plane is the vanishing point of $AB$. 
7.3 Camera Model for Projective View

• How to create a perspective view of a scene in OpenGL?

• How to control the camera’s position and orientation in OpenGL?
7.3 Camera Model for Projective View

Conceptual model of 3D viewing:

- Modelview matrix $M_v M_m$
- Projection matrix $M_p$
- Clip
- Perspective division
- Normalized device coordinates
- Viewport matrix

Output: Window (Device) coordinates
7.3 Camera Model for Projective View

Define **Viewing (Eye, or Camera)**

**Coordinate System:**
(specification of a 3D view)
(Positioning and pointing the camera)

```c
glMatrixMode ( GL_MODELVIEW );
glLoadIdentity ( );
gluLookAt ( eye.x, eye.y, eye.z, look.x, look.y, look.z, up.x, up.y, up.z);
```
7.3 Camera Model for Projective View

\[
\begin{align*}
\mathbf{n} &= \mathbf{EYE} - \mathbf{LOOK} \\
\mathbf{u} &= \mathbf{UP} \times \mathbf{n} \\
\mathbf{v} &= \mathbf{n} \times \mathbf{u}
\end{align*}
\]
Define the view volume:
(create a camera model)

```c
glMatrixMode ( GL_PROJECTION );
glLoadIdentity ( );
gluPerspective ( viewAngle, aspectRatio, N, F );
```
7.3 Camera Model for Projective View

\[ (w, h, -F) \]

\[ (w_1, h_1, -N) \]

\[ N > 0, \ F > 0 \]
7.4 Building Viewing Matrix

View Pipeline

\[ M_v M_m \rightarrow M_p \rightarrow \text{Clip} \rightarrow \text{Perspective division} \]

- Modelview matrix
- Projection matrix
- Eye coordinates
- Clip coordinates
7.4 Building Viewing Matrix

Canonical View Volume

- Parallel: $x = \pm 1$, $y = \pm 1$, $z = \pm 1$

- Perspective: $x = z$, $x = -z$, $y = z$, $y = -z$,
  
  $z = -z_{\text{min}}$, $z = -1$
7.4 Building Viewing Matrix

**Modelview Matrix** ($M_v M_m$):

- **Modeling part** ($M_m$):
  - embodies all the modeling transformations for the object

- **Viewing part** ($M_v$):
  - accounts for the WC to VC transformation set by the camera’s **position** and **orientation**

accumulate all the modeling transformations into a single matrix

translation + rotation
### 7.4 Building Viewing Matrix

where

\[
(d_x, d_y, d_z) = (-\text{eye}_x, -\text{eye}_y, -\text{eye}_z)
\]
7.4 Building Viewing Matrix

**Projection Matrix** ($M_p$):

$$M_p = \text{scaling2}$$

* translation

* perspective transformation

* scaling1

* shearing

$$M_p = M_{s2} \ast M_t \ast M_{pt} \ast M_{s1} \ast M_{sh}$$
Shearing: so that window center would coincide with \((0, 0, -N)\)

\[
M_{sh} = \begin{bmatrix}
1 & 0 & a & 0 \\
0 & 1 & b & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

\[
a = \frac{r + l}{2N} \quad b = \frac{t + b}{2N}
\]
7.4 Building Viewing Matrix

**Scaling 1**: so user defined truncated view volume would coincide with the canonical view volume for perspective projection.
7.4 Building Viewing Matrix

Perspective Transformation: convert CVV for perspective projection to a quasi-CVV for parallel projection

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & \frac{F}{F - N} & \frac{0}{N} \\
0 & 0 & \frac{F - N}{F - N} & \frac{0}{0}
\end{bmatrix}
\]

also called a homography
7.4 Building Viewing Matrix

**Translation**: translate center of the quasi-CVV to the origin (0,0,0)

\[
M_t = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1/2 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]
7.4 Building Viewing Matrix

**Scaling 2**: scale z-direction by 2 to get the CVV for parallel projection

\[
M_{s2} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]
7.4 Building Viewing Matrix

\[ M_p = ? \]

\[ M_p = M_{s2} \cdot M_t \cdot M_{pt} \cdot M_{s1} \cdot M_{sh} \]

\[
\begin{bmatrix}
\frac{F}{w} & 0 & 0 & 0 & 0 \\
0 & \frac{F}{h} & 0 & 0 & 0 \\
0 & 0 & \frac{F + N}{F - N} & \frac{2FN}{F - N} & 0 \\
0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]
7.5 GL_PROJECTION & GL_MODELVIEW

GL_PROJECTION:
- applied to every point that comes after it

GL_MODELVIEW:
- applied to every point in a particular model
GL_PROJECTION

```c
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-1, 1, -1, 1, -1.0, 1.0);
glTranslate(100, 100, 100);
glRotateF(45.0, 1.0, 0.0, 0.0);

Gl_Projection_Matrix = identity_matrix * orthographic_matrix
* Translation_matrix * Rotation_matrix
```
GL_MODELVIEW

```c
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-1, 1, -1, 1, -1.0, 1.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslate(box_x, box_y, box_z);
// draw box here

// draw bottle here
```
GL Matrices

Model Coordinates
- Model Matrix
  - World Coordinates
    - View Matrix
      - Camera Coordinates
        - Projection Matrix
          - Homogeneous Coordinates
GL_MODELVIEW

For the box:

\[
\text{Box\_Vertices\_Matrix} = \text{Projection\_Ortho\_Matrix} \times \text{Projection\_Translation\_Matrix} \times \text{Box\_Translation\_Matrix}
\]

For the bottle:

\[
\text{Bottle\_Vertices\_Matrix} = \text{Projection\_Ortho\_Matrix} \times \text{Projection\_Translation\_Matrix} \times \text{Bottle\_Translation\_Matrix}
\]
7.6 Clipping in Homogeneous Coordinates

What does $M_{pt}$ do?

A $\rightarrow$ G
B $\rightarrow$ E
C $\rightarrow$ F
D $\rightarrow$ G
7.6 Clipping in Homogeneous Coordinates

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & F & N \\
0 & 0 & F - N & F - N
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
= 
\begin{bmatrix}
x \\
y \\
Fz + N \\
F - N
\end{bmatrix}
= 
\begin{bmatrix}
-x/z \\
-y/z \\
Fz + N \\
-z(F - N)
\end{bmatrix}
\]

If \( z > 0 \) then \( \frac{Fz + N}{-z(F - N)} < -1 \)

If \( z < -1 \) then \( \frac{Fz + N}{-z(F - N)} < -1 \)

If \( \frac{-N}{F} < z < 0 \) then \( \frac{Fz + N}{-z(F - N)} > 0 \)

If \( -1 < z < \frac{-N}{F} \) then \( 0 > \frac{Fz + N}{-z(F - N)} > -1 \)
7.6 Clipping in Homogeneous Coordinates

Now consider the following example:

\[ M_{pt} \text{ maps } P \text{ and } Q \text{ both into points in region } G \]
7.6 Clipping in Homogeneous Coordinates

- If $P'Q'$ is clipped against CVV after the perspective division, the clipping algorithm would have the line segment discarded. But $R'S'$ is actually inside CVV.
- The reason that this happens is because the division performed for $P'$ changes the sign of z-component from positive to negative.

Remedy: perform clipping before perspective division, i.e., clip in homogeneous coordinates, then perform perspective division.
7.6 Clipping in Homogeneous Coordinates

- Basic idea:

  Instead of using \( \frac{X}{W}, \frac{Y}{W}, \frac{Z}{W} \),
  for the clipping process
7.6 Clipping in Homogeneous Coordinates

Why?

A point (after the *perspective division*) is inside the CVV for parallel projection if

\[-1 \leq \frac{X}{W} \leq 1, \quad -1 \leq \frac{Y}{W} \leq 1, \quad -1 \leq \frac{Z}{W} \leq 1\]
Clipping in Homogeneous Coordinates

are called **Boundary Coordinates (BC’s)**.

<table>
<thead>
<tr>
<th>$W+X$</th>
<th>$x=-1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W-X$</td>
<td>$x=1$</td>
</tr>
<tr>
<td>$W+Y$</td>
<td>$y=-1$</td>
</tr>
<tr>
<td>$W-Y$</td>
<td>$y=1$</td>
</tr>
<tr>
<td>$W+Z$</td>
<td>$z=-1$</td>
</tr>
<tr>
<td>$W-Z$</td>
<td>$z=1$</td>
</tr>
</tbody>
</table>
7.6 Clipping in Homogeneous Coordinates

If $W > 0$, this means the boundary coordinates (BC’s) must all be positive.

| Boundary coordinate | homogeneous value\(\begin{align*} W + X &> 0 \hline W - X &> 0 \hline W + Y &> 0 \hline W - Y &> 0 \hline W + Z &> 0 \hline W - Z &> 0 \end{align*}\) | clip plane\(\begin{align*} x = -1 \hline x = 1 \hline y = -1 \hline y = 1 \hline z = -1 \hline z = 1 \end{align*}\) |
7.6 Clipping in Homogeneous Coordinates

If $W < 0$, then all the BC’s must be negative.

<table>
<thead>
<tr>
<th>Boundary coordinate</th>
<th>homogeneous value</th>
<th>clip plane</th>
</tr>
</thead>
<tbody>
<tr>
<td>$BC_0$</td>
<td>$W + X &lt; 0$</td>
<td>$x = -1$</td>
</tr>
<tr>
<td>$BC_1$</td>
<td>$W - X &lt; 0$</td>
<td>$x = 1$</td>
</tr>
<tr>
<td>$BC_2$</td>
<td>$W + Y &lt; 0$</td>
<td>$y = -1$</td>
</tr>
<tr>
<td>$BC_3$</td>
<td>$W - Y &lt; 0$</td>
<td>$y = 1$</td>
</tr>
<tr>
<td>$BC_4$</td>
<td>$W + Z &lt; 0$</td>
<td>$z = -1$</td>
</tr>
<tr>
<td>$BC_5$</td>
<td>$W - Z &lt; 0$</td>
<td>$z = 1$</td>
</tr>
</tbody>
</table>
Clip a Line Segment in Homogeneous Coordinates

How to clip a line segment in homogeneous coordinates?

Use Cyrus-Beck clipper:

Input:

\[ A = \left( A_x, A_y, A_z, A_w \right) \]

\[ B = \left( B_x, B_y, B_z, B_w \right) \]

\[ L(t) = A + (B - A) * t \]
Clip a Line Segment in Homogeneous Coordinates

- Compute $BC$’s for A and B
- Compute *outcodes* for A and B
- Perform “trivial rejection” test
- Perform “trivial acceptance” test
Clip a Line Segment in Homogeneous Coordinates

If A and B are on different sides of $x=1$, then compute parameter of the intersection point as follows:

$$t = \frac{A_w - A_x}{(A_w - A_x) - (B_w - B_x)}$$

Then update related items’ values
Clip a Line Segment in Homogeneous Coordinates

For \( x=1 \), we consider: \( W - X = 0 \)

with

\[
W = A_w + (B_w - A_w) \times t \\
X = A_x - (B_x - A_x) \times t
\]

Solving it to get

\[
t = \frac{A_w - A_x}{(A_w - A_x) - (B_w - B_x)}
\]
End of 3D Viewing