Viewing

Aside: Transforms and OpenGL

- Transforms go onto matrix “stacks”
- All vertices xformed by top matrices
Two stacks. Why?

- Separate modeling & viewing xforms

3D viewing process

1. Define objects
2. Compose scene
3. Define lighting
4. Cull, Project
5. Clip
6. Hidden surface removal
7. Rasterize, shade

**Local coordinate space**

- Modeling transformation

**World coordinate space**

- View transformation

**View space**

**Screen space**
View Space
(AKA Camera Coordinate Space)

- What do we need to specify to define what is seen?
  - Extrinsic
  - Intrinsic

Specifying a basic view
Translate $C$ to origin

$$\begin{pmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & -c_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = T(-C)$$

Rotate UVN→XYZ

We want to take $u$ into $(1, 0, 0)$  
v into $(0, 1, 0)$  
n into $(0, 0, 1)$

First derive $n$, $u$, and $v$ from user input:
Rotate UVN

\[
\begin{pmatrix}
  u_x & u_y & u_z & 0 \\
  v_x & v_y & v_z & 0 \\
  n_x & n_y & n_z & 0 \\
  0   & 0   & 0   & 1
\end{pmatrix} = R_{UVN}
\]

ViewSpace Operations

- Backface culling
- Projection
Backface culling

- $N_p$: the polygon normal
- $N$: the view direction vector
- $N_p \cdot N$

Projections

- 3D points project onto view plane where projector (line to COP) intersects VP
- Perspective Proj.
  - COP in world
- Parallel Proj.
  - COP at infinity
Viewing

The resulting view volumes

- Parallel
  - Infinite parallelepiped
- Perspective
  - Semi-infinite pyramid

- Limit them
  - Front and back clipping planes

Perspective View Volume (fig 5.6)
Defining the Perspective View

- \( x_s/d = x_p/z_p \)
- \( T_{\text{persp}} = \)
  
  \[
  \begin{pmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 0 \\
  0 & 0 & 1/d & 0 \\
  \end{pmatrix}
  \]

Parallel (Orthographic) View

- \( T_{\text{ort}} = \)
  
  \[
  \begin{pmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 1 \\
  \end{pmatrix}
  \]
ScreenSpace

- Clipping
- Hidden surface removal (z-buffering) and rendering (rasterization+shading)
- Done in a “canonical volume”
  - Simplifies efficient implementation

Canonical View Volumes
Simple Perspective Transform

1) Scale sides to 45 degrees

\[
\begin{pmatrix}
\frac{d}{h} & 0 & 0 & 0 & 0 \\
0 & \frac{d}{h} & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0
\end{pmatrix}
\]

Simple Perspective Transform

2) Map to canonical parallel volume

\[
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & f/(f-d) & -df/(f-d) & 0 \\
0 & 0 & 1 & 0 & 0
\end{pmatrix}
\]
Z accuracy over [0..1]

\[ z_s = \frac{f(1-d/z_v)}{(f-d)} \]

Figure 5.11 Illustrating the distortion in three-dimensional screen space due to the \( z_s \) to \( z_v \) transformation.

Advanced Viewing: PHIGS

- Two coordinate systems
  - World reference coordinate system (WRC)
  - Viewing reference coordinate system (VRC)
Arbitrary view reference point

- Specify viewplane, view coords (WRC)
  - View Reference Point (VRP)
  - View Plane Normal (VPN)
  - View Up Vector (VUV)
- Specify window on the view plane (VRC)
  - Max and min u,v values (window center (CW))
  - Projection Reference Point (PRP)
    - Ignore VPD from book for now (but understand it!)

Specifying a view

[Diagram showing the relationship between WRC, VRP, VPN, VUP, VRC, and PRP]
Normalizing Transformation for Perspective Views

1. Translate VRP to origin
2. Rotate the VRC system so that VPN become z-axis, u become x-axis and v become y-axis
3. Translate so that the CoP given by the PRP is at origin
4. Shear such that the center line of the view volume becomes the z-axis
5. Scale so that the view volume becomes the canonical view volume

1. Translate VRP to origin

\[
\begin{pmatrix}
1 & 0 & 0 & -\text{VRP}_x \\
0 & 1 & 0 & -\text{VRP}_y \\
0 & 0 & 1 & -\text{VRP}_z \\
0 & 0 & 0 & 1
\end{pmatrix} = \text{T}(-\text{VRP})
\]
2. Rotate VRC

We want to take $u$ into $(1, 0, 0)$
$v$ into $(0, 1, 0)$
n into $(0, 0, 1)$

First derive $n$, $u$, and $v$ from user input:

2. Rotate VRC (cont.)

$\begin{pmatrix}
  u_x & u_y & u_z & 0 \\
v_x & v_y & v_z & 0 \\
n_x & n_y & n_z & 0 \\
0 & 0 & 0 & 1
\end{pmatrix} = R_{VRC}$
3. Translate PRP to the origin

\[
\begin{pmatrix}
1 & 0 & 0 & -PRP_u \\
0 & 1 & 0 & -PRP_v \\
0 & 0 & 1 & -PRP_n \\
0 & 0 & 0 & 1
\end{pmatrix}
= T(-PRP)
\]

4. Shear such that the center line of the view volume becomes the z-axis

Direction of projection (DoP) = CW - PRP

The center line of the view volume is DoP
Shear (cont.)

Multiply DoP with a matrix to get \((0,0,\text{DoP}_z)\)

We want \(\text{SH} \times \text{DoP} = (0,0,\text{DoP}_z)\)

\[
\text{SH} = \begin{pmatrix}
1 & 0 & \text{SH}_x & 0 \\
0 & 1 & \text{SH}_y & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

\(\text{SH}_x = -\text{DoP}_x/\text{DoP}_z\)

\(\text{SH}_y = -\text{DoP}_y/\text{DoP}_z\)

5. Scale

\[
y = \frac{v_{\text{max}} - v_{\text{min}}}{2}
\]

\[
z = -\text{PRP}_n + F
\]

\[
z = -\text{PRP}_n + B
\]

\[
y = -\frac{(v_{\text{max}} - v_{\text{min}})}{2}
\]

NOTE: \(\text{VRP}'_n = -\text{PRP}_n\)
5. Scale (cont.)

Scale is done in two steps:
1. First scale in x and y
   \[ \text{xscale} = 2 \frac{\text{PRP}_n}{(\text{umax} - \text{umin})} \]
   \[ \text{yscale} = 2 \frac{\text{PRP}_n}{(\text{vmax} - \text{vmin})} \]
2. Scale everything uniformly such that the back clipping plane becomes \( z = -1 \)
   \[ \text{xscale} = \frac{1}{(-\text{PRP}_n + B)} \]
   \[ \text{yscale} = \frac{1}{(-\text{PRP}_n + B)} \]
   \[ \text{zscale} = \frac{1}{(-\text{PRP}_n + B)} \]

Total Composite Transformation

\[ \mathbf{N}_{\text{per}} = \mathbf{S}_{\text{per}} \mathbf{SH}_{\text{per}} \mathbf{T}(-\text{PRP}) \mathbf{R} \mathbf{T}(-\text{VRP}) \]

Use this to transform from the viewing to the world space, then project onto the viewplane.