Viewing

Aside: Transforms and OpenGL

- Transforms go onto matrix “stacks”
- All vertices xformed by top matrices

Two stacks. Why?

- Separate modeling & viewing xforms

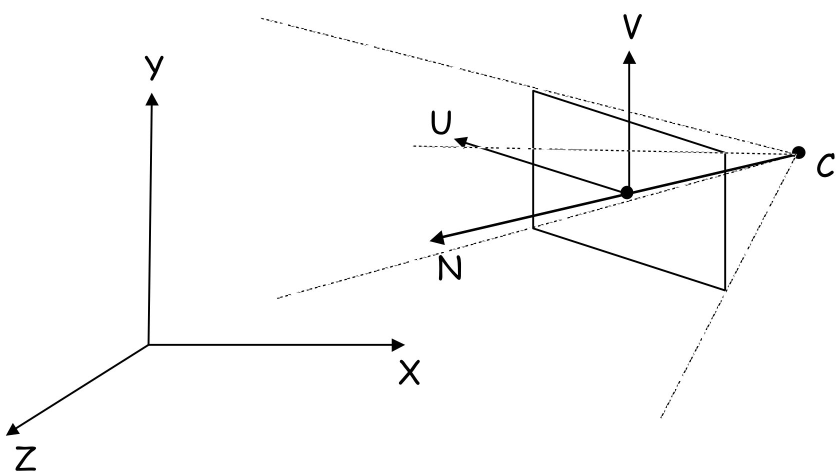
3D viewing process

Define objects	Local coordinate space
	↓ Modeling transformation
Compose scene Define lighting	World coordinate space
	↓ View transformation
Cull, Project	View space
	↓
Clip Hidden surface removal Rasterize, shade	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

$$\begin{pmatrix} 1 & 0 & 0 & -C_X \\ 0 & 1 & 0 & -C_y \\ 0 & 0 & 1 & -C_Z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \mathbf{T(-C)}$$

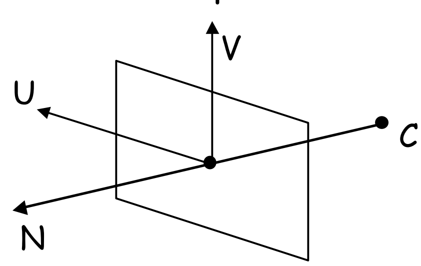
Rotate UVN->XYZ

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

$$\begin{pmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \mathbf{R}_{UVN}$$

ViewSpace Operations

- Backface culling
- Projection

Backface culling

- N_p: the polygon normal
- N: the view direction vector

- $N_p \cdot N$

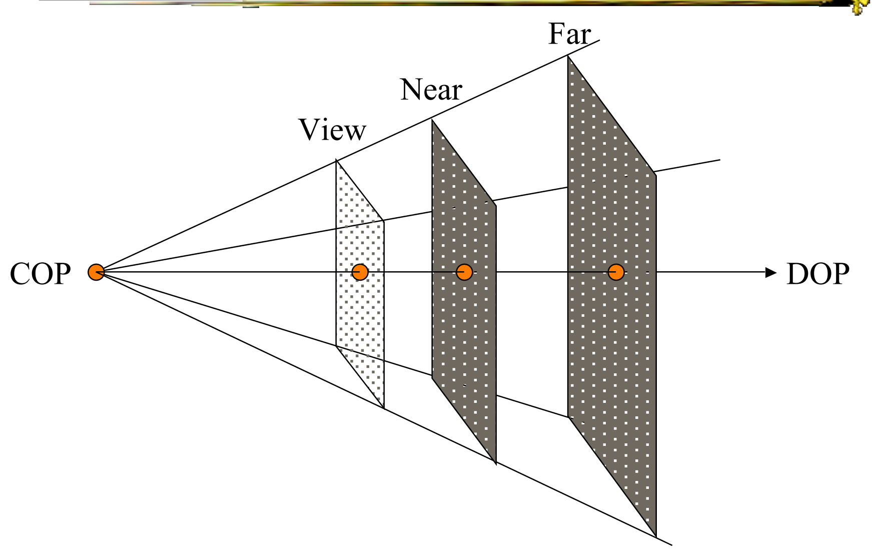
Projections

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

The resulting view volumes

- Parallel
  - Infinite parallelpiped
- Perspective
  - Semi-infinite pyramid
- Limit them
  - Front and back clipping planes

Perspective View Volume (fig 5.6)



Defining the Perspective View

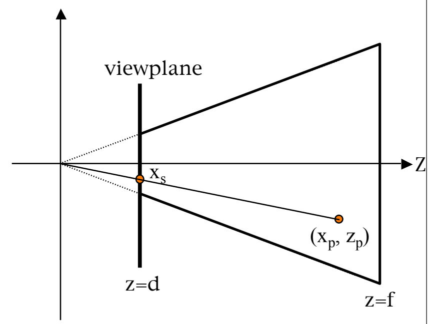
- $x_s/d=x_p/z_p$
- T_{persp} =

  (1 0 0 0)

  (0 1 0 0)

  (0 0 1 0)

  (0 0 1/d 0)



Parallel (Orthographic) View

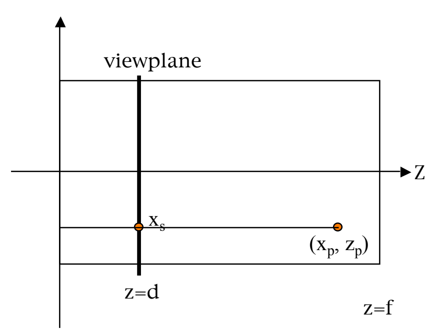
- T_{ort} =

  (1 0 0 0)

  (0 1 0 0)

  (0 0 0 0)

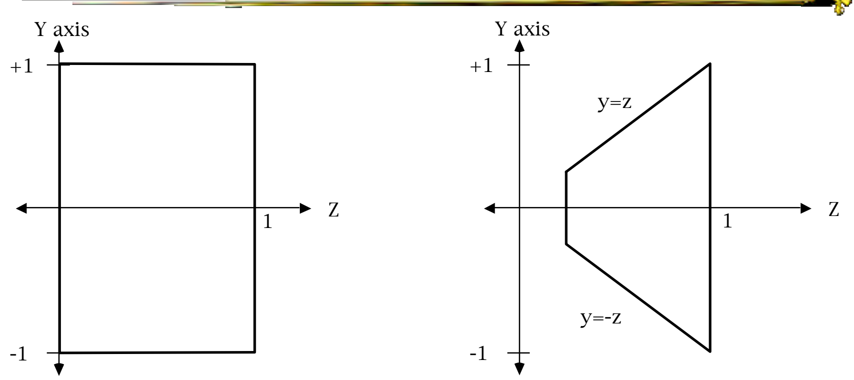
  (0 0 0 1)



ScreenSpace

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

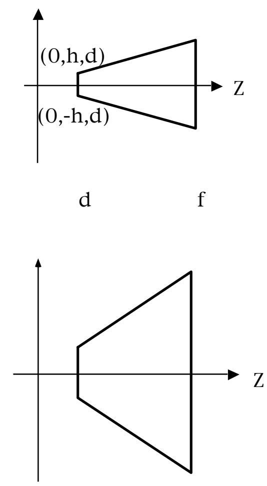
Canonical View Volumes



Simple Perspective Transform

1) Scale sides to 45 degrees

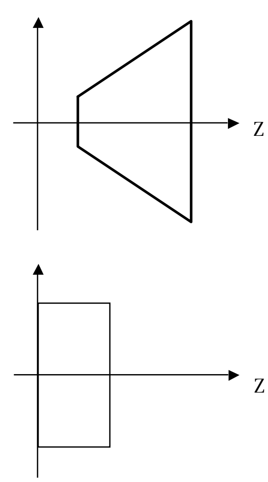
$$\begin{pmatrix} d/h & 0 & 0 & 0 \\ 0 & d/h & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Simple Perspective Transform

2) Map to canonical parallel volume

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & f/(f-d) & -df/(f-d) \\ 0 & 0 & 1 & 0 \end{pmatrix}$$



Z accuracy over [0..1]

- $z_s = (f\,(1-d/z_v))\,/\,(f-d)$

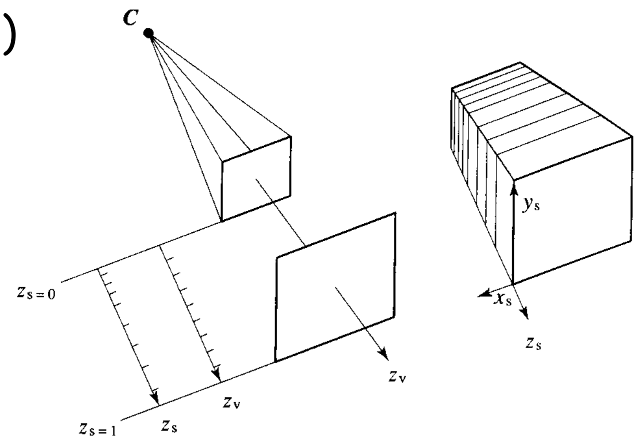


Figure 5.11
Illustrating the distortion in three-dimensional screen space due to the z_v to z_s transformation.

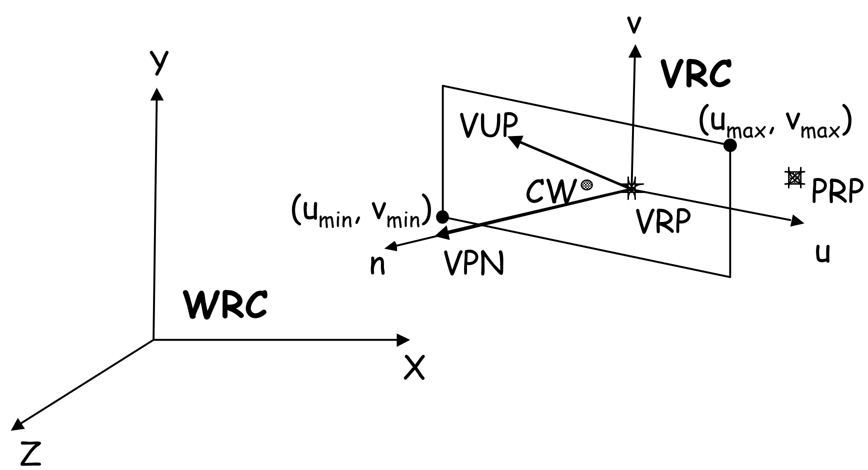
Advanced Viewing: PHIGS

- 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



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

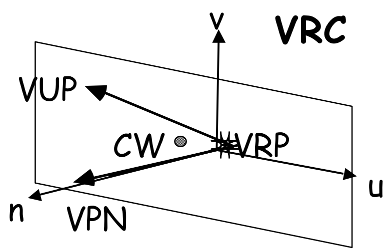
1. Translate VRP to origin

$$\begin{pmatrix} 1 & 0 & 0 & -VRP_x \\ 0 & 1 & 0 & -VRP_y \\ 0 & 0 & 1 & -VRP_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \mathbf{T(-VRP)}$$

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:



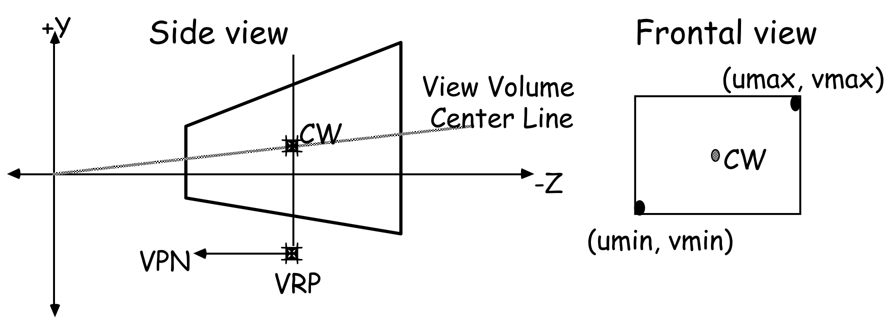
2. Rotate VRC (cont.)

$$\begin{pmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \mathbf{R_{VRC}}$$

3. Translate PRP to the origin

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

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 $(0,0,DoP_z)$

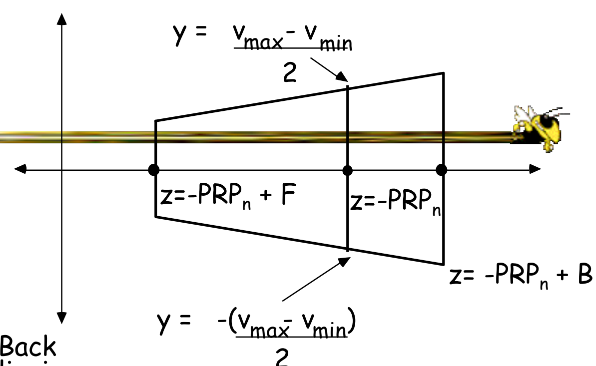
We want SH*DoP = $(0,0,DoP_z)$

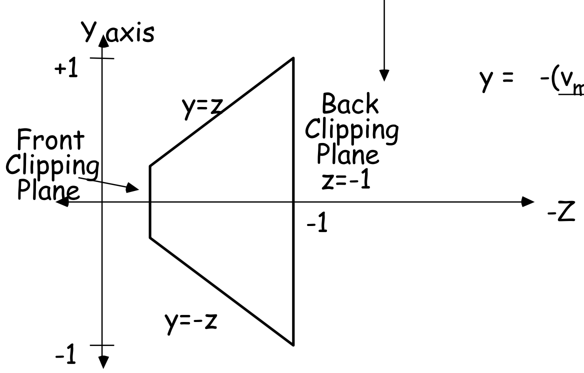
$$SH = \begin{pmatrix} 1 & 0 & SHx & 0 \\ 0 & 1 & SHy & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$SHx = -DoPx/DoPz$

$SHy = -DoPy/DoPz$

5. Scale





NOTE: $VRP'_n=-PRP_n$

5. Scale (cont.)

Scale is done in two steps:

1. First scale in x and y

   xscale = 2 PRP_n/(umax - umin)

   yscale = 2 PRP_n/(vmax - vmin)

2. Scale everything uniformly such that the back clipping plane becomes z =-1

   xscale = 1 / ($-PRP_n$ + B)

   yscale = 1 / ($-PRP_n$ + B)

   zscale = 1 / ($-PRP_n$ + B)

Total Composite Transformation

$N_{per} = S_{per}\ SH_{per}\ T(-PRP)\ R\ T(-VRP)$

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