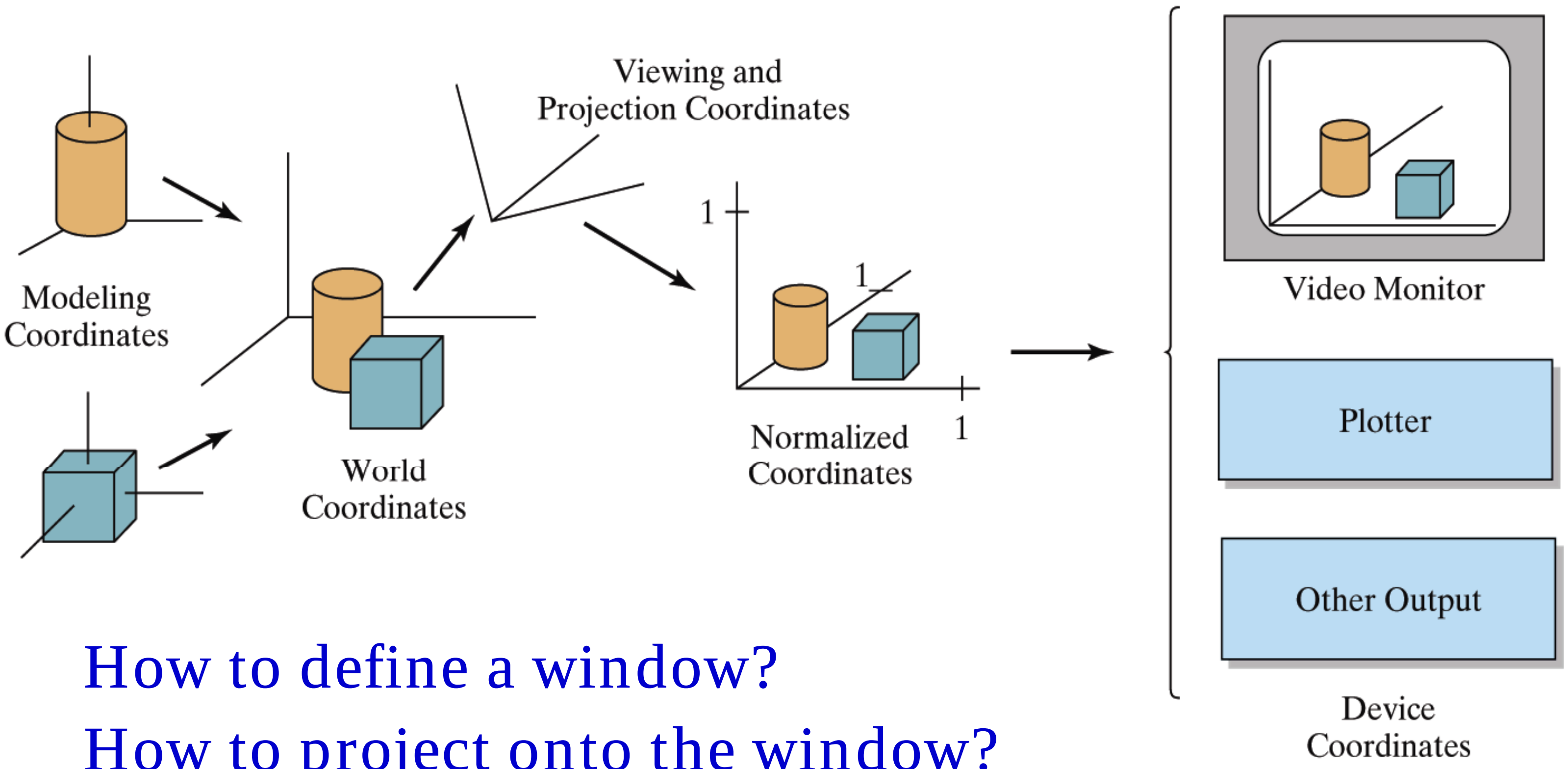
Three-Dimensional Viewing

Chap 7, 2008 Spring

Yeong Gil Shin

Viewing Pipeline



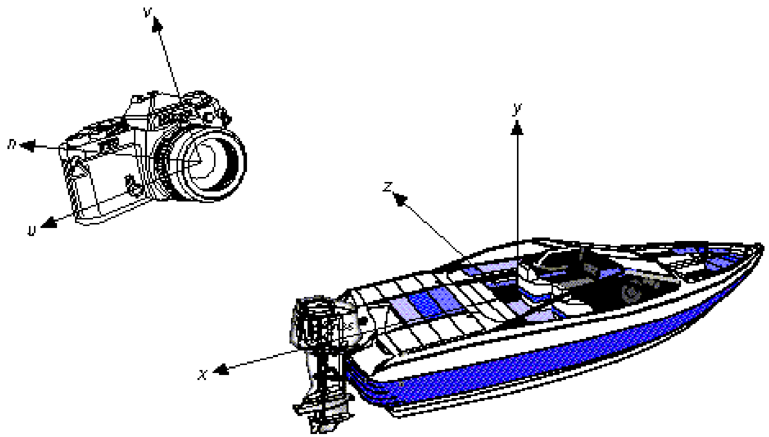
How to define a window?

How to project onto the window?

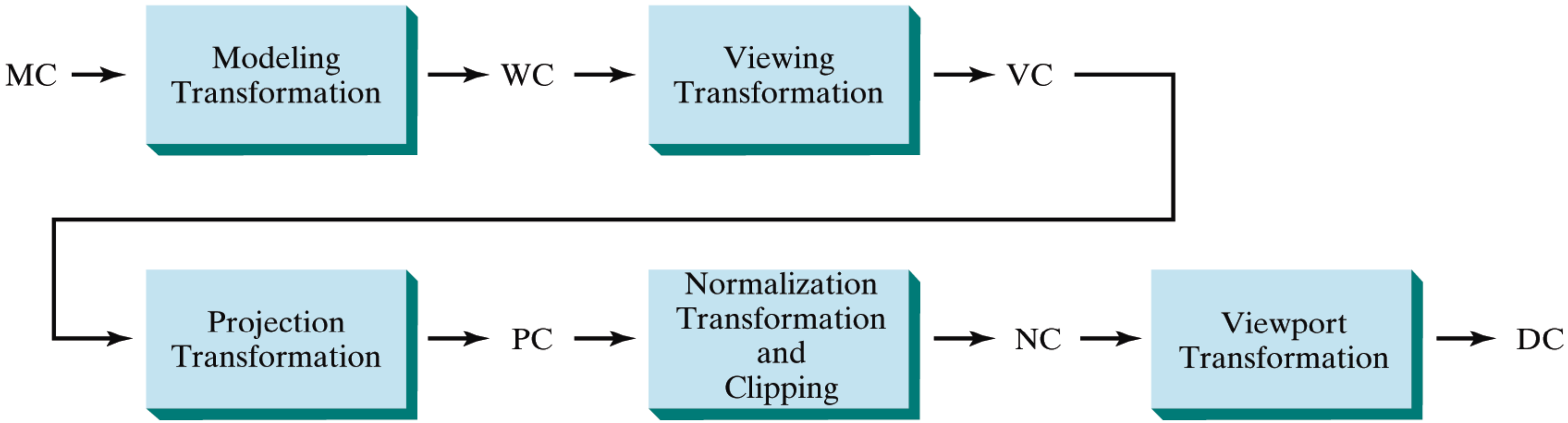
Rendering

"Create a picture (in a synthetic camera)"

- Specification of projection type
- Specification of viewing parameters:
  - viewer's eye, viewing plane (viewing coordinates)
- Clipping in 3D: window, view volume
- Projection : the transformation of points from a coordinate system in *n* dimensions to a coordinate system in *m* dimensions where *m*<*n*
- Display: view port



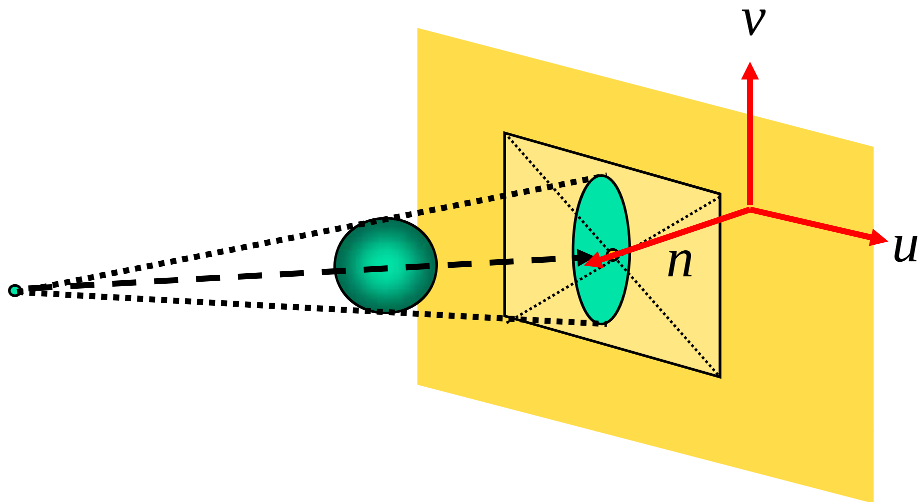
General 3D Viewing Pipeline



- Modeling coordinates (MC)
- World coordinates (WC)
- Viewing coordinates (VC) – VRC, camera position
- Projection coordinates (PC) – window, projection type
- Normalized coordinates (NC)
- Device coordinates (DC) – view port in a screen

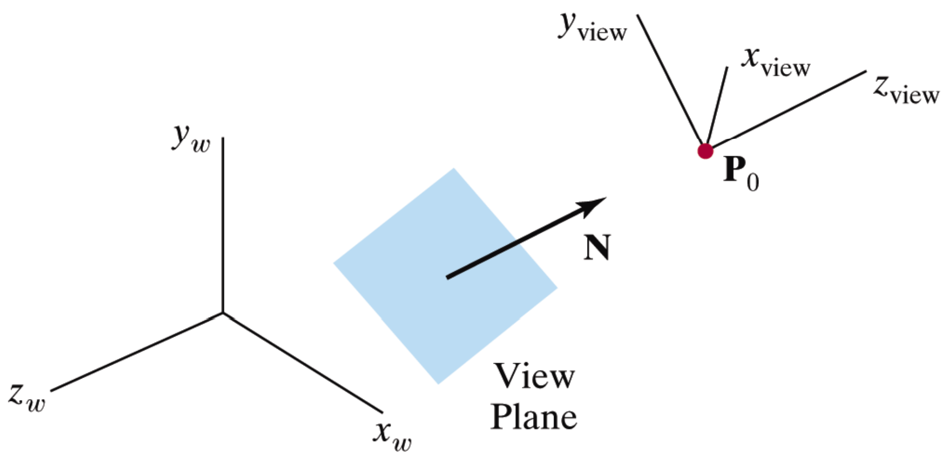
Projections

- Projection coordinate system:
  - left-handed – Core, **DirectX**
    - why? - screen coordinate system is left-handed
  - right-handed - GKS, PHIGS, **OpenGL**, DirectX
- Projection plane (view plane): viewing surface where objects are projected.



Viewing-Coordinate Parameters

- View reference point (VRP)
  - **The viewing origin in WC :** $\mathbf{P}_0 = (x_0, y_0, z_0)$
  - **Eye position, camera position**
- View-plane (Projection plane)
  - **Locates on z_{view} axis :** z_{vp}
  - **Perpendicular to** z_{view}
- Viewing Coordinate : uvn
  - **Defined by** P_0, N, VUP



Viewing-Coordinate Parameters

- How we specify a view plane normal vector **N**

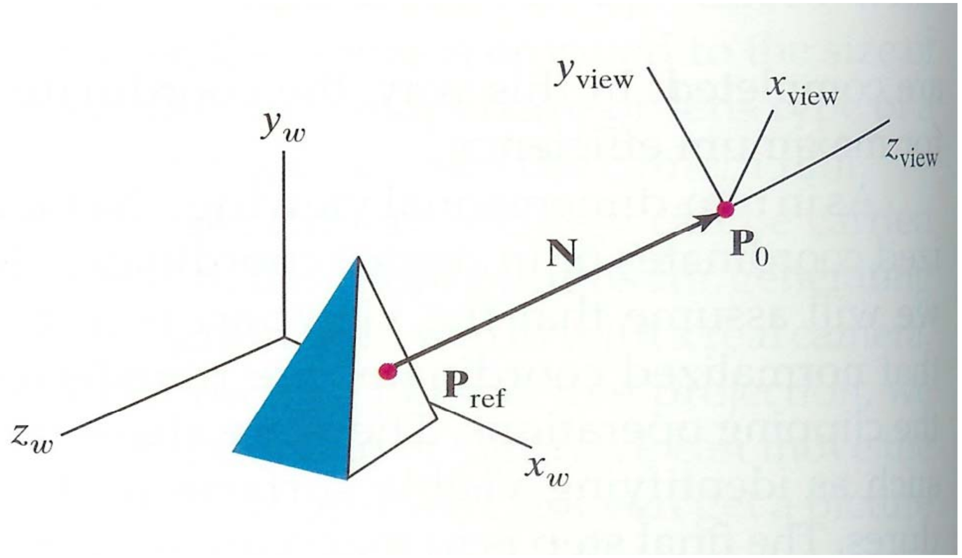
  [Way I] The origin of WC to a selected point position

  [Way II] The direction from a reference point P_{ref}

  to the viewing origin: $\mathbf{N} = \mathbf{P_0} - \mathbf{P_{ref}}$

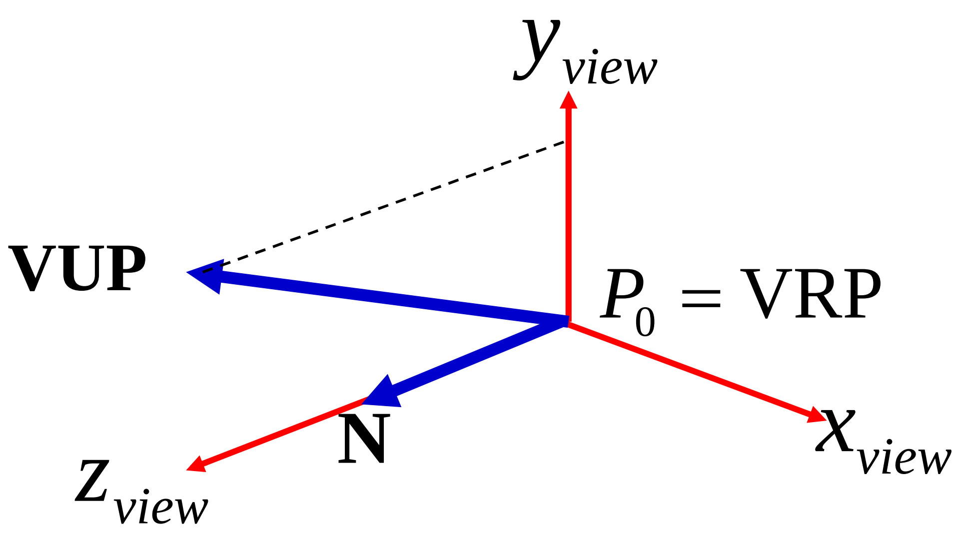
  P_{ref} : look-at point

  Viewing direction: –N



Viewing-Coordinate Parameters

- View-up vector: **VUP**
  - Specified in the world coordinates
  - Used to establish the positive direction for the y_{view} axis
  - **VUP** should be perpendicular to **N**, but it can be difficult to a direction for **VUP** that is precisely perpendicular to **N**



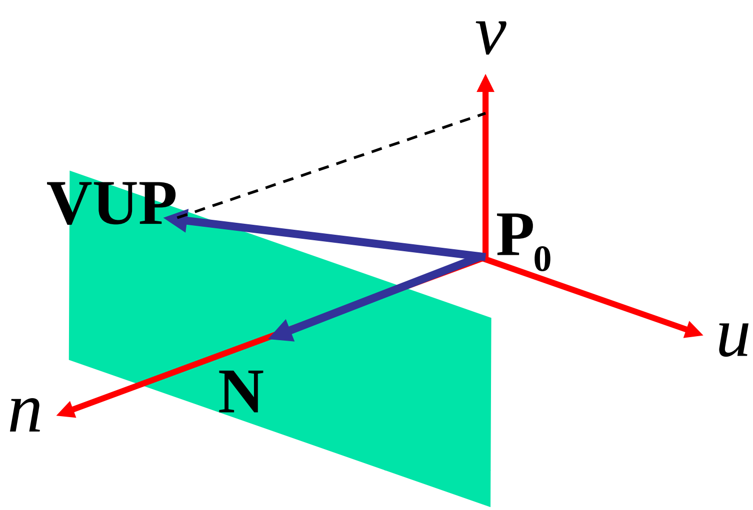
Viewing-Coordinate Reference Frame (VRC)

- The camera orientation is determined by viewing reference frame $\mathbf{x_{view}}$, $\mathbf{y_{view}}$, $\mathbf{z_{view}}$ (or **uvn)**
- The origin of the viewing reference frame : $\mathbf{P_0}$ **(= VRP)**
- **uvn** is called View Reference Coordinate (VRC)

$$\mathbf{n} = \frac{\mathbf{N}}{\|\mathbf{N}\|} = (n_x, n_y, n_z)$$

$$\mathbf{u} = \frac{\mathbf{VUP} \times \mathbf{n}}{\|\mathbf{VUP}\|} = (u_x, u_y, u_z)$$

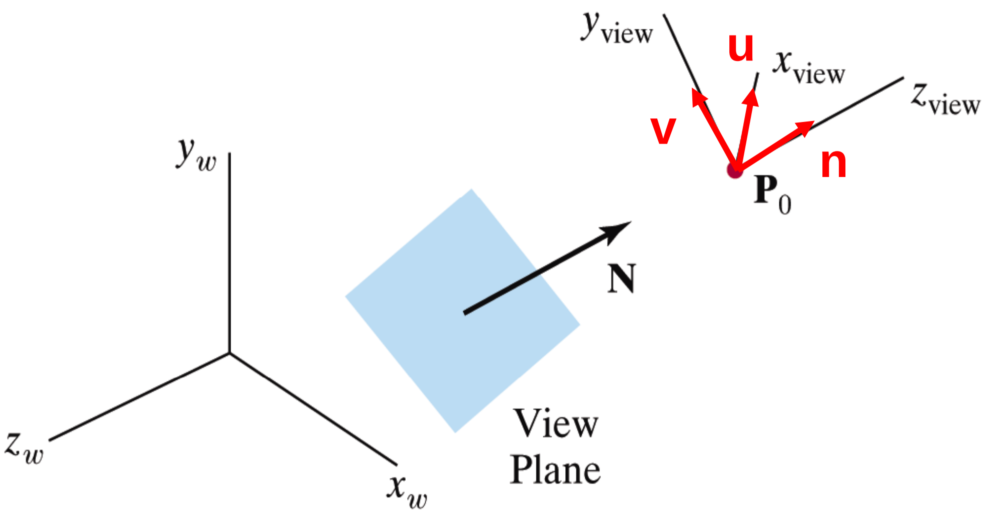
$$\mathbf{v} = \mathbf{n} \times \mathbf{u} = (v_x, v_y, v_z)$$



View plane w.r.t. VRC

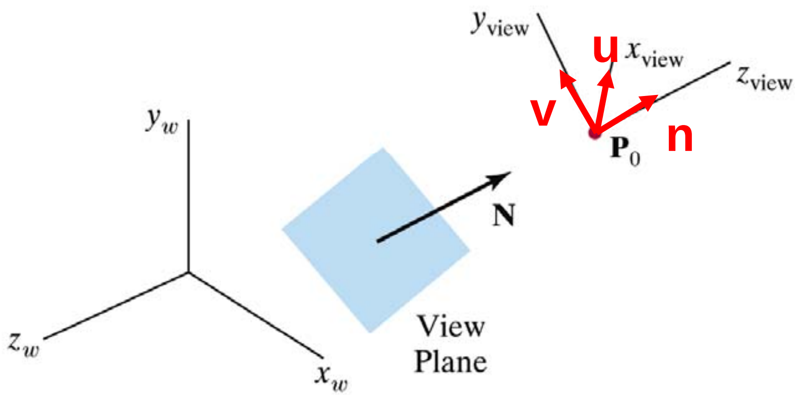
World-to-Viewing Transformation

- Transformation from WC to VRC
  - Translate the viewing-coordinate origin to the world-coordinate origin
  - Apply rotations to align the **u**, **v**, **n** axes with the world x_w, y_w, z_w axes, respectively



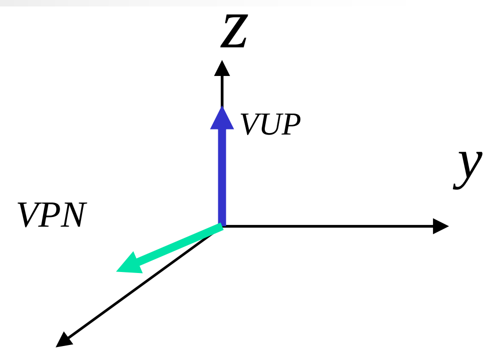
World-to-Viewing Transformation

$$\mathbf{R} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$\mathbf{M}_{wc,vc} = \mathbf{R}\cdot\mathbf{T} = \begin{bmatrix} u_x & u_y & u_z & -\mathbf{u}\cdot\mathbf{P}_0 \\ v_x & v_y & v_z & -\mathbf{v}\cdot\mathbf{P}_0 \\ n_x & n_y & n_z & -\mathbf{n}\cdot\mathbf{P}_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Example of VRC → WC



Given $VPN = \begin{pmatrix} -6 & -8 & -7.5 \end{pmatrix}$

$VUP = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$

$$R_z = \frac{R_z}{\|R_z\|} = \frac{1}{12.5}(-6,-8,-7.5) = (0.48,0.64,0.60)$$

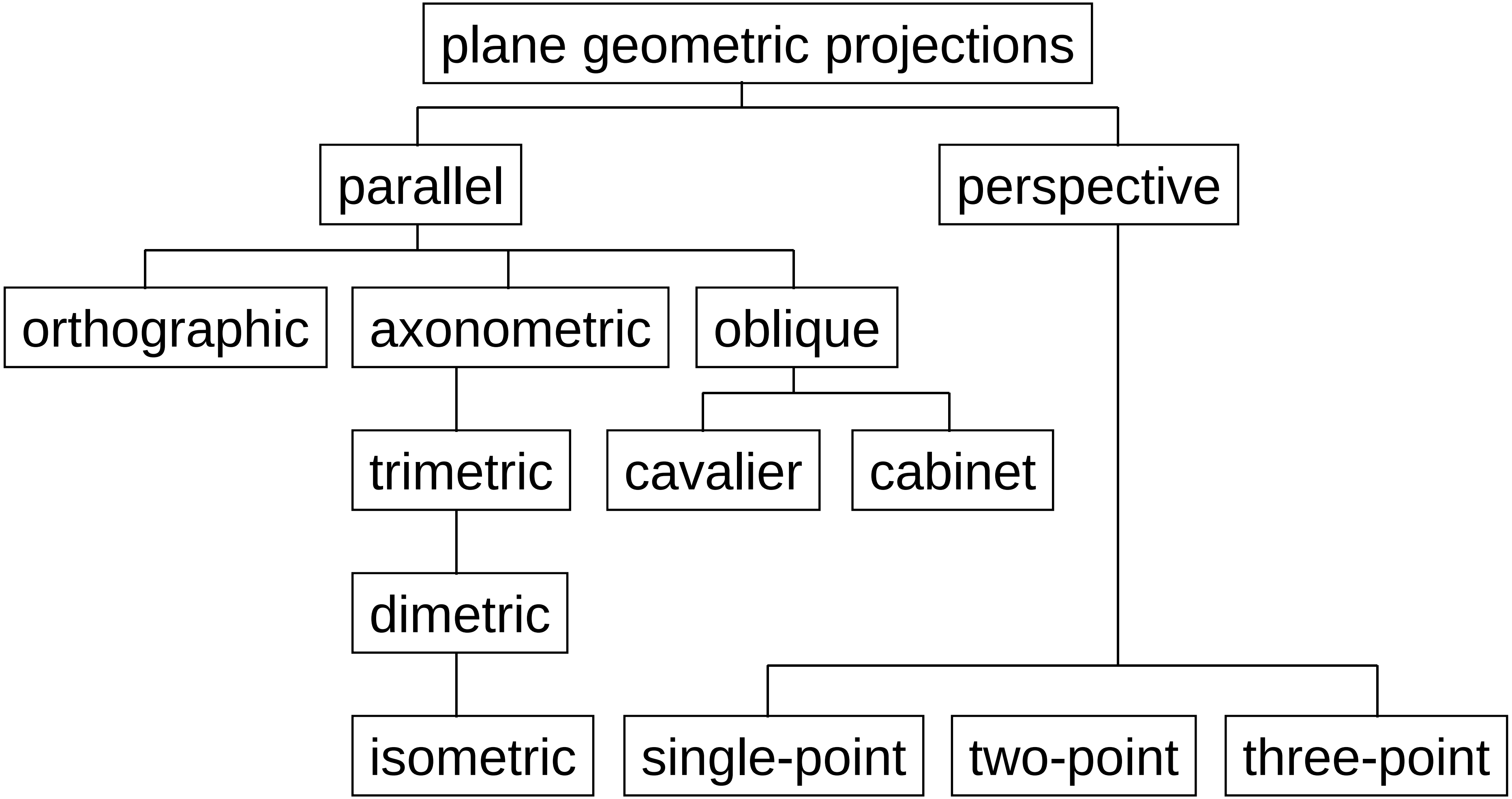
$$VUP \times R_z = \frac{1}{12.5}\det\begin{vmatrix} i & j & k \\ 0 & 0 & 1 \\ 6.0 & 8.0 & 7.5 \end{vmatrix} = \frac{1}{12.5}\begin{pmatrix} -8.0 & 6.0 & 0.0 \end{pmatrix}$$

$$R_x = \frac{VUP \times R_z}{\|VUP \times R_z\|} = (-0.8,0.6,0.0)$$

$$R_y = R_z \times R_x = (-0.35,-0.48,0.8)$$

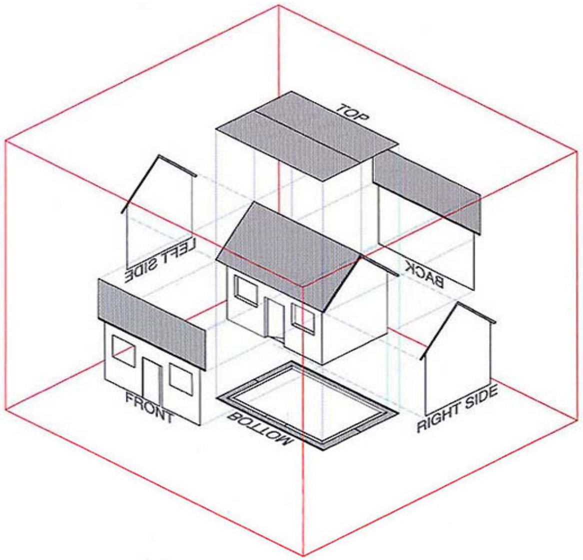
$$\mathbf{R} = \begin{bmatrix} \mathrm{R}_x \\ \mathrm{R}_y \\ \mathrm{R}_z \end{bmatrix}$$

Hierarchy of plane geometric projections



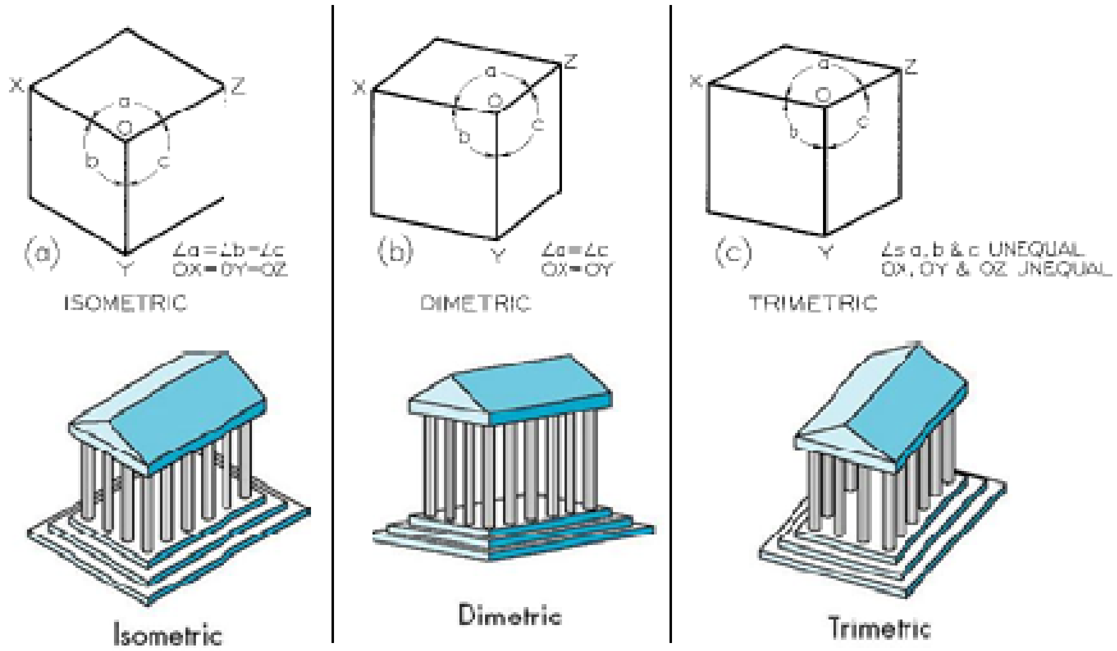
Parallel Projections

- Parallel direction of projection (DOP)
- Direction of projection (DOP) same for all points
- The *parallel projection* of the point (x,y,z) on the xy-plane gives $(x + az, y + bz, 0)$
  - When $a = b = 0$, the projection is said to be *orthographic* or *orthogonal*. Otherwise, it is *oblique*.
- Preserves relative dimension
- Orthographic parallel projections
  - The direction of projection is normal to the projection plane
  - Architectural, engineering drawings



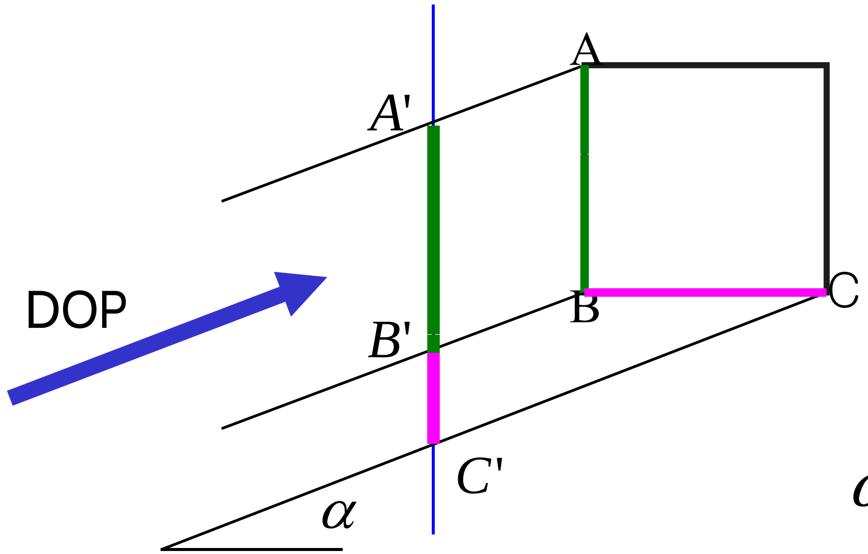
Axonometric Parallel Projection

- Orthographic parallel projection that displays more than one face of an object
- Projection plane intersects each principal axis
- Classify by how many identical angles of a corner of a projected cube.
  - Three: isometric
  - Two: dimetric
  - None : trimetric



Oblique Parallel Projections

- Parallel projection of which DOP (Direction of projection) is not perpendicular to the projection plane
- Only faces of the object parallel to the projection plane are shown true size and shape

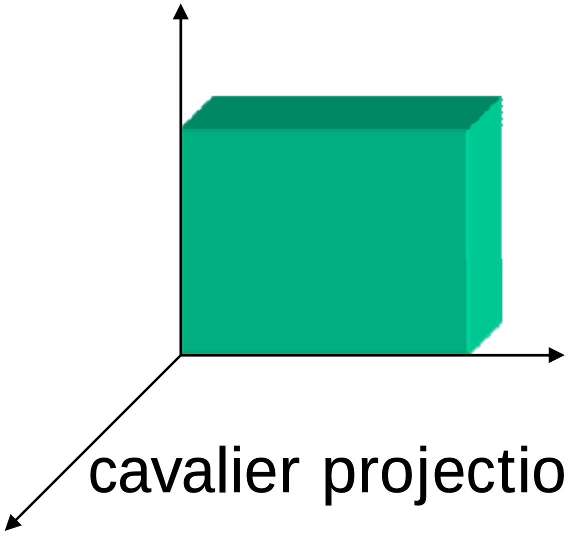


$$\overline{AB} = \overline{A'B'}$$

$$\overline{B'C'} = \frac{1}{2}\overline{BC}$$

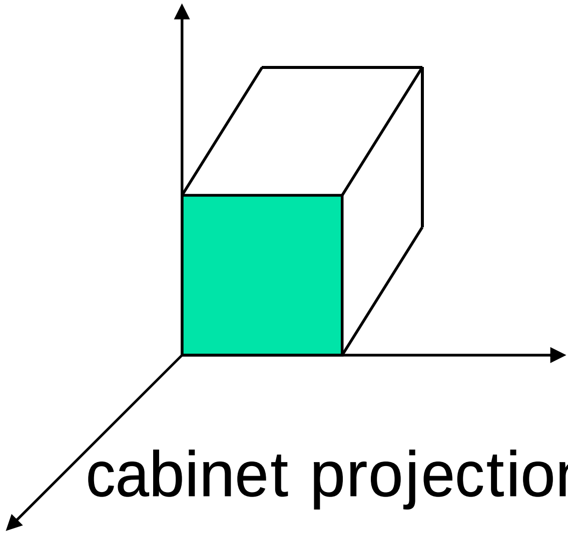
$$\alpha = \arctan(1/2) = 26.565°$$

Oblique Parallel Projections



cavalier projections

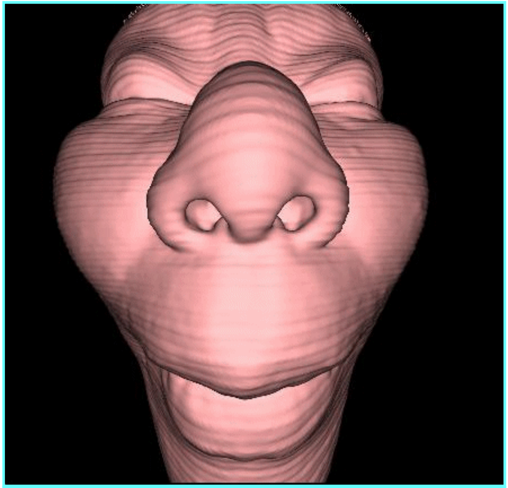
DOP≈45°



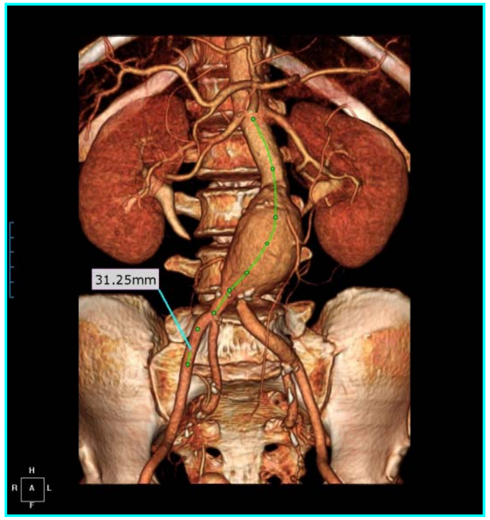
cabinet projections

DOP≈63.4°

Perspective vs. Parallel Projections



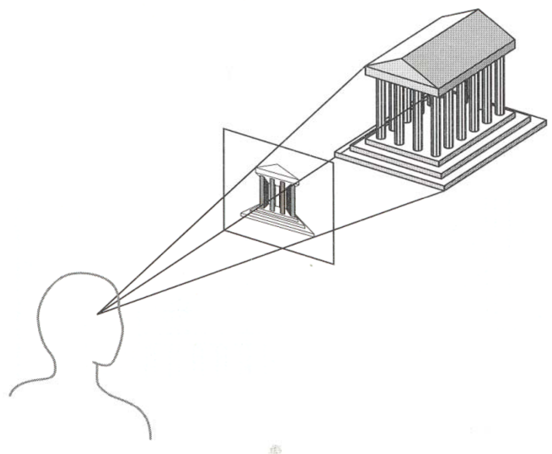
Perspective

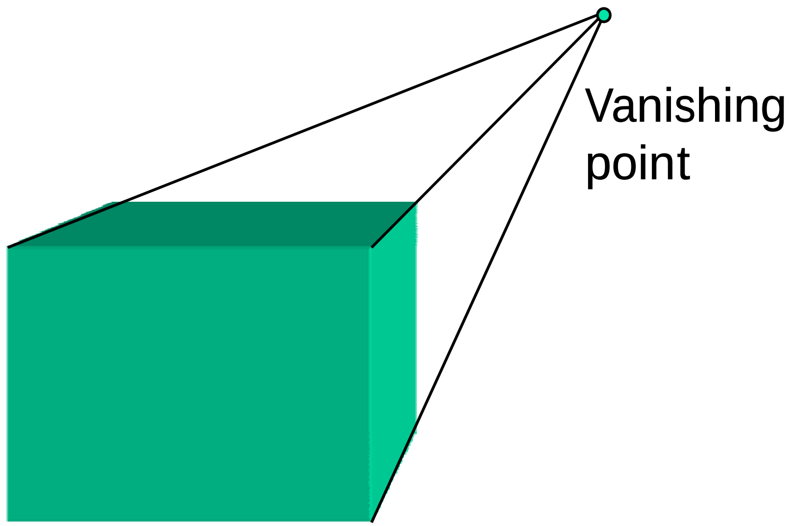


Parallel

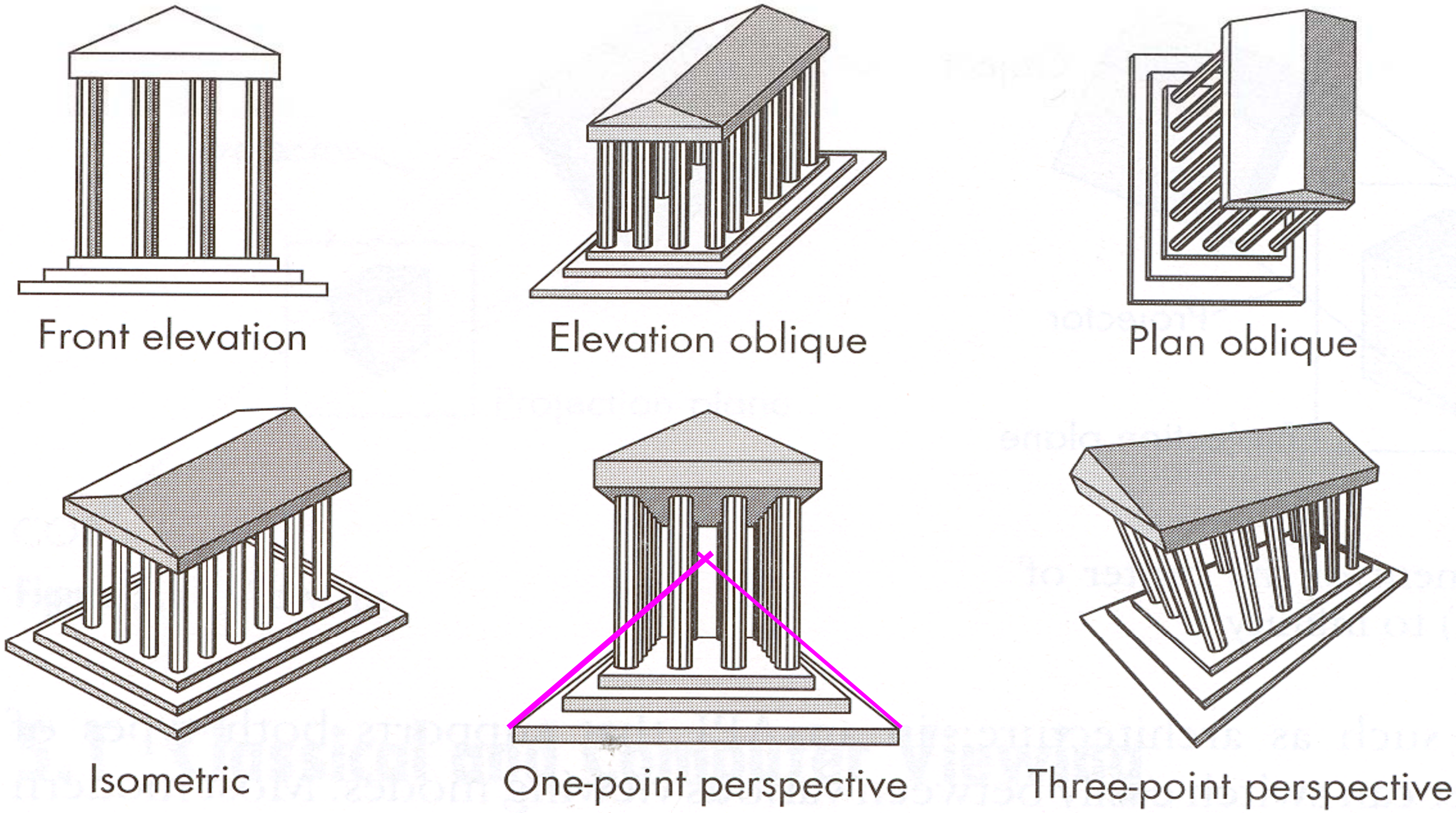
Perspective Projections

- Lines that are not parallel to the projection plane converge to a a single point in the projection (the <u>vanishing point)</u>
- Lines parallel to one of the major axis come to a vanishing point, these are called (principle) axis vanishing points. Only three axis vanishing points in 3D space.



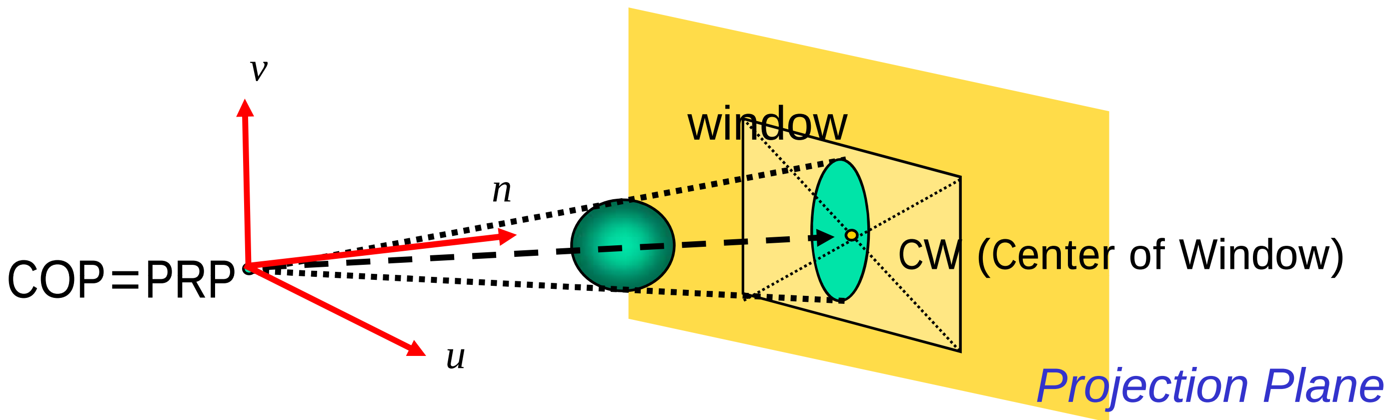


Projection Types



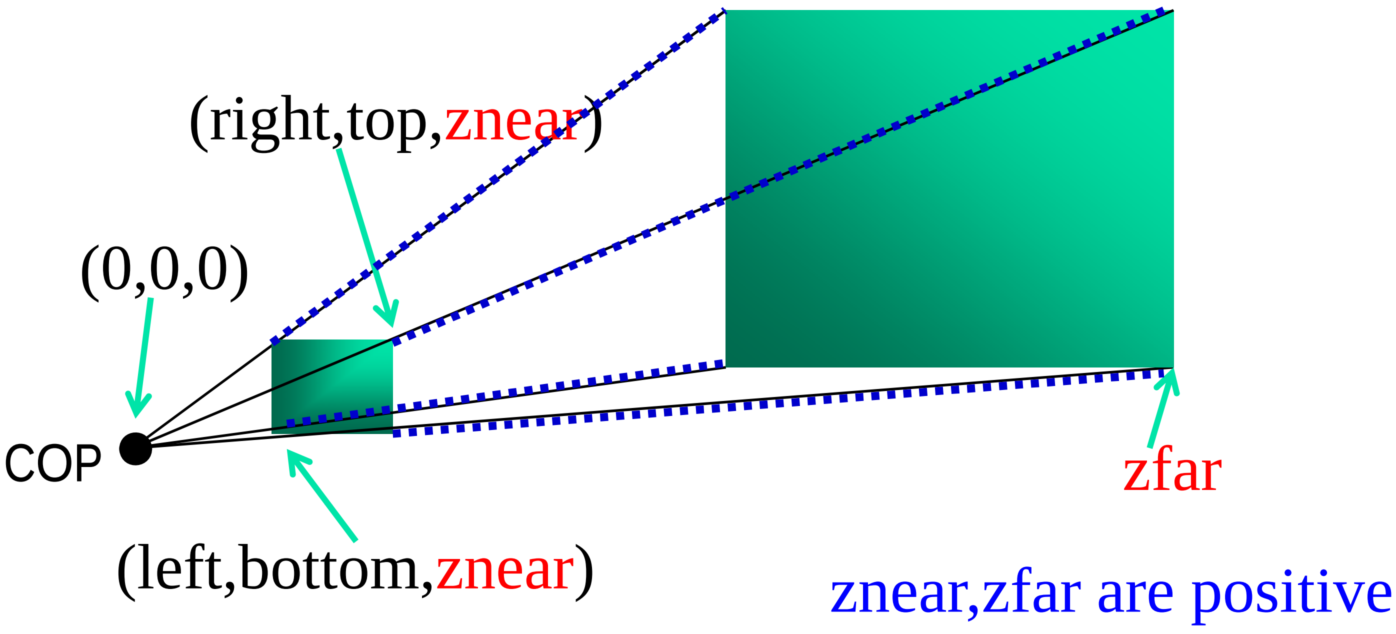
Projections

- Window
  - a rectangular region on the projection plane
- Projection Reference Point (PRP)
  - PRP is specified in the VRC system
  - In general, (0,0,0)
- Direction of projection (DOP): in a parallel projection, PRP $\Rightarrow$ CW
- Center of projection (COP): in a perspective projection, PRP = COP

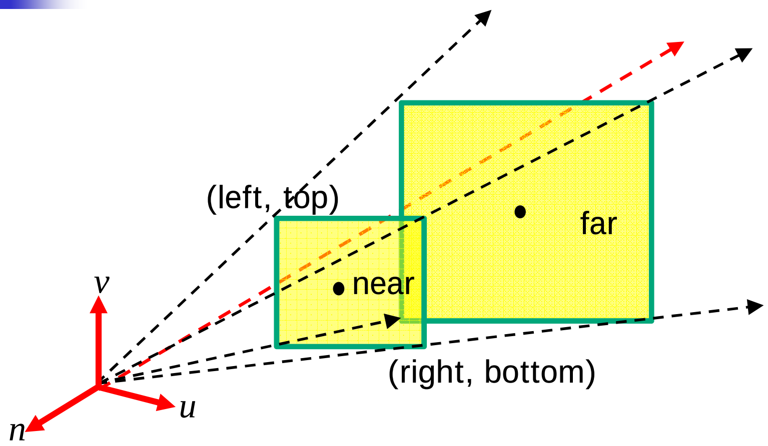


View Volume (View Frustum)

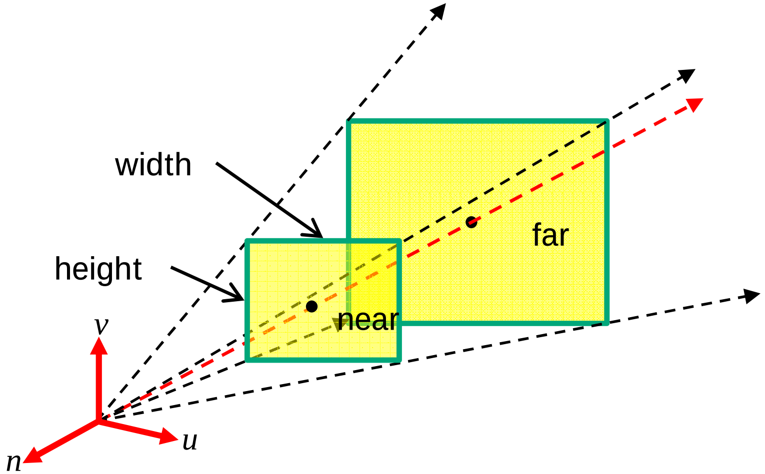
- 3D clipping region where we can see
- Defined by front and back clipping planes (which are parallel to view plane)



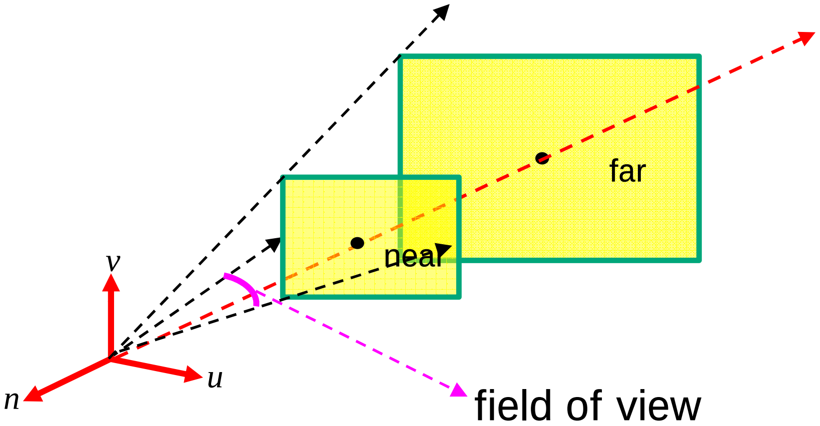
Create a View Volume



CW is not aligned n-axis
(left, right, top, bottom, near, far)



CW is aligned n-axis
(width, height, near, far)



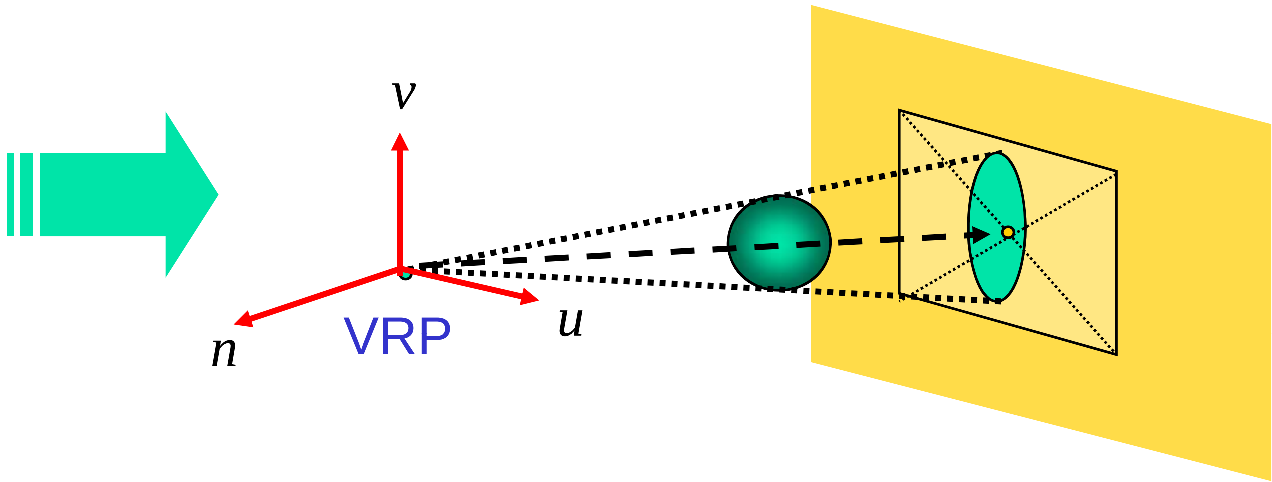
CW is aligned n-axis
(field of view, aspect, near, far)
aspect = width/height

3D Viewing

1) Establishing a View Reference Coordinate System

User supply the following parameters

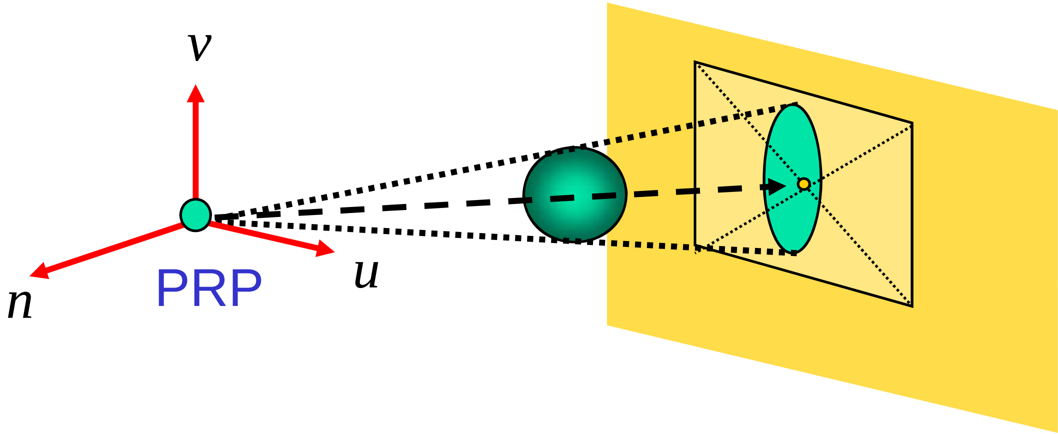
- the view reference point (VRP) - in WC
- VPN - a vector in WC
- VUP - a vector in WC



3D Viewing

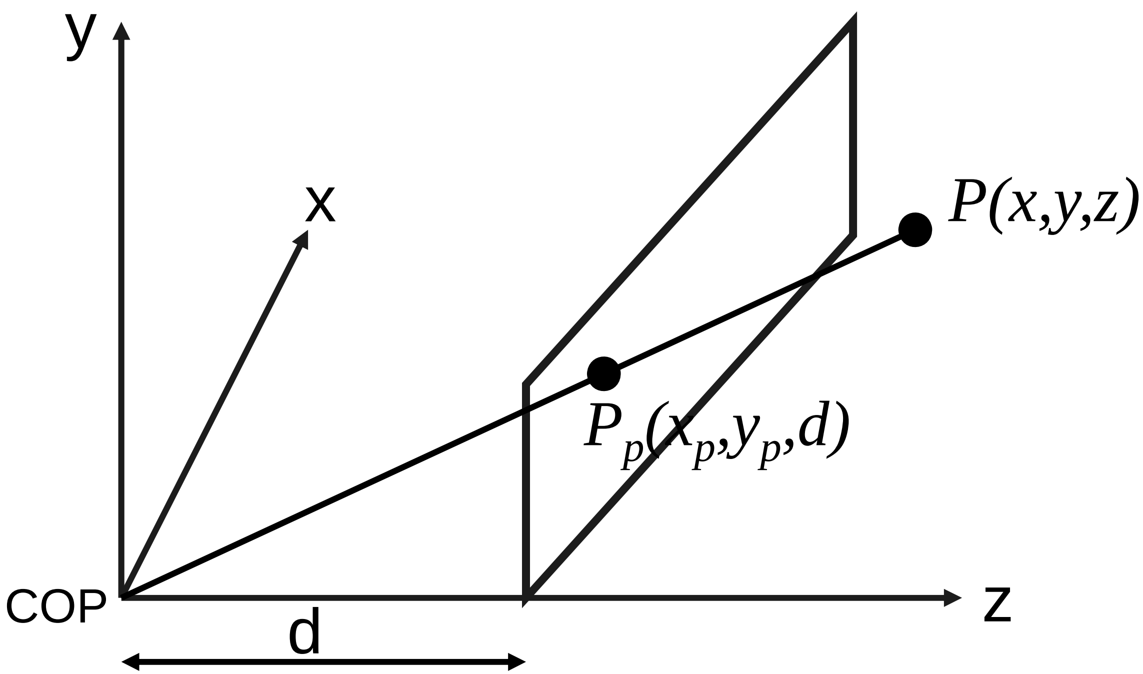
2) View Mapping (WC ➔ VRC ⇒ NPC)
   - the projection type
   - the Projection Reference Point (PRP)
   - a view plane distance (= z_{vp})
     - In DirectX, view plane is identical to near plane
   - a back plane and a front plane distance

3) device-dependent transformation



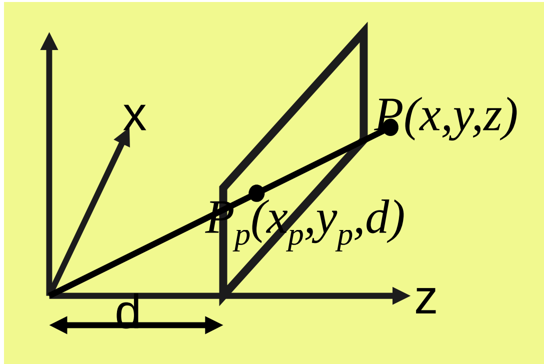
Planar Geometric Projections

- Mathematics of Planar Geometric Projections



Centre of projection at the origin
Projection plane at z=d

Planar Geometric Projections



$$\frac{x_p}{d} = \frac{x}{z}; \qquad \frac{y_p}{d} = \frac{y}{z}$$

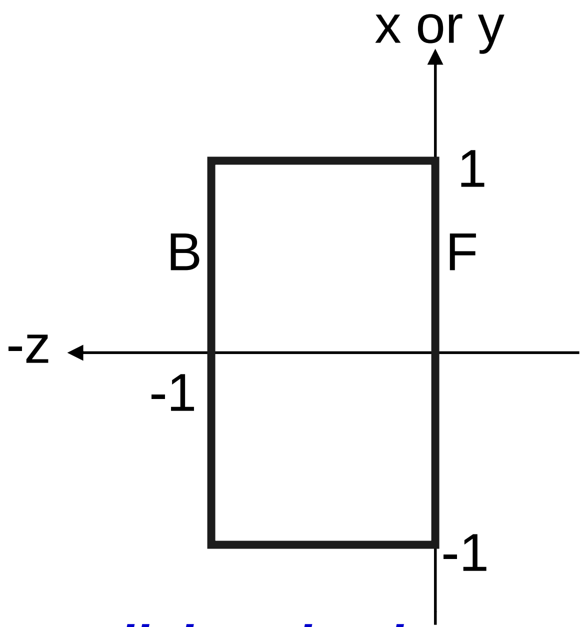
$$\Rightarrow x_p = \frac{d \cdot x}{z} = \frac{x}{z/d}; \qquad y_p = \frac{d \cdot y}{z} = \frac{y}{z/d}$$

$$\Rightarrow \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = M_{per} \cdot P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

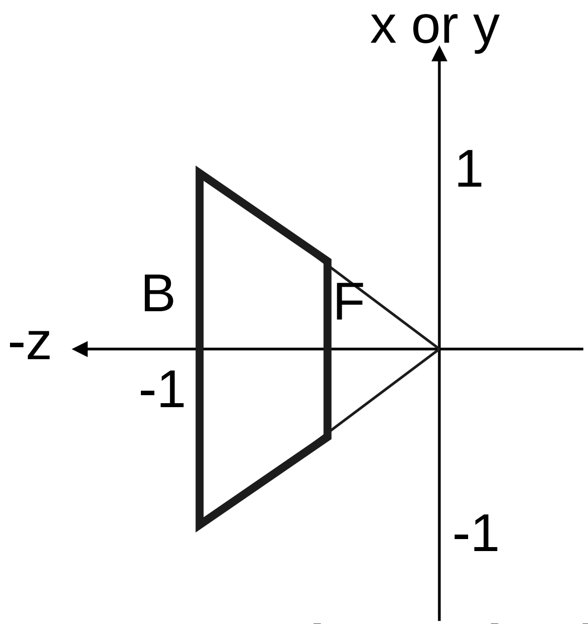
$$\Rightarrow \left(\frac{X}{W}, \frac{Y}{W}, \frac{X}{W} \right) = (x_p, y_p, z_p) = \left(\frac{x}{z/d}, \frac{y}{z/d}, d \right)$$

Normalizing Transformation

- Transform an arbitrary parallel- or perspective-projection view volume into the normalized or canonical view volume (in DirectX)



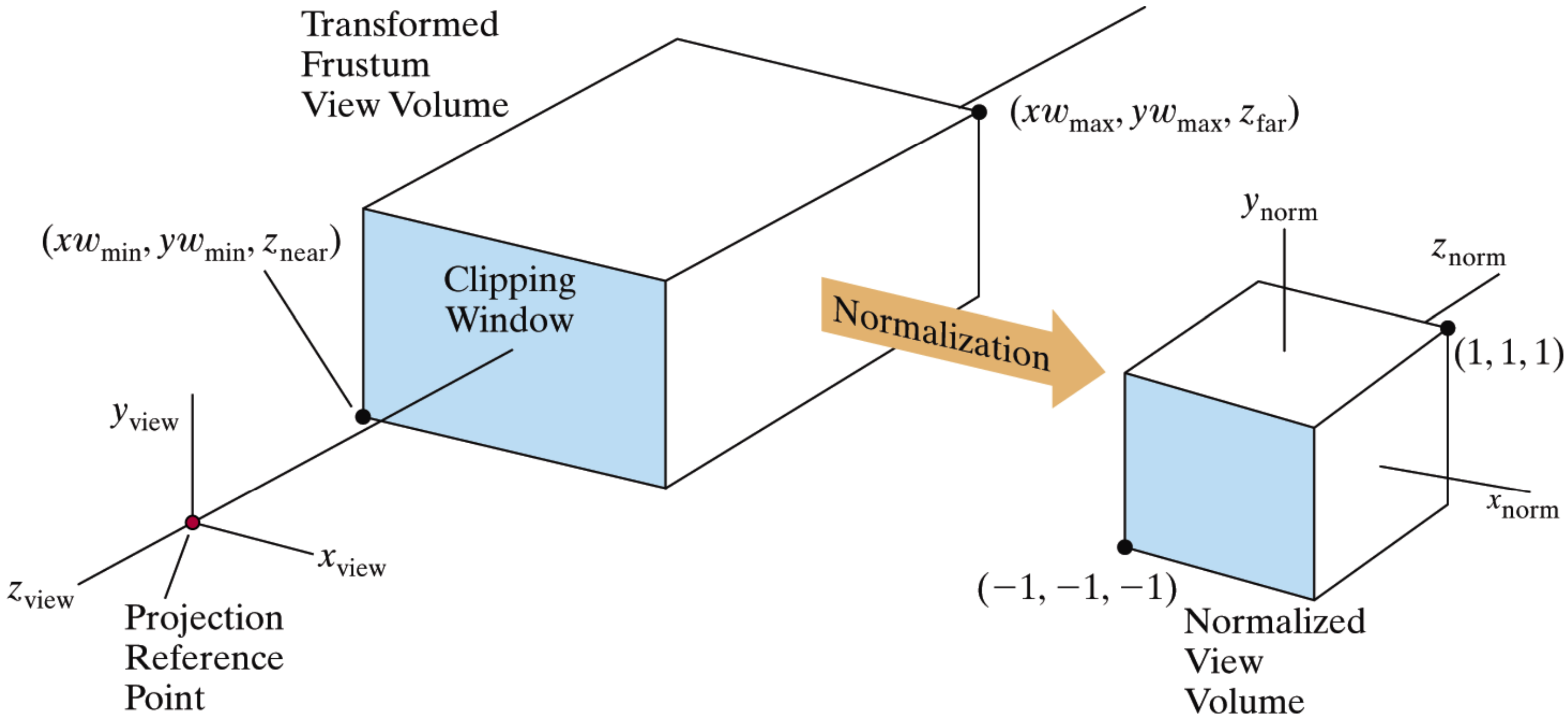
parallel-projection canonical view volume



perspective-projection canonical view volume

Normalizing Transformation

OpenGL



Normalizing transformation for parallel projections

1. Translate VRP to the origin of the WC
2. Rotate VRC such that *n axis = z axis, u axis = x axis and v axis = y axis*.

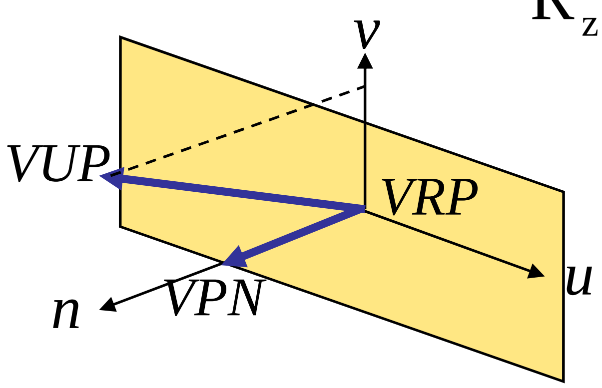
   (By 1 & 2, transformation from WCS to VRC)
3. Shear so the direction of projection parallel to the z axis. (not necessary for orthographic projections)
4. *Translate and scale into the parallel-projection canonical view volume*

Normalizing transformation for parallel projections

[Step2]

$$R_z(\theta)R_y(\phi)R_x(\alpha)$$

Or use orthogonal matrix properties



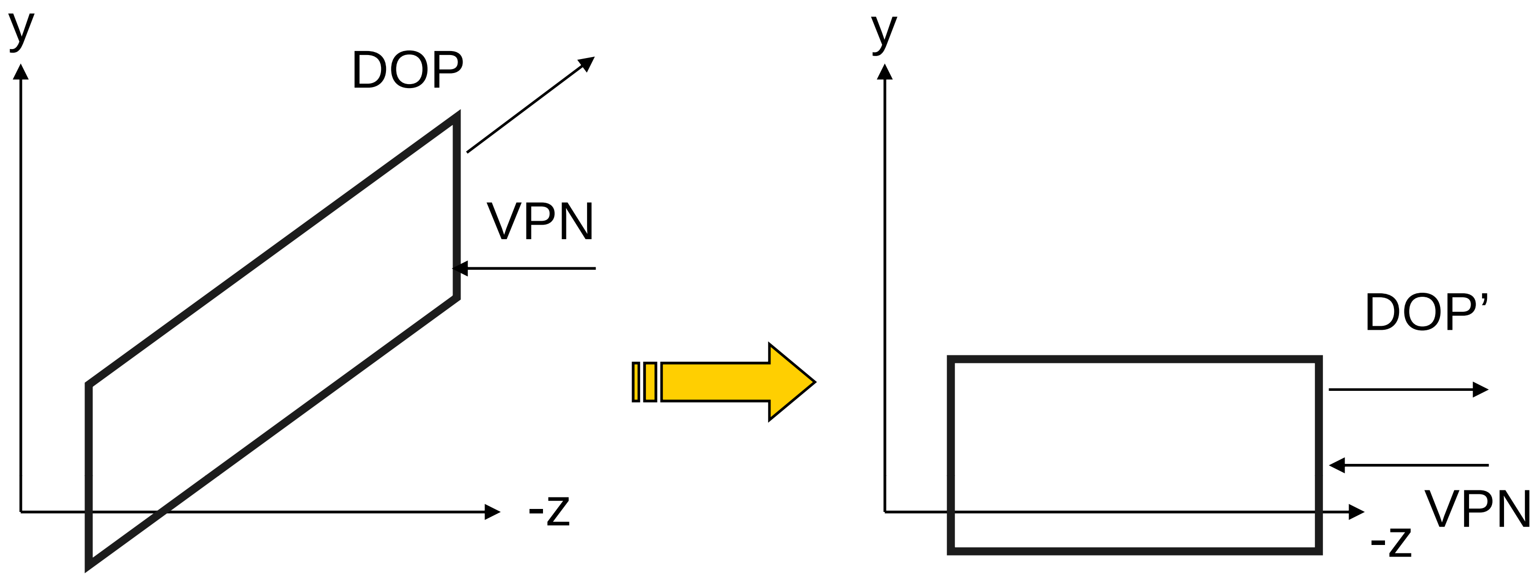
$$\mathrm{R_z} = \frac{\mathrm{VPN}}{\|\mathrm{VPN}\|} \quad \mathrm{R_x} = \frac{\mathrm{VUP} \times \mathrm{R_z}}{\|\mathrm{VUP} \times \mathrm{R_z}\|} \quad \mathrm{R_y} = \mathrm{R_z} \times \mathrm{R_x}$$

$$\mathrm{R} = \begin{bmatrix} r_{1x} & r_{2x} & r_{3x} & 0 \\ r_{1y} & r_{2y} & r_{3y} & 0 \\ r_{1z} & r_{2z} & r_{3z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Normalizing transformation for parallel projections

[Step 3] shearing

- after step2, VRC = WC
- DOP = CW - PRP



Normalizing transformation for parallel projections

- z-component of DOP is invariant.

$$SH_z(a,b) = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

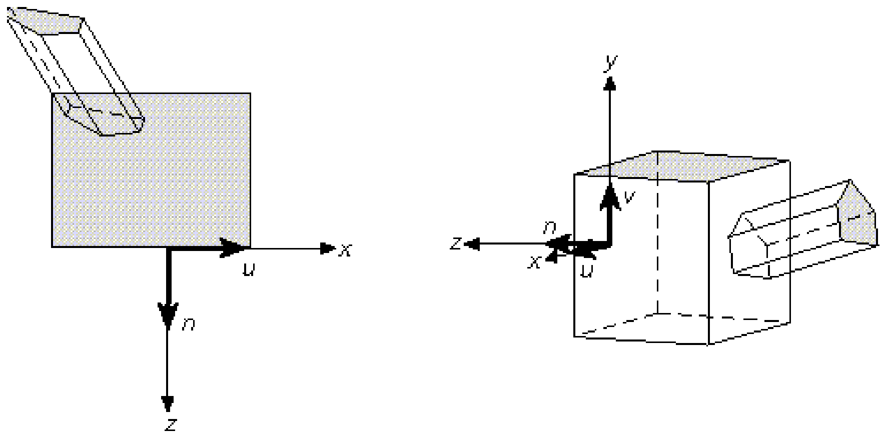
$$DOP_{wc} = CW - PRP = \begin{pmatrix} dop_x & dop_y & dop_z \end{pmatrix}$$

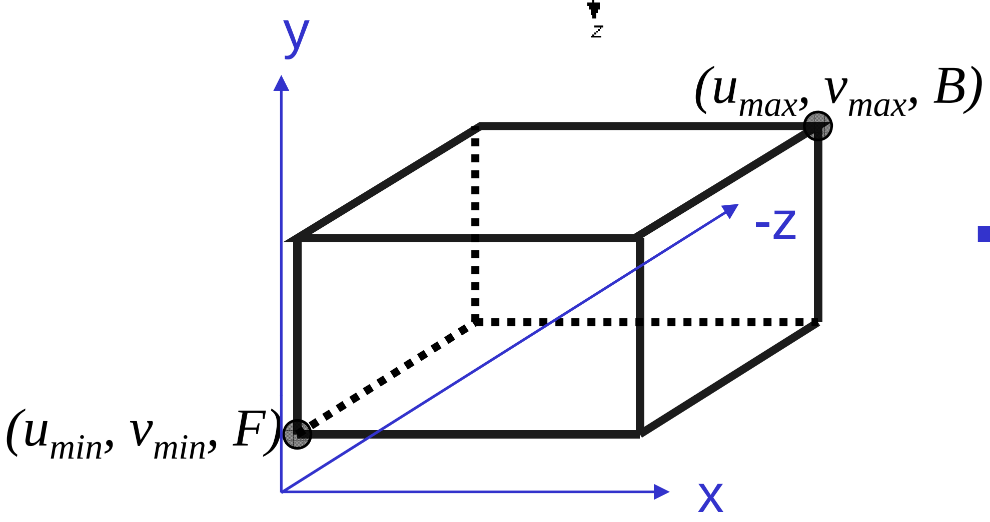
$$DOP' = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot DOP_{wc} = \begin{bmatrix} 0 \\ 0 \\ dop_z \\ 1 \end{bmatrix}$$

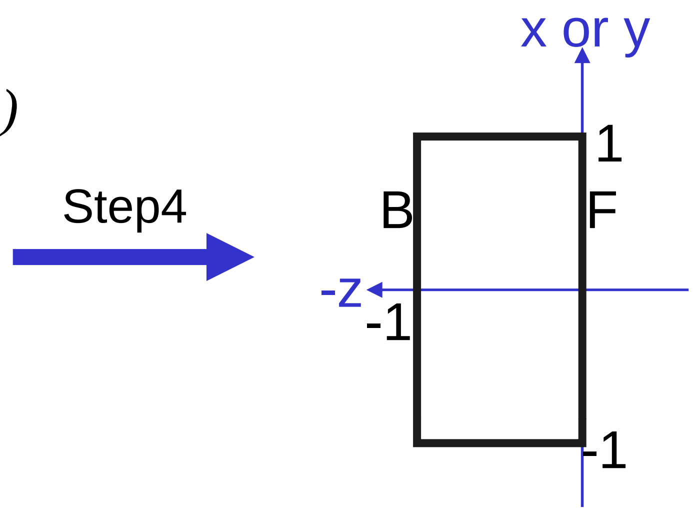
$$a = -\frac{dop_x}{dop_z}, \qquad b = -\frac{dop_y}{dop_z}$$

Normalizing transformation for parallel projections

- View volume after transformation steps 1 to 3







Normalizing transformation for parallel projections

[Step 4] translate and scale

1. Translate the front center of the view volume

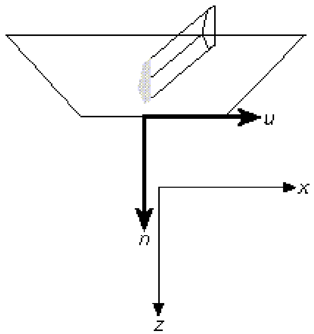
$$T_{par} = T\left(-\frac{u_{\max}+u_{\min}}{2}, -\frac{v_{\max}+v_{\min}}{2}, -F\right)$$

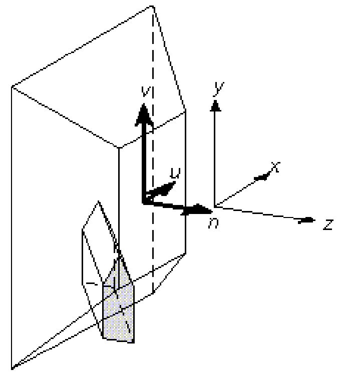
2. Scale to the $2 \times 2 \times 1$ size

$$S_{par} = S\left(\frac{2}{u_{\max}-u_{\min}}, \frac{2}{v_{\max}-v_{\min}}, \frac{1}{F-B}\right)$$

Normalizing transformation for perspective projection

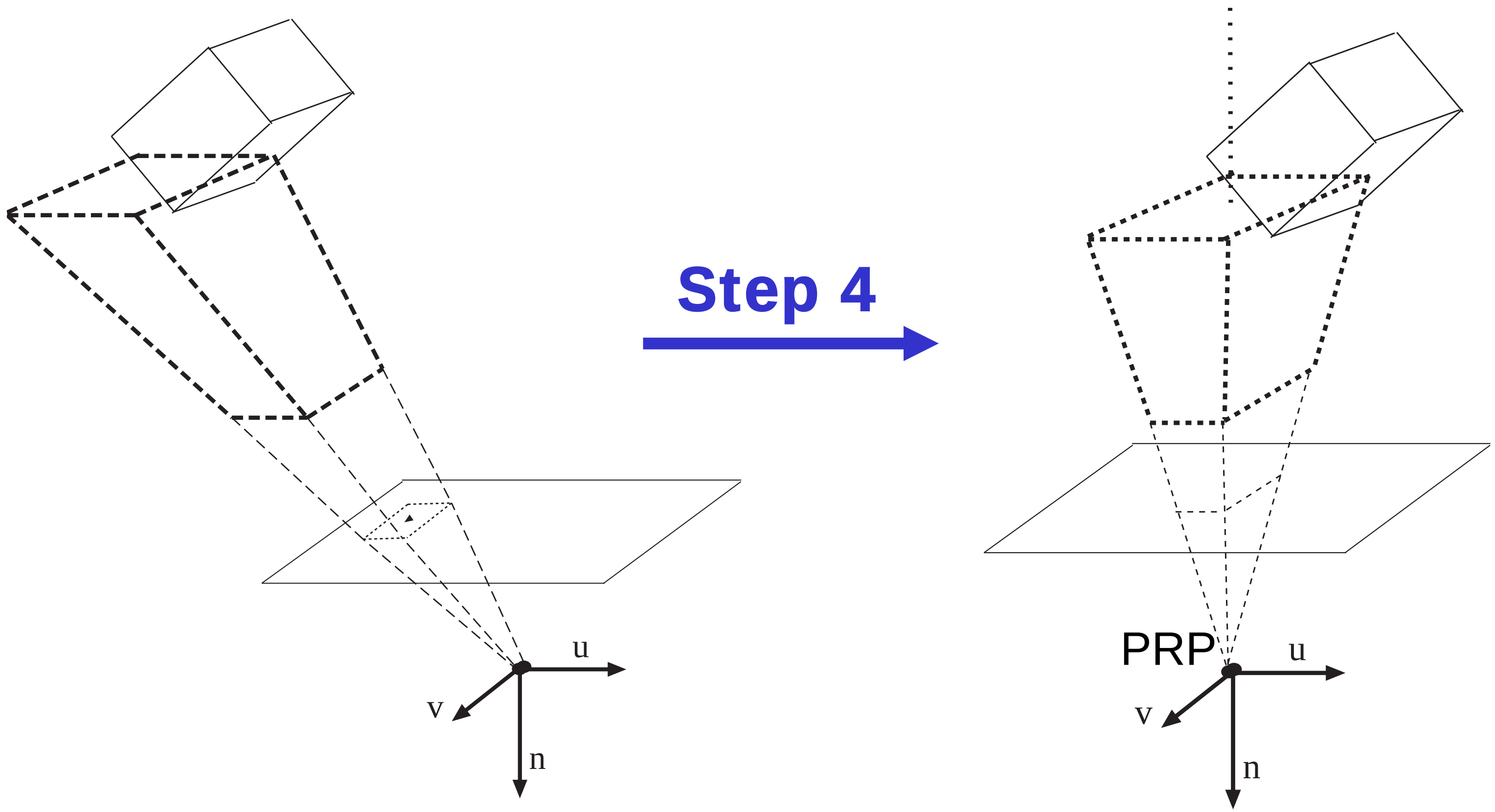
1. Translate VRP to the origin of the WC: T(–VRP)
2. Rotate VRC such that n axis = z axis, u axis = x axis and v axis = y axis
3. Translate such that PRP=(prp_u, prp_v, prp_n) is at the origin: T(–PRP)
4. Shear so the center line of the view volume becomes the z-axis
5. (*Scale such that the view volume becomes the canonical perspective view volume*.)





Normalizing transformation for perspective projection

After step1,2,3



Normalizing transformation for perspective projection

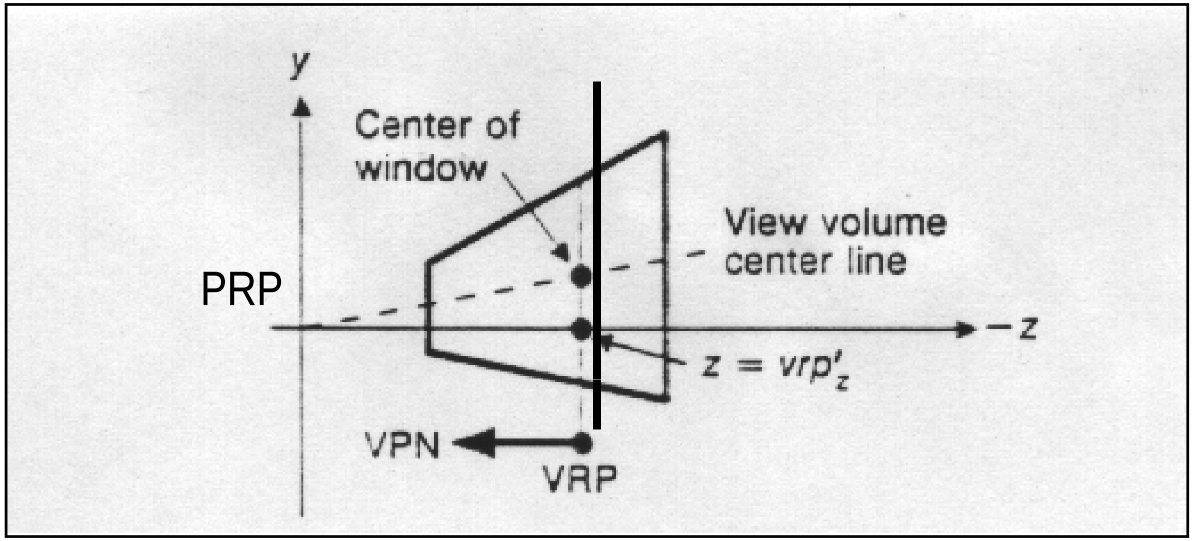
[Step 4] shearing

shear so that CW – PRP is into –z axis

$\Rightarrow$ $SH_{per} = Sh_{par}$

<u>Another Way</u>: VRP' = SH_{per} T(-PRP) $[0\ 0\ 0\ 1]^T$

z component of VRP': $vrp_z' = -prp_n$



Normalizing transformation for perspective projection

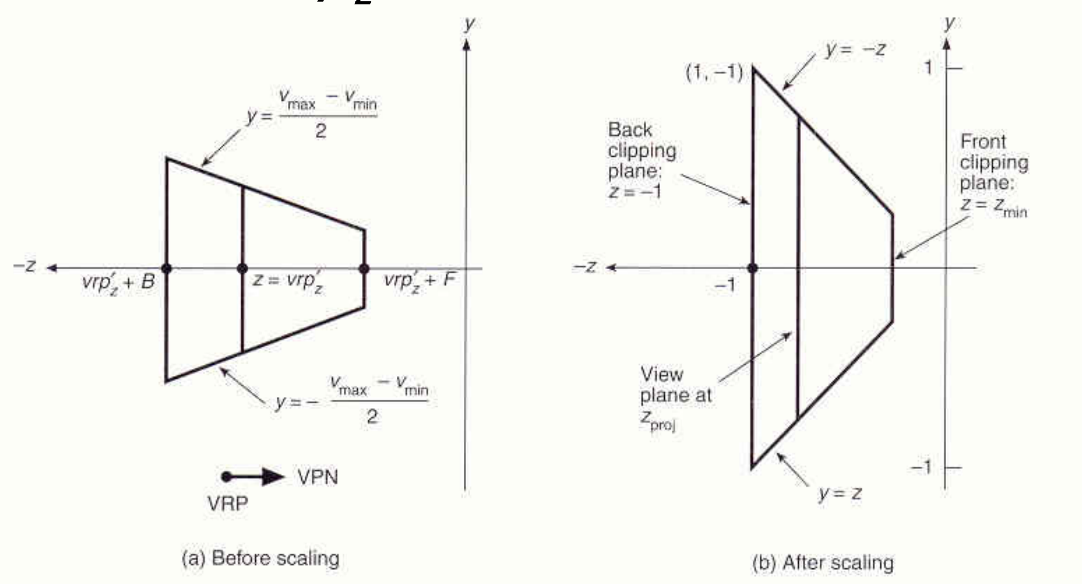
[Step 5] scale

1. Scale x and y to give the sloped planes bounding the view-volume unit slope.

   ⇨ Scale the window so its half-height and half-width are both $-vrp_z'$

$$\text{x scale}: \frac{-2 \cdot vrp'_z}{(u_{\max} - u_{\min})}$$

$$\text{y scale}: \frac{-2 \cdot vrp'_z}{(v_{\max} - v_{\min})}$$



Normalizing transformation for perspective projection

2. Scale uniformly all three axes such that the back clipping plane $z = vrp'_z + B$ becomes –1.

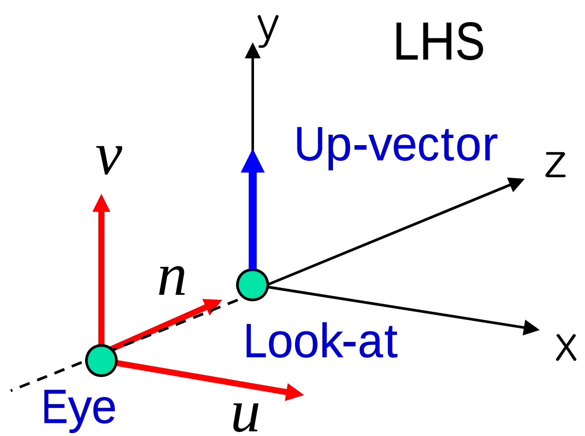
   $\Rightarrow$ scale factor: $-1/(vrp'_z + B)$

Perspective scale transformation

$$S_{per} = S\left(\frac{2vrp'_z}{(u_{\max} - u_{\min})(vrp'_z + B)}, \frac{2vrp'_z}{(v_{\max} - v_{\min})(vrp'_z + B)}, \frac{-1}{(vrp'_z + B)}\right)$$

DirectX: Viewing Transformation

- How to define VRC
  - **Eye-Point (= VRP)**
  - **Look-At Position**
    - **Look-At Position – Eye-Point ➔ N**
  - **Up-Vector (➔ v)**



Syntax

D3DXMATRIX *D3DXMatrixLookAtLH(
D3DXMATRIX **pOut*, CONST D3DXVECTOR3 **pEye*,
CONST D3DXVECTOR3 **pAt*, CONST D3DXVECTOR3 **pUp*);

Parameters

pOut : Pointer to the D3DXMATRIX structure that is the result of the operation.
pEye : Pointer to the D3DXVECTOR3 structure that defines the eye point.
pAt : Pointer to the **D3DXVECTOR3** structure that defines the camera look-at target.
pUp : Pointer to the **D3DXVECTOR3** structure that defines the current world's up, usually [0, 1, 0].

DirectX: Perspective Projection Transformation

When CW aligns n-axis

Syntax

D3DXMATRIX *D3DXMatrixPerspectiveFovLH(
  D3DXMATRIX **pOut*, FLOAT *fovy*, FLOAT *Aspect*, FLOAT *zn*, FLOAT *zf*);

Parameters

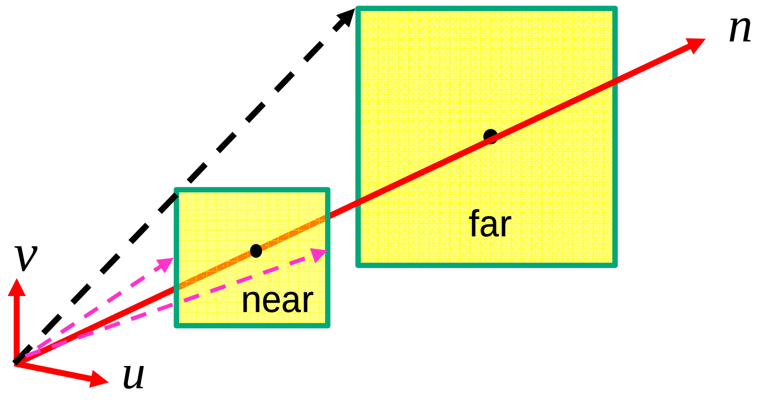
pOut : Pointer to the D3DXMATRIX structure that is the result of the operation.

fovy : Field of view in the *y* direction, in radians.

Aspect : Aspect ratio, defined as view space width divided by height.

zn : Z-value of the near view-plane.

zf : Z-value of the far view-plane.



DirectX: Perspective Projection Transformation

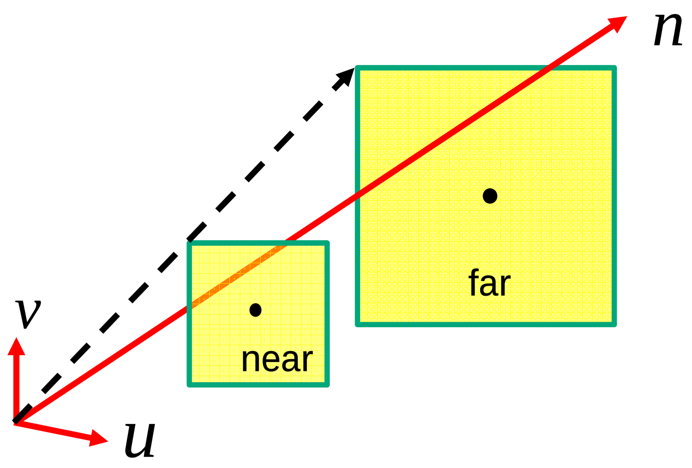
When CW does not align n-axis

Syntax

D3DXMATRIX *D3DXMatrixPerspectiveOffCenterLH
(D3DXMATRIX **pOut*, FLOAT *l*, FLOAT *r*, FLOAT *b*, FLOAT *t*,
FLOAT *zn*, FLOAT *zf*);

Parameters

pOut : Pointer to the D3DXMATRIX structure that is the result of the operation.
l : Minimum x-value of the view volume.
r : Maximum x-value of the view volume.
b : Minimum y-value of the view volume.
t : Maximum y-value of the view volume.
zn : Minimum z-value of the view volume.
zf : Maximum z-value of the view volume.



DirectX: Orthographic Parallel Projection Transformation

Syntax

D3DXMATRIX *WINAPI D3DXMatrixOrthoLH(
D3DXMATRIX **pOut*, FLOAT *w*, FLOAT *h*, FLOAT *zn*, FLOAT *zf*);

Parameters

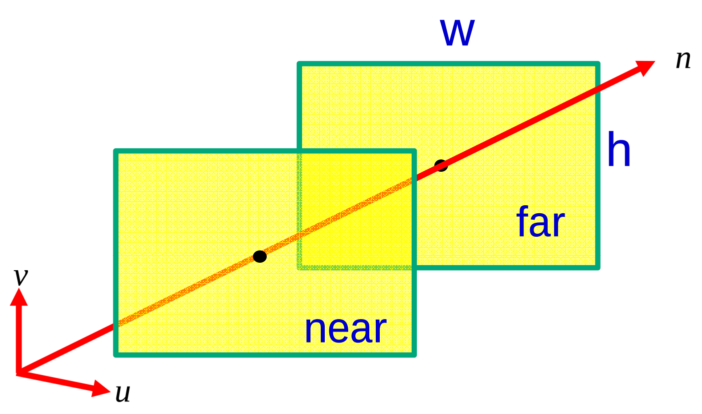
pOut : Pointer to the D3DXMATRIX structure that contains the resulting matrix.

w : Width of the view volume.

h : Height of the view volume.

zn : Minimum z-value of the view volume which is referred to as z-near.

zf : Maximum z-value of the view volume which is referred to as z-far.



DirectX: Oblique Parallel Projection Transformation

Syntax

D3DXMATRIX *D3DXMatrixOrthoOffCenterLH
(D3DXMATRIX **pOut*, FLOAT *l*, FLOAT *r*, FLOAT *b*, FLOAT *t*,
FLOAT *zn*, FLOAT *zf*);

Parameters

pOut : Pointer to the D3DXMATRIX structure that is the result of the operation.

l : Minimum x-value of view volume.

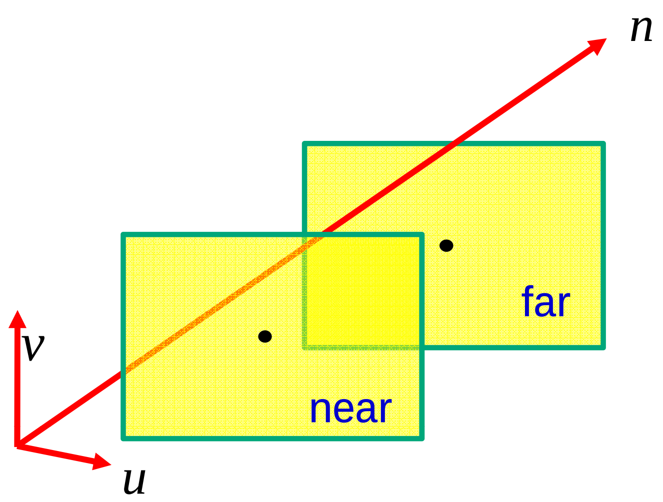
r : Maximum x-value of view volume.

b : Minimum y-value of view volume.

t : Maximum y-value of view volume.

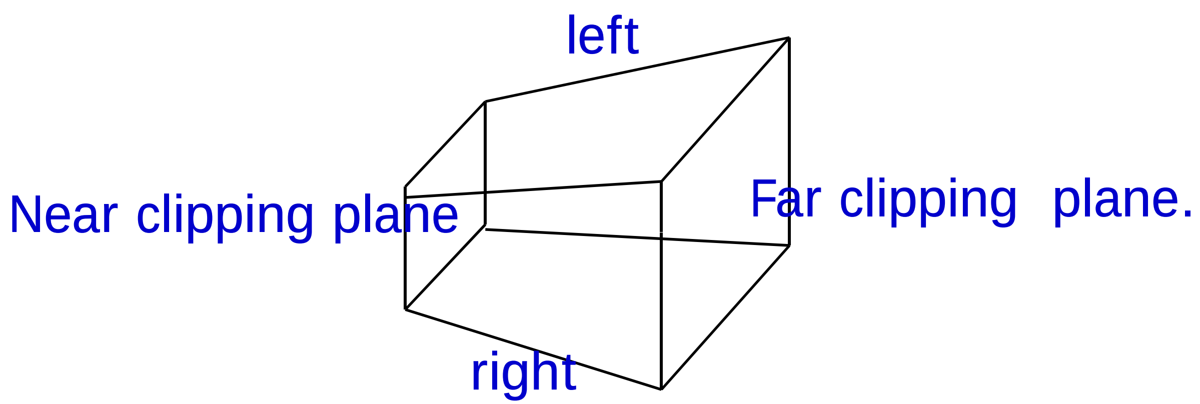
zn : Minimum z-value of the view volume.

zf : Maximum z-value of the view volume.



3D Clipping

- For orthographic projection, view volume is a box
- For perspective projection, view volume is a *frustum*



$\Longleftrightarrow$ Need to calculate intersection with 6 planes

3D Clipping

- Clipping is efficiently done on the normalized view volume.
- The canonical parallel projection view volume is defined by:

$$-1 \le x \le 1,\ -1 \le y \le 1\ ,\ -1 \le z \le 0$$

- Clip primitives against this view volume

3D Region Coding for Clipping

- 3-D Extension of 2-D Cohen-Sutherland Algorithm

For parallel-projection canonical view volume :

Bit 1 - point is above view volume: $y > 1$

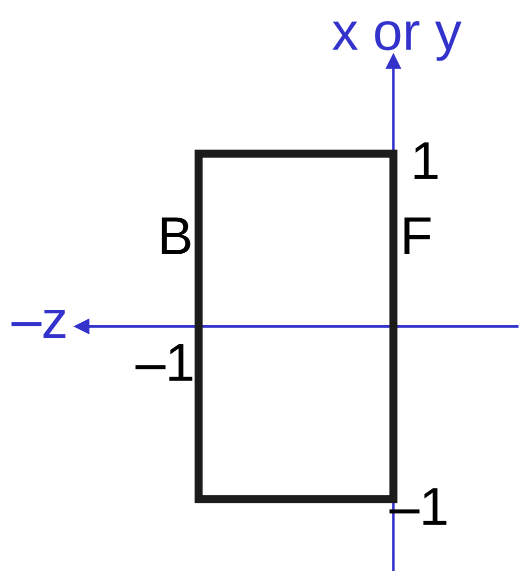
Bit 2 - point is below view volume: $y < -1$

Bit 3 - point is right of view volume: $x > 1$

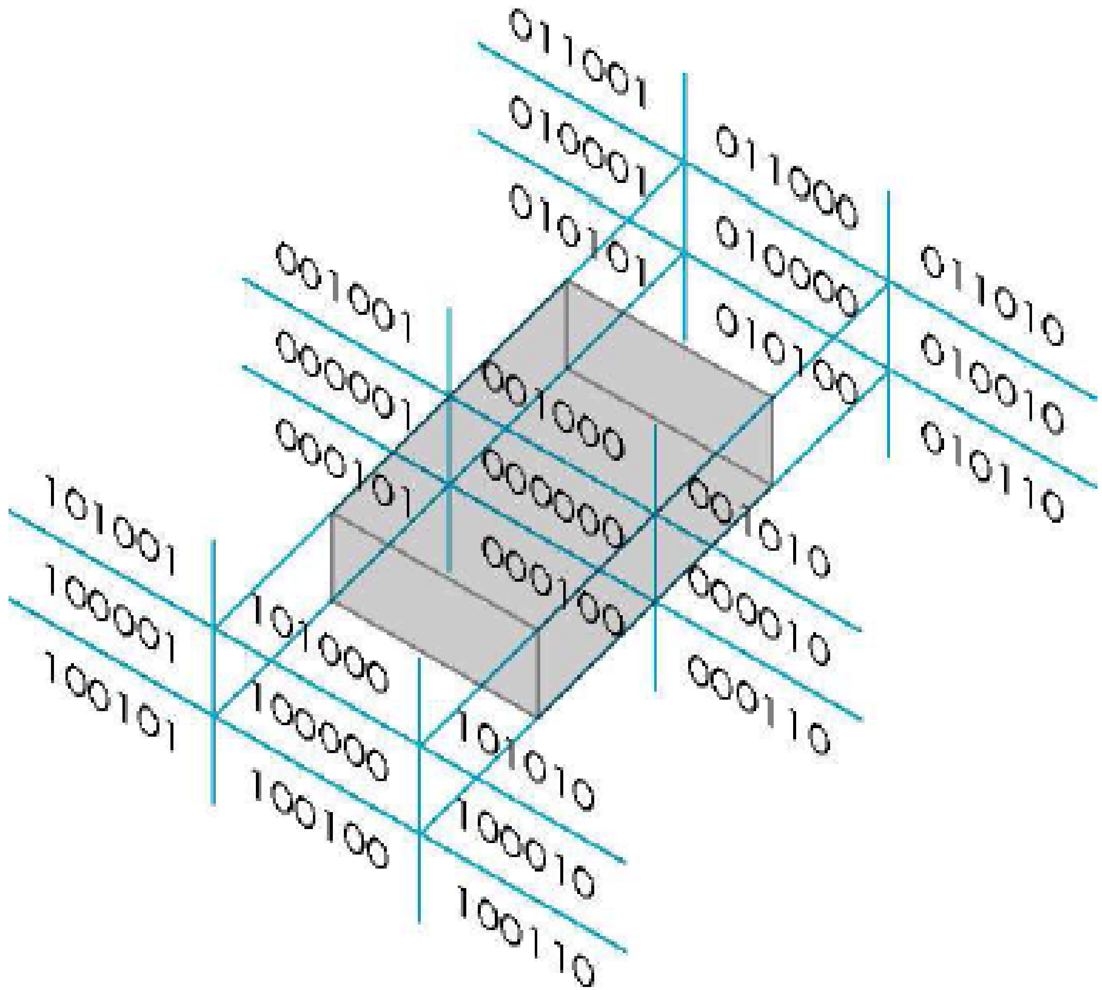
Bit 4 - point is left view volume: $x < -1$

Bit 5 - point is behind view volume: $z < -1$

Bit 6 - point is in front of view volume: $z > 0$

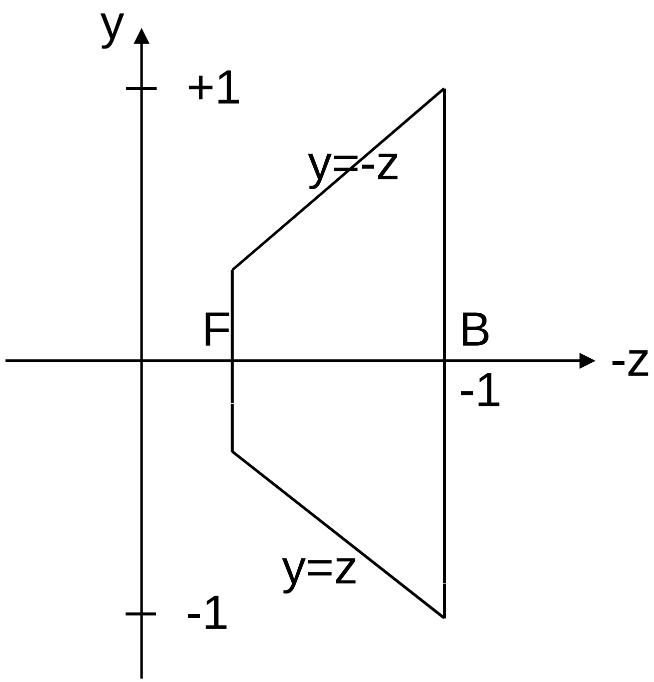


3D Region Coding for Clipping



3D Region Coding for Clipping

For perspective-projection canonical view volume



Bit 1 - Point is above view volume $y > -z$

Bit 2 - Point is below view volume $y < z$

Bit 3 - Point is right of view volume $x > -z$

Bit 4 - Point is left of view volume $x < z$

Bit 5 - Point is behind view volume $z < -1$

Bit 6 - Point is in front of view volume $z > z_{min}$

Clipping and Homogeneous Coordinates

- Efficient to transform frustum into perspective canonical view volume – unit slope planes
- Even better to transform to parallel canonical view volume
  - Clipping must be done in homogeneous coordinates
  - We do not need to $[X,Y,Z,W] \Rightarrow [x,y,z,1]$ for the clipped region
- Points in homogeneous coordinate can appear with $-W$ and cannot be clipped properly in 3D

Clipping and Homogeneous Coordinates

- 3D parallel projection volume is defined by:

$$-1 \le x \le 1,\ -1 \le y \le 1\ ,\ -1 \le z \le 0$$

- Replace by X/W,Y/W,Z/W:

$$-1 \le X/W \le 1,\ -1 \le Y/W \le 1\ ,\ -1 \le Z/W \le 0$$

- Corresponding plane equations are :

$$X = -W,\ X = W,\ Y = -W,\ Y = W,\ Z = -W,\ Z = 0$$

- If W>0 , multiplication by W does not change sign.

$$W > 0:\ -W \le X \le W,\ -W \le Y \le W,\ -W \le Z \le 0$$

- However if W<0, need to change sign :

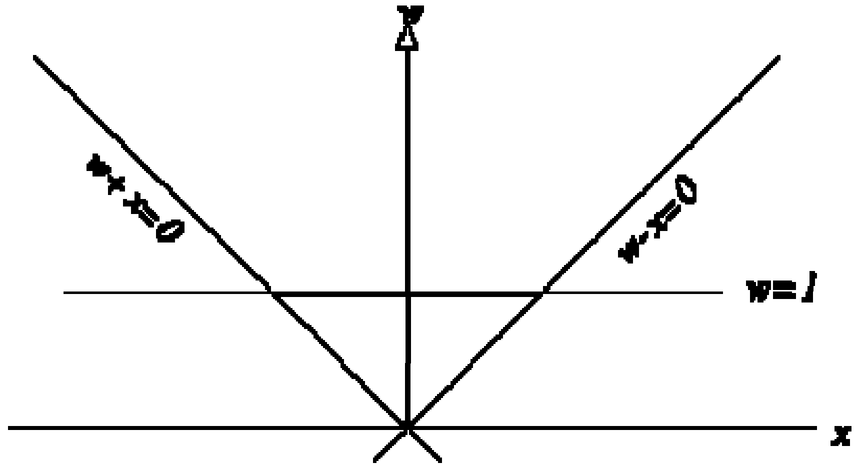
$$W < 0:\ -W \ge X \ge W,\ -W \ge Y \ge W,\ -W \ge Z \ge 0$$

Clipping and Homogeneous Coordinates

- For the canonical parallel projection volume:

  $-1 \le x \le 1,\ -1 \le y \le 1,\ -1 \le z \le 0$

- To clip to $x = -1$ (left):
  - Homogeneous coordinate: Clip to $\mathbf{X}/\mathbf{W} = -1$
  - Homogeneous plane: $\mathbf{W}+\mathbf{X} = 0$
  - Point is visible if $\mathbf{W}+\mathbf{X} > 0$

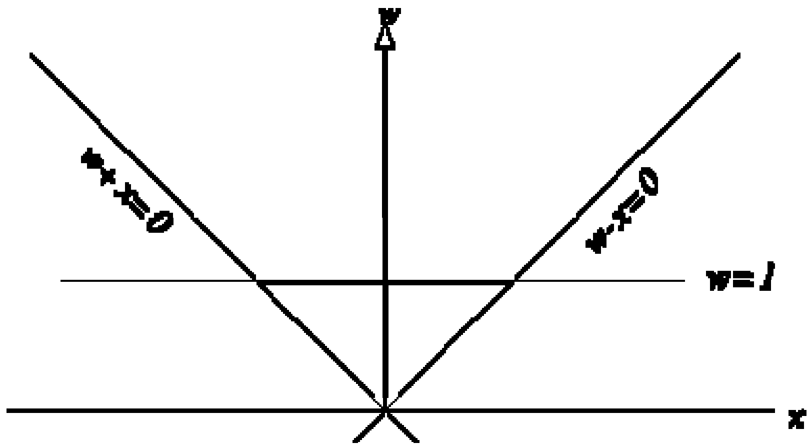


Clipping and Homogeneous Coordinates

- The intersection of the line segment with a clipping plane:

$$P = (1-\alpha)P_1 + \alpha P_2 \quad \text{and} \quad w + x = 0$$

$$\Longrightarrow \quad [(1-\alpha)w_1 + \alpha w_2] + [(1-\alpha)x_1 + \alpha x_2] = 0$$



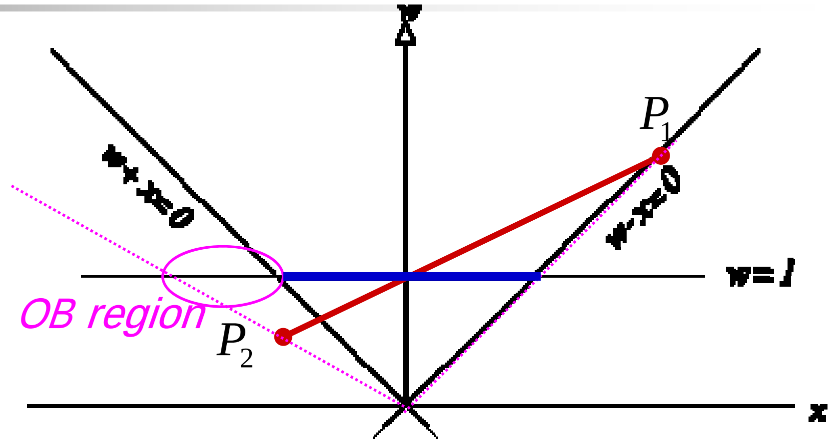
$$\Longrightarrow \quad \alpha = \frac{x_1 + w_2}{(w_1 + x_1) - (w_2 + x_2)}$$

- Repeat for remaining boundaries : other Near and Far clipping planes

Clipping and Homogeneous Coordinates (example)

$$P_1 = [2, y_1^*, z_1^*, 2]$$

$$P_2 = [-1, y_2^*, z_2^*, \tfrac{1}{2}]$$

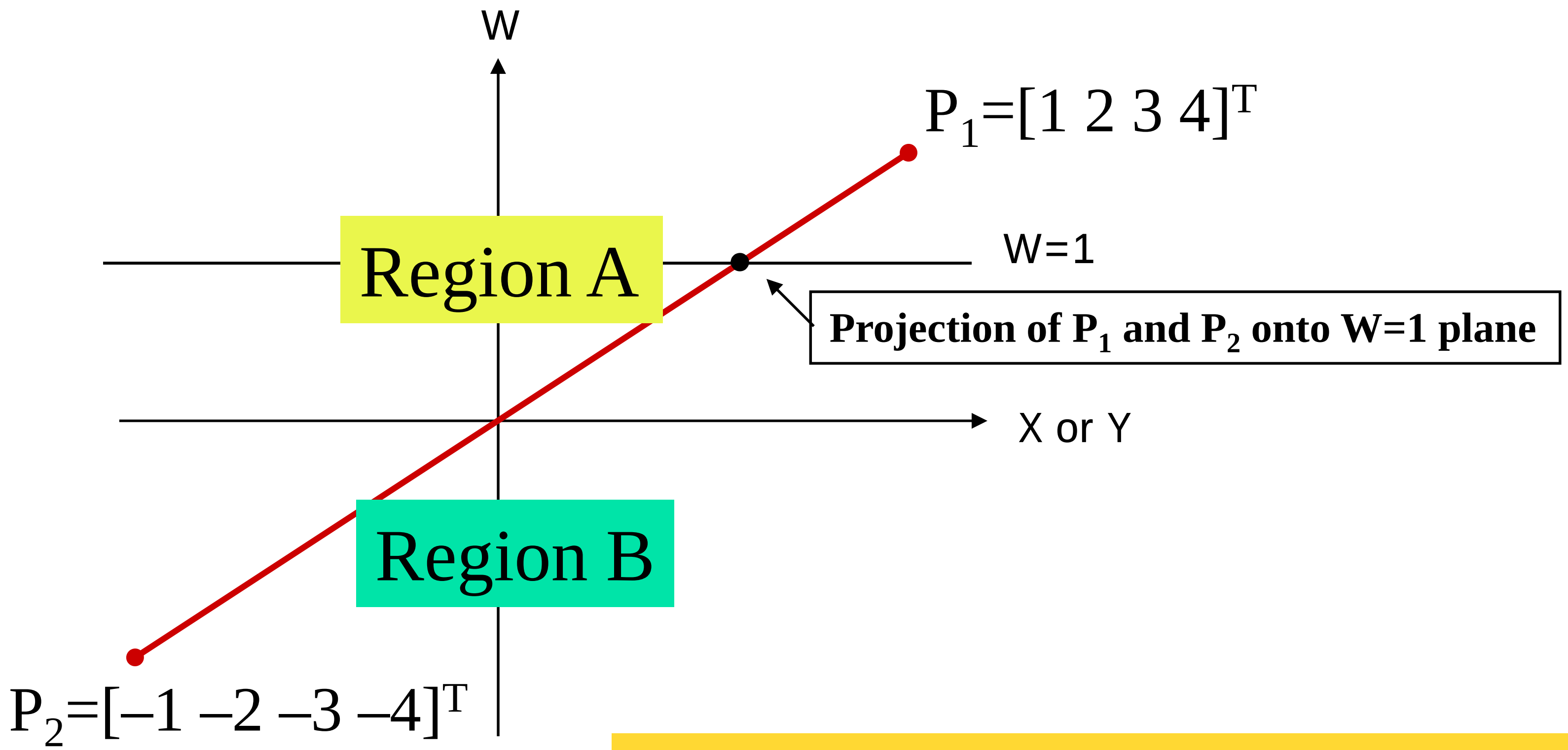


$$\Rightarrow \quad \alpha = \frac{x_1 + w_2}{(w_1 + x_1) - (w_2 + x_2)} = \frac{2+2}{-(\tfrac{1}{2} - 2) - (-1 - 2)} = \tfrac{8}{9}$$

$$\Rightarrow \quad P^* = [-\tfrac{2}{3}, y_1^* + \tfrac{8}{9}(y_2^* - y_1^*), z_1^* + \tfrac{8}{9}(z_2^* - z_1^*), \tfrac{2}{3}]$$

$$\Rightarrow \quad \text{Projected x-coordinate of } P^* = x/w^* = -1$$

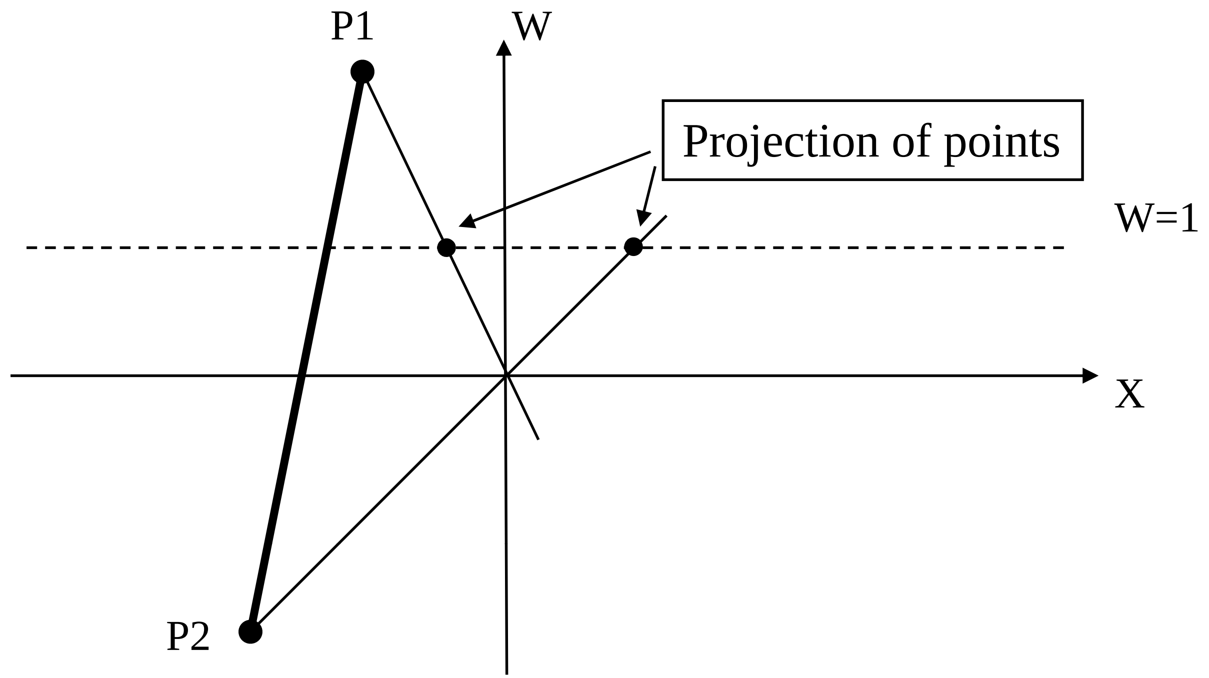
Points in Homogeneous Coordinates



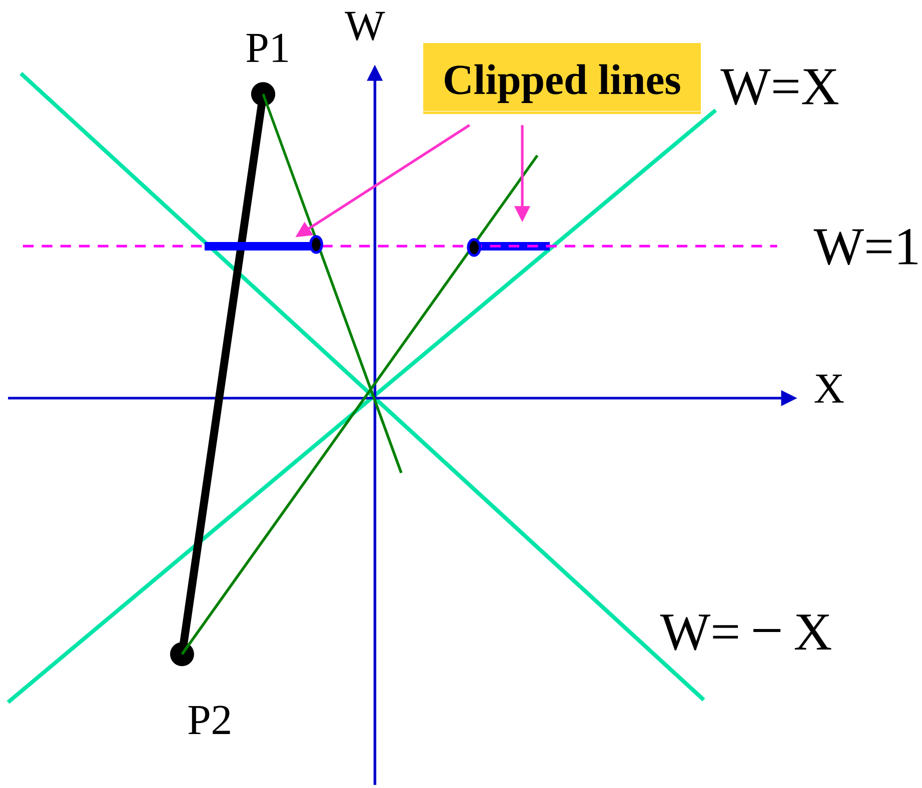
Need to consider both regions when Performing clipping

Lines in Homogeneous Coordinates

- Could clip twice – once for region B, once for region A.
  - Expensive
- Check for negative W values and negate points before clipping

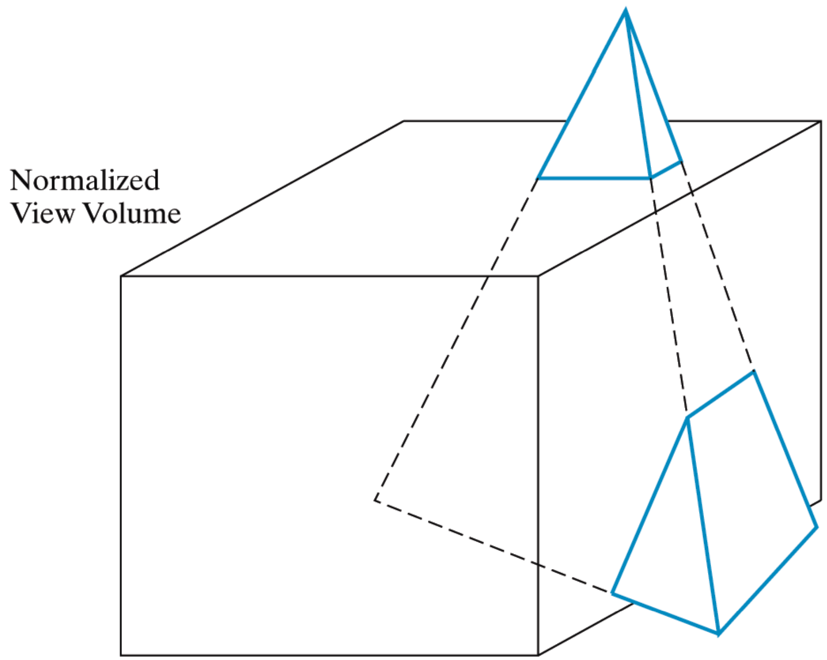


Lines in Homogeneous Coordinates



3D Polygon Clipping Algorithms

- Bounding box or sphere test for early rejection
- Sutherland-Hodgman and Weiler-Atherton algorithms can be generalized



What's Next