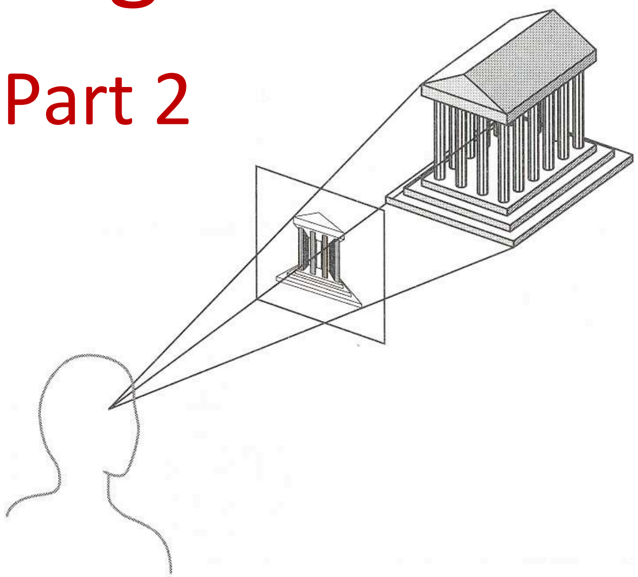
FACULTY OF ENGINEERING AND ARCHITECTURE

E016712: Computer Graphics

Viewing in 3D

Part 2



Lecturers: Aleksandra Pizurica and Danilo Babin

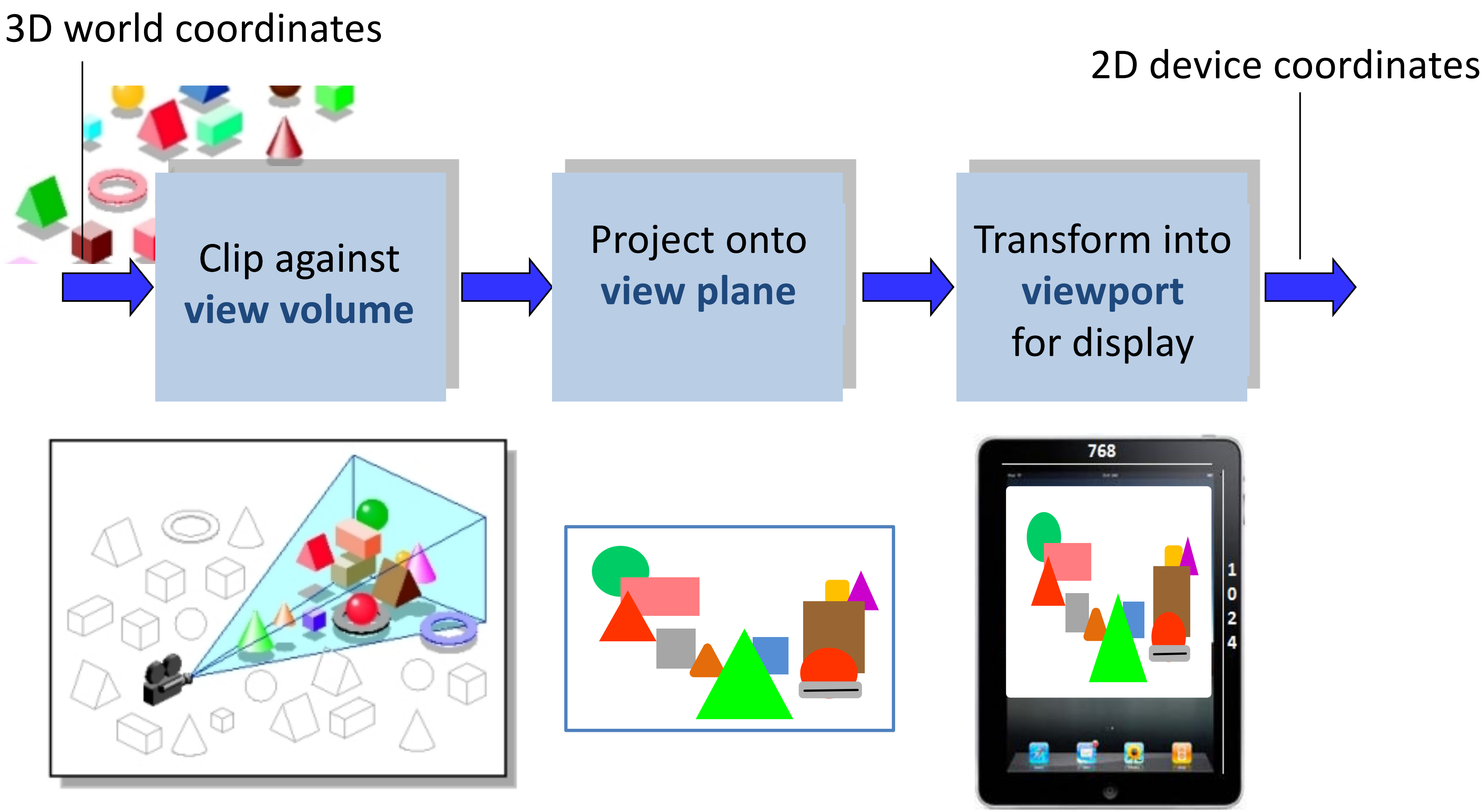
GHENT UNIVERSITY

Overview

- Mathematics of geometric projections
- Implementing planar geometric projections
- Normalizing projections
- Clipping against canonical view volumes

Partially based on:
[FvDFH] J. Foley, A. van Dam, S. Feiner and J. Hughes: *Computer Graphics: Principles and Practice*, Addison-Wesley, 1996.

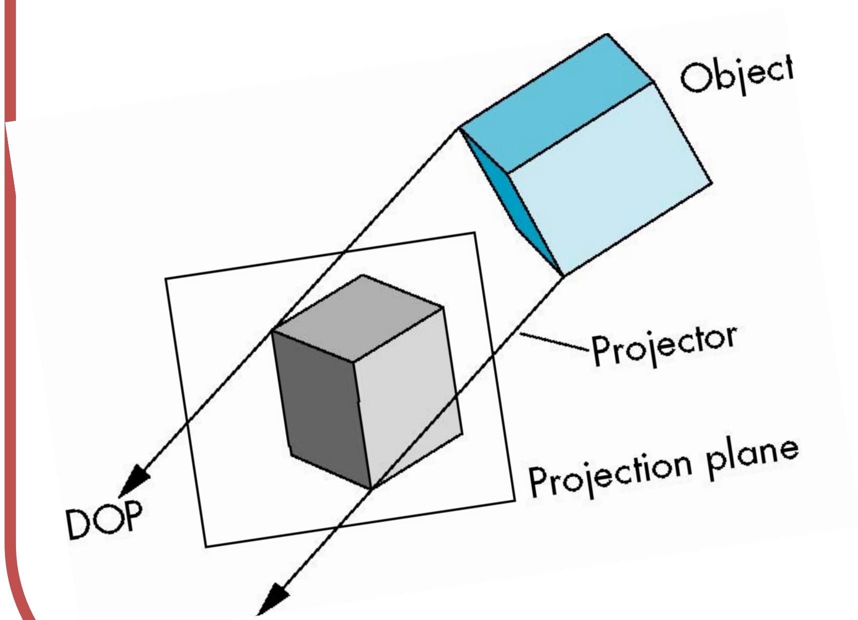
Reminder: Conceptual model of 3D viewing



Reminder: Planar Geometric Projections

Parallel projection

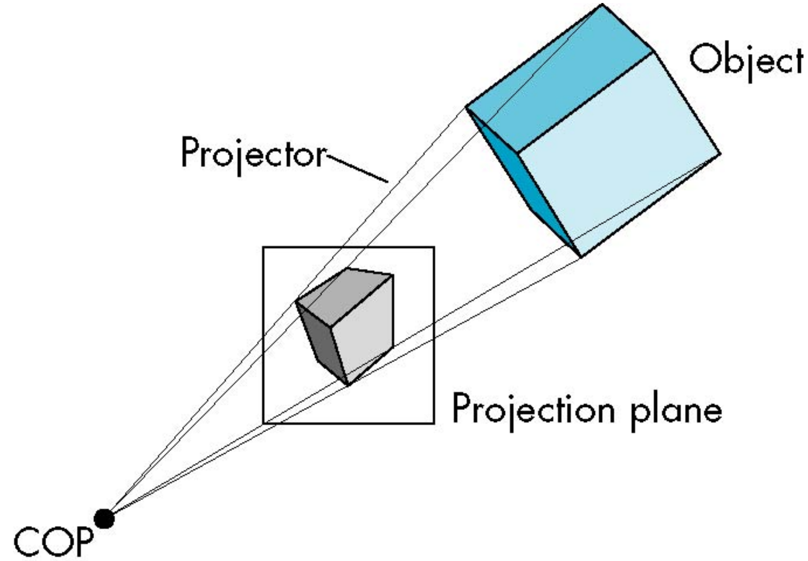
Mainly for technical drawings (accurate measures)



DOP – direction of projection

Perspective projection

More natural, not convenient for accurate measurements



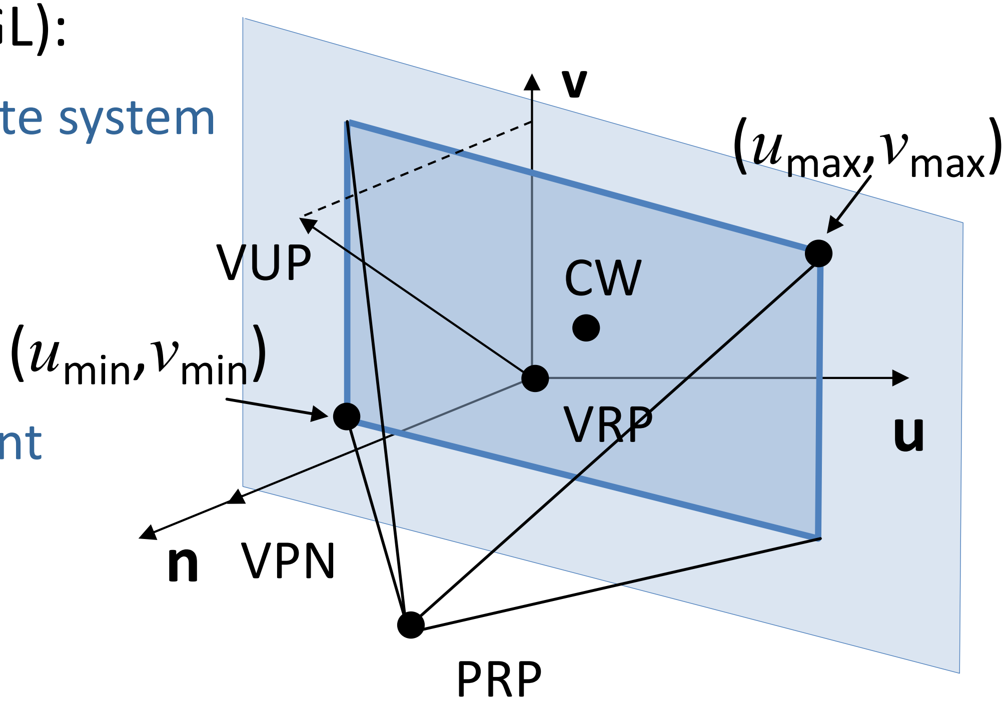
COP – center of projection

Projection plane = view plane = picture plane

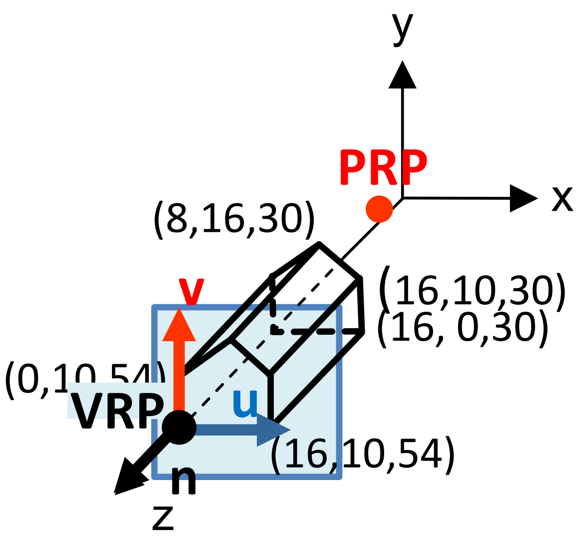
COP = Projection Reference Point (PRP) = eye = view point

Reminder View Reference System

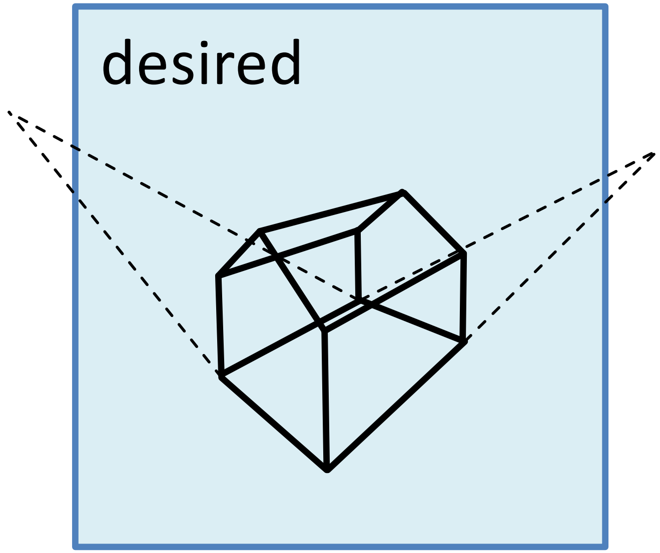
- 3D objects that we are viewing "live" in a **World Coordinate System – WCS** (right-handed coordinate system with x, y and z axes)
- In camera space, we defined **Viewing Reference System** with unit vectors $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{n}$ (which is also a right-handed coordinate system)
- Common notation in graphics application program interfaces - API (PHIGS, OpenGL):
  - VRC – view reference coordinate system
  - VRP – view reference point
  - VPN – view plane normal
  - VUP – view up vector
  - PRP – projection reference point
  - CW – center of **window**



Reminder: Specifying arbitrary view in 3D

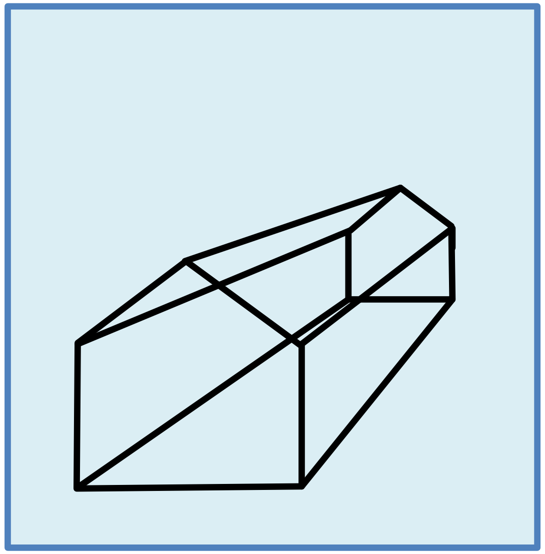


Which parameters?



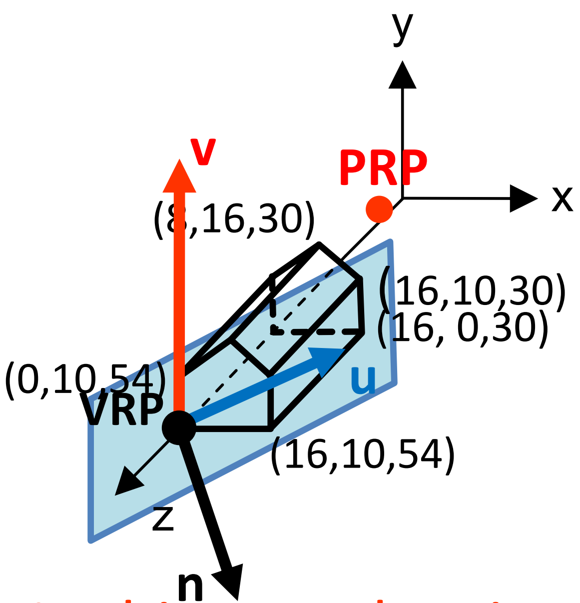
Try the following:

VRP (WC):	(16,0,54)
VPN (WC):	**(0,0,1)**
VUP (WC):	(0,1,0)
PRP (VRC):	(20,25,20)
Window (VRC):	(-20,20,-5,35)
Projection type:	perspective

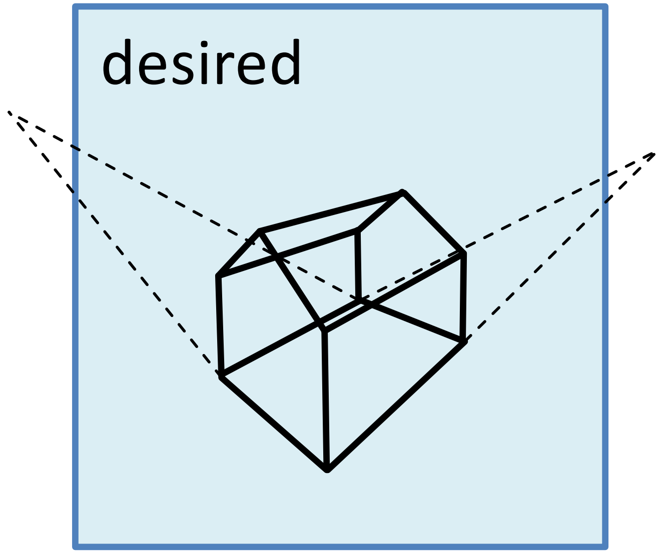


Need to reorient the view plane!

Reminder: Specifying arbitrary view in 3D

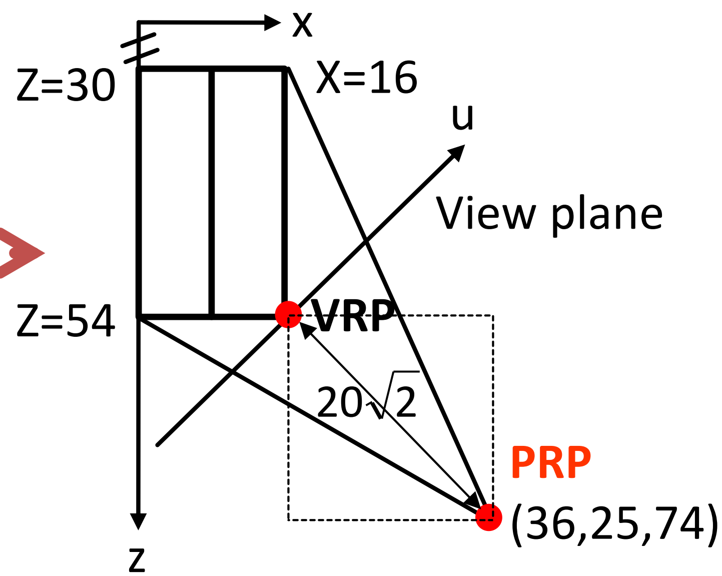


Which parameters?



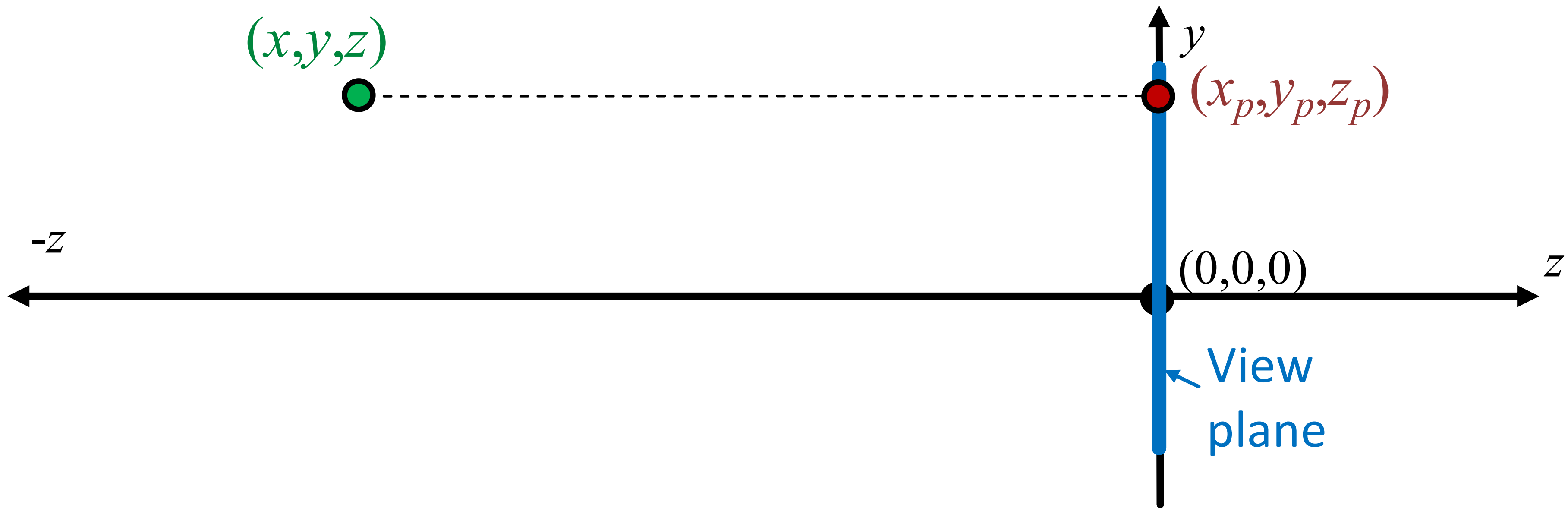
In this case the view plane shouldn't be parallel to x-y plane!

VRP (WC):	(16,0,54)
VPN (WC):	(**1**,0,1)
VUP (WC):	(0,1,0)
PRP (VRC):	$(0,25,20\sqrt{2})$
Window (VRC):	(-20,20,-5,35)
Projection type:	perspective



Mathematics of Geometric Projections

Orthographic projection matrix

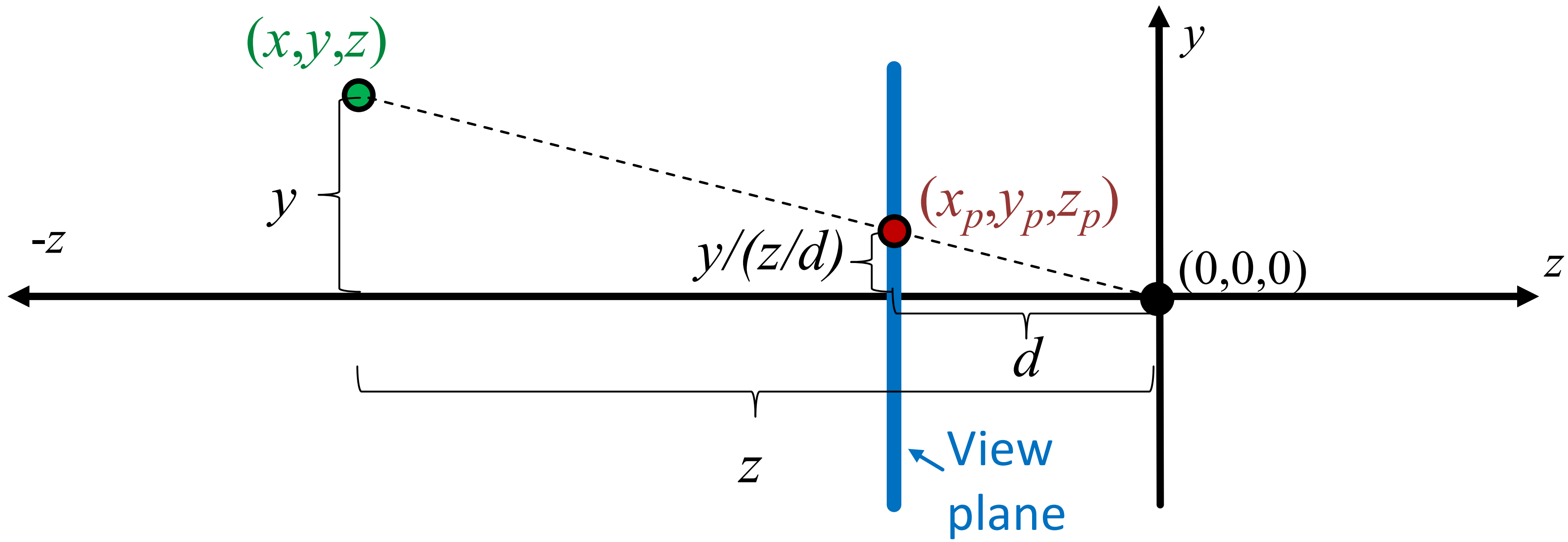


In homogeneous coordinates:

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}; \quad \mathbf{M}_{\text{ort}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \qquad \mathbf{q} = \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = \mathbf{M}_{\text{ort}}\,\mathbf{p} = \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix}$$

Going back to 3D coordinates: $(x_p, y_p, z_p) = \left(\frac{X}{W}, \frac{Y}{W}, \frac{Z}{W} \right) = (x, y, 0)$

Perspective projection matrix (1)



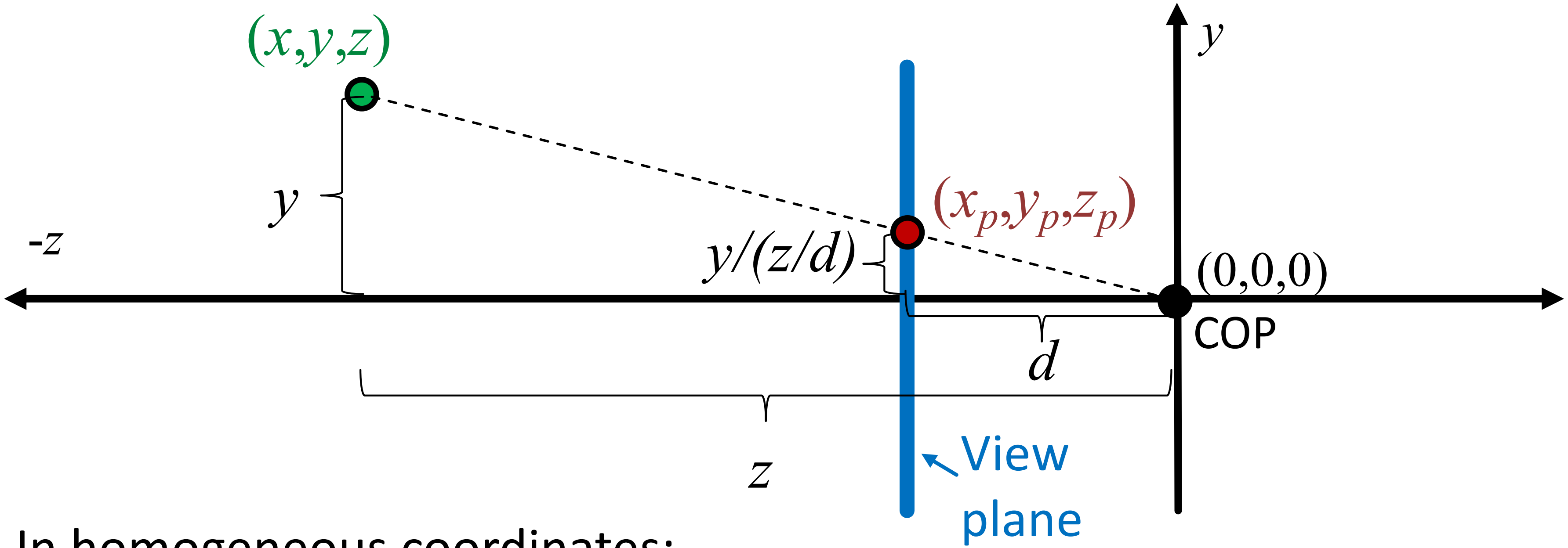
$$z_p = d$$

$$\frac{x_p}{d} = \frac{x}{z} \Rightarrow x_p = \frac{x}{z/d}$$

$$\frac{y_p}{d} = \frac{y}{z} \Rightarrow y_p = \frac{y}{z/d}$$

Division along x- and y-axis with z produces **non-uniform foreshortening** – objects that are further away (larger z) appear smaller

Perspective projection matrix (2)

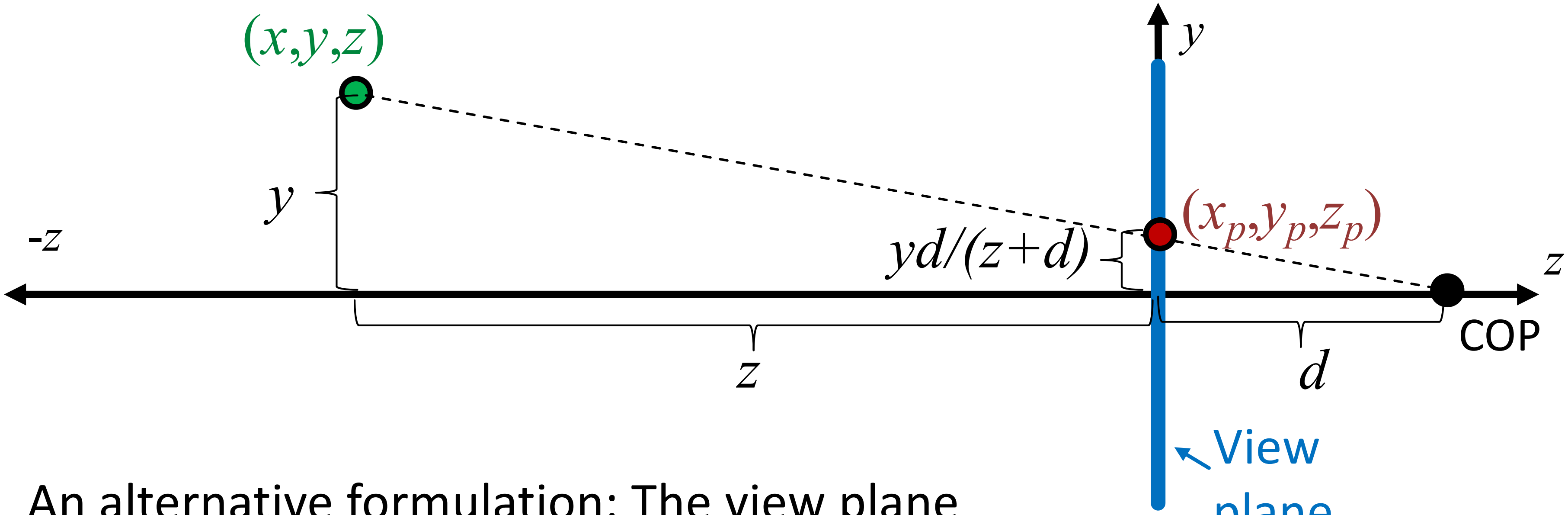


In homogeneous coordinates:

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} ;\ \mathbf{M}_{\text{per}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} ;\ \ \mathbf{q} = \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = \mathbf{M}_{\text{per}}\, \mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix}$$

Going back to 3D coordinates: $(x_p, y_p, z_p) = \left(\frac{X}{W}, \frac{Y}{W}, \frac{Z}{W} \right) = \left(\frac{x}{z/d}, \frac{y}{z/d}, d \right)$

Perspective projection matrix (3)



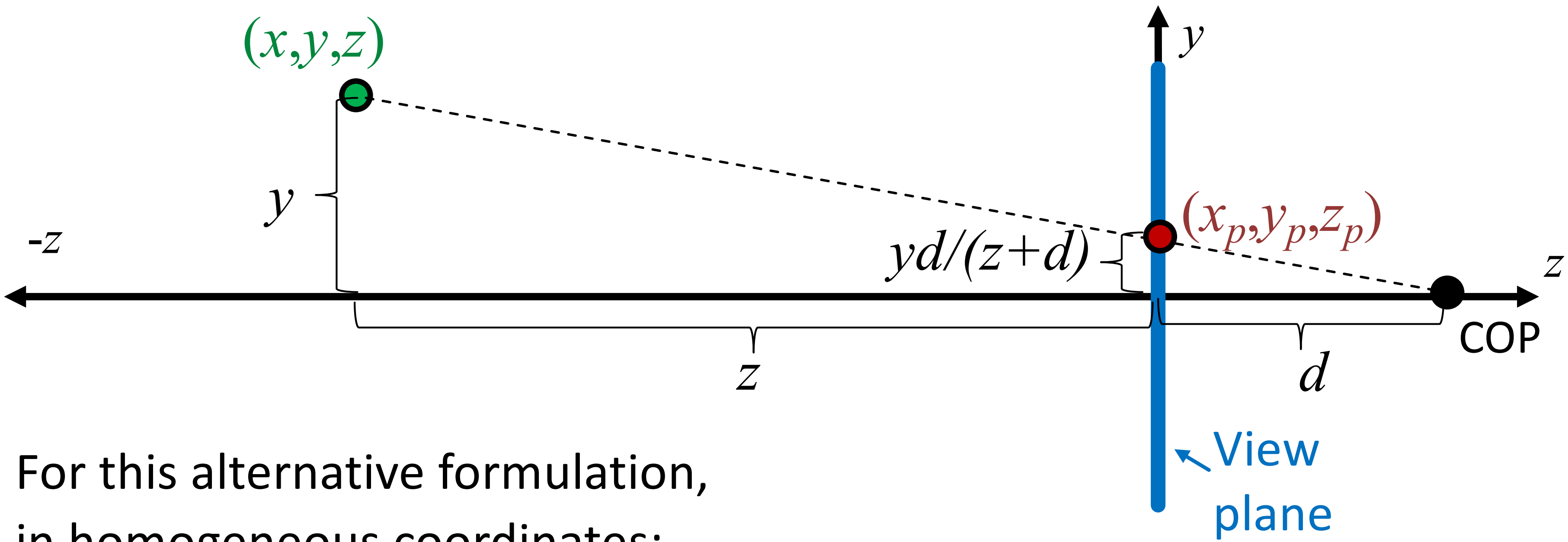
An alternative formulation: The view plane at $z = 0$ and the center of projection at $z=d$.

$$z_p = 0$$

$$\frac{x_p}{d} = \frac{x}{z+d} \Rightarrow x_p = \frac{xd}{z+d} = \frac{x}{z/d+1}$$

$$\frac{y_p}{d} = \frac{y}{z+d} \Rightarrow y_p = \frac{yd}{z+d} = \frac{y}{z/d+1}$$

Perspective projection matrix (4)

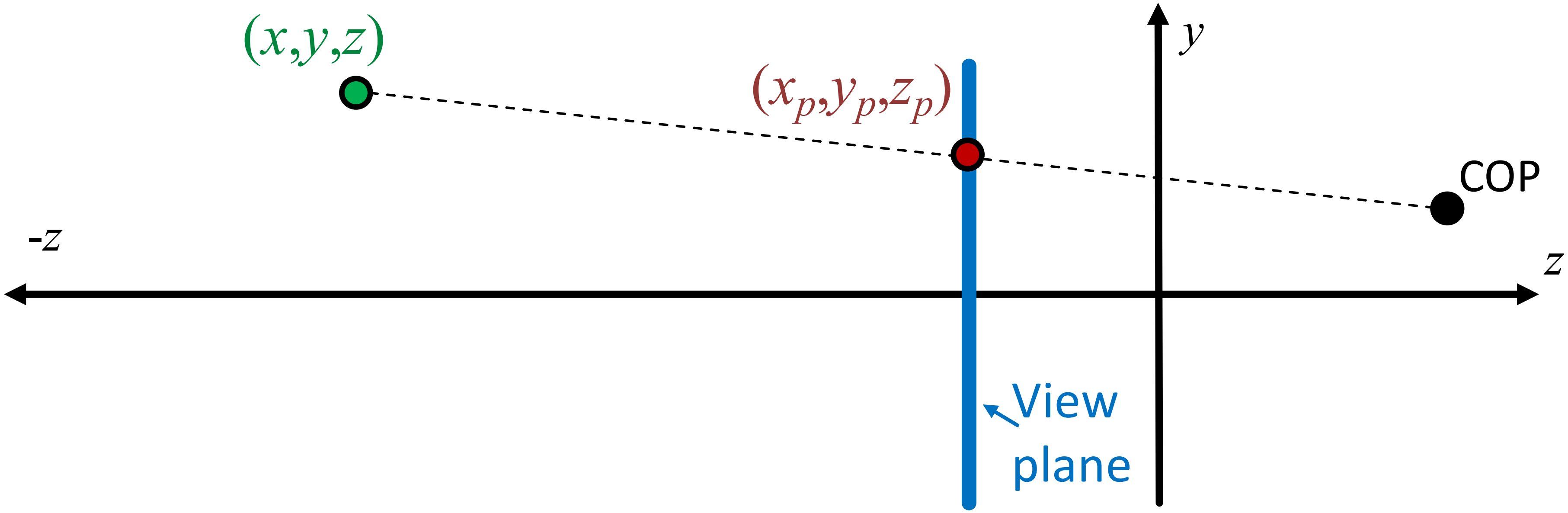


For this alternative formulation, in homogeneous coordinates:

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \; ; \; \mathbf{M}_{\text{per}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/d & 1 \end{bmatrix} \; ; \; \mathbf{q} = \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = \mathbf{M}_{\text{per}}\,\mathbf{p} = \begin{bmatrix} x \\ y \\ 0 \\ z/d+1 \end{bmatrix}$$

In 3D coordinates: $(x_p, y_p, z_p) = \left(\frac{X}{W}, \frac{Y}{W}, \frac{Z}{W} \right) = \left(\frac{x}{z/d+1}, \frac{y}{z/d+1}, 0 \right)$

On more general projection matrices



- We can derive projection matrices for more general cases like this one (COP not in the origin and the view plane not aligned with z=0) and even parameterize so that they apply for both parallel and perspective projection as two special cases (think of it).

- In practice, we will usually need only the "standard" $\mathbf{M}_{\text{ort}}$ and $\mathbf{M}_{\text{per}}$ matrices, because we will first apply a transformation, which brings more complex types of projections to the "simple" ones → see next.

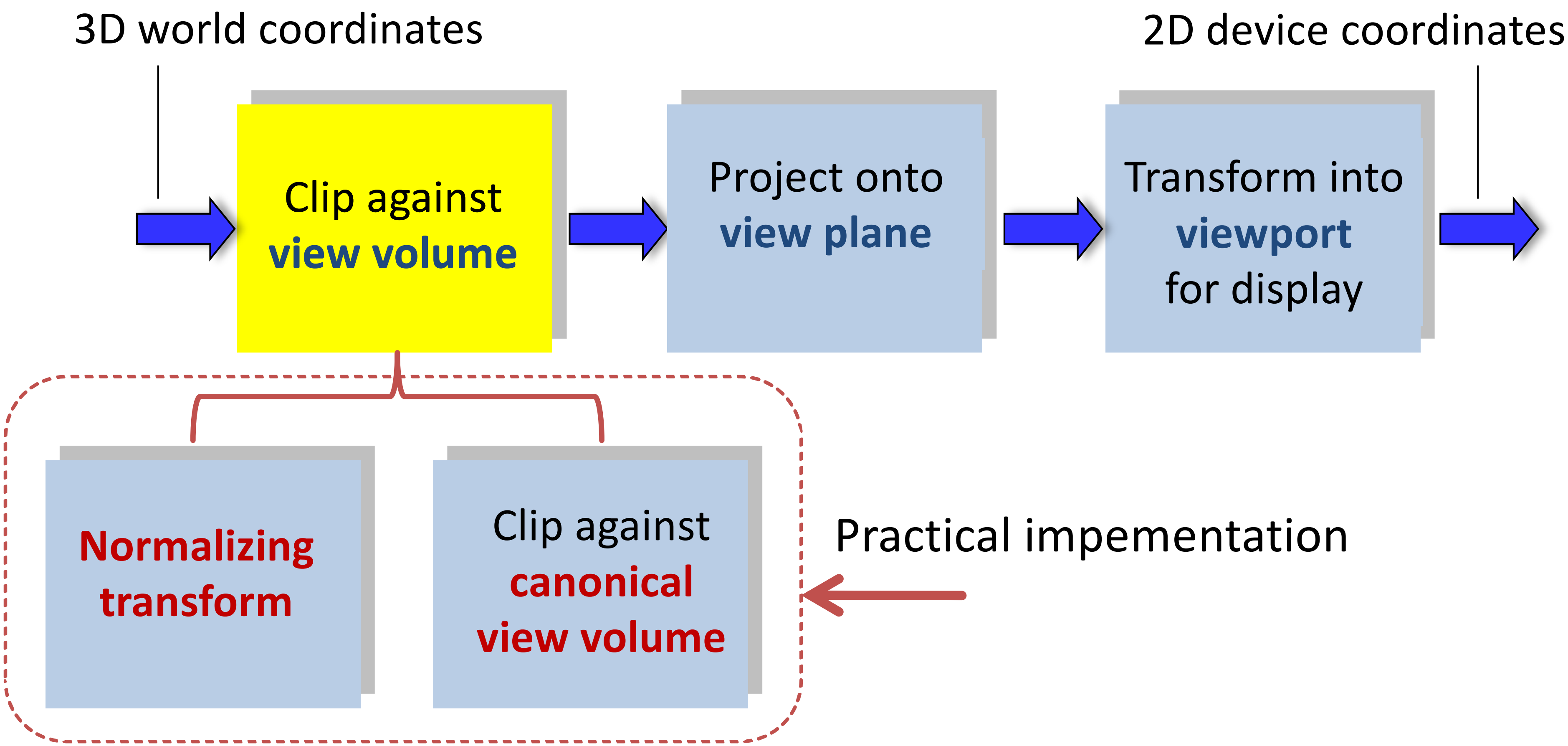
Implementing planar projections

Implementing planar geometric projections

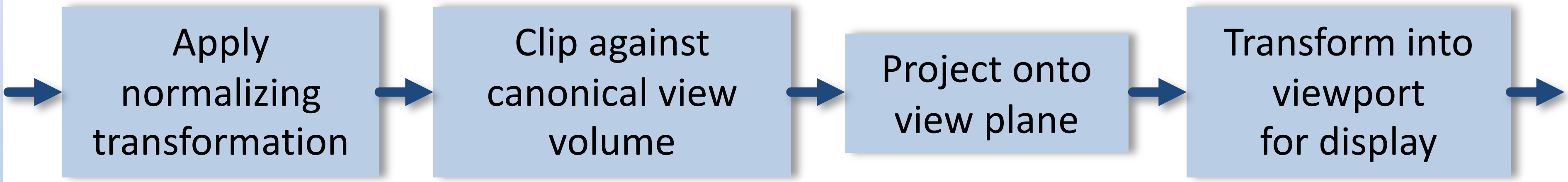
- Recall the conceptual viewing model:
  - Clipping against a view volume → projection → viewport transformation
- How do we actually implement the operation of clipping and projecting?
  - For an arbitrary view volume we would need to calculate the intersections with the 6 planes that define the frustrum.
  - This can be quite intensive computationally!
  - Certain volumes are easier to clip against than others –we will make use of it
  - We will transform arbitrary view volume to a **canonical view volume** using a **normalizing transformation** and then clip against the canonical view volume
  - The canonical view volume and the normalizing transformation differ for parallel and projective view volumes
  - Practical viewing model: normalizing transformation → clipping → projection via projection matrices $\mathbf{M}_{ort}$ (parallel view) or $\mathbf{M}_{per}$ (perspective view)

Practical viewing approach (1)

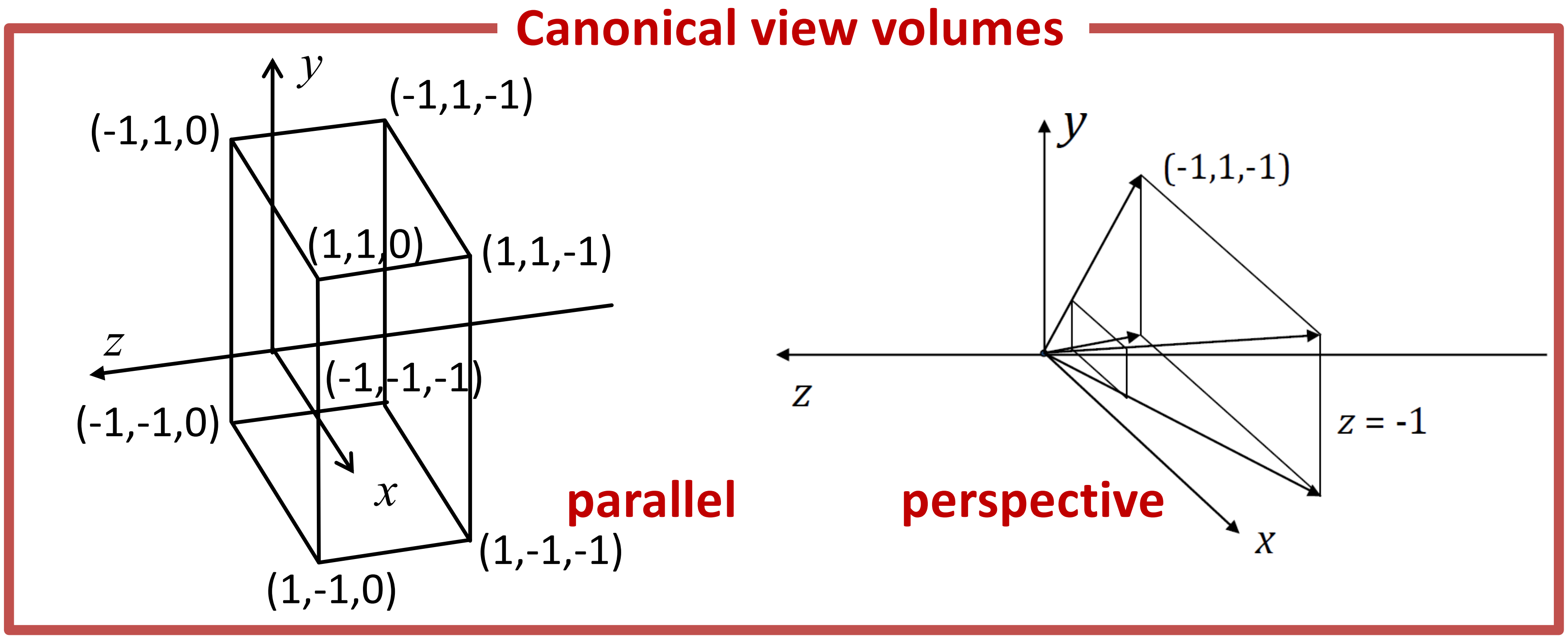
Our conceptual viewing model:



Practical viewing approach (2)

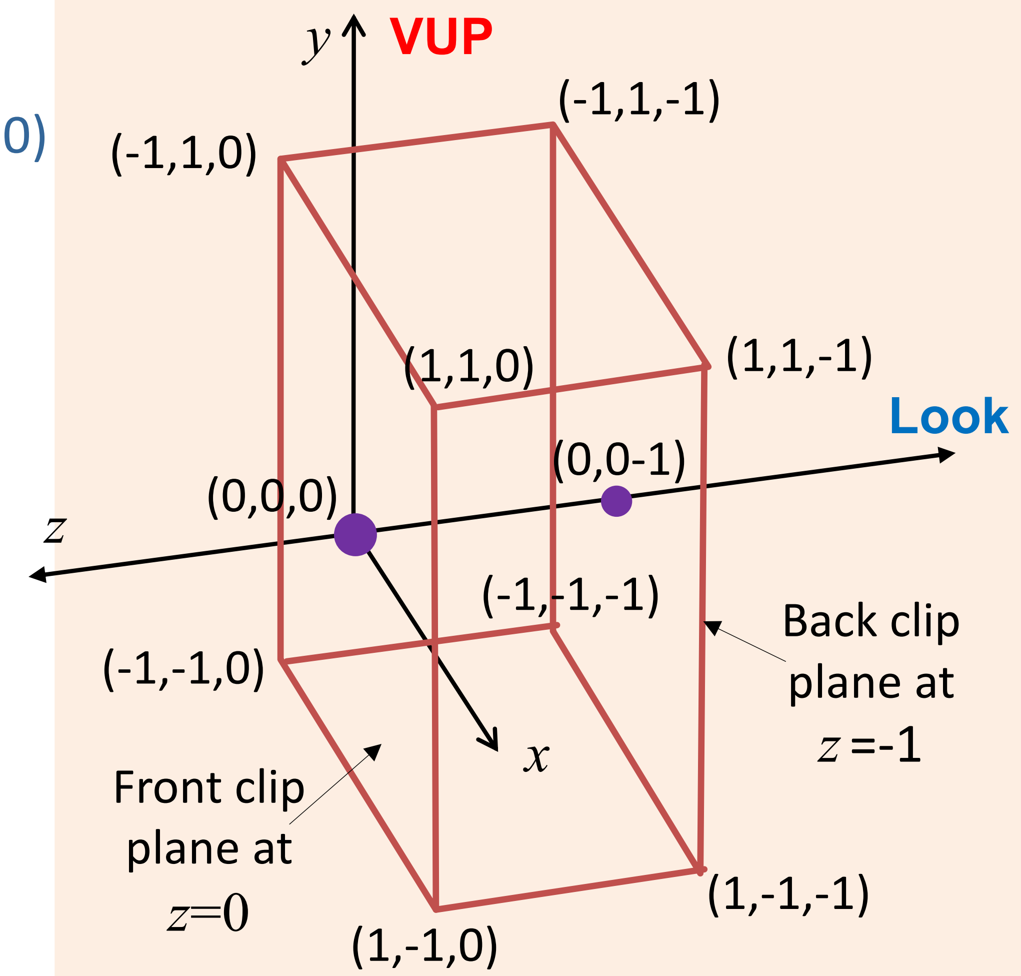


The normalizing transformation will be the most complex operation here



Canonical view volume for parallel projection

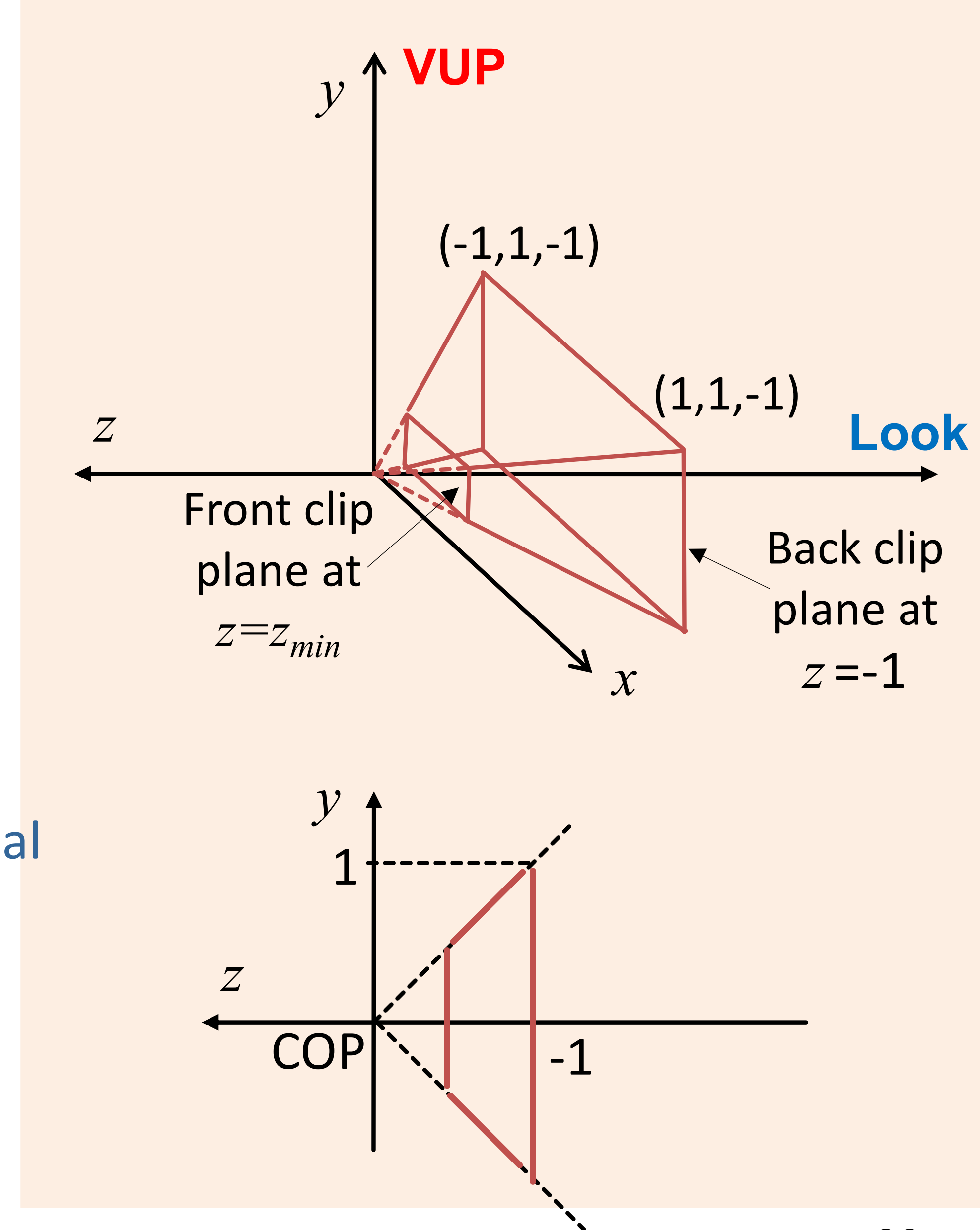
- “Sits” at origin
  - near clipping face centered at (0,0,0)
- “Looks” along negative z-axis
  - Look vector (0,0,-1)
- Oriented up right
  - VUP vector (0,1,0)
- Normalized viewing window
  - $-1 \leq x \leq 1$ and $-1 \leq y \leq 1$
- Front and back clipping planes:
  - front plane at z=0
  - back plane at z=-1



- Other conventions exist! in OpenGL: $x = \pm 1, \;\; y = \pm 1, \;\; z = \pm 1$

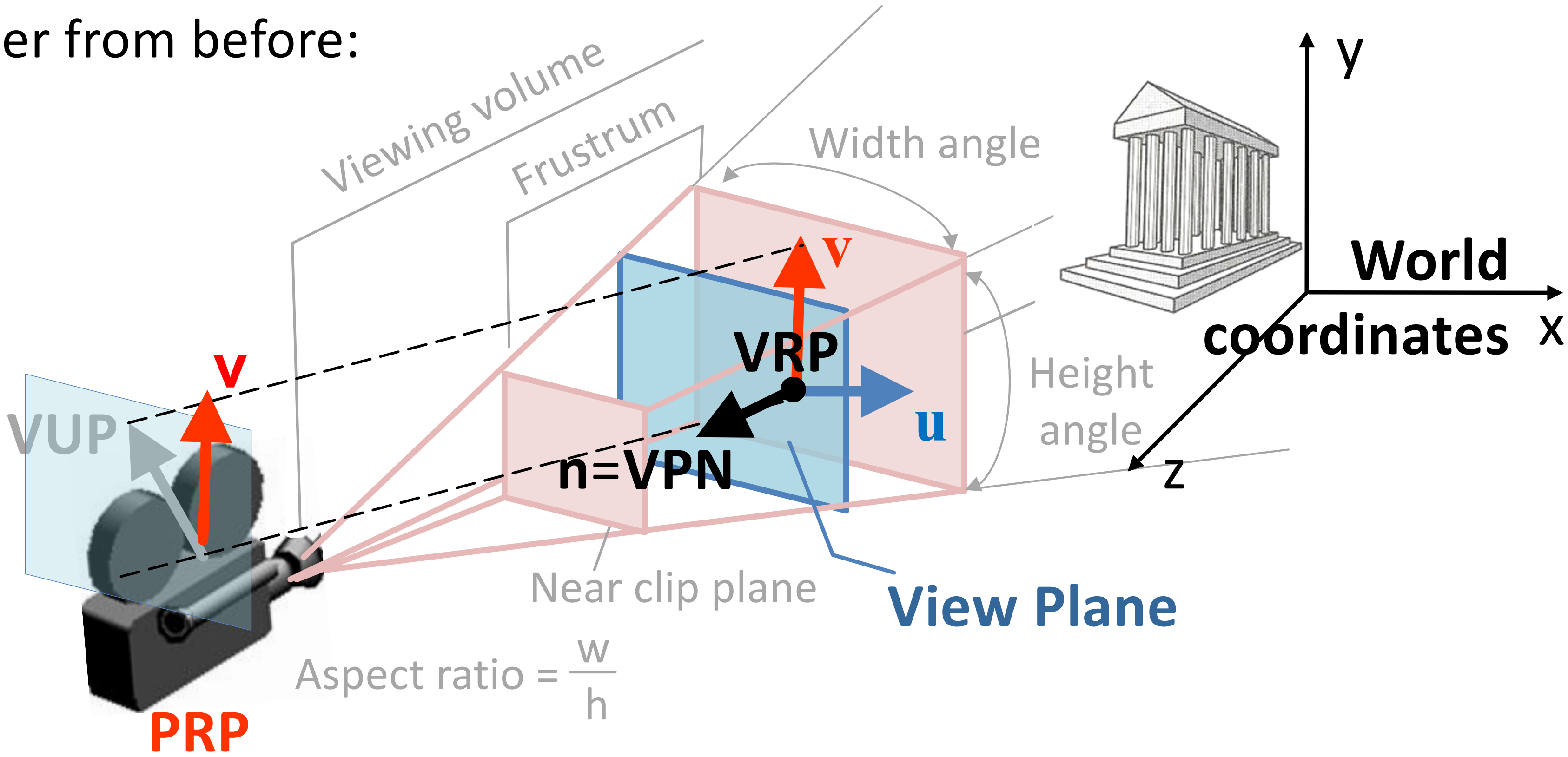
Canonical view volume for perspective projection

- Center of projection at origin
  - COP=(0,0,0)
- "Looks" along negative z-axis
  - Look vector (0,0,-1)
- Oriented up right
  - VUP vector (0,1,0)
- Front and back clipping planes:
  - front plane: z_{min} - determined after normalization (depends on the original view volume parameters)
  - back plane at z=-1
- Back clipping plane bounds:
  - $-1 \leq x \leq 1$ and $-1 \leq y \leq 1$



View Reference System (VRC)- reminder

Remember from before:



- View Reference Point (VRP) specifies the view plane together with VPN
- View Reference System (VRC) is the right-handed system **u-v-n**
  - **n** is VPN (View Plane Normal)
  - **v** is the projection of the View UP (VUP) vector in the view plane
  - **u** is defined such that **u**, **v** and **n** form a right-handed coordinate system

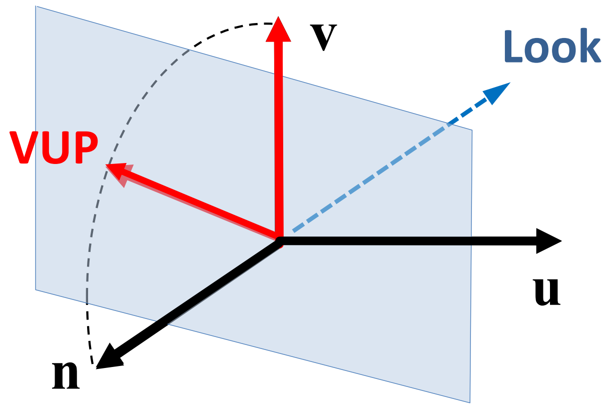
VRC explicitly defined: finding **u**, **v** and **n**

- We already defined before the viewing reference system **u-v-n**
- Let us now define the **u-v-n** vectors explicitly in terms of the virtual camera parameters
- Note: we determine the **world coordinates** of the **u-v-n** vectors

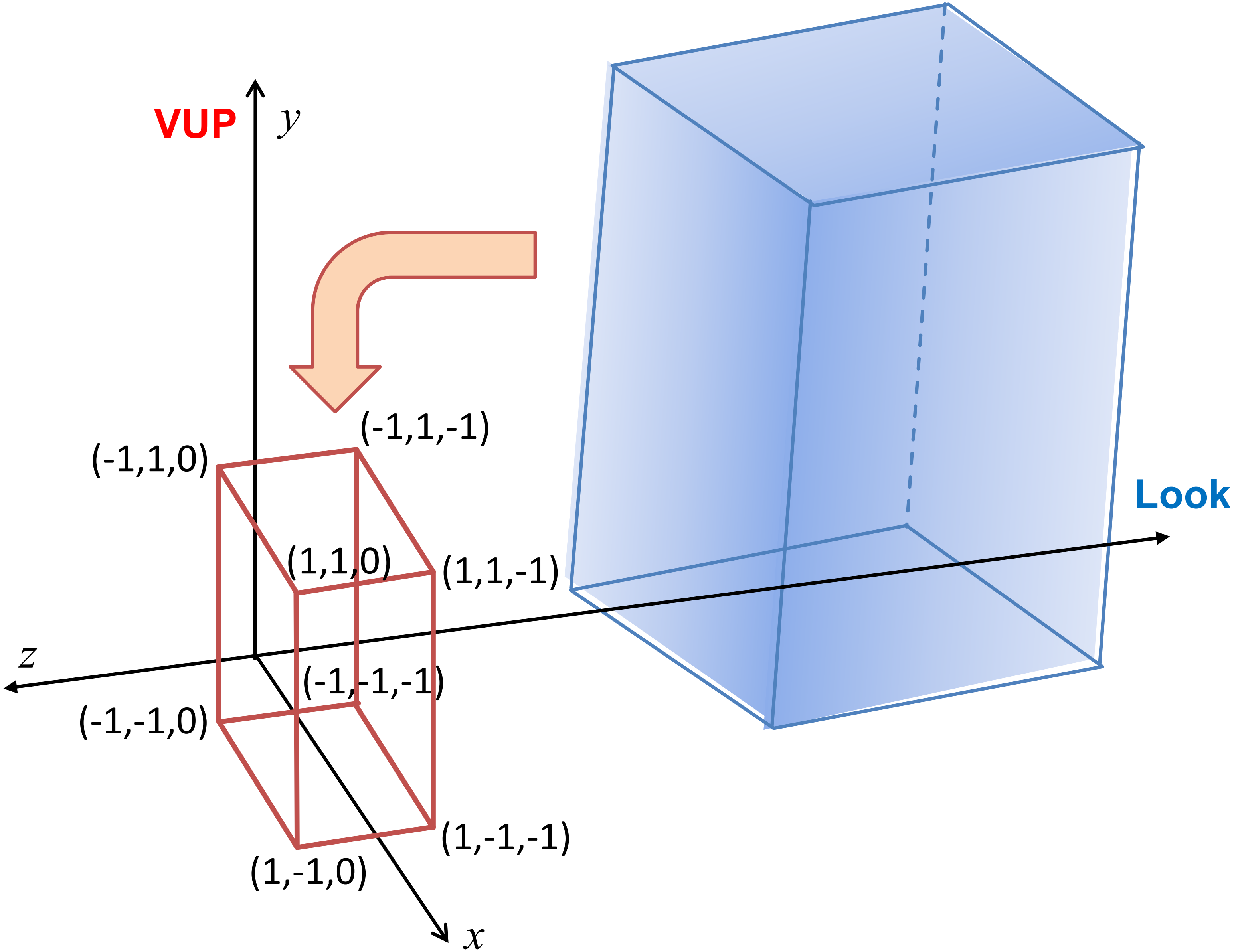
$$\mathbf{n} = [n_x \; n_y \; n_z]^{\mathrm{T}} = \frac{\mathbf{VPN}}{\|\mathbf{VPN}\|} = \frac{-\mathbf{Look}}{\|\mathbf{Look}\|}$$

$$\mathbf{v} = [v_x \; v_y \; v_z]^{\mathrm{T}} = \frac{\mathbf{VUP} - (\mathbf{n} \cdot \mathbf{VUP})\mathbf{n}}{\|\mathbf{VUP} - (\mathbf{n} \cdot \mathbf{VUP})\mathbf{n}\|}$$

$$\mathbf{u} = [u_x \; u_y \; u_z]^{\mathrm{T}} = \frac{\mathbf{v} \times \mathbf{n}}{\|\mathbf{v} \times \mathbf{n}\|}$$



Normalizing parallel projection (1)



Normalizing parallel projection (2)

- Steps to convert a parallel projection view volume into the canonical view volume:
  1. Translate VRP to origin
  2. Rotate VRC such that the n-axis (VPN) aligns with the z-axis, the u-axis becomes the x-axis and the v axis becomes the y axis
  3. Shear such that the Direction of Projection (DoP) becomes parallel to the z-axis
  4. Translate the front center of the view volume to the origin
  5. Scale such that the view volume becomes bounded by the planes

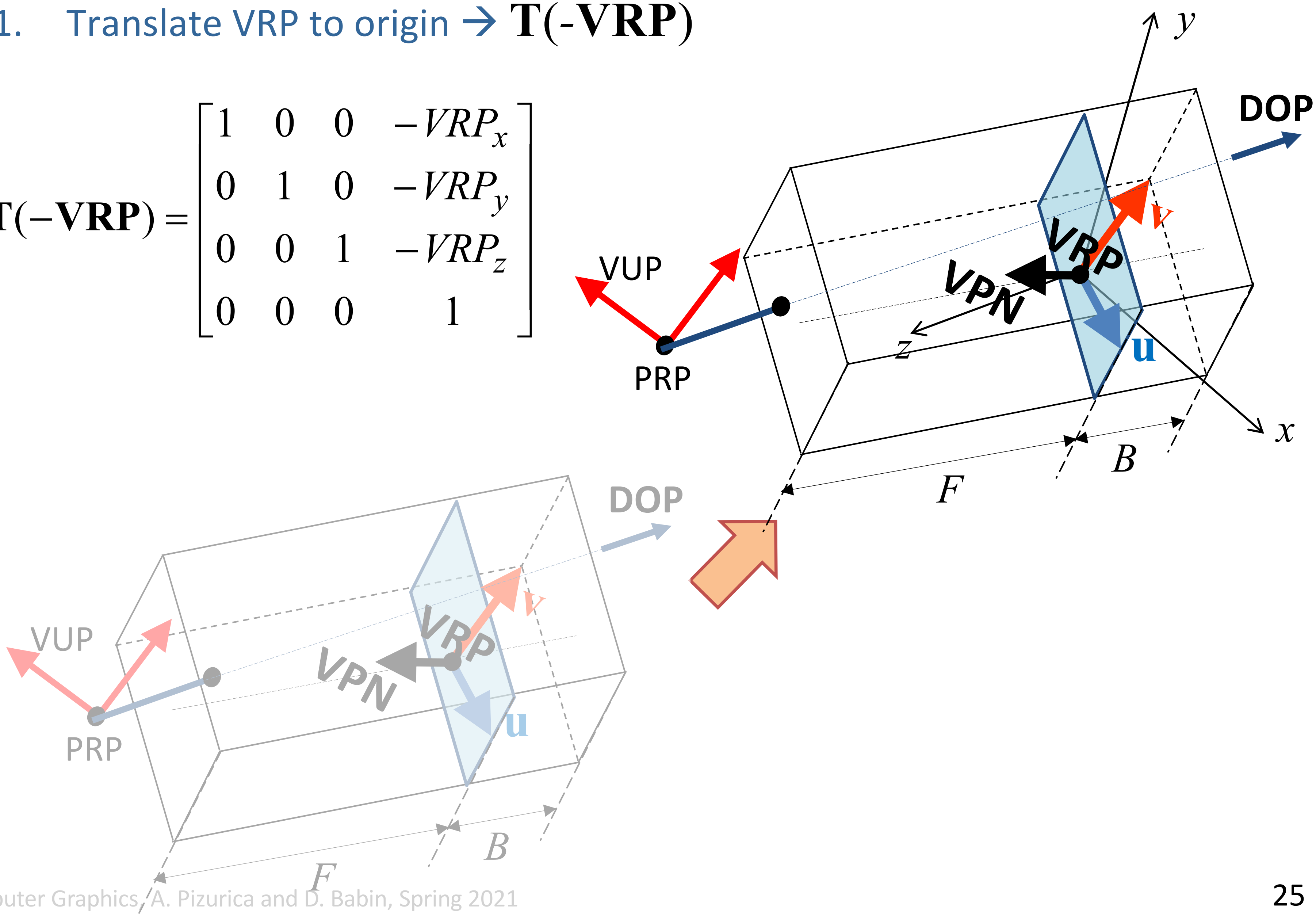
     x=-1, x=1, y=-1, y=1, z=0, z=-1

- Note that steps 4 and 5 will change when canonic view volume is defined differently, e.g. as in OpenGL (centered in the origin, with $x = \pm 1,\ y = \pm 1,\ z = \pm 1$). But the principle remains the same!

Normalizing parallel projection (3)

1. Translate VRP to origin → **T(-VRP)**

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



Normalizing parallel projection (4)

1. Translate VRP to origin → $\mathbf{T(\text{-}VRP)}$
2. Rotate VRC system so that u-v-n axes align with x-y-z axes, resp.

$$\left.\begin{aligned}\mathbf{Ru} &= [1\ 0\ 0]^{\mathrm{T}}\\ \mathbf{Rv} &= [0\ 1\ 0]^{\mathrm{T}}\\ \mathbf{Rn} &= [0\ 0\ 1]^{\mathrm{T}}\end{aligned}\right\}\ \mathbf{R}[\mathbf{u}\ \mathbf{v}\ \mathbf{n}] = \mathbf{I}\ \Rightarrow\ \mathbf{R} = [\mathbf{u}\ \mathbf{v}\ \mathbf{n}]^{-1} = [\mathbf{u}\ \mathbf{v}\ \mathbf{n}]^{\mathrm{T}} = \begin{bmatrix} u_x & u_y & u_z & 0\\ v_x & v_y & v_z & 0\\ n_x & n_y & n_z & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

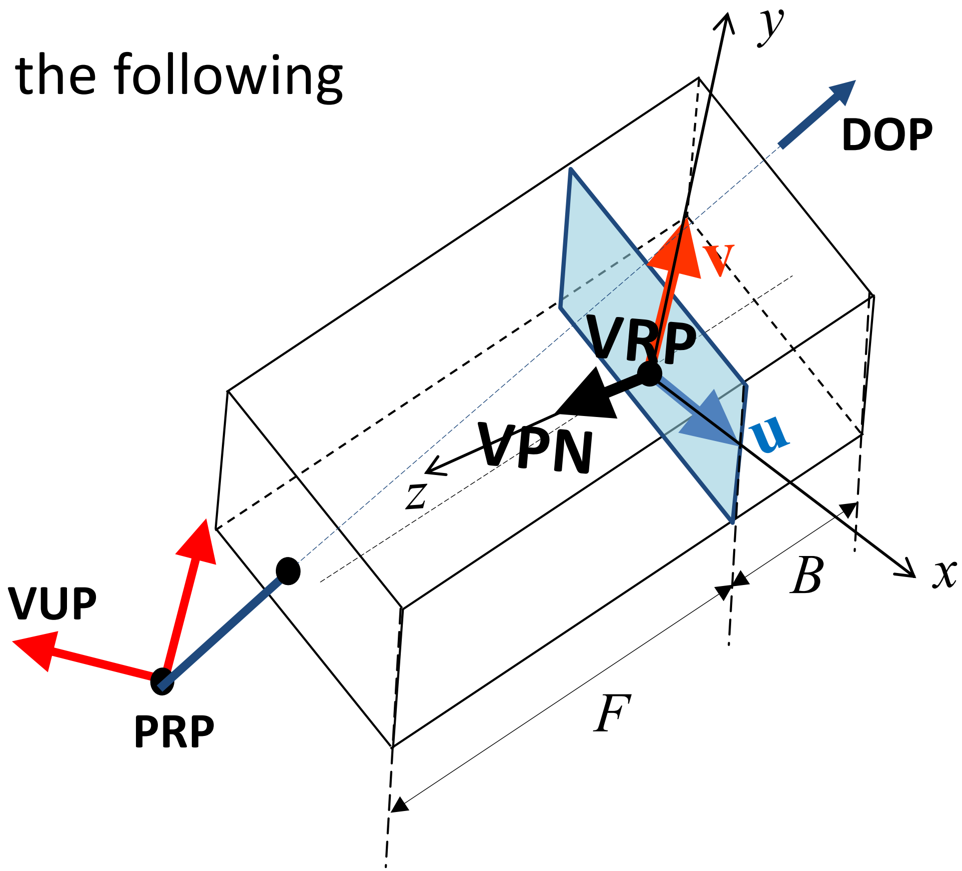
Note: The two steps so far RT(-VRP) should yield the equivalent result as the transformation from VRC to WC that we showed in the lesson Part 1 on 3D viewing. Verify this (taking into account slightly different notation).

Normalizing parallel projection (5)

After steps 1-2 we have in general the following

Remember that for oblique projection

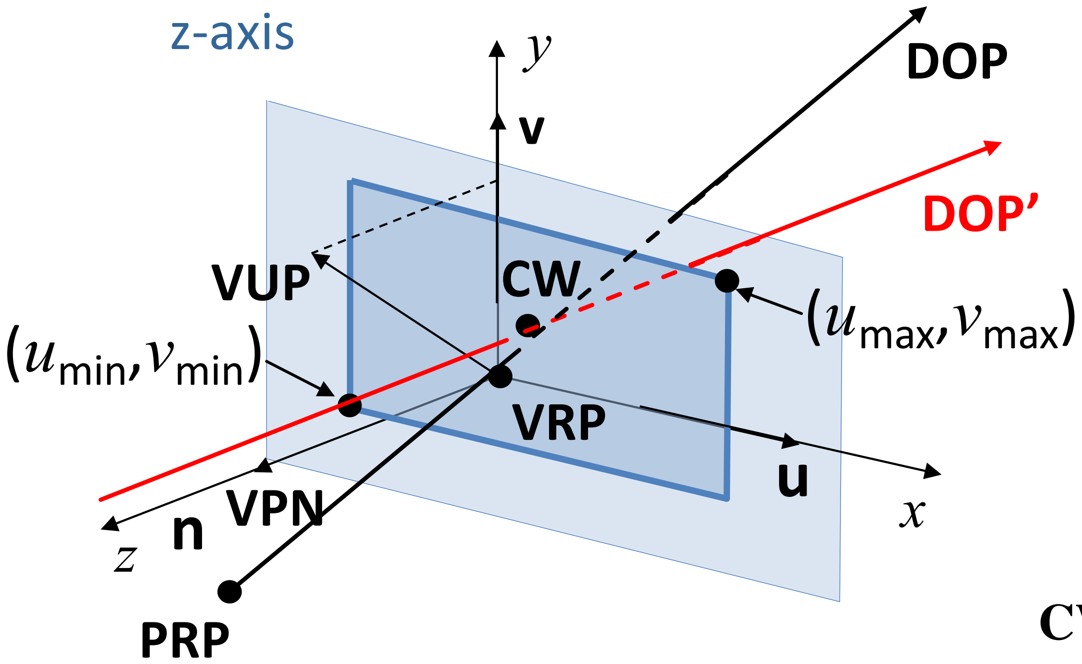
VPN ≠ –DOP



What do we need to do to make DOP parallel to the z- axis? Obviously we cannot rotate the whole volume. Why?

Normalizing parallel projection (5)

3. Shear such that the Direction of Projection (DoP) becomes parallel to the z-axis



$$\mathbf{DOP} = \begin{bmatrix} dop_x \\ dop_y \\ dop_z \\ 1 \end{bmatrix} = \mathbf{CW} - \mathbf{PRP}$$

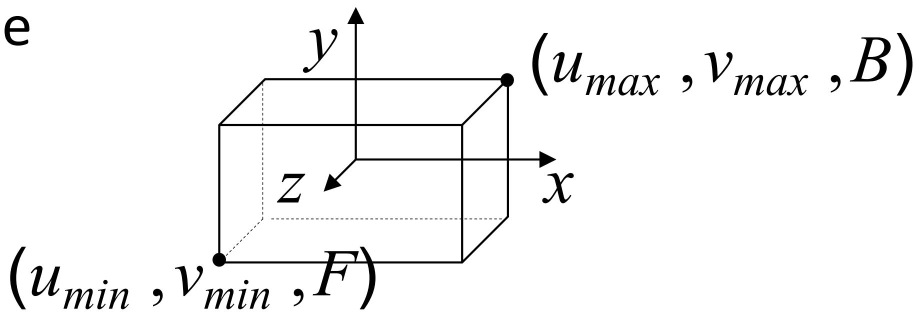
$$\mathbf{CW} = \begin{bmatrix} \frac{u_{\max} + u_{\min}}{2} \\ \frac{v_{\max} + v_{\min}}{2} \\ 0 \\ 1 \end{bmatrix} \qquad \mathbf{PRP} = \begin{bmatrix} prp_u \\ prp_v \\ prp_n \\ 1 \end{bmatrix}$$

$$\mathbf{DOP'} = [0 \quad 0 \quad dop_z \quad 0]^T = \mathbf{H}_{\text{par}}\mathbf{DOP}$$

$$\mathbf{H}_{\text{par}} = \begin{bmatrix} 1 & 0 & H_x & 0 \\ 0 & 1 & H_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad H_x = -\frac{dop_x}{dop_z}, \quad H_y = -\frac{dop_y}{dop_z}$$

Normalizing parallel projection (6)

Steps 1-3: result in a view volume



4. Translate the front center of the view volume to the origin

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

5. Scale such that the view volume becomes bounded by the planes
   x=-1; x=1; y=-1; y=1; z=0; z=-1

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

The resulting normalizing transform: $\mathbf{N}_{\text{par}} = \mathbf{S}_{\text{par}} \cdot \mathbf{T}_{\text{par}} \cdot \mathbf{H}_{\text{par}} \cdot \mathbf{R} \cdot \mathbf{T}(-\mathbf{VRP})$

Normalizing perspective projection (1)

- Steps to convert a perspective projection view volume into canonical view:
  1. Translate VRP to origin → **T**(-**VRP**)
  2. Rotate VRC to align VPN with the z-axis → rotation **R**
  3. Translate PRP (i.e., center of projection) to the origin **T**(-**PRP**)
  4. Shear such that the center line of view volume becomes the z-axis

     → Shear matrix is the same as for the parallel case – think why
  5. Scale such that the view volume becomes the canonical perspective view volume → apply scaling matrix $\mathbf{S}_{per}$

     5.1 Scale in x and y such that the side planes make 45-degree angles with the axes (→ make the width and the height of the window twice the z-coordinate of the view plane VRP_z)

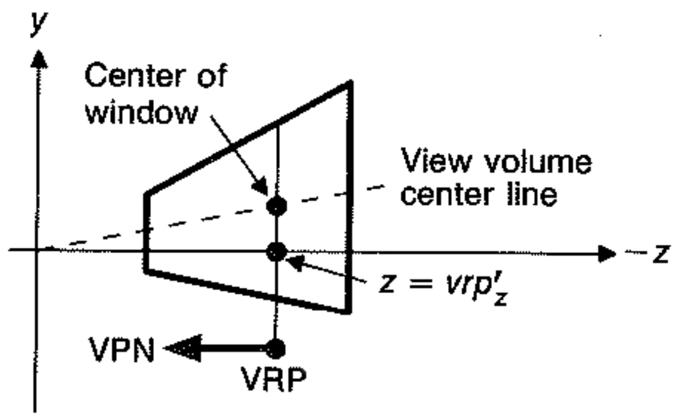
     5.2 Scale along all three axes so that the back clipping plane becomes z=1 (scale in all dir by $1/(VRP_z+n_{max})$) Derive $\mathbf{S}_{per}$ as an exercise!

The normalizing transform: $\mathbf{N}_{per} = \mathbf{S}_{per} \cdot \mathbf{H}_{par} \cdot \mathbf{T}(-\mathbf{PRP}) \cdot \mathbf{R} \cdot \mathbf{T}(-\mathbf{VRP})$

Normalizing perspective projection (2)

Steps 1,2 and 4 are the same as in the parallel case, and step 3 is a standard translation. We need to consider extra only the step 5

5. Scale such that the view volume becomes the canonical perspective view volume → apply scaling matrix $\mathbf{S}_{per}$



VRP' is **VRP** transformed by the previous steps

Divide into 2 steps (see previous slide). Show that

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

Find z_{min} as a function of **VRP'**, F and B

Exercise!

Clipping Against Canonic View Volume

- Clipping against the canonic view volume for parallel projection (cube):

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 of view volume	x<-1
bit 5:point is behind view volume	z<-1
bit 5:point is in front of view volume	z>0

- As in 2D, a line is trivially accepted if both endpoints have a code of all zeros, and is trivially rejected if the bit-per-bit logical **and** of the codes is not all zeros
- The intersection calculation use parametric representation of a line from point P_0 to P_1:

$$P_0(x_0, y_0, z_0)$$
$$P_1(x_1, y_1, z_1)$$

$$x = (x_1 - x_0)t + x_0$$
$$y = (y_1 - y_0)t + y_0$$
$$z = (z_1 - z_0)t + z_0, \ 0 \le t \le 1$$

Exercise!

Clipping Against Canonic View Volume

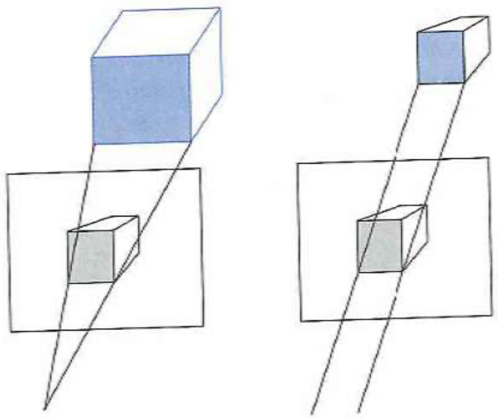
- Clipping against the canonic view volume for perspective projection (cube):

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 5:point is in front of view volume	$z>z_{min}$

- Application analogous to that for parallel canonic view volume

Exercise!

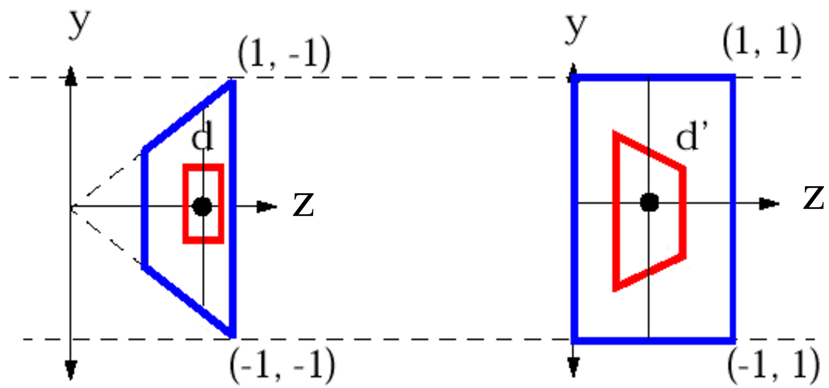
Clipping made even easier (1)



- Clipping against the parallel canonic view volume is easier than for the perspective
- Hardware implementations typically provide a single clip procedure
- Idea: convert all projections into orthogonal projections
  - Equivalent to distorting the object such that the orthogonal projection of the distorted object is the same as the desired projection of the non distorted (original) object

Clipping made even easier (1)

- Perspective normalization needs an additional step: transforming the perspective canonic view volume into the parallel one



- We will represent this transformation as an additional matrix in homogeneous coordinates
- In order to avoid errors, the clipping in this case needs to be done in homogeneous coordinates

Perspective-to-parallel canonical view volume (1)

- It can be shown that the transformation from the perspective-projection canonical view volume to the parallel-projection canonical view volume is

$$\mathbf{M}_{\text{cpp}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{1}{1+z_{\min}} & \dfrac{-z_{\min}}{1+z_{\min}} \\ 0 & 0 & -1 & 0 \end{bmatrix}; \quad z_{\min} \neq -1$$

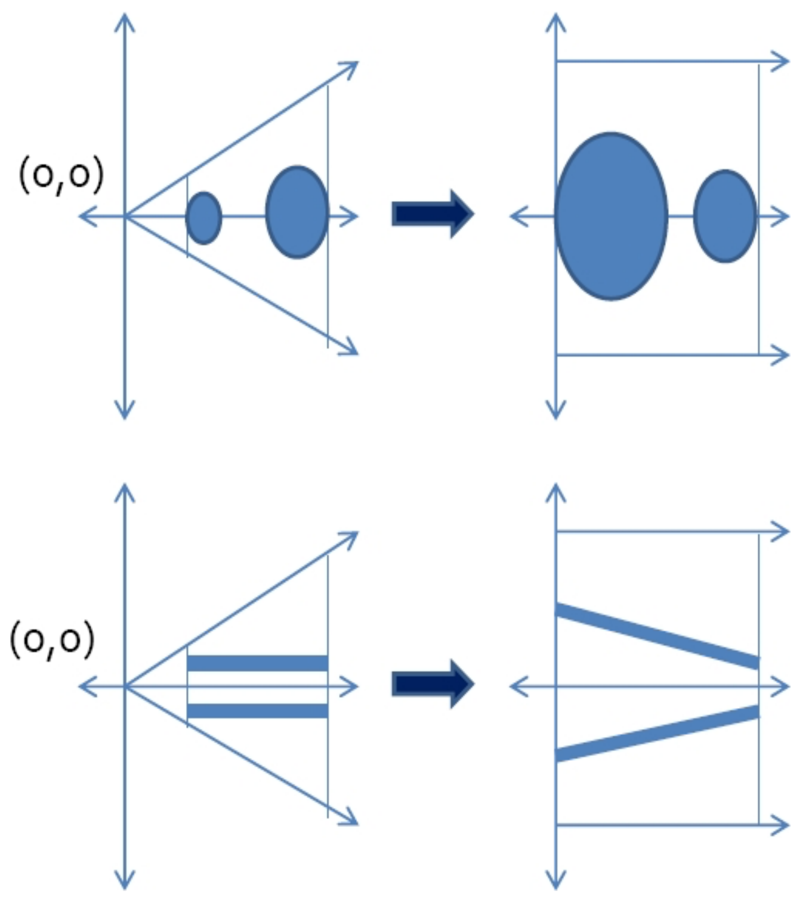
- The final normalizing transformation for the perspective view volume is now

$$\mathbf{N}'_{per} = \mathbf{M}_{\text{cpp}}\mathbf{S}_{\text{per}} \cdot \mathbf{H}_{\text{par}} \cdot \mathbf{T}(-\mathbf{PRP}) \cdot \mathbf{R} \cdot \mathbf{T}(-\mathbf{VRP})$$

Exercise!

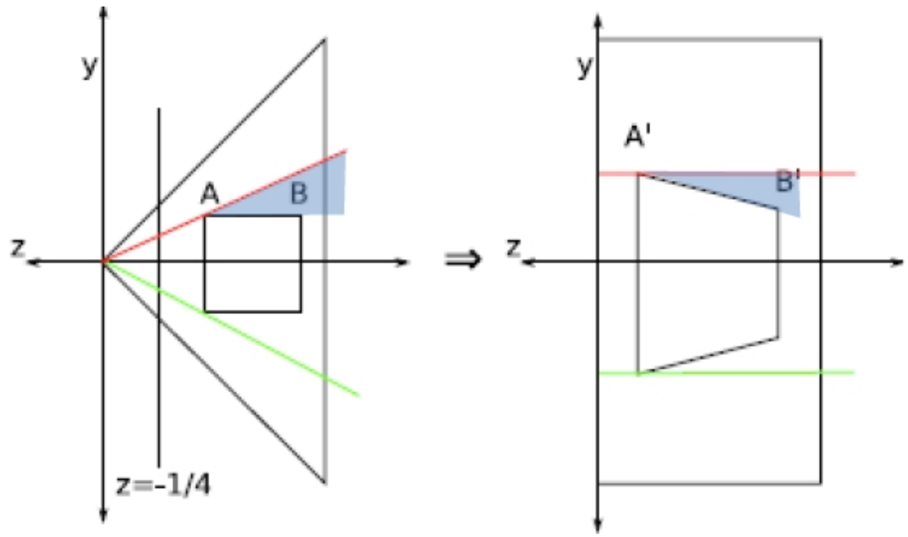
Perspective-to-parallel canonical view volume (2)

- After applying the previous transformation, the orthographic projection yields the perspective view. Why does it work?
  - The key is **unhinging** step
- The example in the top of the figure shows:
  - The closer the object to the near clip plane the more it is enlarged
  - Result: closer objects appear larger
- The example in the bottom part of the figure
  - After unhinging, parallel lines "fan out" at the near clip plane
  - Result: converging lines



Example from *Introduction to Computer Graphics* by Andries Van Dam

Perspective-to-parallel canonical view volume (3)



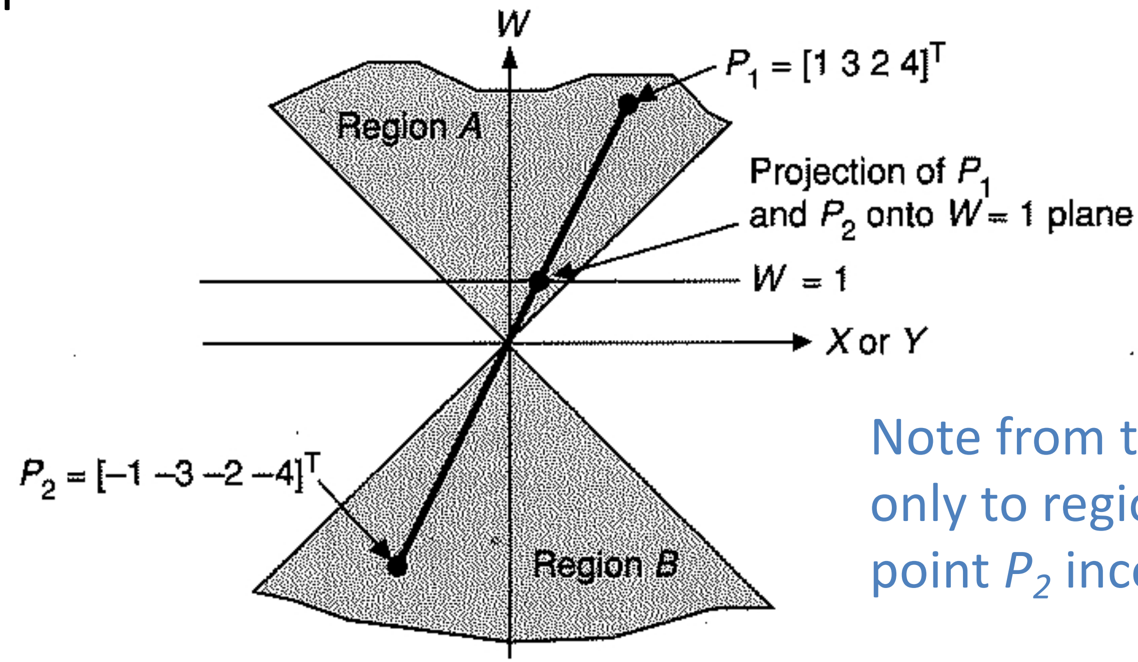
Example from *Introduction to Computer Graphics*
by Andries Van Dam

- Another way to demonstrate the effect of creating perspective by unhinging is to use occlusion (where some elements in the scene are blocked by others)
- The figure demonstrates that all points that were visible/obscured in the perspective view remain visible/obscured after unhinging and parallel projection

Clipping in homogeneous coordinates (1)

- After the unhinging transformation, the clipping must be done in homogeneous coordinates, otherwise errors can occur

- Also, some other computer graphics operations (like generation of curves, the use of rational parametric splines etc.) can result in points with negative W component in the homogeneous representation



FvDFH book, Fig. 6.57

Note from this example: if we clip only to region A, than we discard point P_2 incorrectly!

Clipping in homogeneous coordinates (2)

- The canonical view volume for parallel projection is given by

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

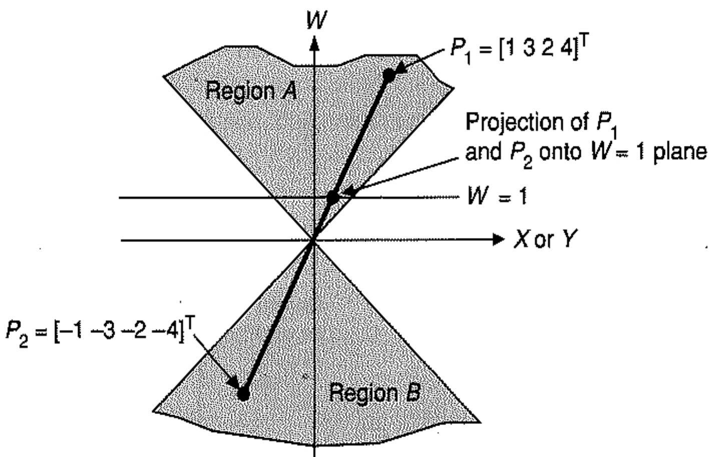
- The corresponding inequalities in homogeneous coordinates are

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

- We must consider separately the cases with $W > 0$ and $W < 0$:

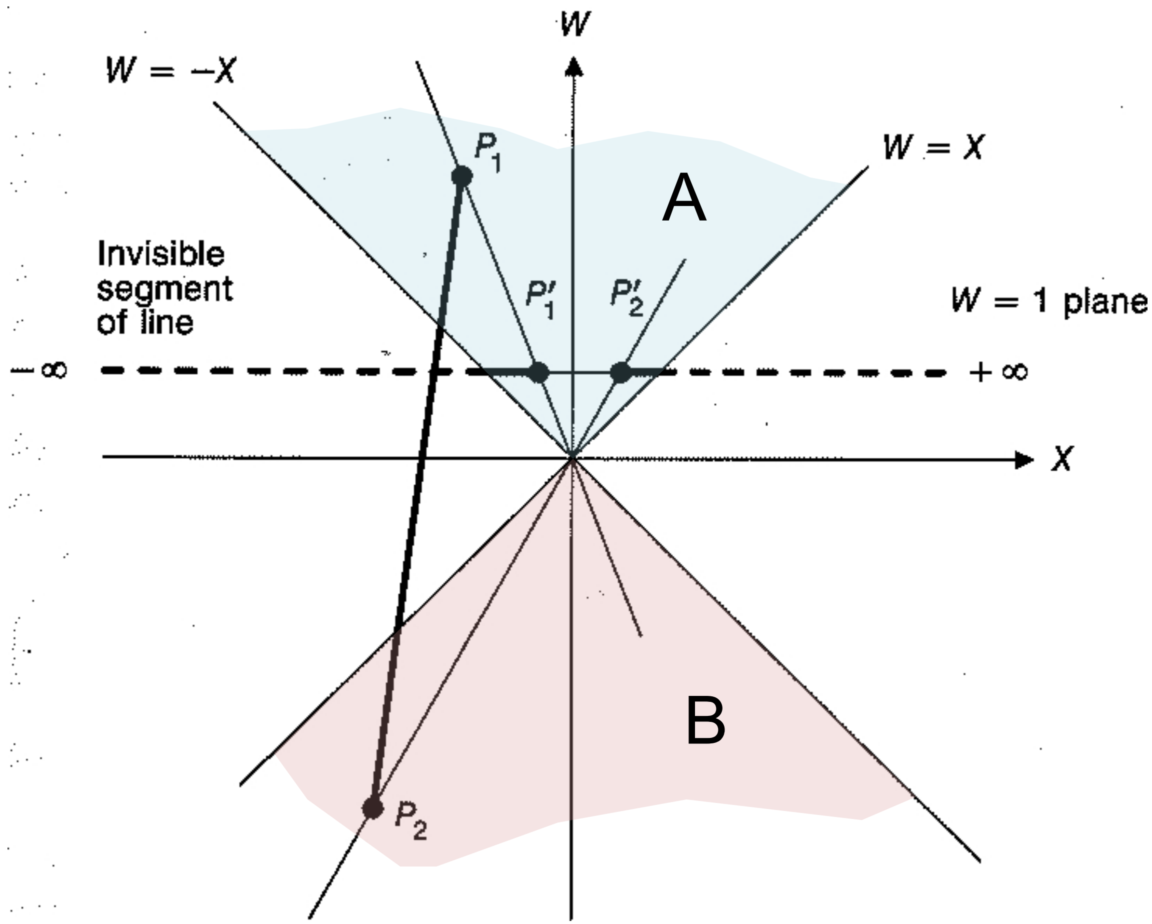
$$W > 0: -W \le X \le W, \ -W \le Y \le W, \ -W \le Z \le 0,$$

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



FvDFH book, Fig. 6.57

Clipping in homogeneous coordinates (3)



- In this example, the end points P_1P_2 have opposite values of W. The projection onto the plane $W=1$ yields two segments. We can clip against region A, then negate both end points clip again against A.

Summary

- We learned how to realize practically projections and clipping
- Crucial step: normalization to canonic view volumes for parallel and perspective projections
- After this view normalization the projection and clipping are relatively simple operations
- It is useful to simplify the clipping operation for perspective projection further and it is also useful to have a unique clipping operation for all the projections in hardware
- For this reason, we do an additional normalization: perspective to parallel canonic view volume
- The clipping in this case needs to be done in homogeneous coordinates