Viewing Transform

Intended Learning Outcomes

- Able to set up a camera coordinate system
- Understand the properties of different projection methods
- Able to set up the required projection matrices and use appropriate OpenGL commands to realize the projection
- Describe the operation and function of clipping

Image generation process

- A camera has its own coordinate system $\mathbf{X}^{(VC)}$-$\mathbf{Y}^{(VC)}$-$\mathbf{Z}^{(VC)}$, called the viewer coordinate system
- viewer coordinate system is alternatively called camera coordinate system
- To generate an image, we first need to define a camera, then transforming the 3D scene from world coordinate system (WC) to viewing coordinate system (VC)

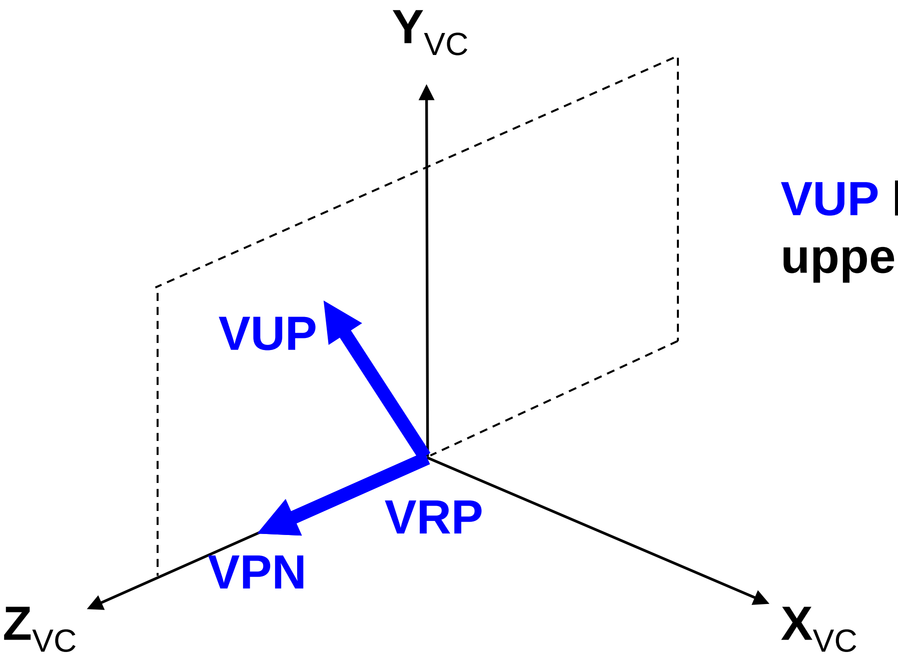
$$\mathbf{X}^{(WC)}\text{-}\mathbf{Y}^{(WC)}\text{-}\mathbf{Z}^{(WC)} \rightarrow \mathbf{X}^{(VC)}\text{-}\mathbf{Y}^{(VC)}\text{-}\mathbf{Z}^{(VC)}$$

- Then projecting each point to a view plane

To specify a viewer coordinate system, we need to specify three vectors

- View Reference Point (**VRP**): origin of the viewing coordinate system (i.e. physical location of the camera)
- View Plane Normal (**VPN**): a vector giving the pointing direction of the camera (i.e. +ve $Z^{(VC)}$ axis of the camera $X^{(VC)}$-$Y^{(VC)}$-$Z^{(VC)}$)
- View UP Vector (**VUP**) : a vector defining what is the upward direction for the film (image)

- Note 1: These vectors do not need to be unit vector
- Note 2: These vectors are in WC



VUP lies on the upper Z_{VC} – Y_{VC} plane

$\mathbf{Z}_{VC}$ = |**VPN**| (unit vector in WC)

$\mathbf{X}_{VC}$ = | **VUP** X **VPN** | (unit vector in WC)

$\mathbf{Y}_{VC} = \mathbf{Z}_{VC} \times \mathbf{X}_{VC}$ (unit vector in WC)

Note: | | is used in the notes to denote normalization to unit vector

Transformation from WC to VC

- $\mathbf{P}^{(VC)} = \mathbf{M}_{VC \leftarrow WC}\, \mathbf{P}^{(WC)}$

- Applying coordinate system transformation method 1:

$$\mathbf{M}_{VC \leftarrow WC} = \begin{pmatrix} \mathbf{X}_{VC} & \mathbf{Y}_{VC} & \mathbf{Z}_{VC} & \mathbf{VRP} \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1}$$

- Note: $\mathbf{X}_{VC}$, $\mathbf{Y}_{VC}$, $\mathbf{Z}_{VC}$ are unit column vector in WC

OpenGL commands

- *glMatrixMode (GL_MODELVIEW);*
- *gluLookAt (x0, y0, z0, xref, yref, zref, Vx, Vy, Vz);*

- **VRP** = (x0, y0, z0)
- **VPN** = (x0, y0, z0) – (xref, yref, zref)
- **VUP** = (Vx, Vy, Vz)

- To remember this, it is convenient to remember (x0, y0, z0) as where the camera is placed, (xref, yref, zref) as where the center of the scene is, and (Vx, Vy, Vz) as a vector that tells where it's up for the camera

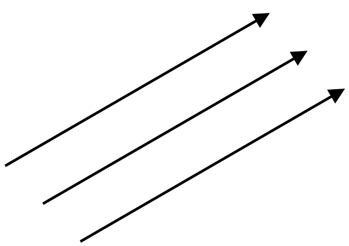
View Plane

- Also called projection plane or image plane
- It is usually a plane defined by Z_{VC} = constant, i.e., parallel to the $\mathbf{X}_{VC}$ – $\mathbf{Y}_{VC}$ plane
- As the name implies, a 3D point (X, Y, Z) in viewer coordinates is projected to a 2D point lying on the view plane
- i.e. 3D becomes 2D

Projections : project $(X, Y, Z)^{(VC)}$ to $(x, y)^{(VC)}$

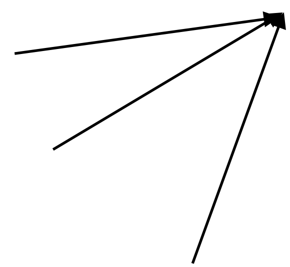
- Two general types: Parallel and Perspective Projections

Parallel projection



all light rays are parallel

Perspective projection



all light rays converge on a common point called projection reference point (PRP)

Parallel projection can be considered as the special case of perspective projection when PRP = ∞

Different properties

Parallel projection	Perspective Projection
coordinate positions are transformed to the projection plane along // lines i.e. center of projection at infinity	coordinate positions are transformed to the projection plane along lines that converge to a point called the center of projection (Projection Reference Point)
preserves relative proportions	does not preserve
use in engineering drafting	use in realistic views

Parallel Projections

- Specify a *projection vector* – a direction vector
- Two types:

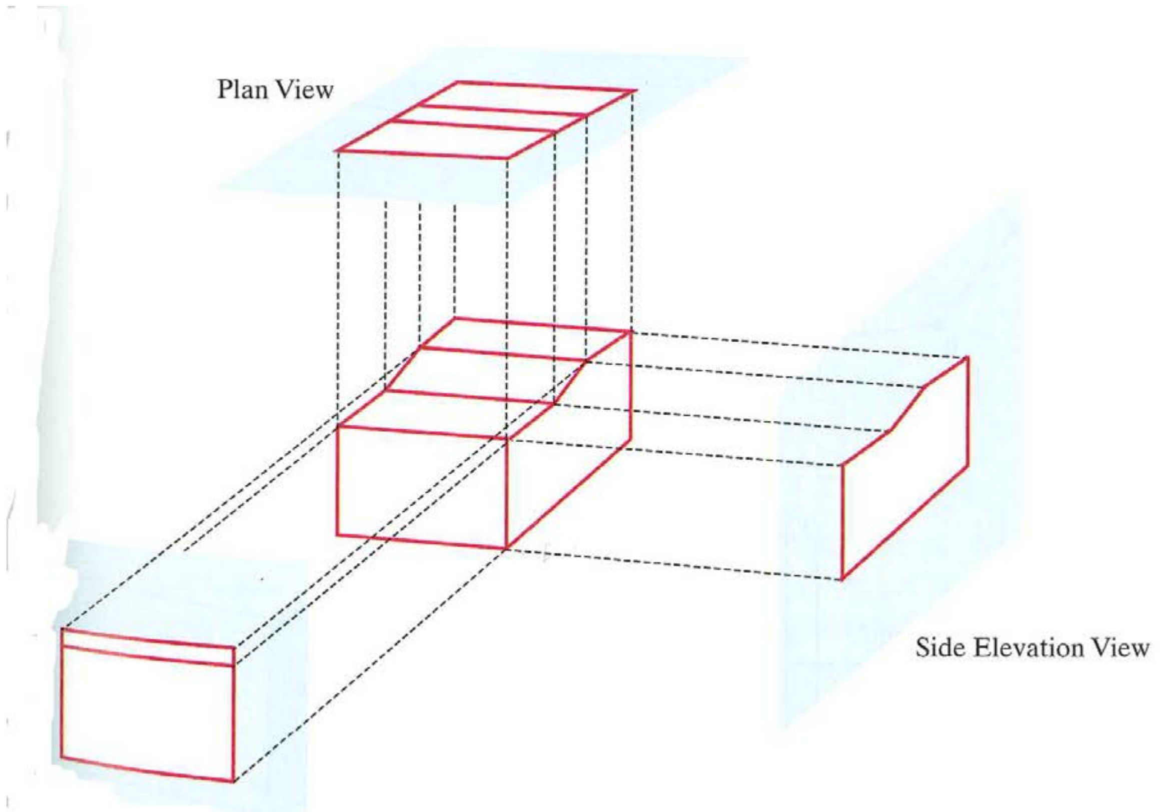
- *Orthographic projection* - projection vector $\perp$ projection plane

- *Oblique projection* - projection vector not $\perp$ to projection plane

Orthographic Projection ($\alpha = 90^o$)

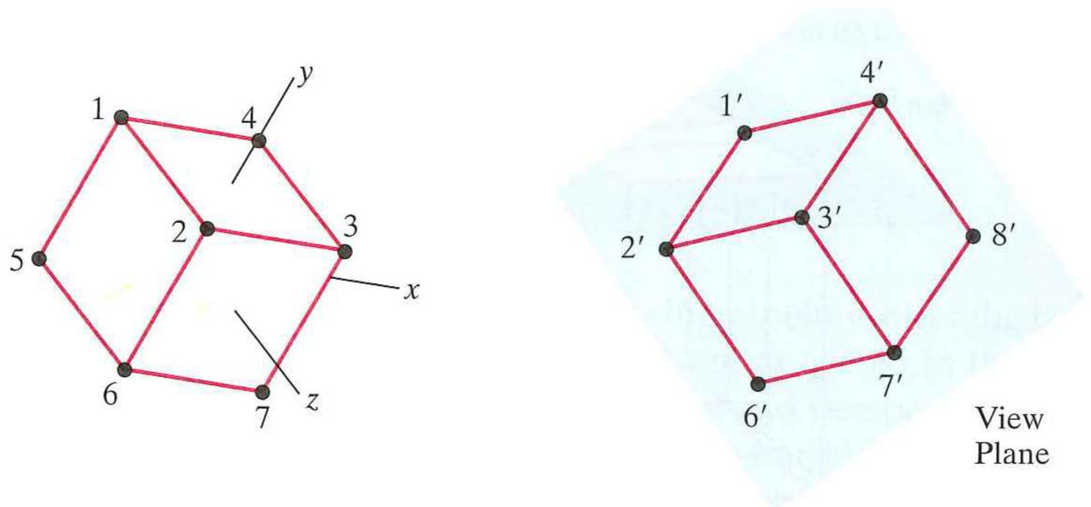
- Two types:
- *Front elevation*, *side elevation*, *rear elevation*, *plan view* - only X-Y, X-Z or Y-Z is shown
- *Isometric projection* - projection vector = ($\pm$1, $\pm$1, $\pm$1)
  - For a cube, each side will be displayed equally
  - The 8 possibilities corresponds to viewing in the 8 octants

Front elevation, side elevation, rear elevation, plan view



front elevation view

Isometric projection

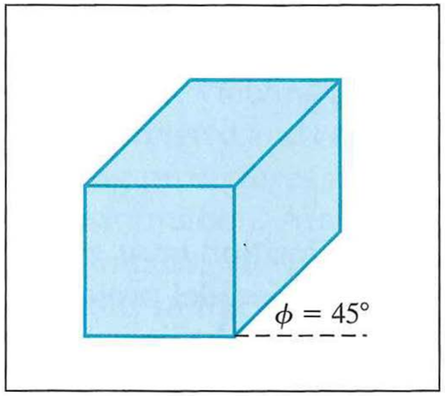


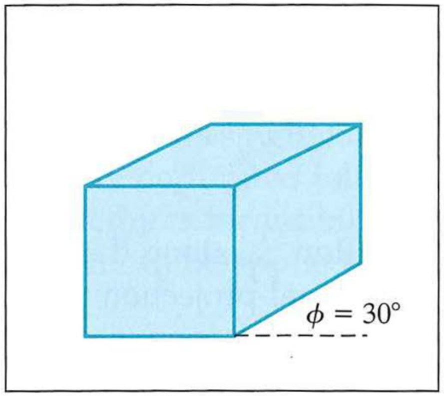
Oblique Projection ($\alpha \neq 90^o$)

- Two types:

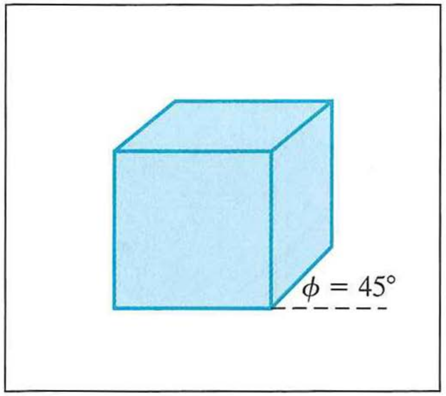
- *Cavalier projection* - projection vector makes an angle α of $\tan^{-1}1$ with projection plane
  - for a cube, length of X axis, Y axis and Z axis will remain the same

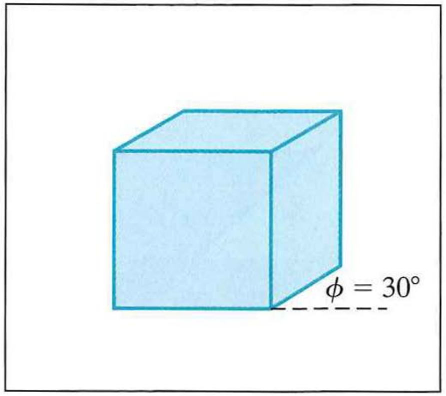
- *Cabinet projection* - projection vector makes an angle α of $\tan^{-1}2$ with projection plane
  - for a cube, length of X axis, Y axis will remain the same; length of Z axis will be halved.



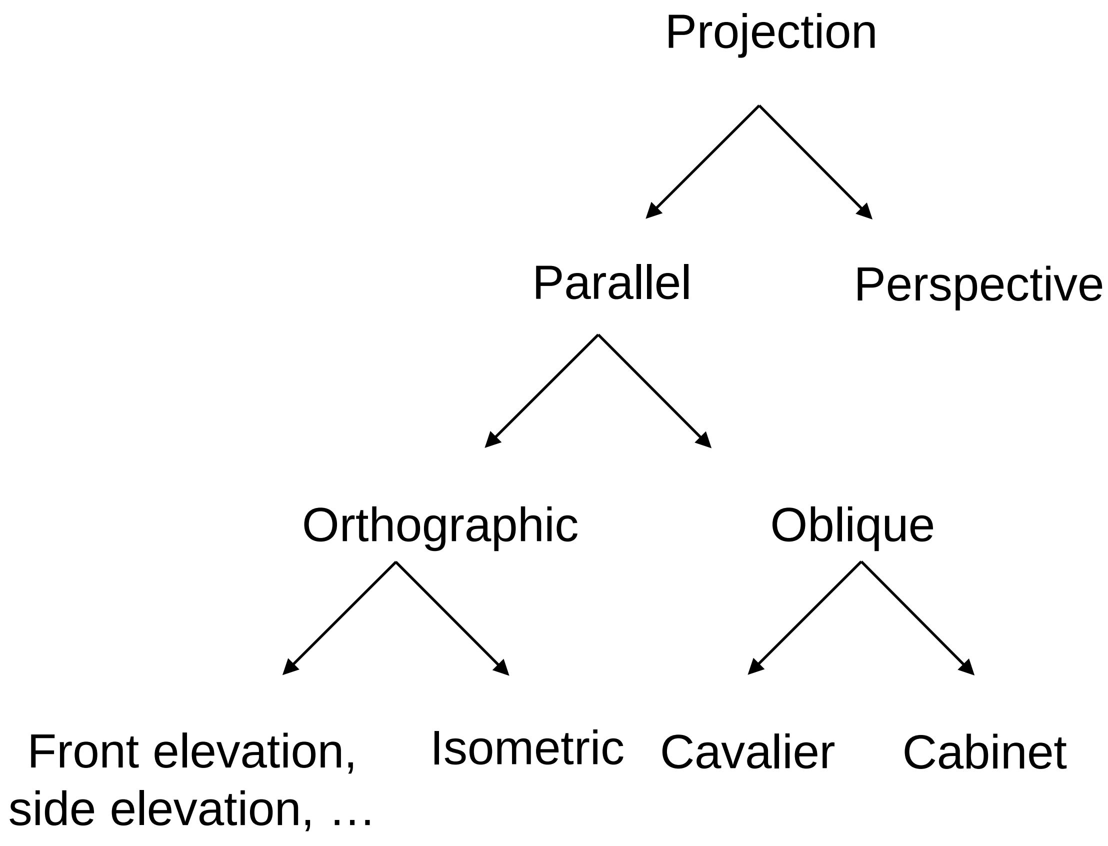


Cavalier Projection





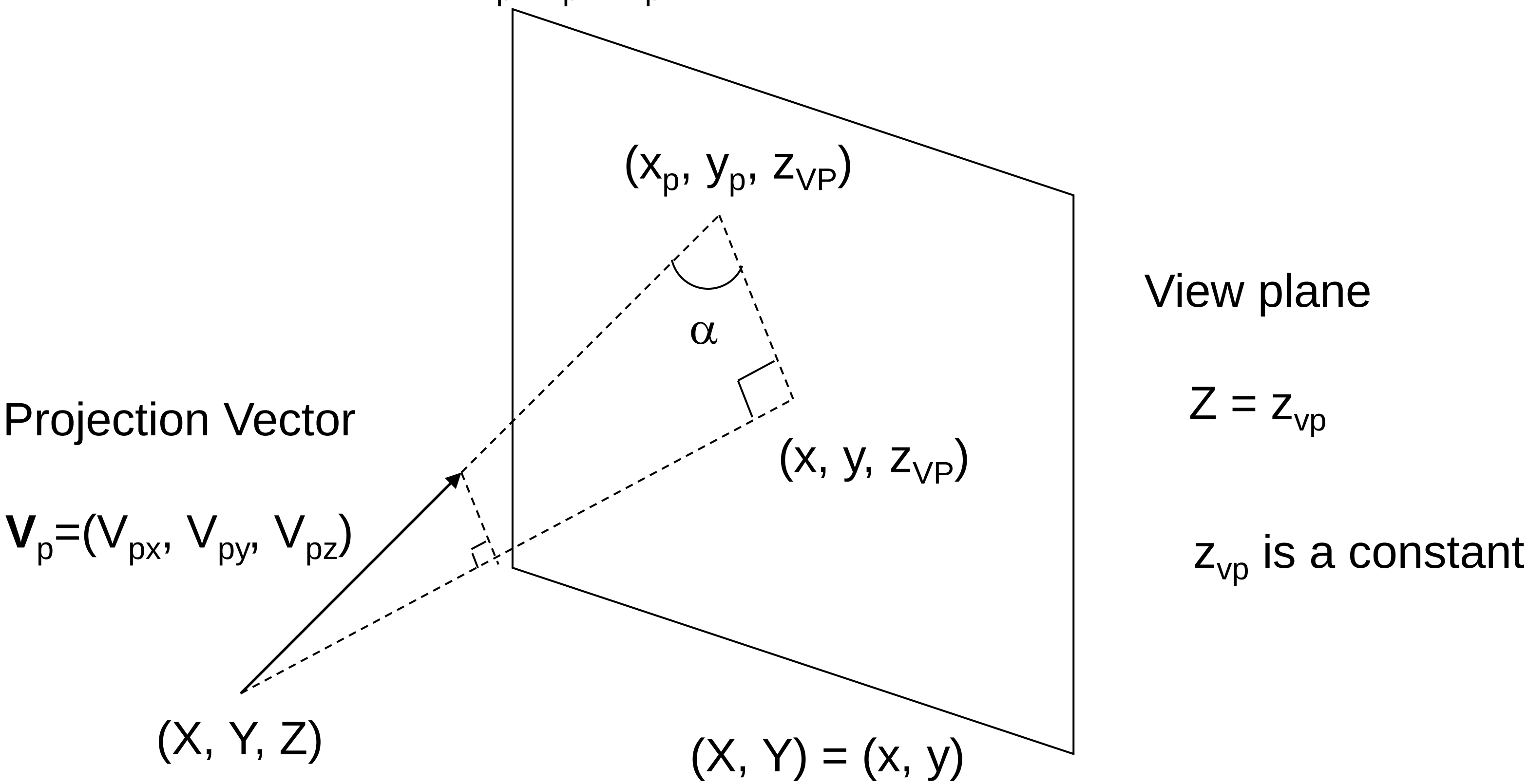
Cabinet Projection


Projection
Parallel
Perspective
Orthographic
Oblique
Front elevation,
side elevation, …
Isometric
Cavalier
Cabinet

4 x 4 Transform for Parallel Projection

All quantities in this slide are already in VC

$\mathbf{P}^{(VC)} = (X, Y, Z)$ $\mathbf{P}^{(im)} = (x_p, y_p, z_{vp})$



$\alpha = 90^o$ Orthographic projection $\alpha \neq 90^o$ Oblique projection

By similar triangles,

$$\frac{x_p - X}{z_{vp} - Z} = \frac{V_{px}}{V_{pz}}$$

$$\frac{y_p - Y}{z_{vp} - Z} = \frac{V_{py}}{V_{pz}}$$

Rearranging,

$$x_p = X + (z_{vp} - Z)\frac{V_{px}}{V_{pz}}$$

$$y_p = Y + (z_{vp} - Z)\frac{V_{py}}{V_{pz}} \qquad (1)$$

- $\mathbf{P}^{(im)} = (x_p, y_p, z_{vp}, 1)$ $\mathbf{P}^{(VC)} = (X, Y, Z, 1)$
- $\mathbf{P}^{(im)} = \mathbf{M}_{parallel}\, \mathbf{P}^{(VC)}$

$$\mathbf{M}_{parallel} = \begin{pmatrix} 1 & 0 & -\dfrac{V_{px}}{V_{pz}} & z_{vp}\dfrac{V_{px}}{V_{pz}} \\ 0 & 1 & -\dfrac{V_{py}}{V_{pz}} & z_{vp}\dfrac{V_{py}}{V_{pz}} \\ 0 & 0 & 0 & z_{vp} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

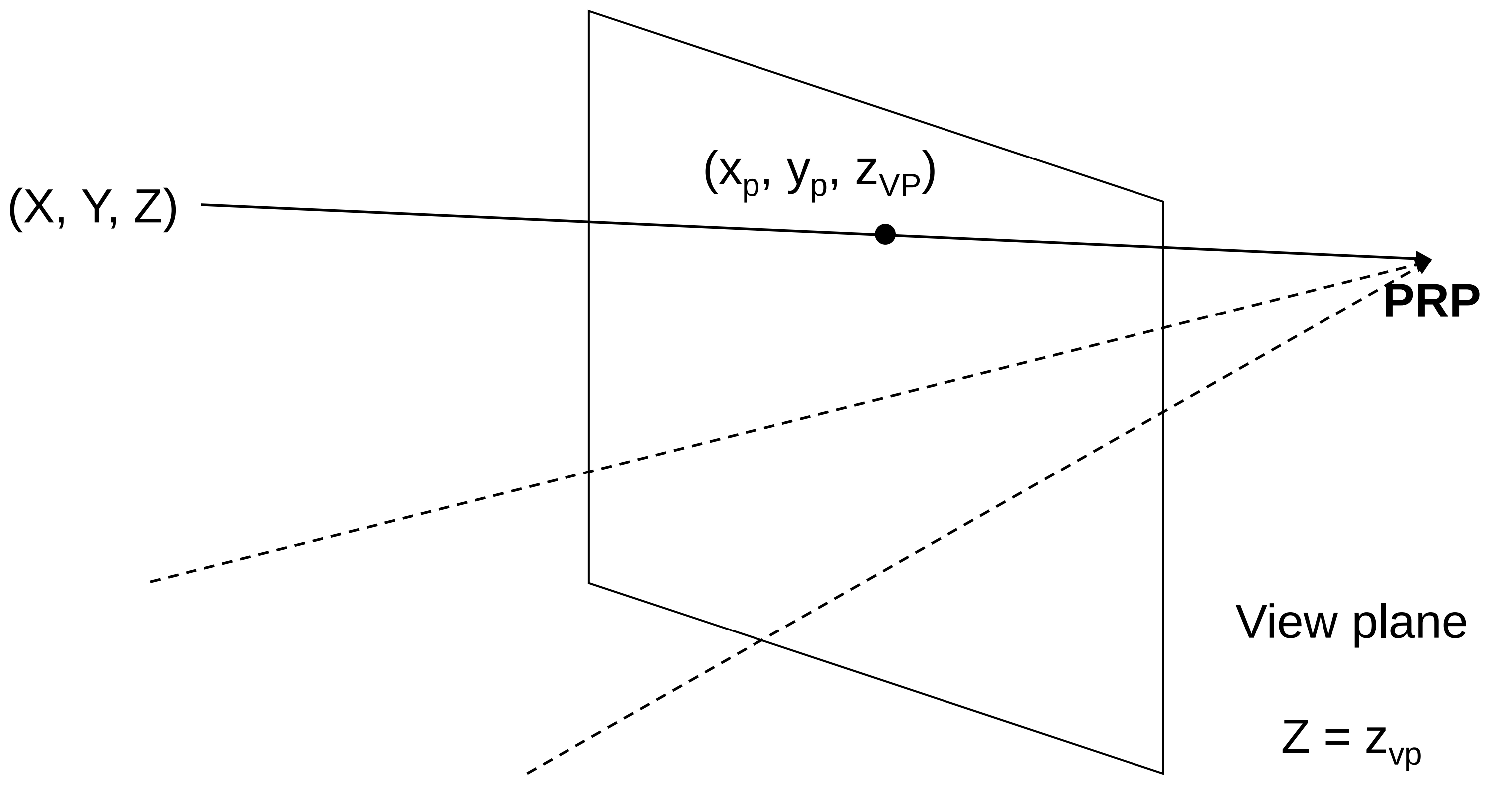
- Verify that the first two rows implement eqn (1) above
- The third row is such that the projected point is at $Z = z_{vp}$. However, this is not maintained; in OpenGL, $\mathbf{p}^{(im)} = (x_p, y_p, Z)$, i.e. the original Z is kept for depth tests

- Third row of eqn (10-13), pg. 350 of text is set to

  0 0 1 0, which achieves the same effect as maintaining the original Z.

Perspective Projection

- ALL light rays goes through the Projection Reference Point (**PRP**), also called center of projection.



(x_p, y_p, z_{VP})
(X, Y, Z)
PRP
View plane
$Z = z_{vp}$

Example:

i) **PRP** = **VRP**

ii) Z = z_{vp} is the view plane

By similar triangles,

$$\frac{x_p}{z_{vp}} = \frac{X}{Z} \qquad \frac{y_p}{z_{vp}} = \frac{Y}{Z}$$

Multiplying each side by z_{vp} yields

$$x_p = \frac{z_{vp} \cdot X}{Z} = \frac{X}{Z / z_{vp}} \qquad (2)$$

$$y_p = \frac{z_{vp} \cdot Y}{Z} = \frac{Y}{Z / z_{vp}}$$

- $\mathbf{P}^{(VC)} = (X, Y, Z, 1)$
- $\mathbf{P}^{(im)} = \mathbf{M}_{perspective}\ \mathbf{P}^{(VC)}$

$$\mathbf{M}_{perspective} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/z_{vp} & 0 \end{bmatrix}$$

- Verify that the first two rows and the fourth row implements eqn (2) above using homogeneous coordinates operation
- Note that the fourth row is not 0 0 0 1 anymore
- The third row is such that the projected point is at Z = z_{vp}. However, this is not maintained; in OpenGL, $\mathbf{p}^{(im)}$ = (x_p, y_p, Z), i.e. the original Z is kept for depth tests

Clipping

- Any object not within the clipping volume does not need to be processed – this eliminates most of the objects at one go
- For a convex clipping volume bounded by planes, one can check whether a point is inside by checking the signs of the plane equations (see Lecture 2).

OpenGL – first set matrix mode

- *glMatrixMode (GL_PROJECTION);*

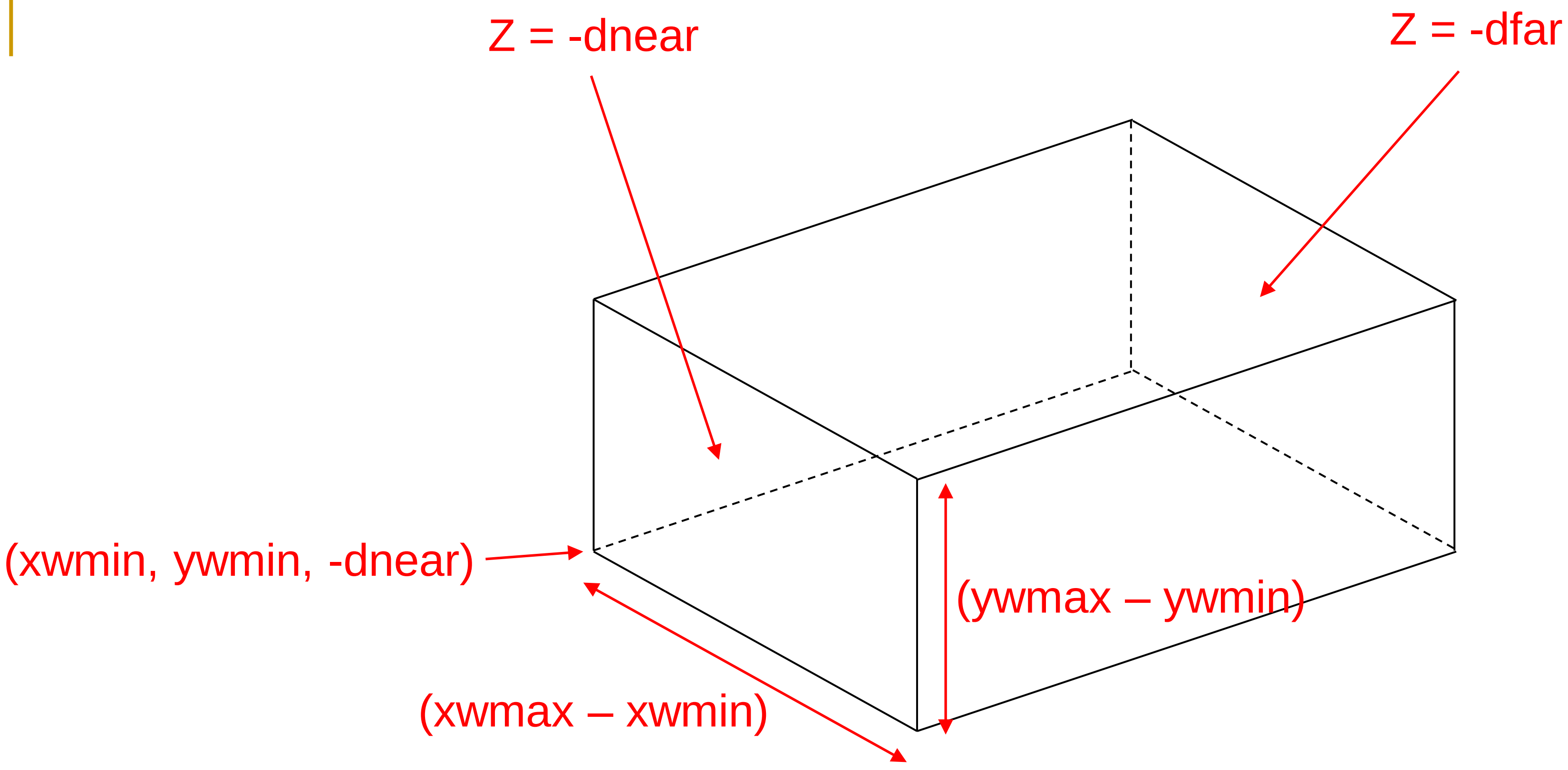
- Note: *GL_PROJECTION* is used as it deals with projection
- There are two 4 x 4 composite transformation matrices: *GL_MODELVIEW* and *GL_PROJECTION*
- A point is pre-multiplied by

  [*GL_PROJECTION*] [*GL_MODELVIEW*]

- *glOrtho* and *gluPerspective* commands may be used

OpenGL – Orthographic projection

- *glOrtho (xwmin, xwmax, ywmin, ywmax, dnear, dfar)*
  - Projection vector $V_p = (0, 0, 1)$
  - Clipping planes: Z = -dnear   Z = -dfar
  - Near clipping plane Z = -dnear also serve as the view plane
  - Only points whose X and Y are in |xwmin, xwmax| and |ywmin, ywmax| respestively are displayed
  - Clipping volume is a rectangular box



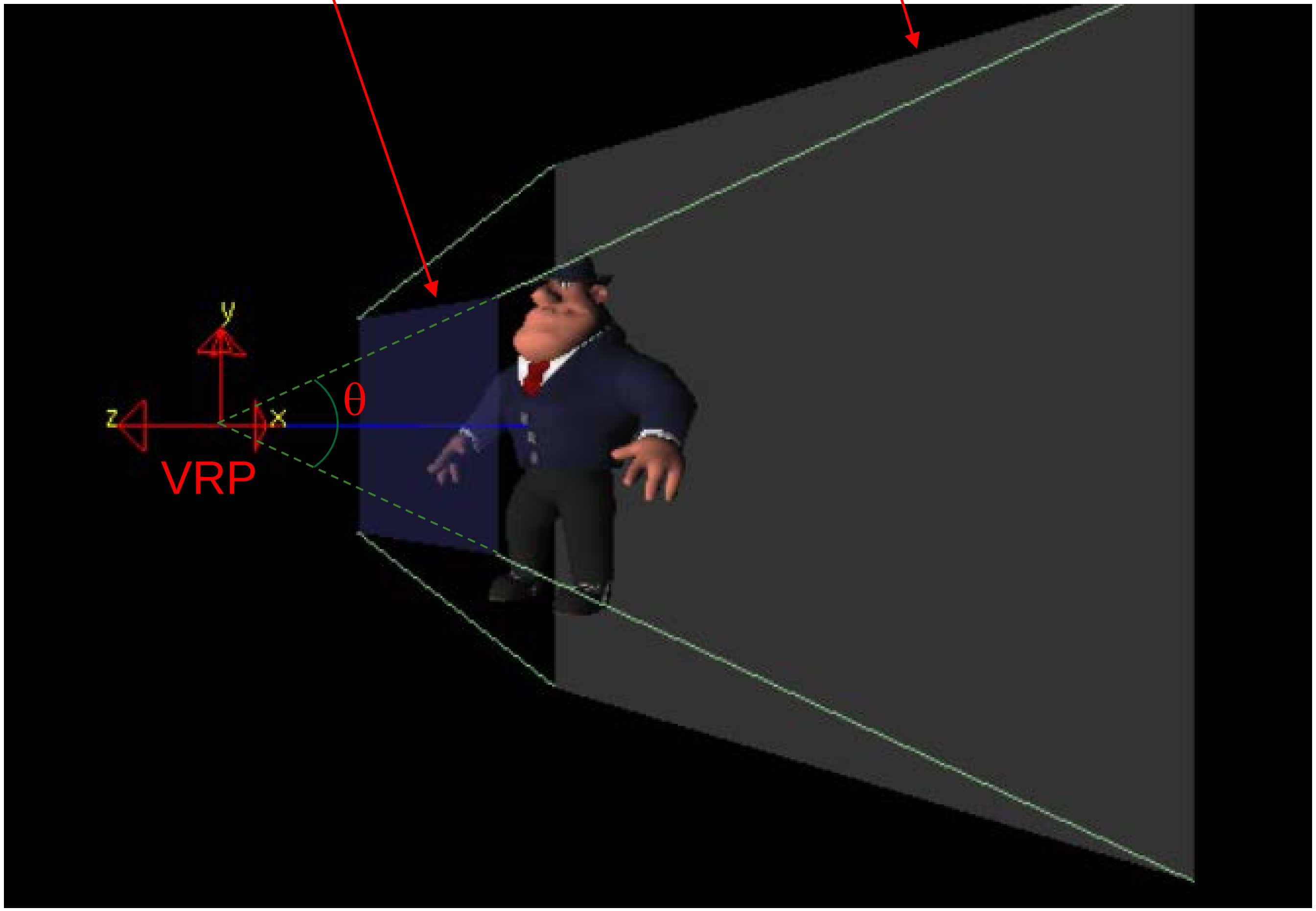
Only objects inside the rectangular shaped clipping volume is further processed

OpenGL – Perspective projection

- *gluPerspective (theta, aspect, dnear, dfar)*
  - **PRP** = **VRP**
  - Z = -*dnear* is the view plane (note the –ve sign)
  - dnear and dfar define the near and far clipping planes Z = -*dnear* and Z = -*dfar* respectively
  - *theta* is the angle of view
  - *aspect* = (width /height)
  - *theta* and *aspect* together determines size of image window
  - clipping volume is a frustum

Z = -dnear

Z = -dfar



aspect = width /height of the blue plane

Only objects inside the frustum shaped clipping volume is further processed

References

- Text : Ch. 10.2–10.7 discusses the viewing transform and the various types of projection
- Text : Ch. 10.8 discusses general perspective projection and then discusses the special case. We only discuss the special case here
- Text : Ch. 10.9–10.10 discusses the OpenGL commands