Viewing, Projection and Viewport Transformations

2016 Spring

National Cheng Kung University

Instructors: Min-Chun Hu 胡敏君

Shih-Chin Weng 翁士欽 (西基電腦動畫)

Classical and Computer Viewing

Viewing

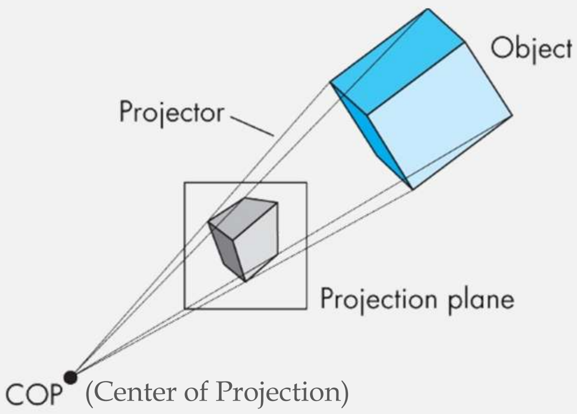
- How to describe our virtual camera?
- What is the relationship between classical viewing techniques and computer viewing?
- How to implement projection?

Outline

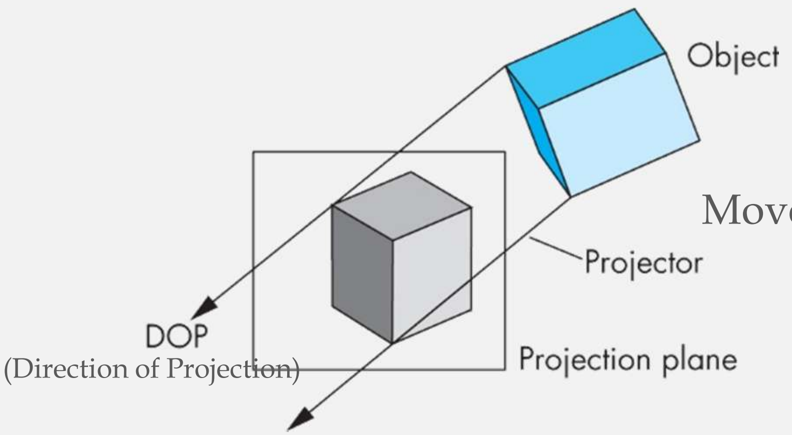
- Classical views
- Computer viewing
- Projection matrices

Classical Viewing

- Viewing requires three basic elements
  - One or more objects
  - A viewer with a projection surface
  - Projectors that go from the object(s) to the projection surface



Perspective View



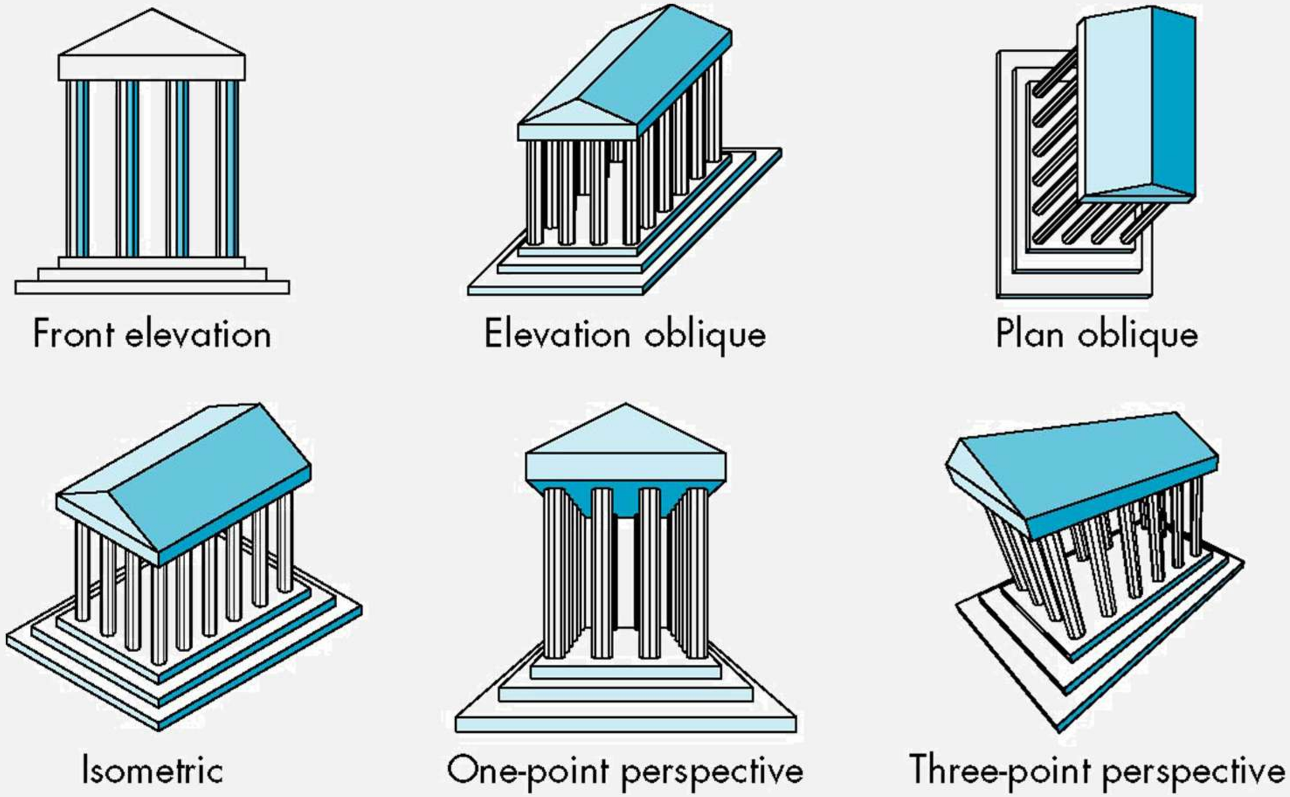
Move the COP to infinity.

Parallel View

Perspective View vs Parallel View

- Classical viewing developed different techniques for drawing each type of projection
- Mathematically parallel viewing is the limiting case of perspective viewing
  - Set the COP to infinity
- Computer graphics treats all projections the same and implements them with a single pipeline

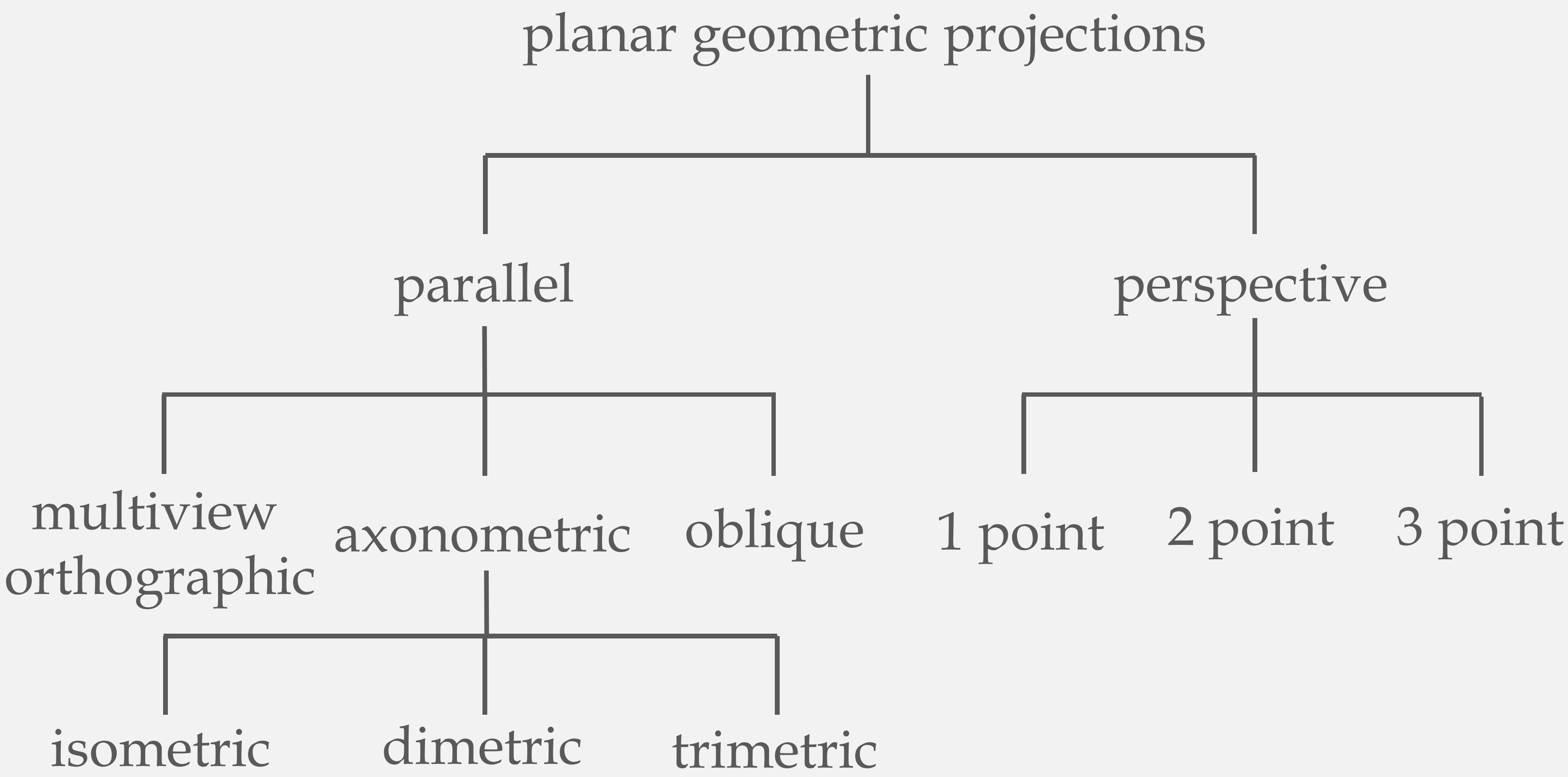
Classical Projections



Planar Geometric Projections

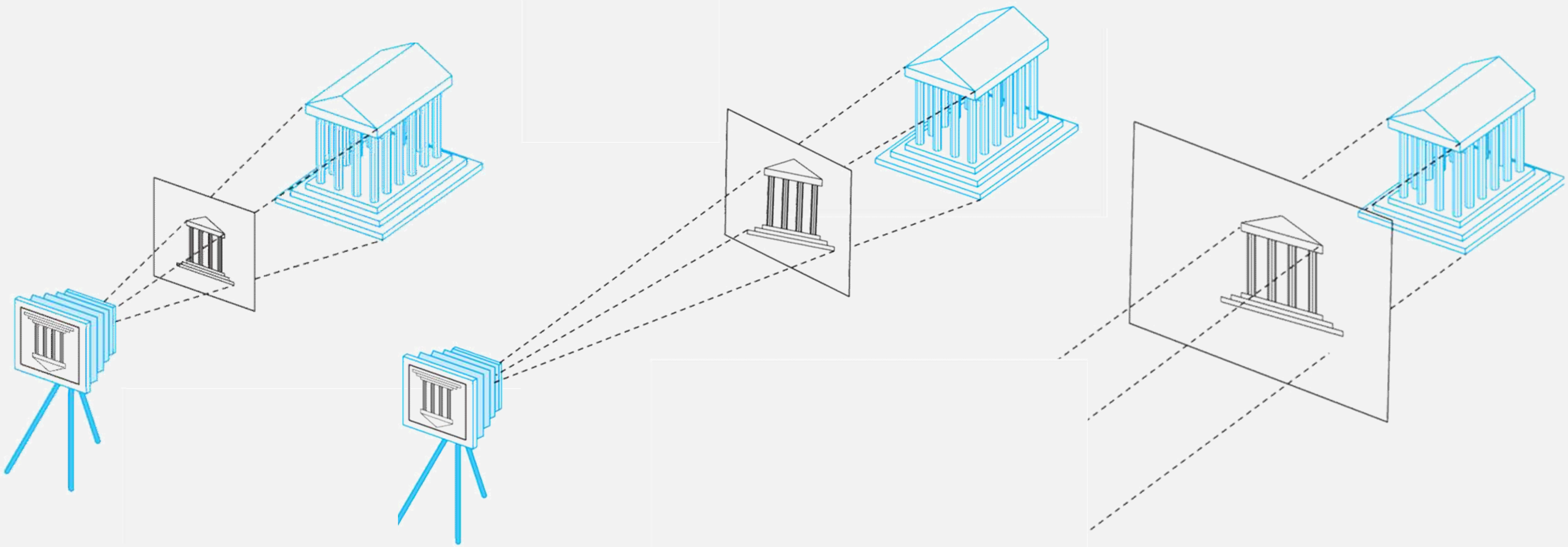
- Standard projections project onto a plane.
- Projectors are lines that either converge at a center of projection or are parallel
- Such projections preserve lines but not necessarily angles

Taxonomy of Planar Geometric Projections

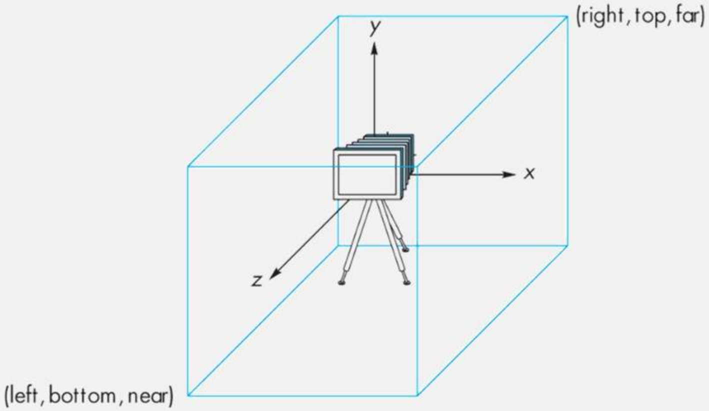


Orthographic View

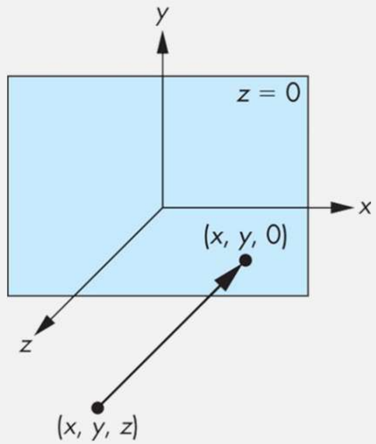
- Projectors are orthogonal to projection surface



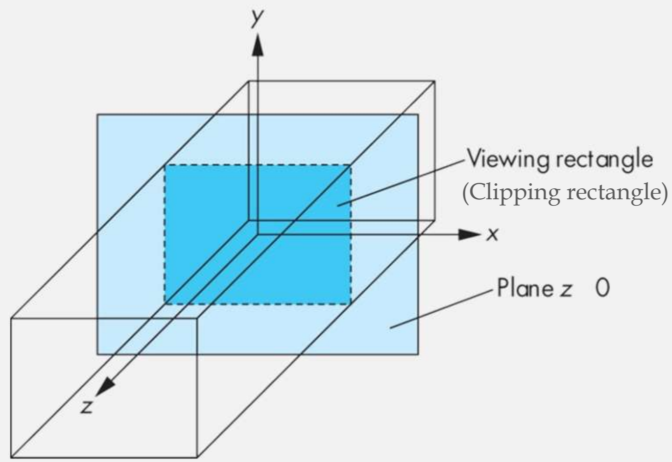
Orthographic View (Cont.)



The default camera and an orthographic view volume.



Orthographic Projection.



Viewing Volume

Multiview Orthographic Projection

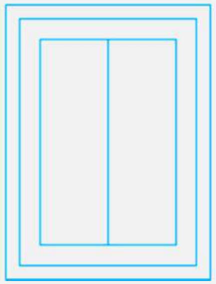
- Projection plane parallel to principal faces
- Usually form front, top, side views

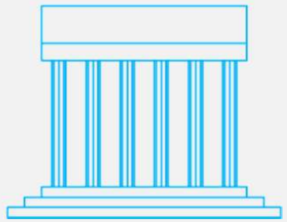
isometric (not multiview orthographic view)

front

in CAD and architecture, we often display three multiviews plus isometric

top





side

Advantages and Disadvantages

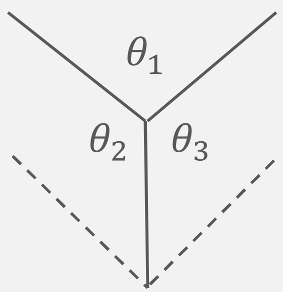
- Preserves both distances and angles
  - Shapes preserved
  - Can be used for measurements
    - Building plans
    - Manuals

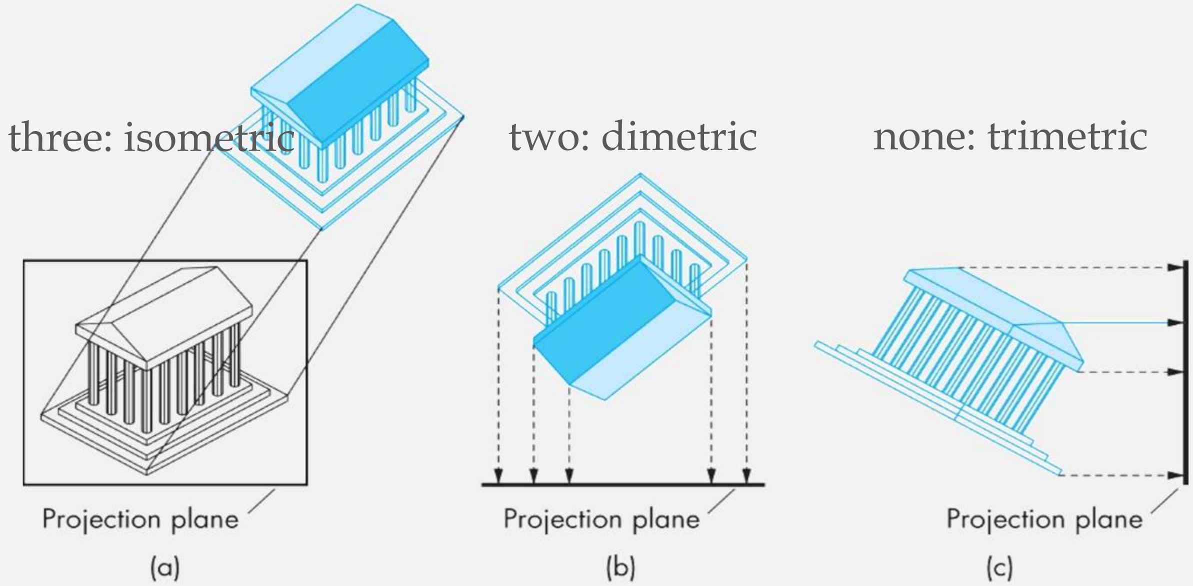
- Cannot see what object really looks like because many surfaces hidden from view
  - Often we add the isometric

Axonometric Projections

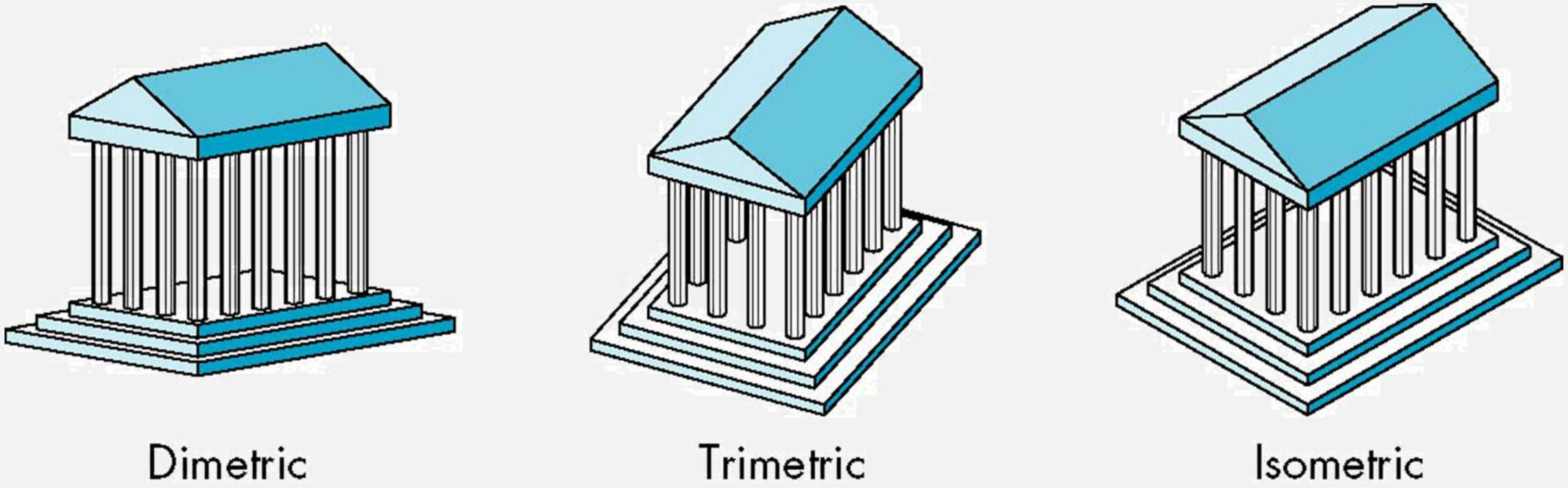
- Projectors are still orthogonal to the projection plane.
- Allow projection plane to move relative to object

Classify by how many angles of a corner of a projected cube are the same:





Types of Axonometric Projections

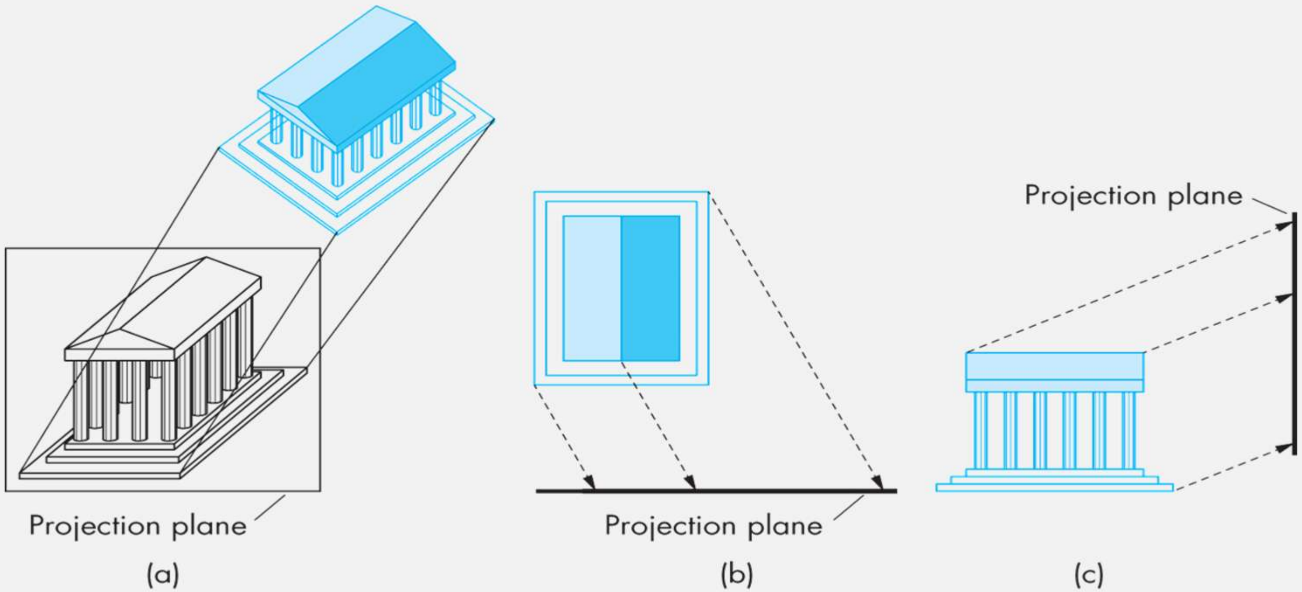


Advantages and Disadvantages

- Lines are scaled (foreshortened) but can find scaling factors
- Lines preserved but angles are not
  - Projection of a circle in a plane not parallel to the projection plane is an ellipse
- Does not look real because far objects are scaled the same as near objects
- Used in CAD applications

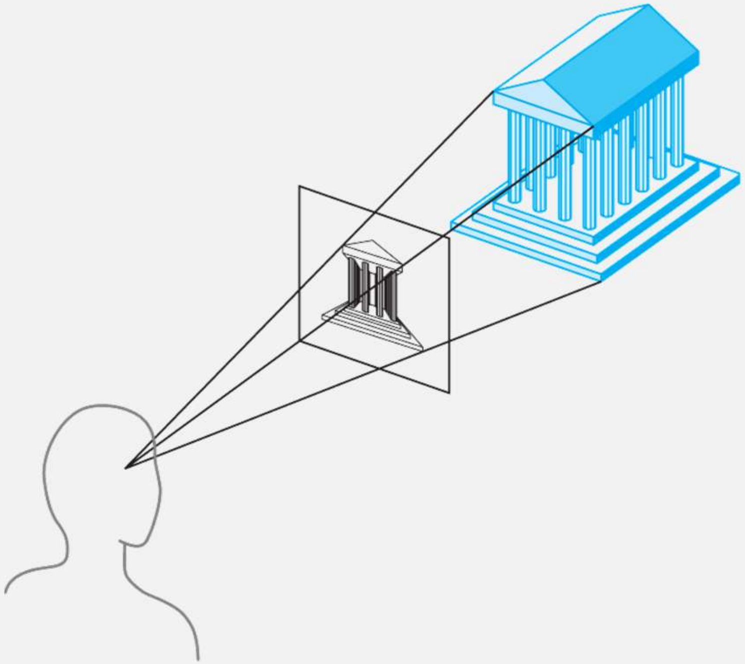
Oblique Projection

- Arbitrary relationship between projectors and projection plane



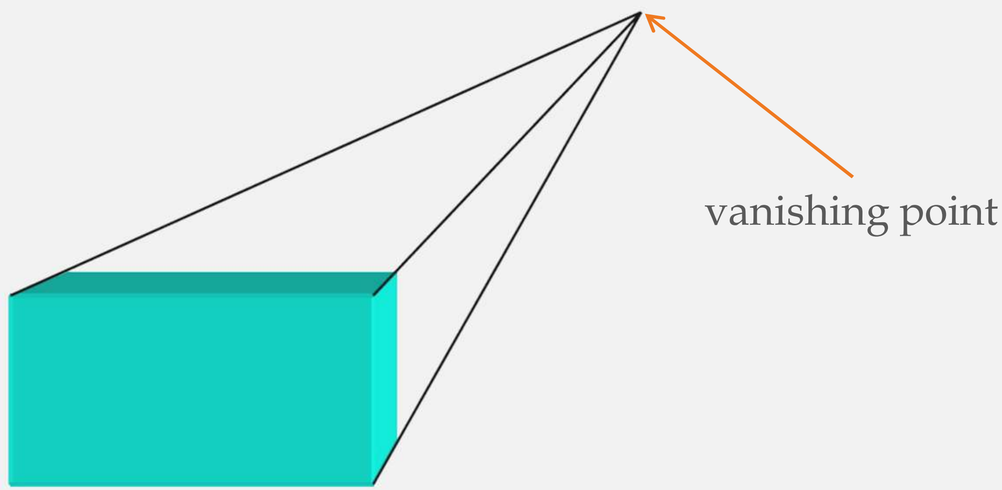
Perspective Projection

- Projectors converge at center of projection
- The viewer is located symmetrically with respect to the projection plane



Vanishing Points

- Parallel lines (not parallel to the projection plan):
  - converge at a single point in the projection (the *vanishing point*)
- Drawing simple perspectives by hand uses these vanishing point(s)



Three-Point Perspective

- No principal face parallel to the projection plane
- Three vanishing points for cube

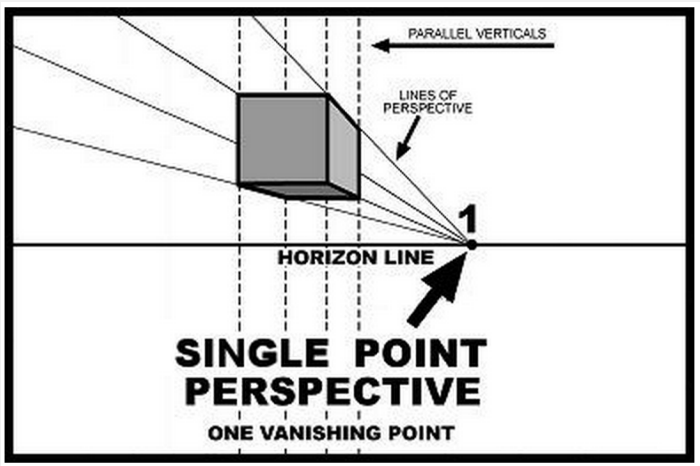
Two-Point Perspective

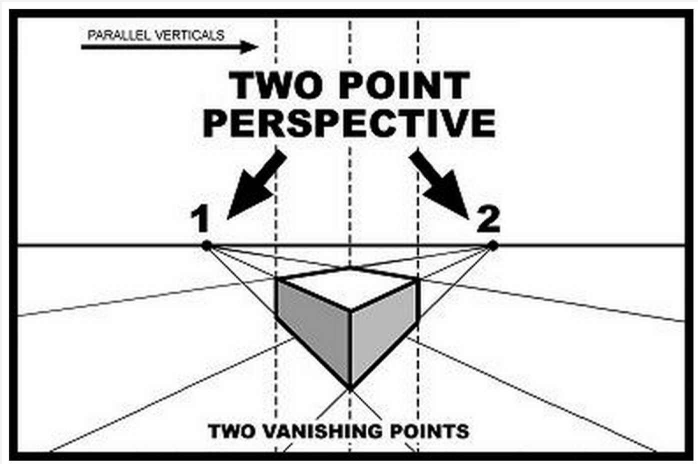
- On principal direction parallel to the projection plane
- Two vanishing points for cube

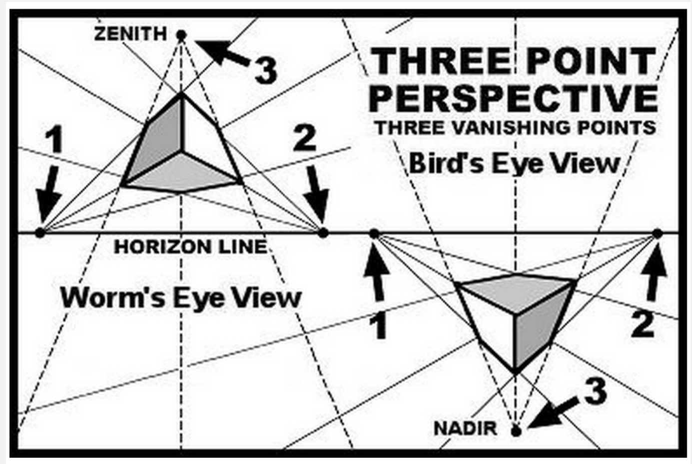
One-Point Perspective

- Two principal faces parallel to the projection plane
- One vanishing point for cube

Perspective Projection Example







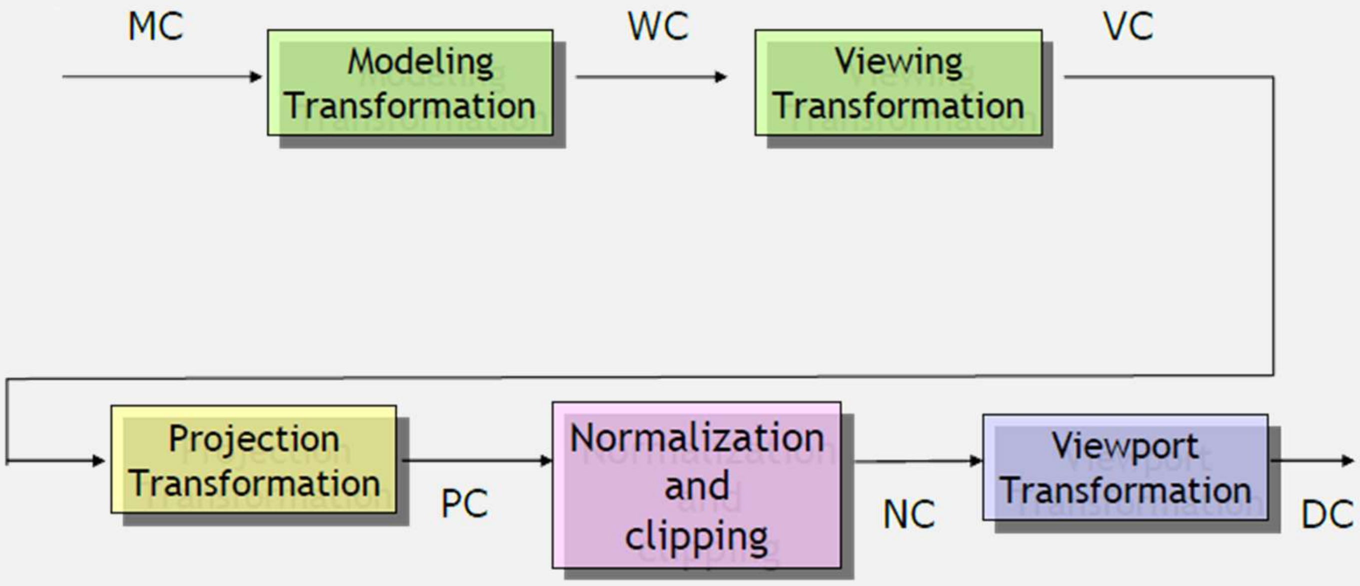
How many vanish points in this painting?

Advantages and Disadvantages

- Diminution:
  - Objects farther from viewer are projected smaller (Looks realistic)
- Nonuniform foreshortening:
  - Equal distances along a line are not projected into equal distances
- Angles preserved only in planes parallel to the projection plane
- More difficult to construct by hand than parallel projections

Viewing with A Computer

- Pipeline View



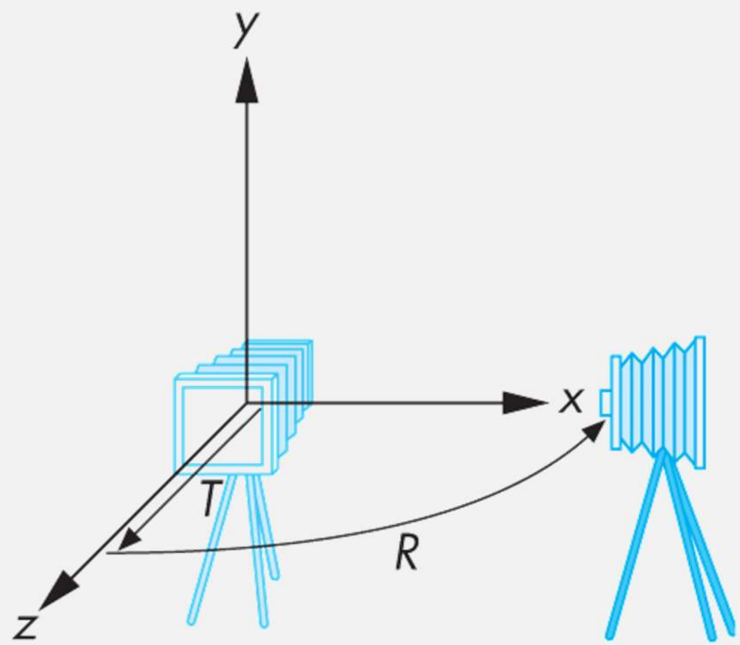
Viewing with A Computer (Cont.)

- Three aspects of the viewing process implemented in the pipeline:
  - Positioning the camera
    - Setting the model-view matrix
  - Selecting a lens
    - Setting the projection matrix: orthogonal or perspective
  - Normalization & Clipping
    - Setting the view volume

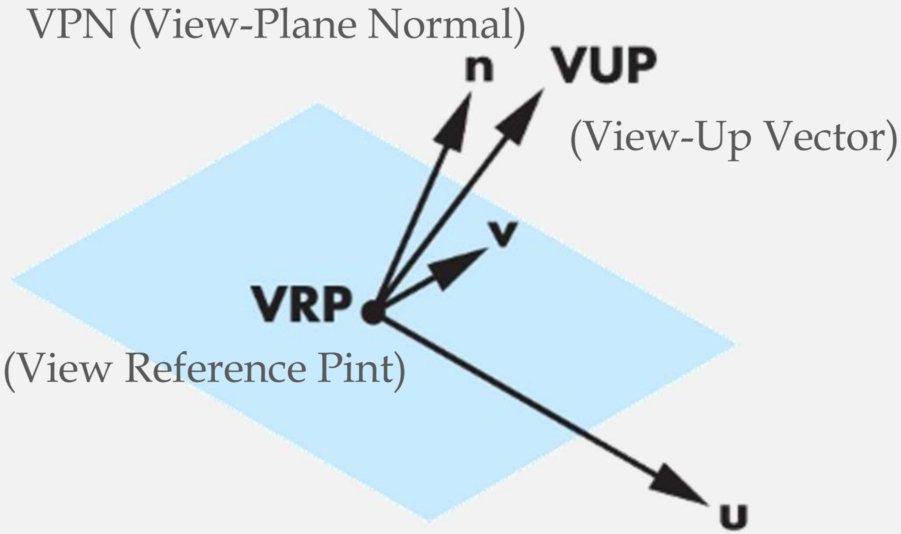
Moving the Camera

- We can move the camera to any desired position by a sequence of rotations and translations

- Example: side view
  - Rotate the camera
  - Move it away from origin
  - View matrix C = TR



How to Obtain the View Matrix ?



Given $\mathbf{VRP} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$, $\mathbf{n} = \begin{bmatrix} n_x \\ n_y \\ n_z \\ 1 \end{bmatrix}$, $\mathbf{v_{up}} = \begin{bmatrix} \mathbf{v_{up}}_x \\ \mathbf{v_{up}}_y \\ \mathbf{v_{up}}_z \\ 1 \end{bmatrix}$

$$\mathbf{v} = \alpha\mathbf{n} + \beta\mathbf{v_{up}}$$

To simplify, set $\beta = 1$ and $\alpha = -\frac{\mathbf{v_{up}}\cdot\mathbf{n}}{\mathbf{n}\cdot\mathbf{n}}$

$$\Rightarrow \mathbf{v} = \mathbf{v_{up}} - \frac{\mathbf{v_{up}} \cdot \mathbf{n}}{\mathbf{n} \cdot \mathbf{n}}\mathbf{n}$$

$$\mathbf{u} = \mathbf{v} \times \mathbf{n}$$

How to Obtain the View Matrix ? (Cont.)

Normalize **u**, **v**, and **n**, and set the rotation matrix as:

$$\mathbf{A} = \begin{bmatrix} u'_x & v'_x & n'_x & 0 \\ u'_y & v'_y & n'_y & 0 \\ u'_z & v'_z & n'_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

What we want is the opposite direction ($\mathbf{A}^{-1}$)that represent the vectors in the original system in the $\mathbf{u}'\mathbf{v}'\mathbf{n}'$ coordinate system. Hence, the rotation matrix of the model-view matrix is:

$$\mathbf{R}' = \mathbf{A}^{-1} = \mathbf{A}^T = \begin{bmatrix} u'_x & u'_y & u'_z & 0 \\ v'_x & v'_y & v'_z & 0 \\ n'_x & n'_y & n'_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finally, by multiplying the translation matrix **T**, we have:

$$\mathrm{C} = \mathrm{R'T'} = \begin{bmatrix} u'_x & u'_y & u'_z & 0 \\ v'_x & v'_y & v'_z & 0 \\ n'_x & n'_y & n'_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x \\ 0 & 1 & 0 & -y \\ 0 & 0 & 1 & -z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} u'_x & u'_y & u'_z & -xu'_x - yu'_y - zu'_z \\ v'_x & v'_y & v'_z & -xv'_x - yv'_y - zv'_z \\ n'_x & n'_y & n'_z & -xn_x - yn'_y - zn'_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

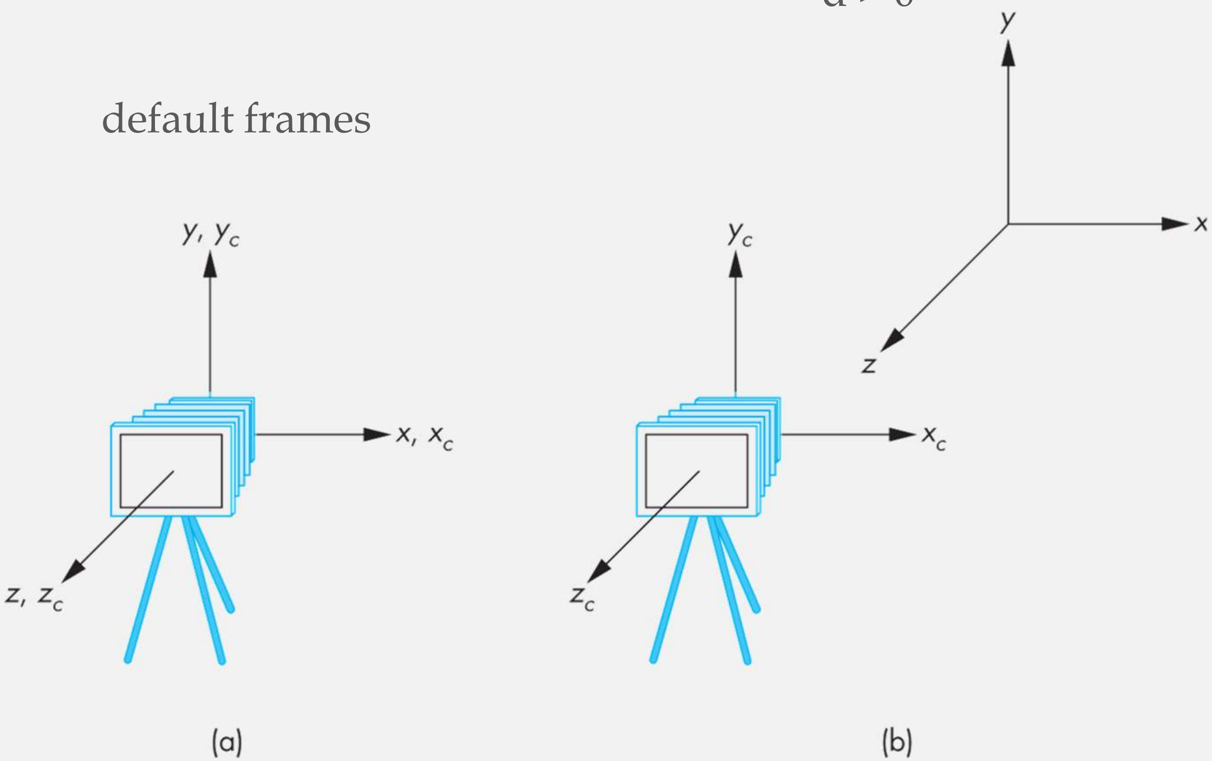
Moving the Camera Frame

- If we want to visualize object with both positive and negative z values we can either
  - Move the camera in the positive z direction
    - Translate the camera frame
  - Move the objects in the negative z direction
    - Translate the world frame
- Both of these views are equivalent and are determined by the model-view matrix
  - Want a translation (`glTranslatef(0.0,0.0,-d);`)
  - `d` > 0

Moving Camera back from Origin

frames after translation by –d
d > 0

default frames

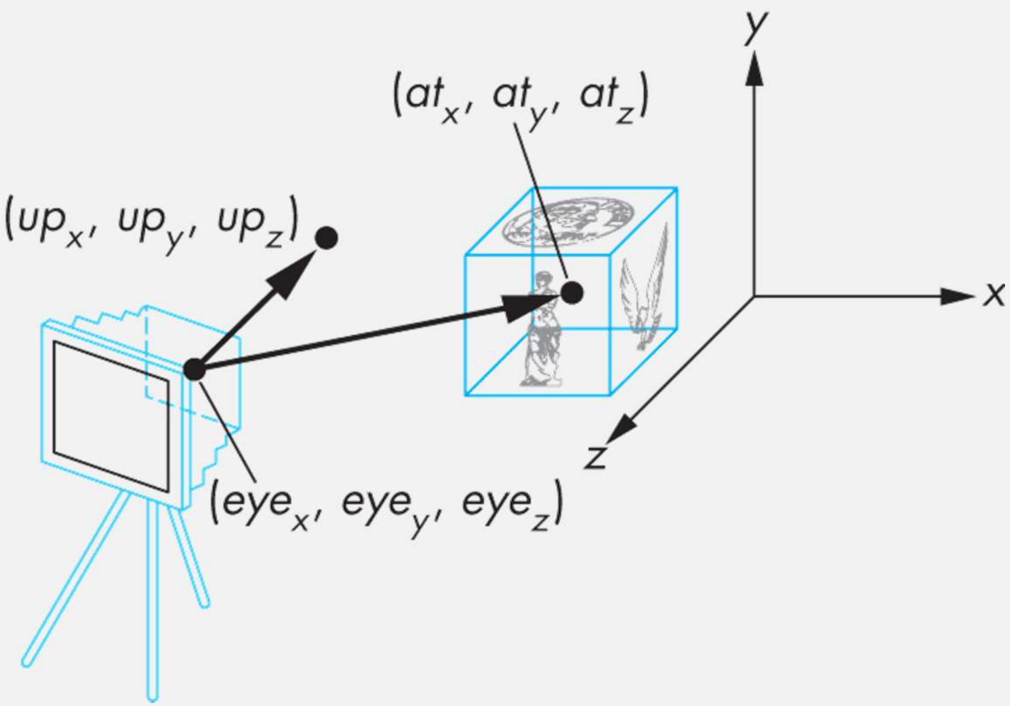


The OpenGL Camera

- In OpenGL, initially the object and camera frames are the same
  - Default model-view matrix is an identity
- The camera is located at origin and points in the negative z direction
- OpenGL also specifies a default view volume that is a cube with sides of length 2 centered at the origin
  - Default projection matrix is an identity

How to Set the Camera Position/Orientation?

- OpenGL: gluLookAt(eye_x, eye_y, eye_z, at_x, at_y, at_z, up_x, up_y, up_z)



$$\mathbf{vpn} = \boldsymbol{a} - \boldsymbol{e}$$

$$\mathbf{n} = \frac{\mathbf{vpn}}{|\mathbf{vpn}|}$$

$$\mathbf{u} = \frac{\mathbf{v_{up}} \times \mathbf{n}}{|\mathbf{v_{up}} \times \mathbf{n}|}$$

$$\mathbf{v} = \frac{\mathbf{n} \times \mathbf{u}}{|\mathbf{n} \times \mathbf{u}|}$$

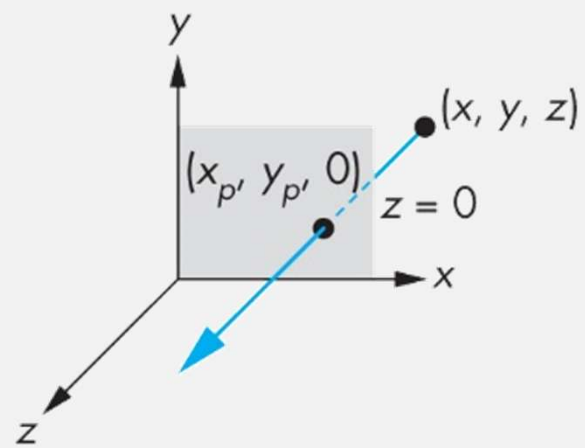
Viewing with A Computer

- Three aspects of the viewing process implemented in the pipeline:
  - Positioning the camera
    - Setting the model-view matrix
  - Selecting a lens
    - Setting the projection matrix: orthogonal or perspective
  - Normalization & Clipping
    - Setting the view volume

Projections and Normalization

- The default projection in the eye (camera) frame is orthogonal

- For points within the default view volume
  - $x_p = x$
  - $y_p = y$
  - $z_p = 0$



Homogeneous Coordinate Representation

- $x_p = x$
- $y_p = y$
- $z_p = 0$
- $w_p = 1$

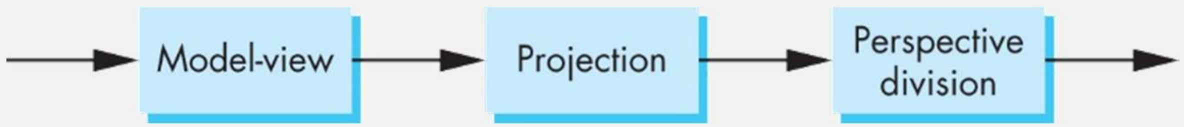
default orthographic projection

$$\mathbf{q} = \mathbf{Mp}$$

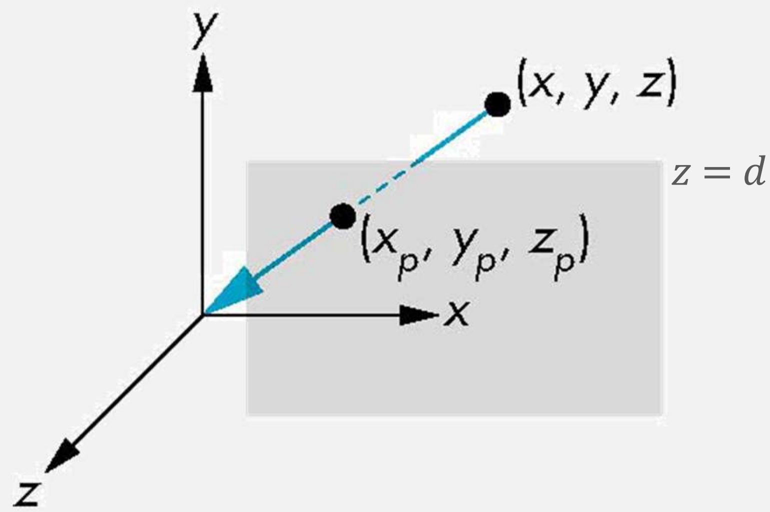
$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In practice, we can let **M= I** and set the *z* term to zero later

Simple Perspective Projections

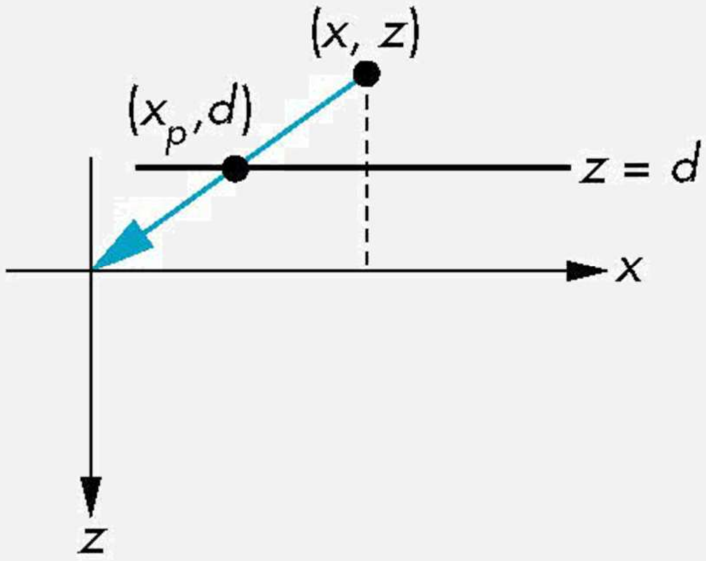


- Center of projection : at the origin
- Projection plane $z = d, d < 0$



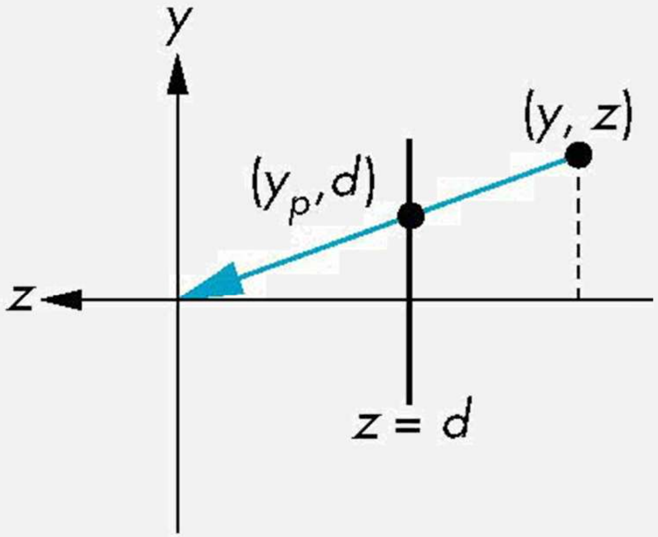
Perspective Equations

Top view



Side view

$$\frac{x}{z/d}$$



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

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

$$z_p = d = \frac{z}{z/d}$$

Homogeneous Coordinate Representation

Consider **q= Mp** where

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

$$\nLeftrightarrow$$

$$\mathrm{q} = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix} \qquad \mathrm{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \qquad \mathrm{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Perspective Division

The desired perspective equations:

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

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

$$z_p = d$$

$$q = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix}$$

- However $w \neq 1$, so we must divide by w to return from homogeneous coordinates. This perspective *division* yields

$$q' = \begin{bmatrix} \dfrac{x}{z/d} \\ \dfrac{y}{z/d} \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$$

Viewing with A Computer (Cont.)

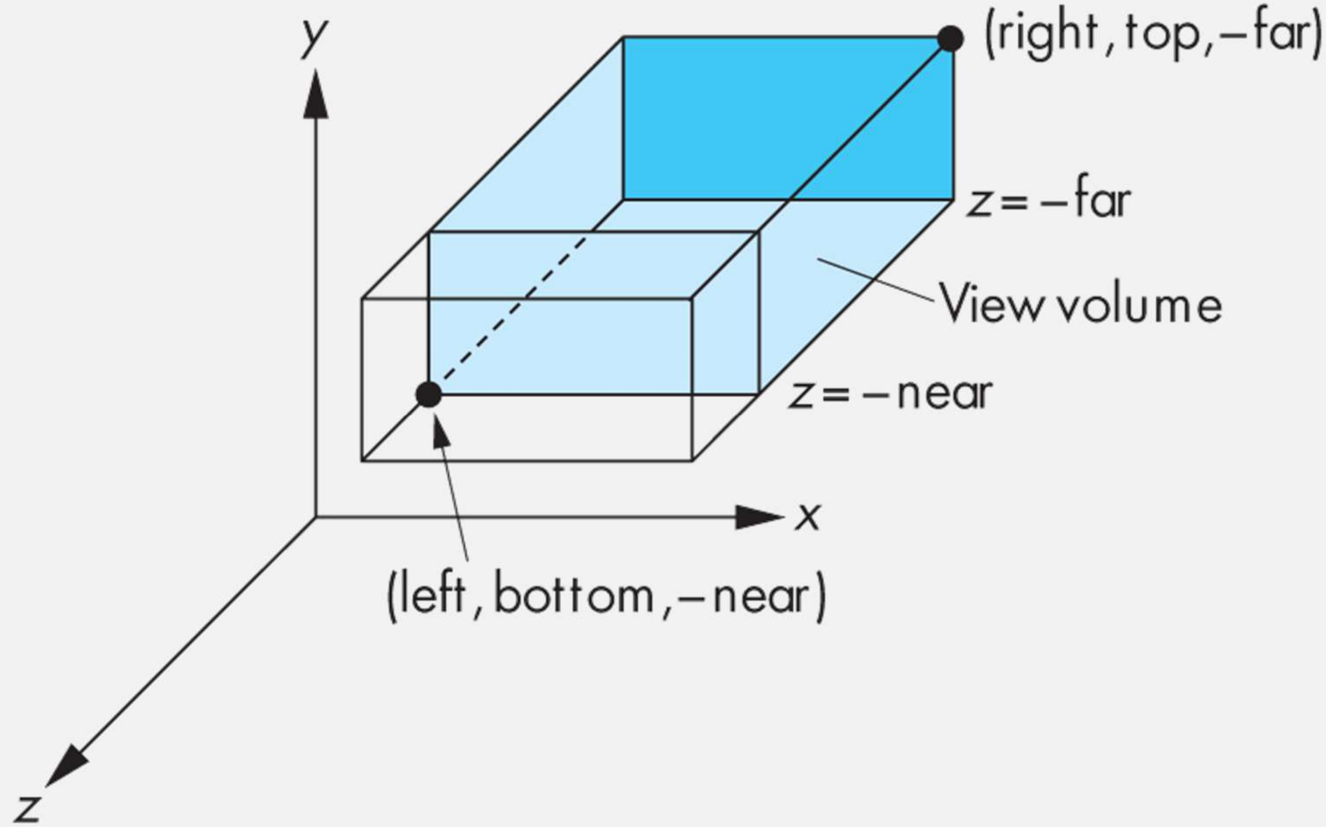
- Three aspects of the viewing process implemented in the pipeline:
  - Positioning the camera
    - Setting the model-view matrix
  - Selecting a lens
    - Setting the projection matrix: orthogonal or perspective
  - Normalization & Clipping
    - Setting the view volume

Taking Clipping into Account

- After the view transformation, a simple projection and viewport transformation can generate screen coordinate.
- However, projecting all vertices are usually unnecessary.
- Clipping with 3D volume.
- Associating projection with clipping and normalization.

Why do we use normalization ?

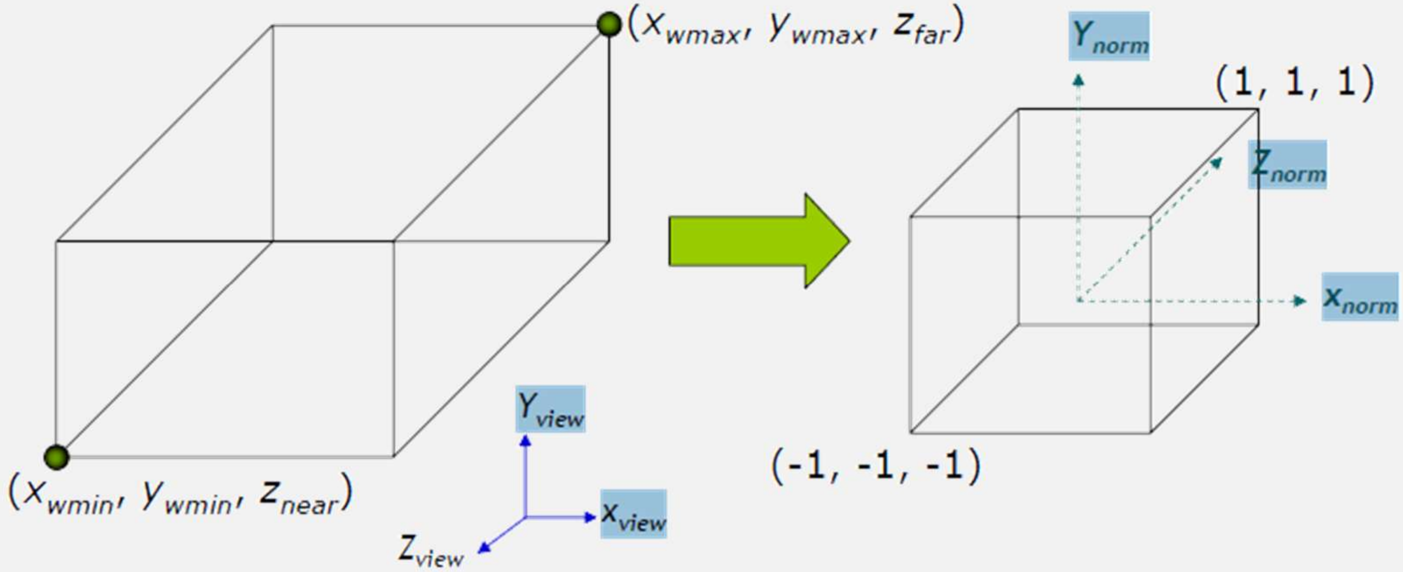
Orthogonal Viewing Volume



Orthogonal Normalization

glOrtho(left,right,bottom,top,near,far)

normalization ⇒ find transformation to convert specified clipping volume to default



Orthogonal Normalization Matrix

- Two steps
  - T: Move center to origin
  - S: Scale to have sides of length 2

$$\mathbf{T} = \mathbf{T}\left(-\frac{(right + left)}{2}, -\frac{(top + bottom)}{2}, -\frac{(far + near)}{2}\right)$$

$$\mathbf{S} = \mathbf{S}\left(\frac{2}{(right - left)}, \frac{2}{(top - bottom)}, \frac{2}{(near - far)}\right)$$

$$\mathbf{P} = \mathbf{ST} = \begin{bmatrix} \frac{2}{xw_{\max} - xw_{\min}} & 0 & 0 & -\frac{xw_{\max} + xw_{\min}}{xw_{\max} - xw_{\min}} \\ 0 & \frac{2}{yw_{\max} - yw_{\min}} & 0 & -\frac{yw_{\max} + yw_{\min}}{yw_{\max} - yw_{\min}} \\ 0 & 0 & \frac{2}{z_{near} - z_{far}} & \frac{z_{near} + z_{far}}{z_{near} - z_{far}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Final Orthogonal Projection

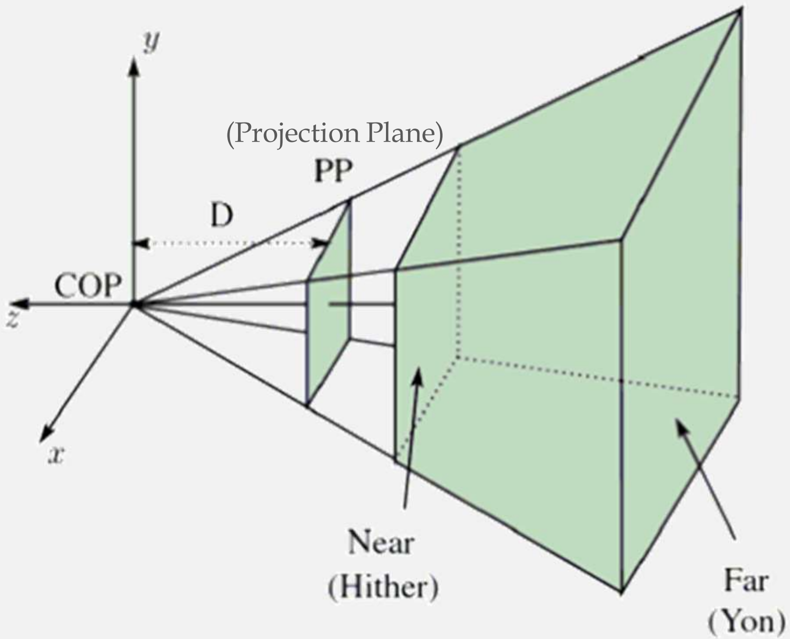
- Set z = 0
- Equivalent to the homogeneous coordinate transformation

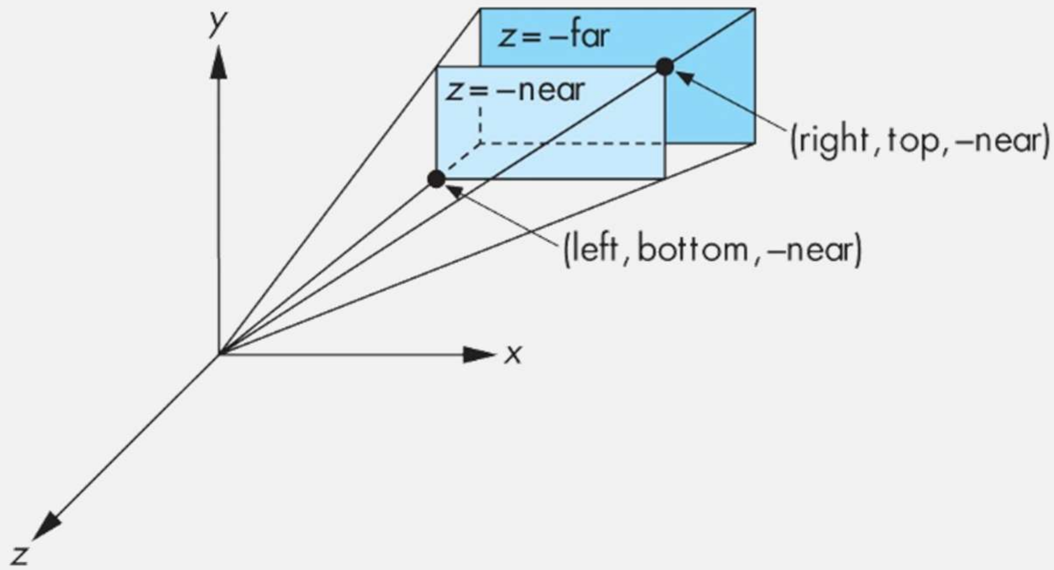
$$\mathbf{M}_{\mathrm{orth}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Hence, general orthogonal projection in 4D is

$$\mathbf{P} = \mathbf{M}_{\mathrm{orth}}\mathbf{ST}$$

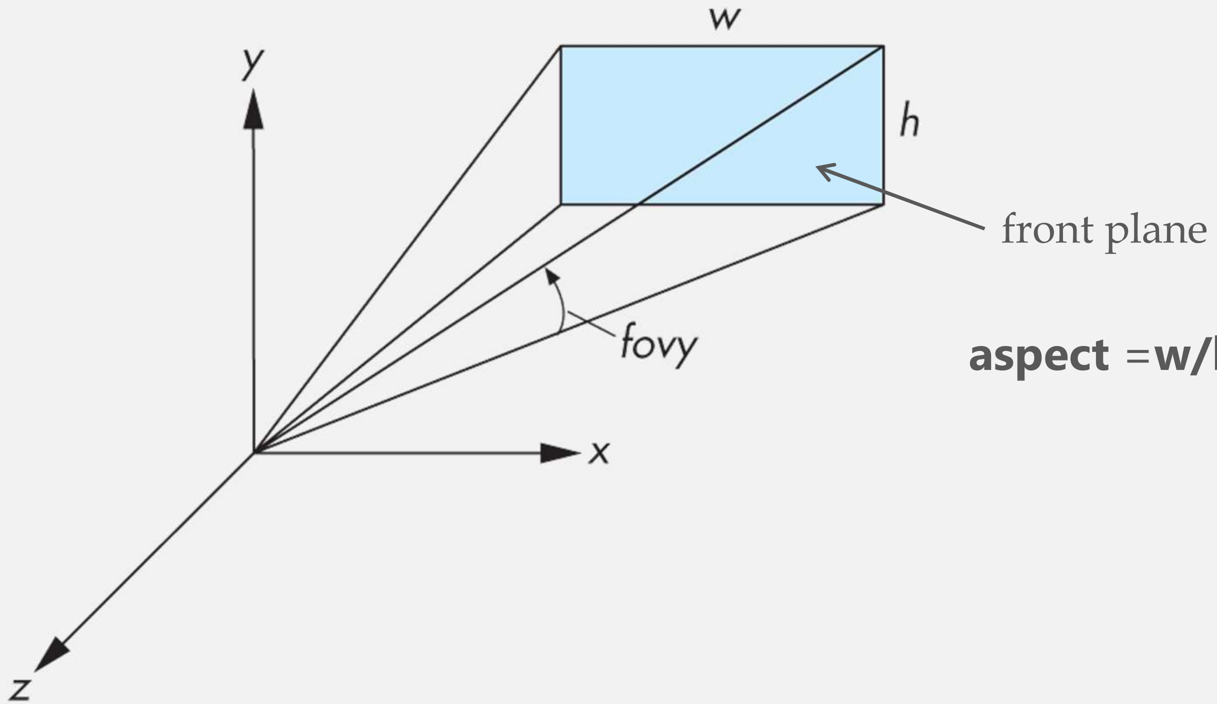
Perspective Viewing Volume





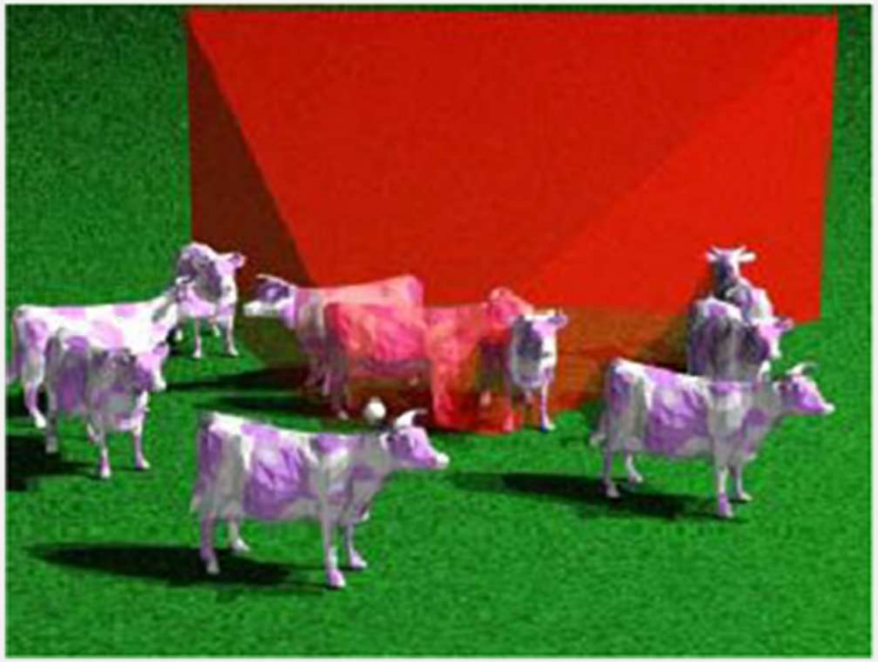
Using Field/Angle of View

- In addition to directly assigning the viewing frustum, assigning field of view may be more user-friendly.

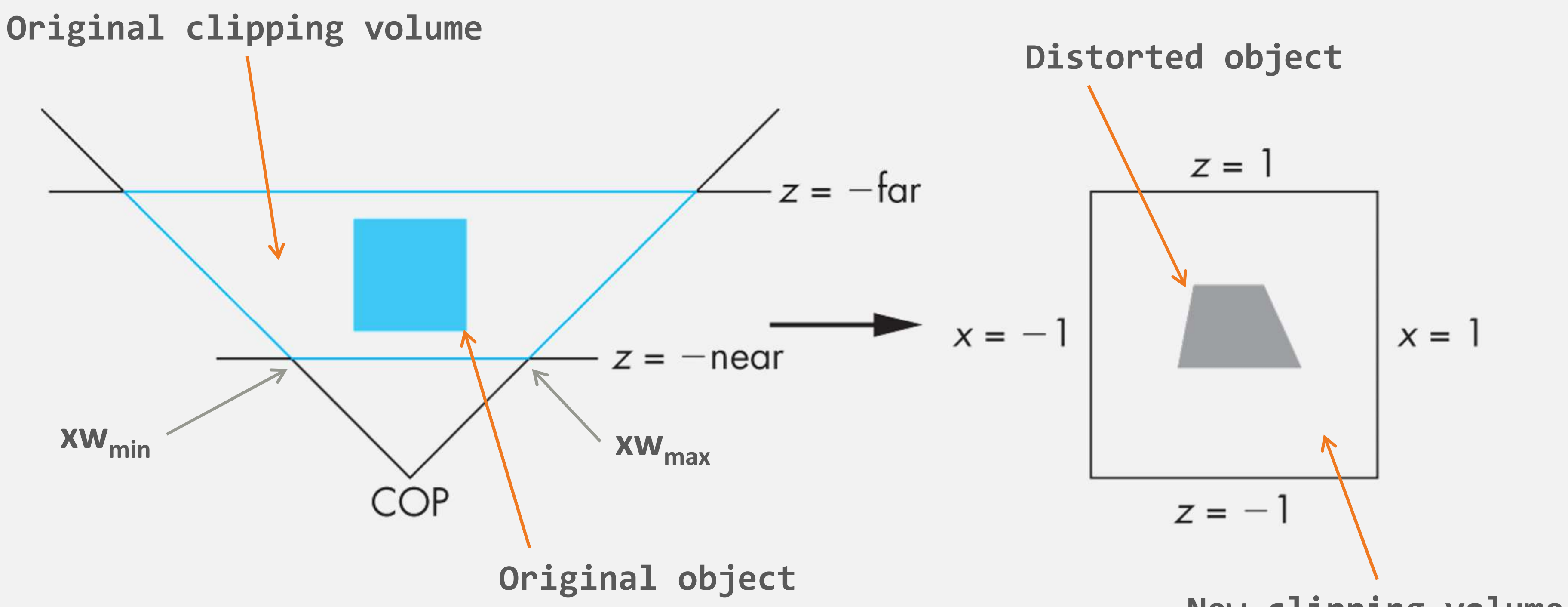


aspect =**w/h**

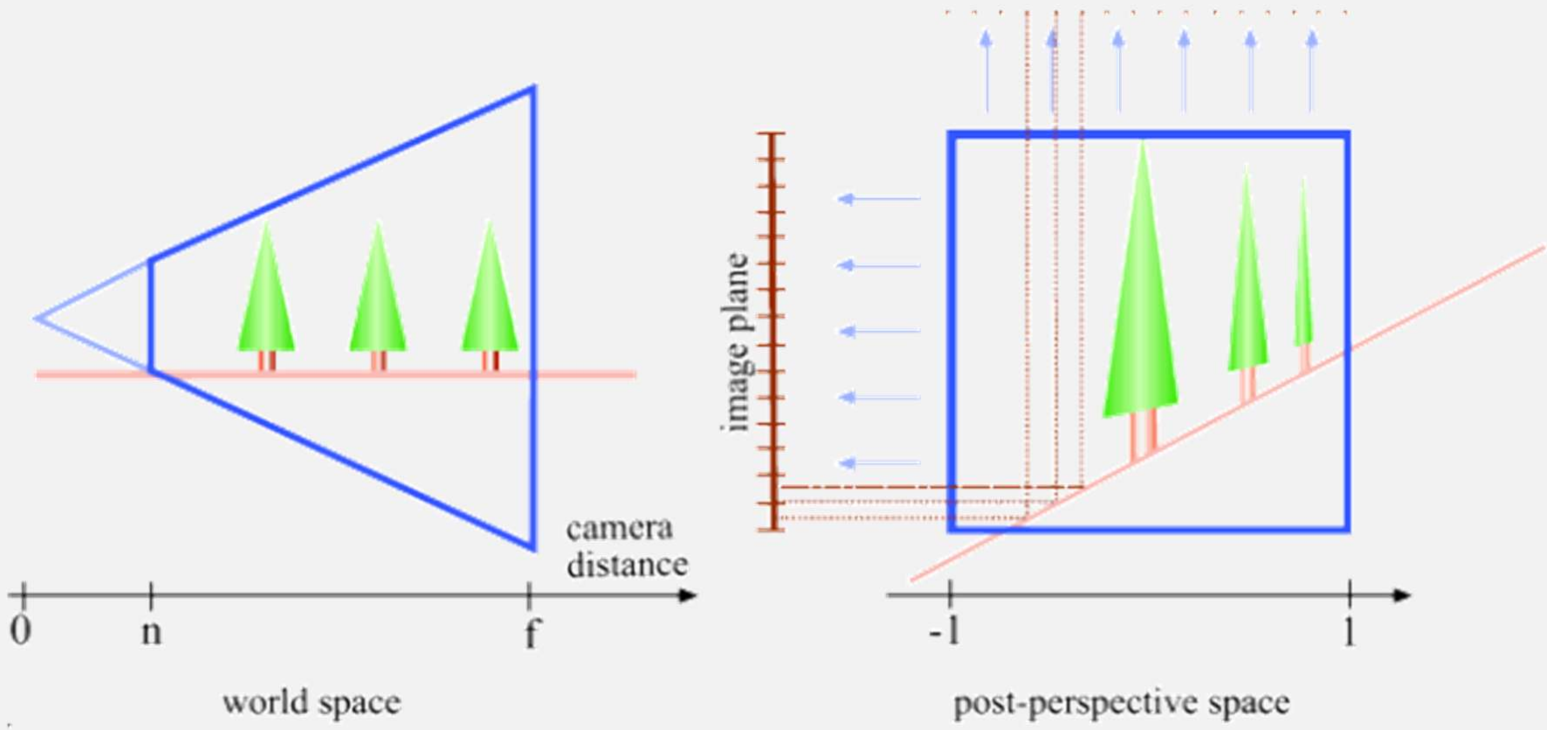
Clipping for Perspective Views



Perspective Normalization

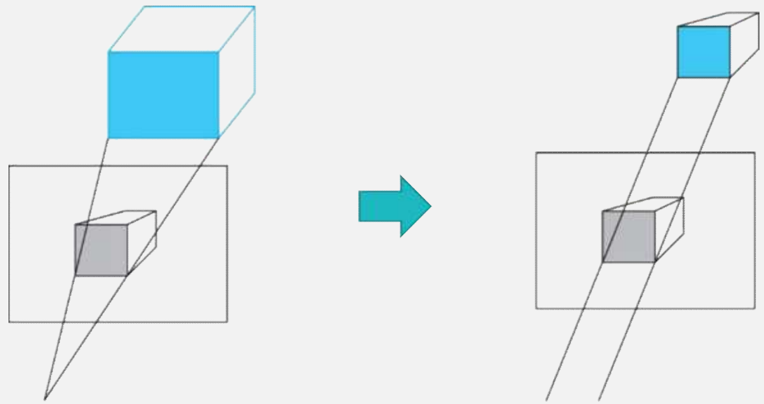


Perspective Normalization (Cont.)



Normalization

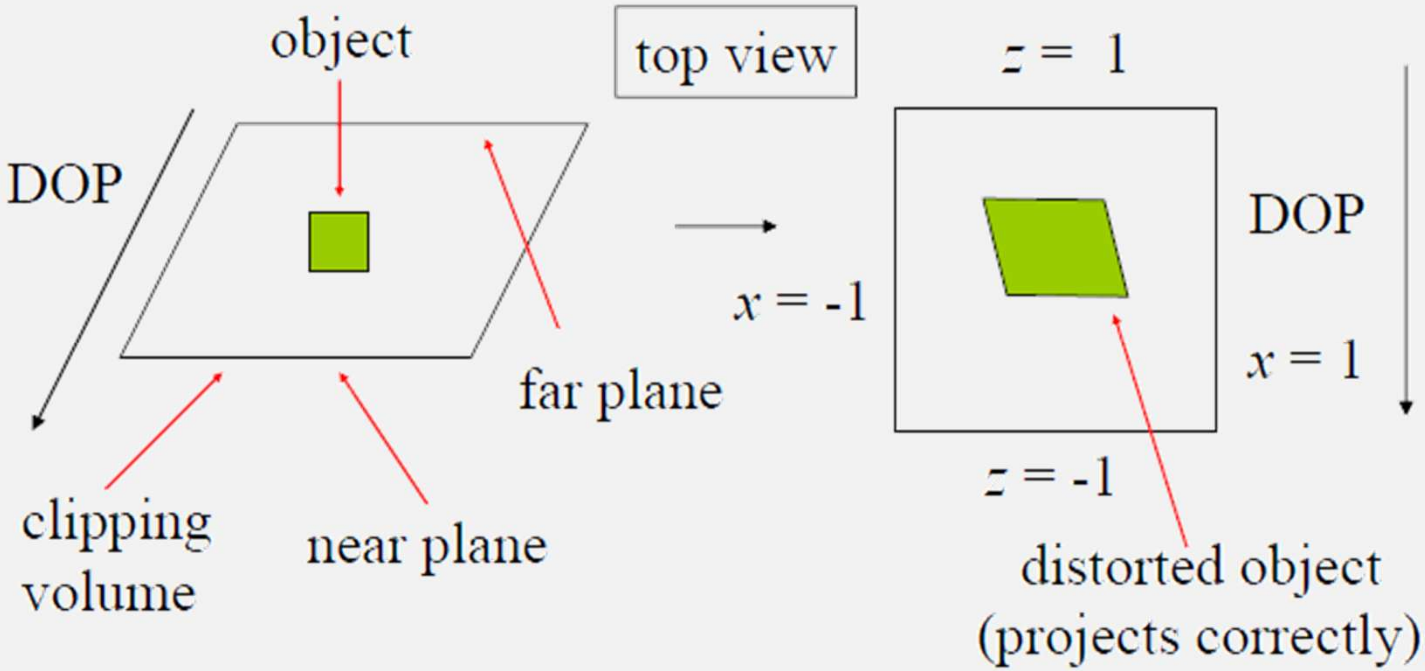
- Rather than derive a different projection matrix for each type of projection, we can **convert all projections to orthogonal projections** with the default view volume



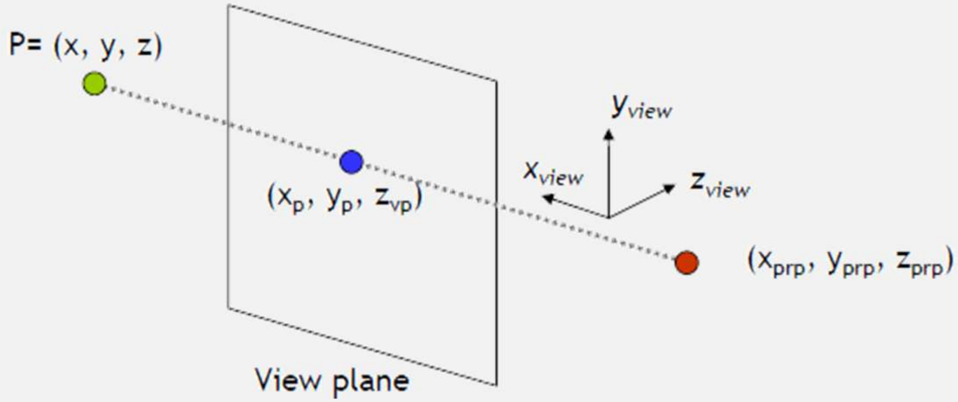
- This strategy allows us to use **standard transformations** in the pipeline and makes for **efficient clipping**

Effect on Clipping

- The projection matrix **P= STH** transforms the original clipping volume to the default clipping volume



Perspective-Projection Transformation



$$x_p = (1-u)x + ux_{prp}$$

$$y_p = (1-u)y + uy_{prp}$$

$u = 0 \sim 1$

$$\boxed{\begin{aligned} x_p &= x\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right) + x_{prp}\left(\frac{z_{vp} - z}{z_{prp} - z}\right) \\ y_p &= y\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right) + y_{prp}\left(\frac{z_{vp} - z}{z_{prp} - z}\right) \end{aligned}}$$

Given $x_{prp} = y_{prp} = z_{prp} = 0$, $z_{vp} = z_{near}$

$$\boxed{\begin{aligned} x_p &= x\left(\frac{-z_{near}}{-z}\right) \\ y_p &= y\left(\frac{-z_{near}}{-z}\right) \end{aligned}}$$

Perspective-Projection Transformation (Cont.)

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$$

$$M_{pers} = \begin{bmatrix} -z_{near} & 0 & 0 & 0 \\ 0 & -z_{near} & 0 & 0 \\ 0 & 0 & s_z & t_z \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

After perspective division, the point (*x*,*y*,*z*,1) goes to

$$x_p = x\left(\frac{-Z_{near}}{-Z}\right)$$

$$y_p = y\left(\frac{-Z_{near}}{-Z}\right)$$

$$z_p = \frac{s_z z + t_z}{-z} = -\left(s_z + \frac{t_z}{z}\right)$$

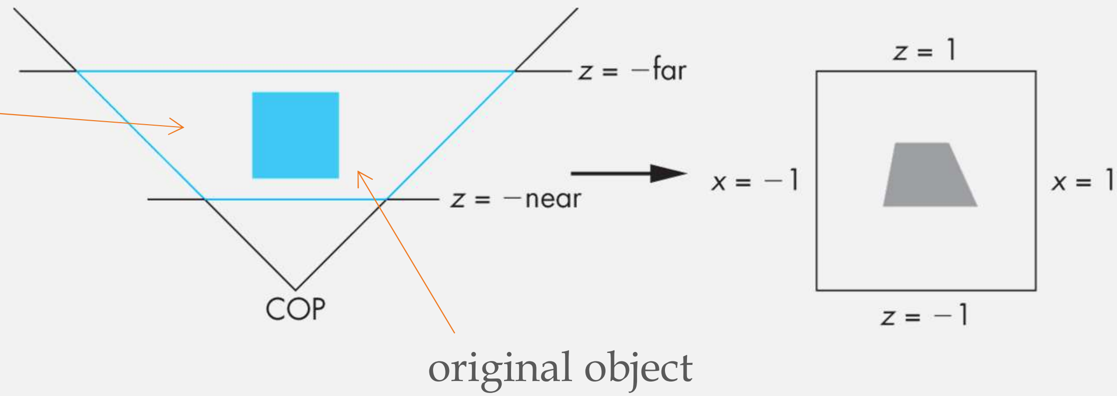
To make $-1 \le z_p \le 1$

$$S_z = \frac{Z_{near} + Z_{far}}{Z_{near} - Z_{far}}$$

$$t_z = \frac{-2Z_{near}Z_{far}}{Z_{near} - Z_{far}}$$

Further Normalization

$$M_{pers} = \begin{bmatrix} -z_{near} & 0 & 0 & 0 \\ 0 & -z_{near} & 0 & 0 \\ 0 & 0 & \dfrac{z_{near}+z_{far}}{z_{near}-z_{far}} & \dfrac{-2z_{near}z_{far}}{z_{near}-z_{far}} \\ 0 & 0 & -a & 0 \end{bmatrix}$$



$$\boxed{M_{normpers} = \begin{bmatrix} -z_{near}\dfrac{2}{xw_{\max}-xw_{\min}} & 0 & 0 & 0 \\ 0 & -z_{near}\dfrac{2}{yw_{\max}-yw_{\min}} & 0 & 0 \\ 0 & 0 & \dfrac{z_{near}+z_{far}}{z_{near}-z_{far}} & \dfrac{-2z_{near}z_{far}}{z_{near}-z_{far}} \\ 0 & 0 & -1 & 0 \end{bmatrix}}$$

Notes

- Normalization let us clip against a simple cube regardless of type of projection

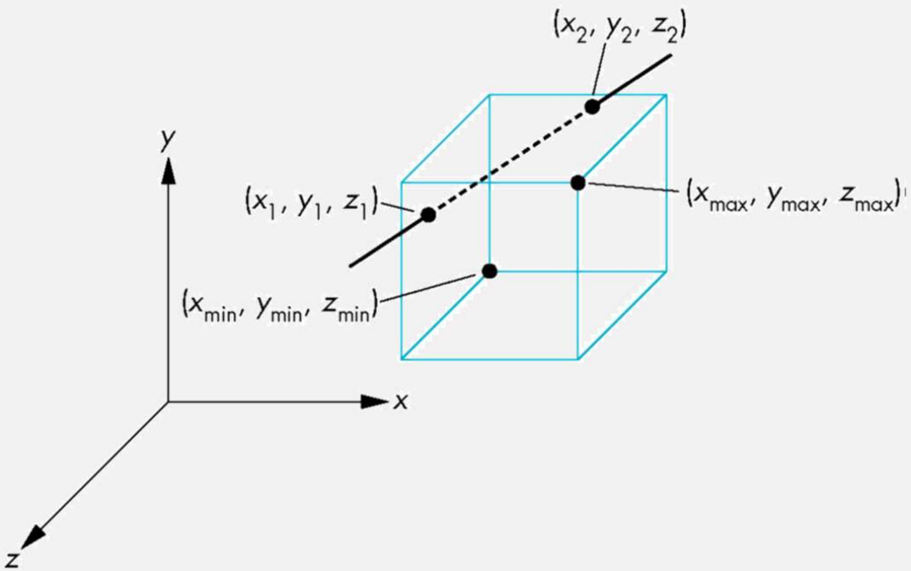
- Delay final “projection” until end
  - Important for ***hidden-surface removal*** to retain depth information as long as possible

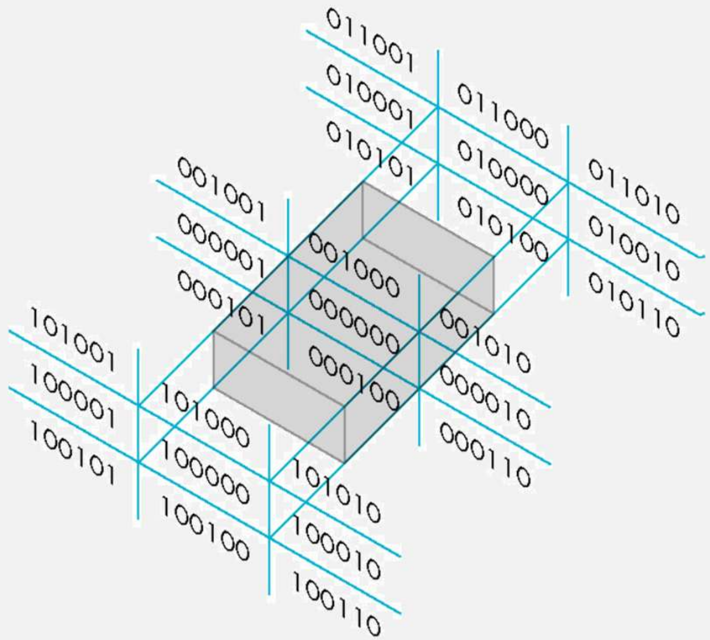
Why do we do it this way?

- Normalization allows for *a single pipeline* for both perspective and orthogonal viewing
- We stay in four dimensional homogeneous coordinates as long as possible to retain three-dimensional information needed for hidden-surface removal and shading
- ***Clipping** is now "easier".*

Cohen-Sutherland Method in 3D

- Use 6-bit outcodes
  - When needed, clip line segment against planes





Cohen-Sutherland Method in 3D (Cont.)

Check for outcodes:

$$-1 \le x_p \le 1,\ -1 \le y_p \le 1,\ -1 \le z_p \le 1$$

Since

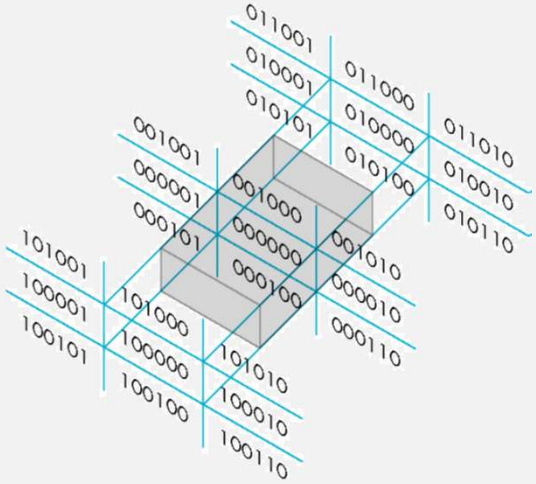
$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \xRightarrow{\text{SRT}\ldots} \ldots\ldots \Rightarrow \begin{bmatrix} x_h \\ y_h \\ z_h \\ h \end{bmatrix} \xRightarrow{\text{Divide } h} \ldots\ldots \Rightarrow \begin{bmatrix} x_h/h \\ y_h/h \\ z_h/h \\ 1 \end{bmatrix} = \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$$

To avoid unnecessary float division, We can check

$$-h \le x_h \le h,\ -h \le y_h \le h,\ -h \le z_h \le h$$

Cohen-Sutherland Method in 3D (Cont.)

- If outcode(A)==outcode(B)==0
  - Accept the whole line segment.
- If(outcode(A) and outcode(B))!=0
  - Reject the line segment.



- Other cases
  - Calculate an intersection (according to outcode bits)
  - Then check outcode again
- Note: use parametric forms

$$x_h = x_{ha} + (x_{hb} - x_{ha})u$$

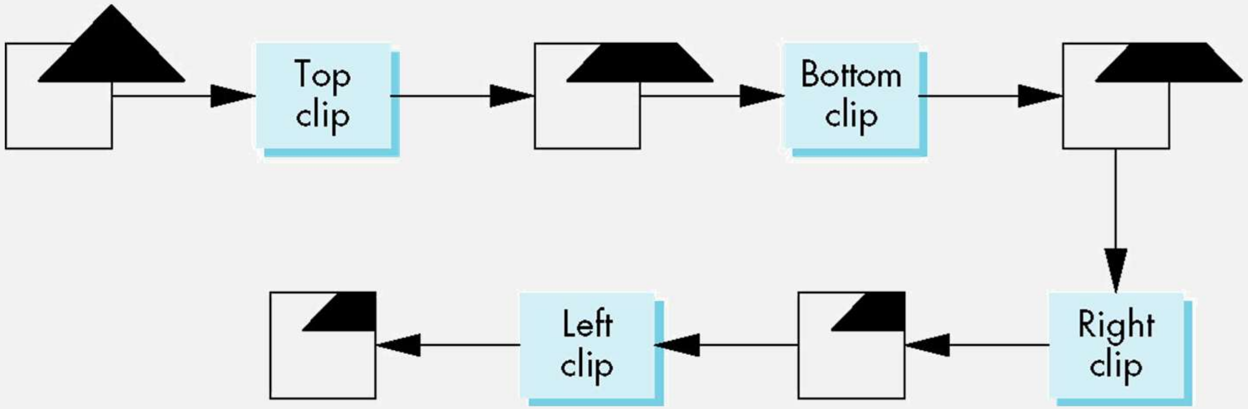
$$y_h = y_{ha} + (y_{hb} - y_{ha})u$$

$$z_h = z_{ha} + (z_{hb} - z_{ha})u$$

$$h = h_a + (h_b - h_a)u$$

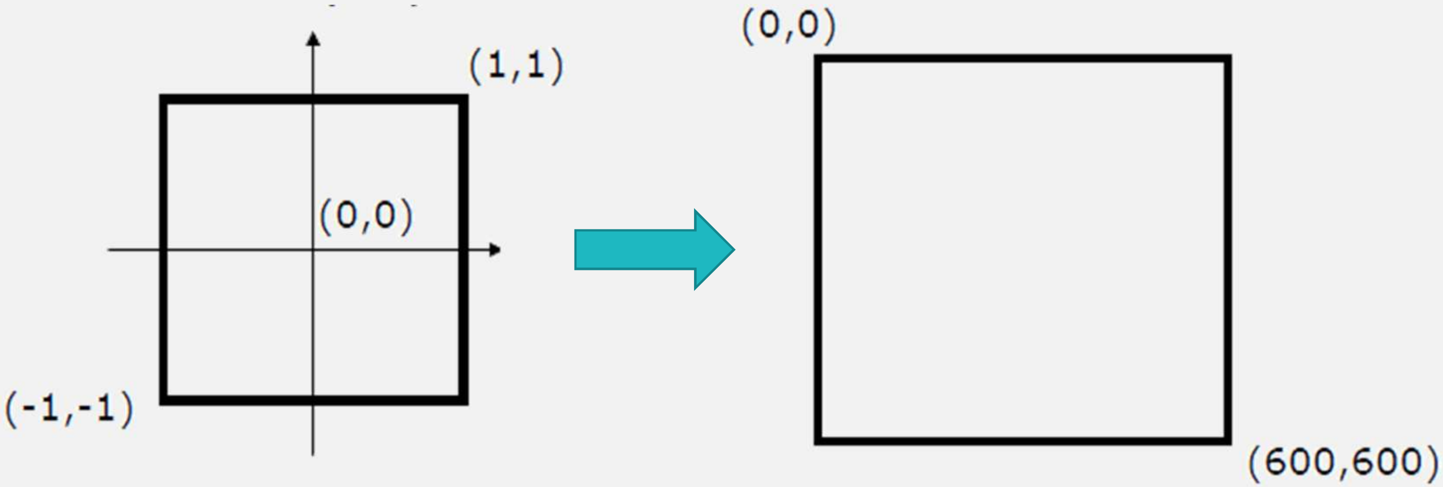
Polygon Clipping in 3D

- Similar to 2D clipping
  - Bounding box
  - Clipping with each clipping plane
  - Etc…..



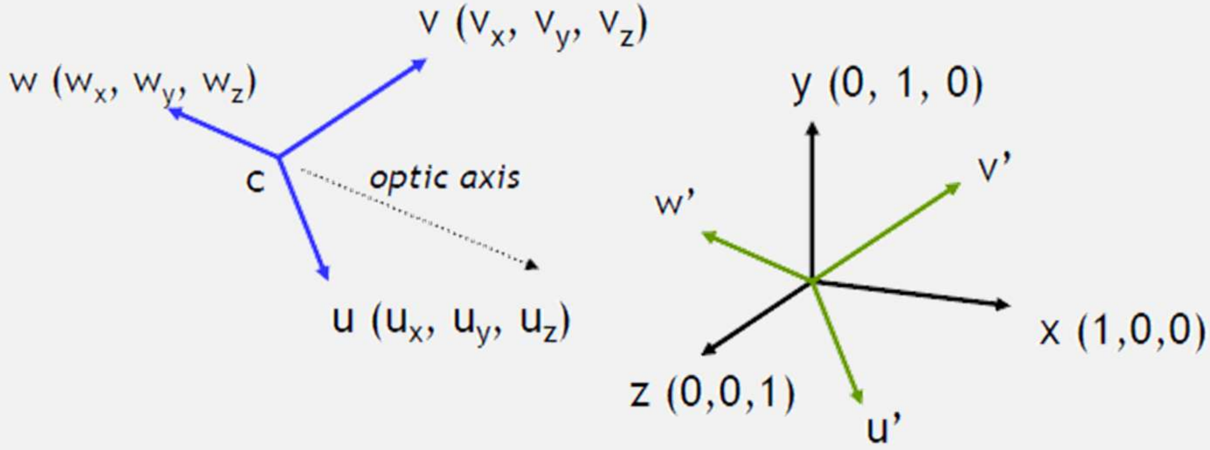
Viewport Transformation

- From the working coordinate to the coordinate of display device.



By 2D scaling and translation

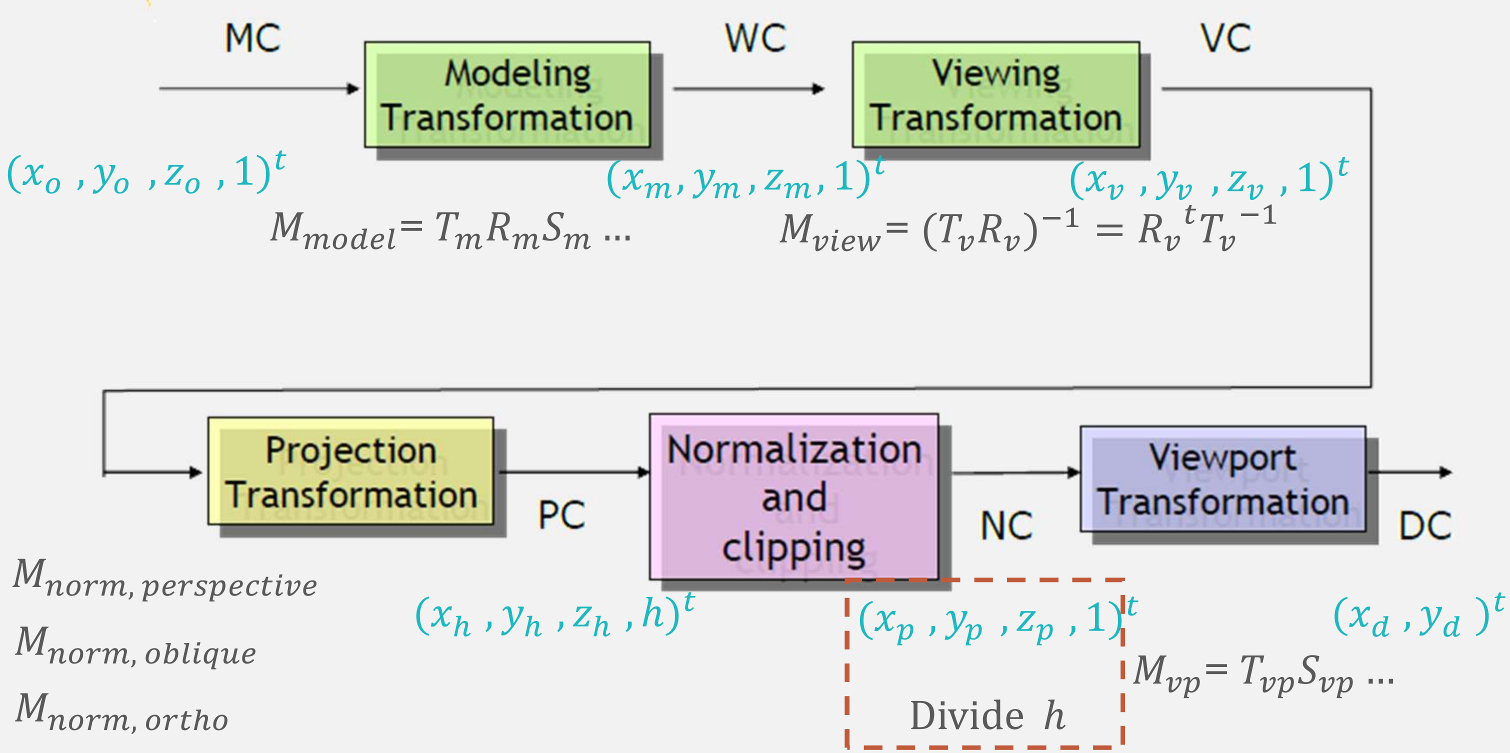
By Coordinate Transformations



$$\begin{bmatrix} x_{wc} \\ y_{wc} \\ z_{wc} \\ 1 \end{bmatrix} = \begin{bmatrix} u'_x & v'_x & w'_x & 0 \\ u'_y & v'_y & w'_y & 0 \\ u'_z & v'_z & w'_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x'_{vc} \\ y'_{vc} \\ z'_{vc} \\ 1 \end{bmatrix} \qquad \begin{bmatrix} x'_{vc} \\ y'_{vc} \\ z'_{vc} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & -c_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{vc} \\ y_{vc} \\ z_{vc} \\ 1 \end{bmatrix}$$

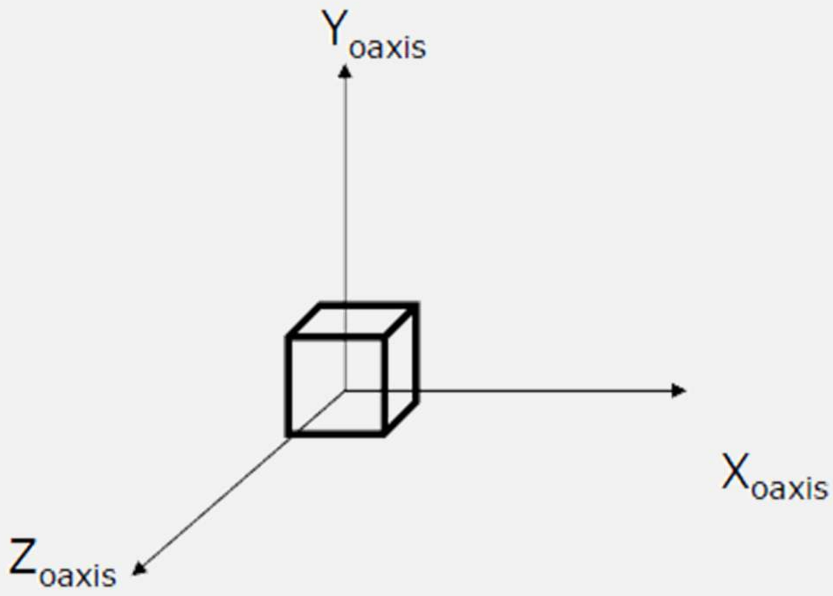
Example

Pipeline View



Loading an Object

- $(x_o\ , y_o\ , z_o\ , 1)^t$

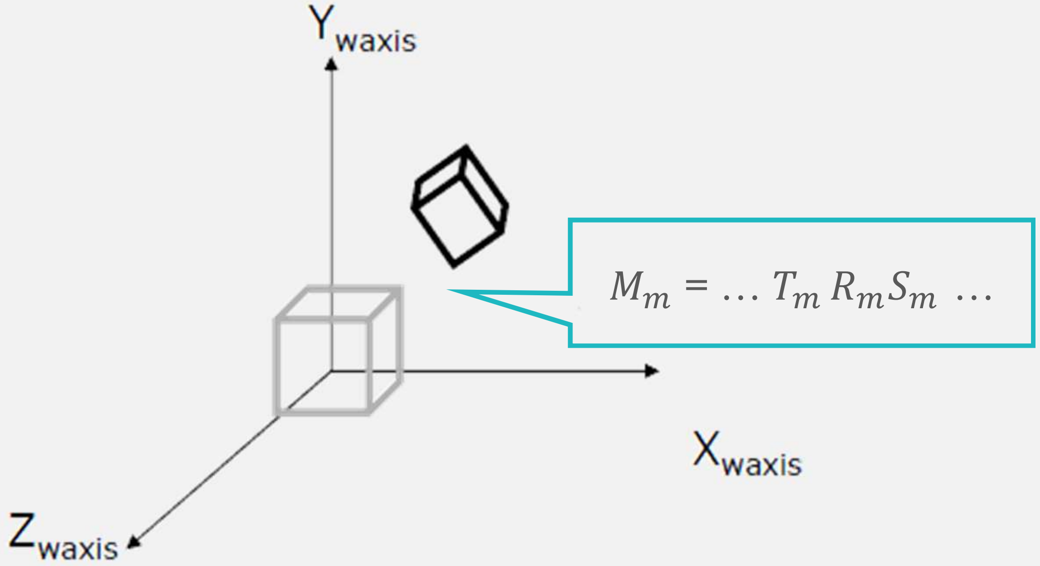


object coordinates

Modeling Transformation

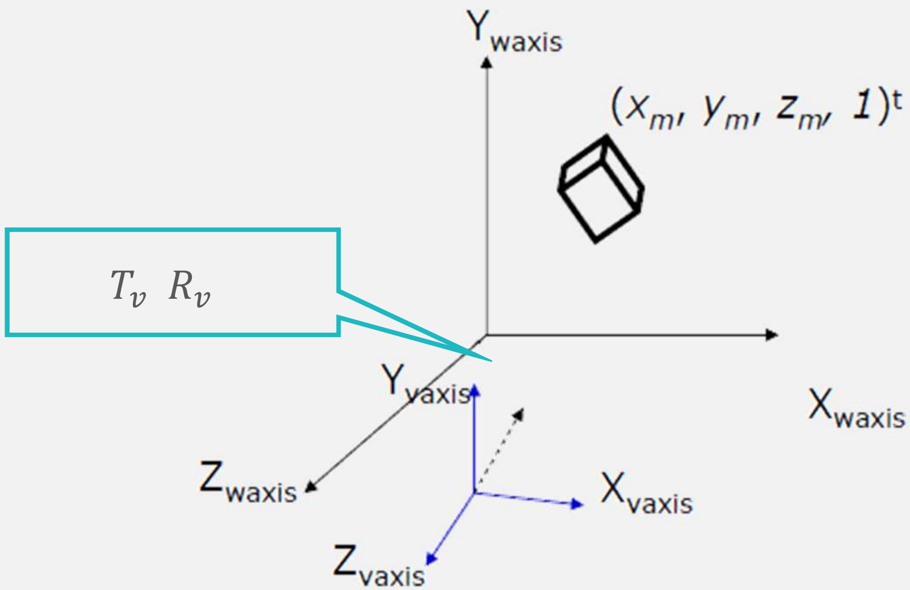
- $(x_m, y_m, z_m, 1)^t = M_m(x_o\ , y_o\ , z_o\ , 1)^t$

  where $M_m = \ldots\ T_m\ R_m S_m\ \ldots$



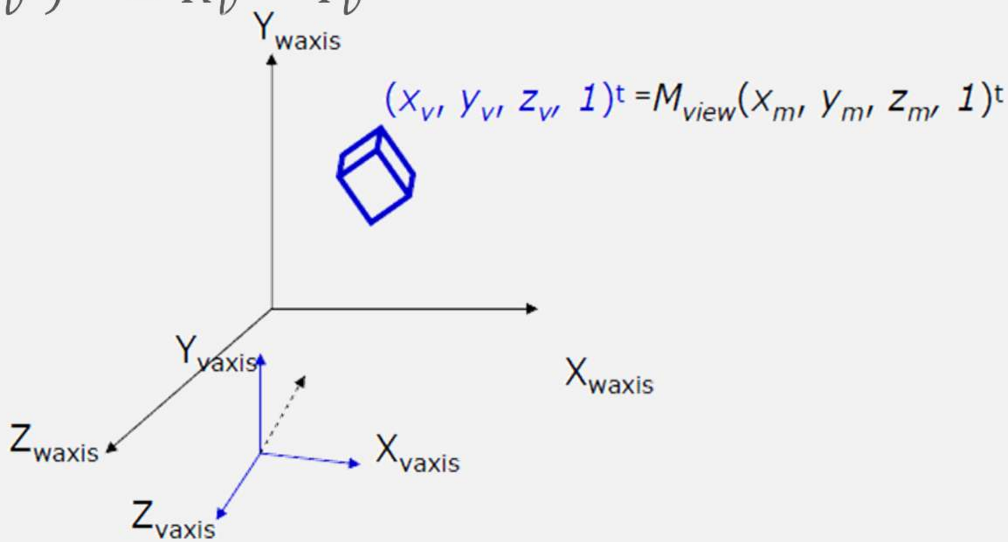
Put a Virtual Camera

- Move a camera from the origin (by $T_v R_v$)

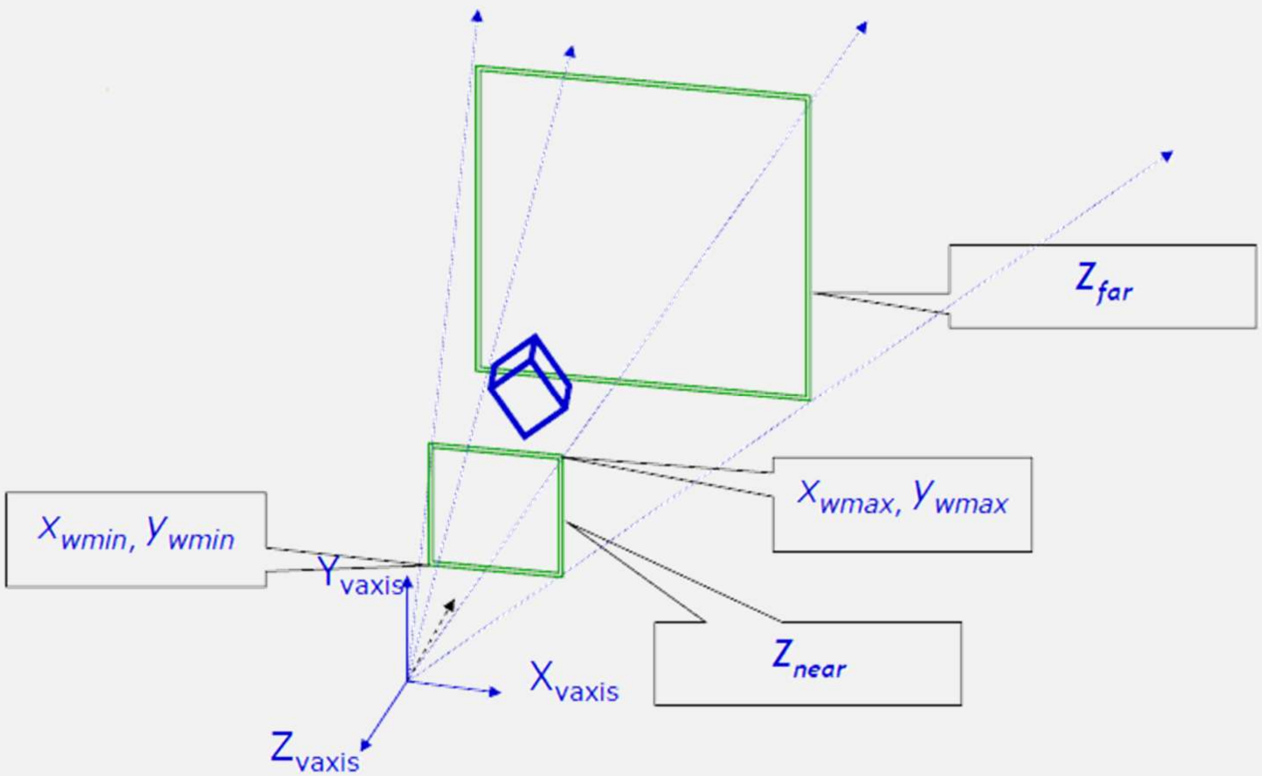


Virtual Camera's Coordinate

- Change the object's coordinate
- $(x_v\ , y_v\ , z_v\ , 1)^t = M_{view}(x_m, y_m, z_m, 1)^t$
- $M_{view} = (T_v\ , R_v\ )^{-1} = R_v^{\ -1} T_v^{\ -1}$



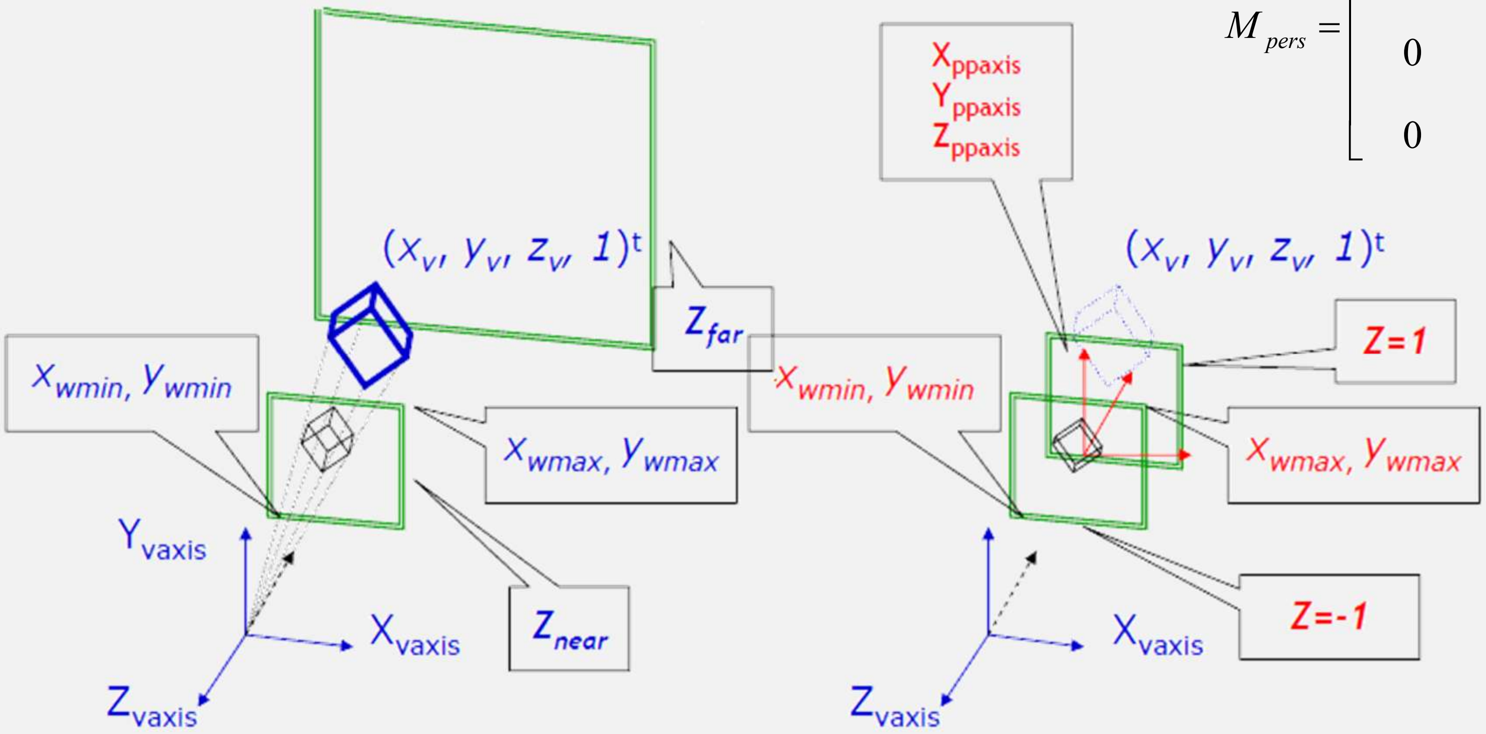
Virtual Camera's Coordinate (Cont.)



Perspective Projection

This matrix is usually combined with the normalization matrix.

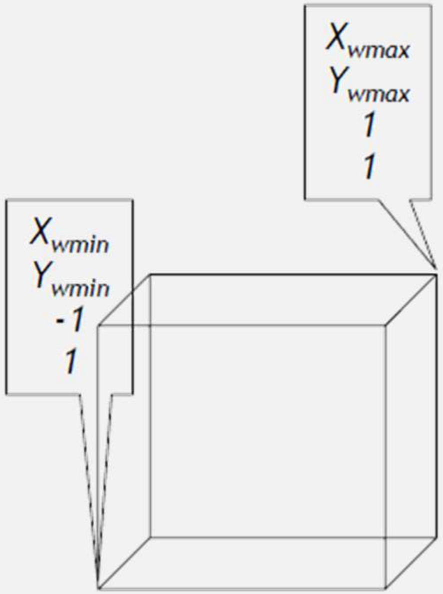
$$M_{pers} = \begin{bmatrix} -z_{near} & 0 & 0 & 0 \\ 0 & -z_{near} & 0 & 0 \\ 0 & 0 & \dfrac{z_{near}+z_{far}}{z_{near}-z_{far}} & \dfrac{-2z_{near}z_{far}}{z_{near}-z_{far}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

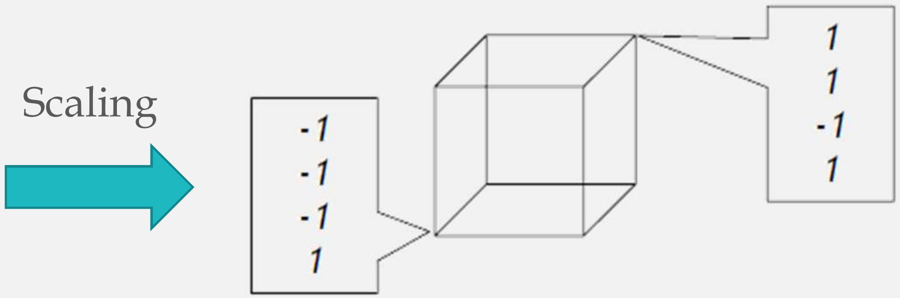


Projection + Normalization

$$M_{normpers} = \begin{bmatrix} -z_{near}\dfrac{2}{xw_{\max} - xw_{\min}} & 0 & 0 & 0 \\ 0 & -z_{near}\dfrac{2}{yw_{\max} - yw_{\min}} & 0 & 0 \\ 0 & 0 & \dfrac{z_{near} + z_{far}}{z_{near} - z_{far}} & \dfrac{-2z_{near}z_{far}}{z_{near} - z_{far}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} \dfrac{2}{xw_{\max} - xw_{\min}} & 0 & 0 & 0 \\ 0 & \dfrac{2}{yw_{\max} - yw_{\min}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} M_{pers}$$

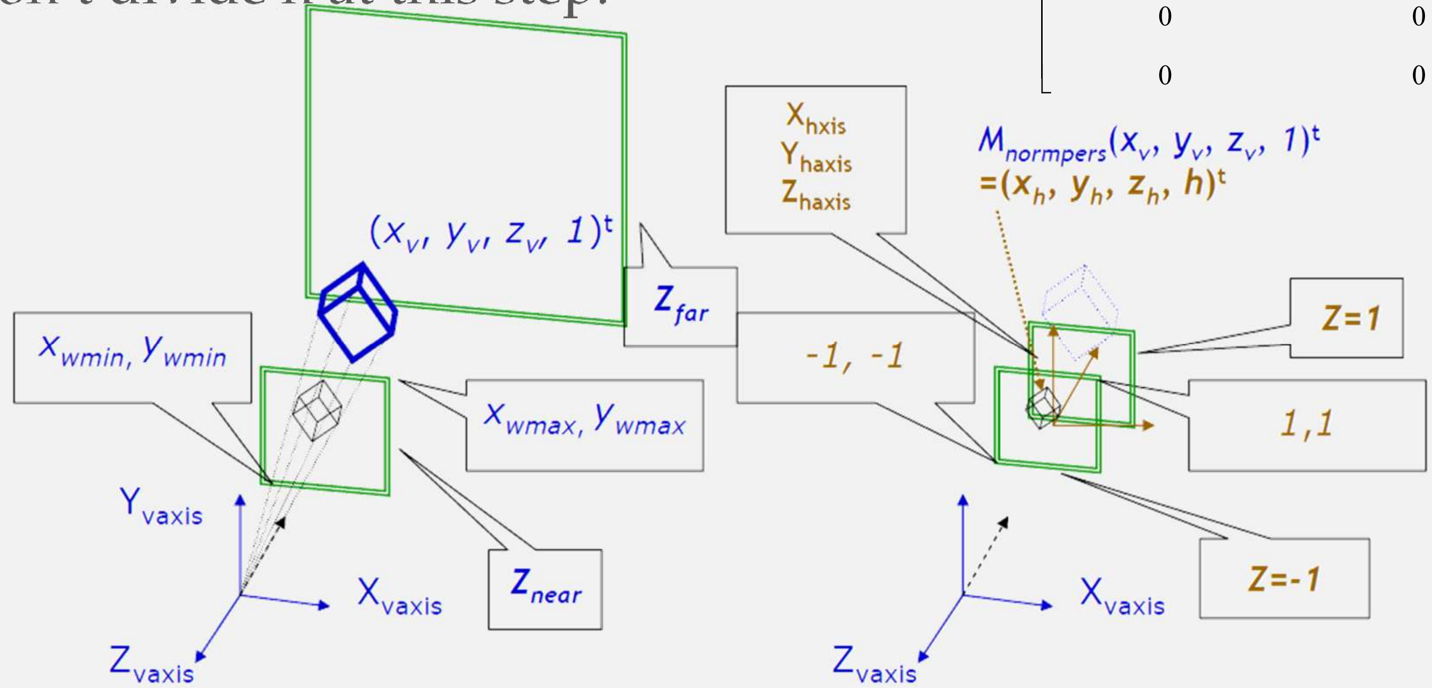




Projection + Normalization (Cont.)

- $(x_h\ , y_h\ , z_h\ , h)^t = M_{normpers}(x_v\ , y_v\ , z_v\ , 1)^t$
- Don’t divide h at this step.

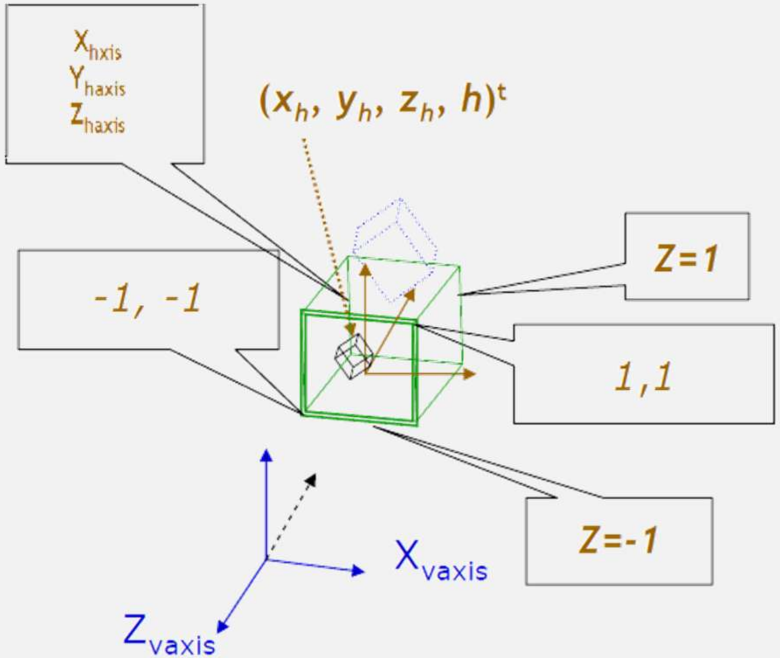
$$M_{normpers} = \begin{bmatrix} -z_{near}\dfrac{2}{xw_{\max} - xw_{\min}} & 0 & 0 & 0 \\ 0 & -z_{near}\dfrac{2}{yw_{\max} - yw_{\min}} & 0 & 0 \\ 0 & 0 & \dfrac{z_{near} + z_{far}}{z_{near} - z_{far}} & \dfrac{-2z_{near}z_{far}}{z_{near} - z_{far}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$



Clipping

- Perform clipping with $(x_h, y_h, z_h, h)^t$
- Avoid unnecessary division - $h \leq x_h \leq h$, - $h \leq y_h \leq h$, - $h \leq z_h \leq h$
- Use parametric forms for intersection

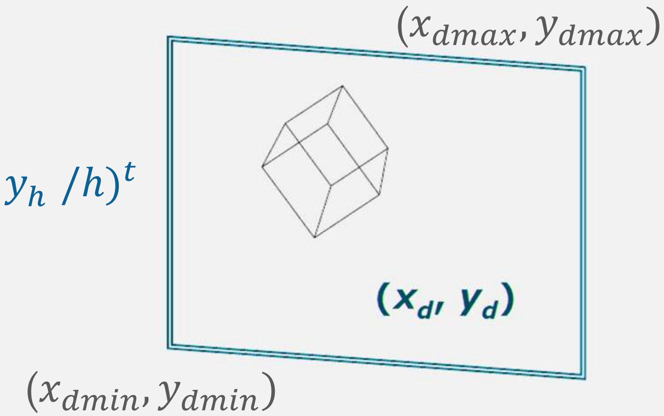
$$x_h = x_{ha} + (x_{hb} - x_{ha})u$$
$$y_h = y_{ha} + (y_{hb} - y_{ha})u$$
$$z_h = z_{ha} + (z_{hb} - z_{ha})u$$
$$h = h_a + (h_b - h_a)u$$

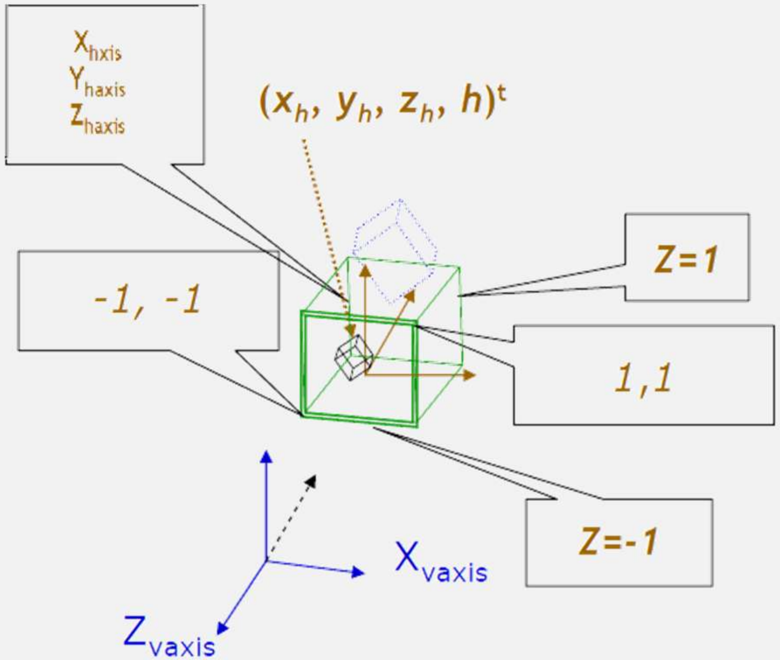


Viewport Transformation

- $(x_d, y_d, z_d, 1)^t = M_{viewport}(x_h, y_h, z_h, h)^t$

  or $(x_d, y_d)^t = {}_{SUB}M_{viewport}(x_p, y_p)^t$, $(x_p, y_p)^t = (x_h/h, y_h/h)^t$



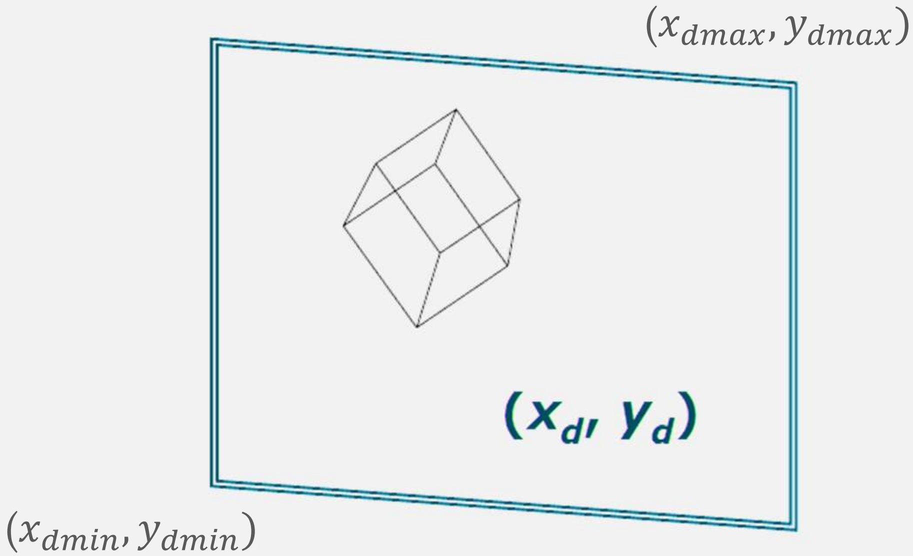


$$M_{viewport} = \begin{bmatrix} \frac{x_{d\max} - x_{d\min}}{2} & 0 & 0 & \frac{x_{d\max} + x_{d\min}}{2} \\ 0 & \frac{y_{d\max} - y_{d\min}}{2} & 0 & \frac{y_{d\max} + y_{d\min}}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rasterization

- Line drawing or polygon filling with

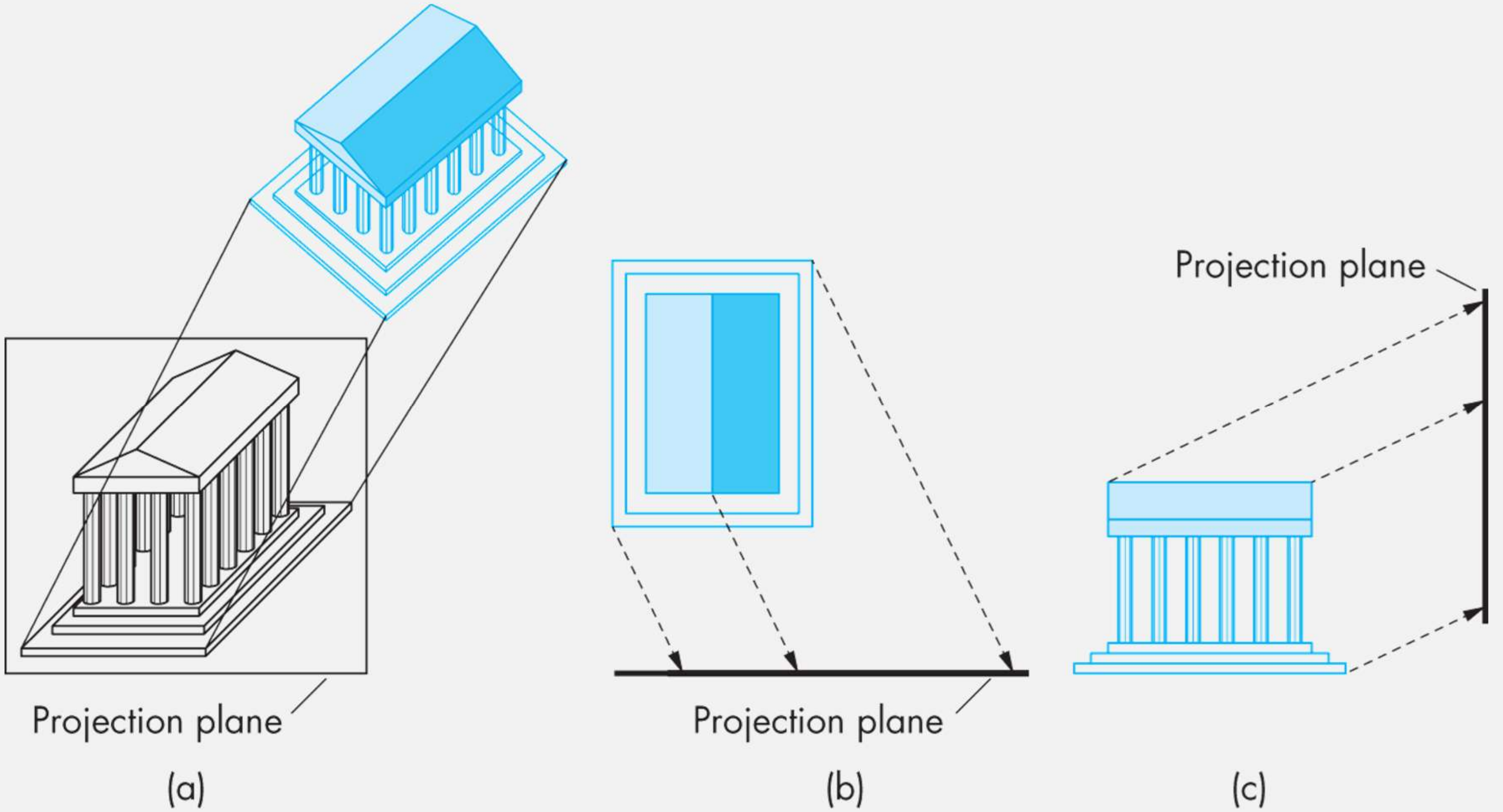
  $(x_d\ , y_d\ , z_d\ , 1)^t$ or $(x_d\ , y_d\ )^t$ and z_h



Appendix

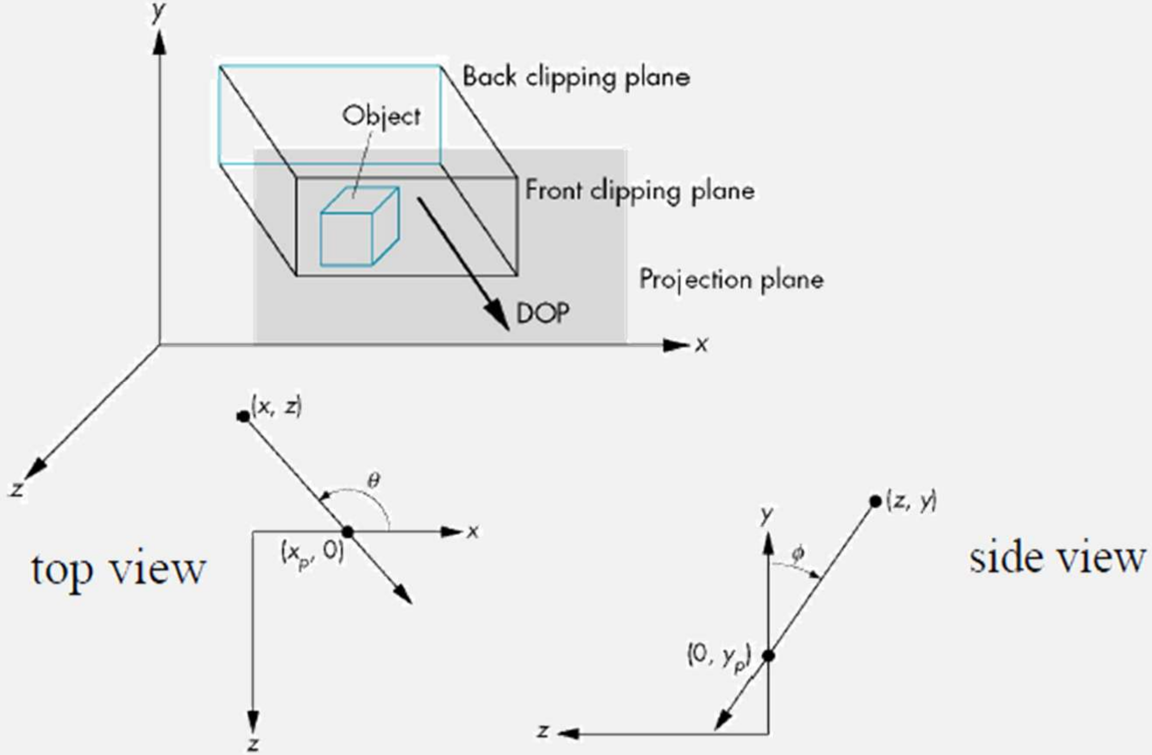
Oblique Parallel Projection

Oblique Parallel Projection



Oblique Parallel Projection (Cont.)

- Oblique Projection = Shear + Orthogonal Projection



Shear Matrix

xy shear (*z* values unchanged)

$$\mathbf{H}(\theta,\phi) = \begin{bmatrix} 1 & 0 & -\cot\theta & 0 \\ 0 & 1 & -\cot\varphi & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Projection matrix $\mathbf{P} = \mathbf{M}_{\text{orth}}\,\mathbf{H}(\theta,\phi)$
- General case: $\mathbf{P} = \mathbf{M}_{\text{orth}}\,\mathbf{STH}(\theta,\phi)$

More General Cases

$$\frac{x_p - x}{z_{vp} - z} = \frac{V_{px}}{V_{pz}}$$

$$\frac{y_p - y}{z_{vp} - z} = \frac{V_{py}}{V_{pz}}$$

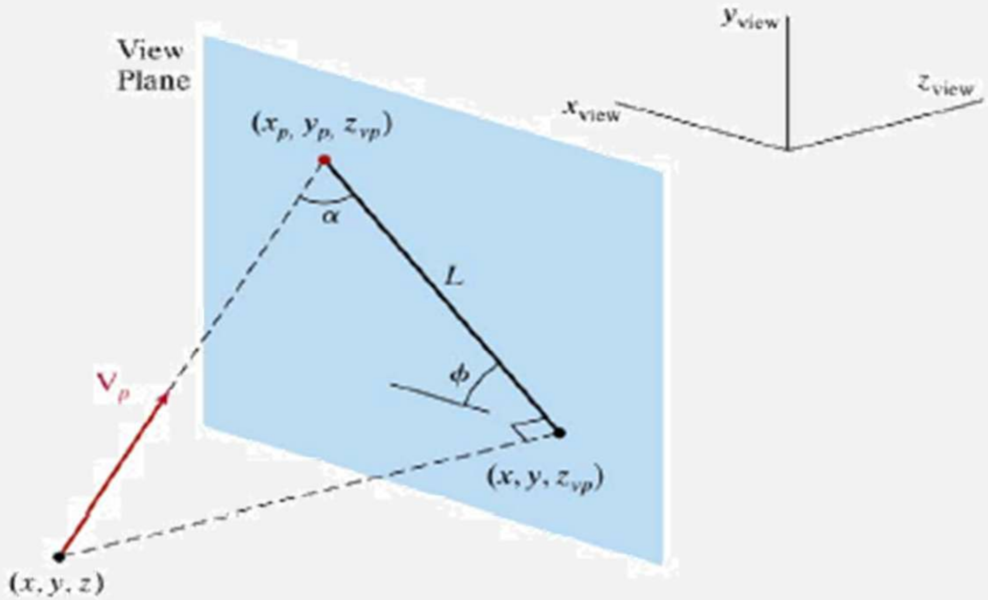


Figure 7-37

Oblique parallel projection of position (x, y, z) to a view plane along a projection line defined with vector $\mathbf{V}_p$.

More General Cases (Cont.)

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

$$y_p = y + (z_{vp} - z)\frac{V_{py}}{V_{pz}}$$

$$M_{oblique} = \begin{bmatrix} 1 & 0 & -\frac{V_{px}}{V_{pz}} & z_{vp}\frac{V_{px}}{V_{pz}} \\ 0 & 1 & -\frac{V_{py}}{V_{pz}} & z_{vp}\frac{V_{py}}{V_{pz}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boxed{M_{oblique,norm} = M_{ortho,norm} M_{oblique}}$$

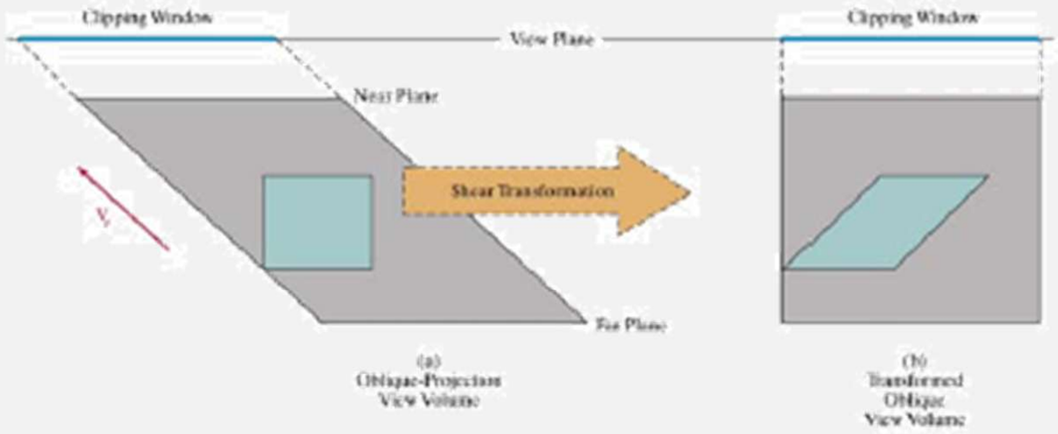
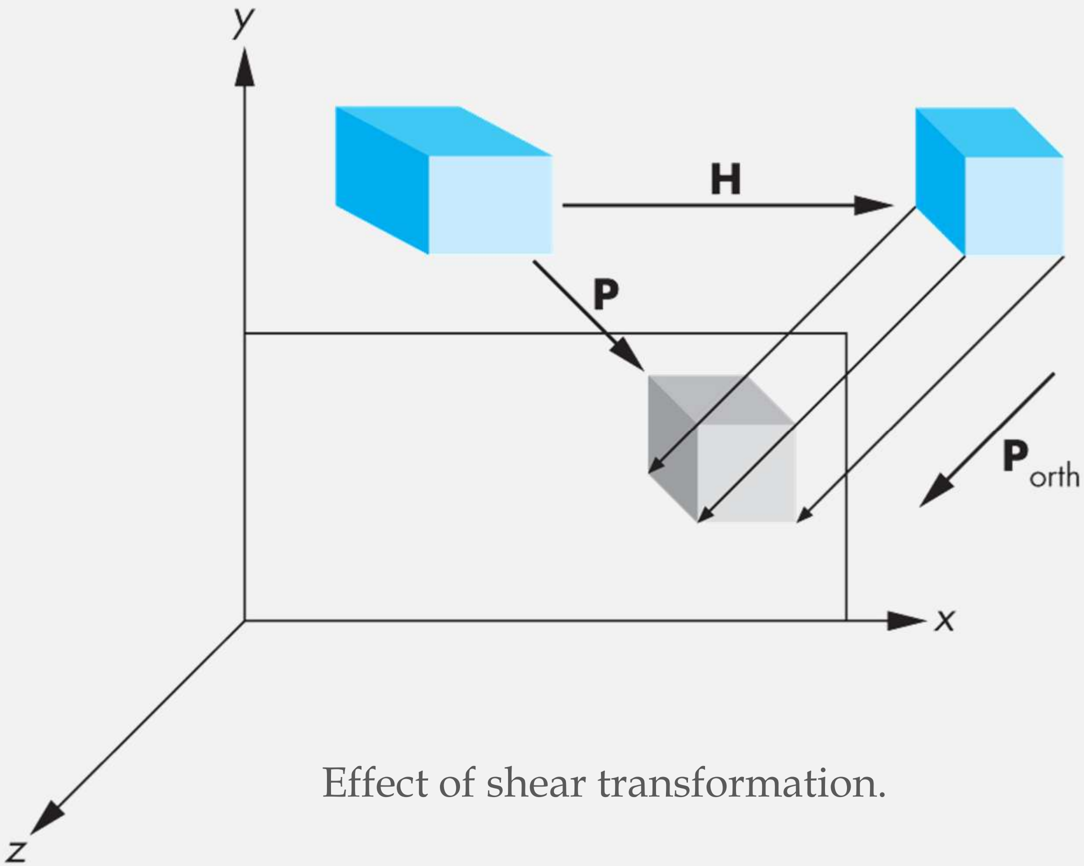


Figure 7-39

Top view of an oblique parallel-projection transformation. The oblique view volume is converted into a rectangular parallelepiped, and objects in the view volume, such as the green block, are mapped to orthogonal-projection coordinates.

Equivalency



Effect of shear transformation.