

Introduction to Computer Graphics

3. Viewing in 3D

I-Chen Lin

National Chiao Tung University

Textbook: E. Angel, D. Shreiner Interactive Computer Graphics, 6th Ed., Pearson
Ref: D.D. Hearn, M. P. Baker, W. Carithers, Computer Graphics with OpenGL, 4th Ed., Pearson

Outline

- ▶ Classical views
- ▶ Computer viewing
- ▶ Projection matrices

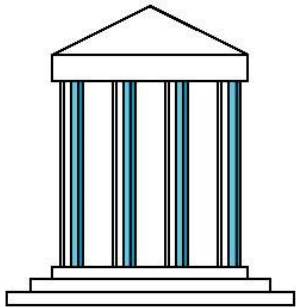
Classical Viewing

- ▶ 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
- ▶ Each object is assumed to be constructed from flat *principal faces*
 - ▶ Buildings, polyhedra, manufactured objects

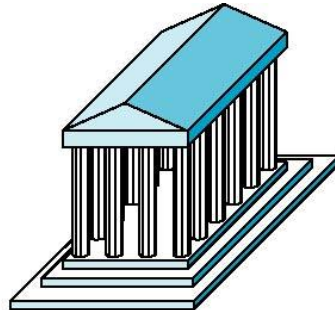
Planar Geometric Projections

- ▶ Standard projections project onto a plane.
- ▶ Projectors are lines that either
 - ▶ converge at a center of projection
 - ▶ are parallel
- ▶ Such projections preserve lines
 - ▶ but not necessarily angles
- ▶ When do we need non-planar projections?

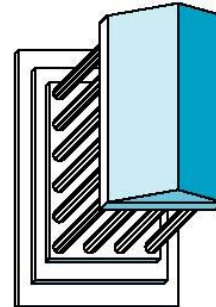
Classical Projections



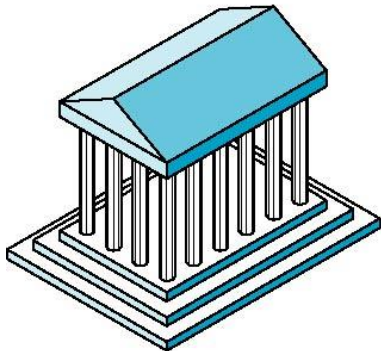
Front elevation



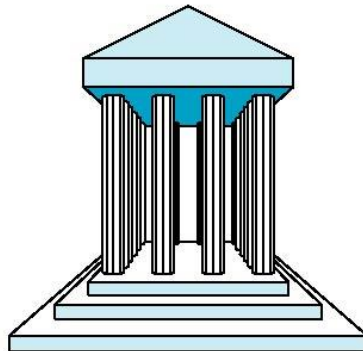
Elevation oblique



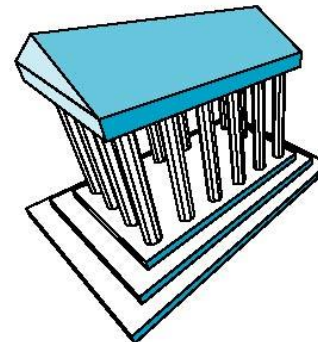
Plan oblique



Isometric



One-point perspective

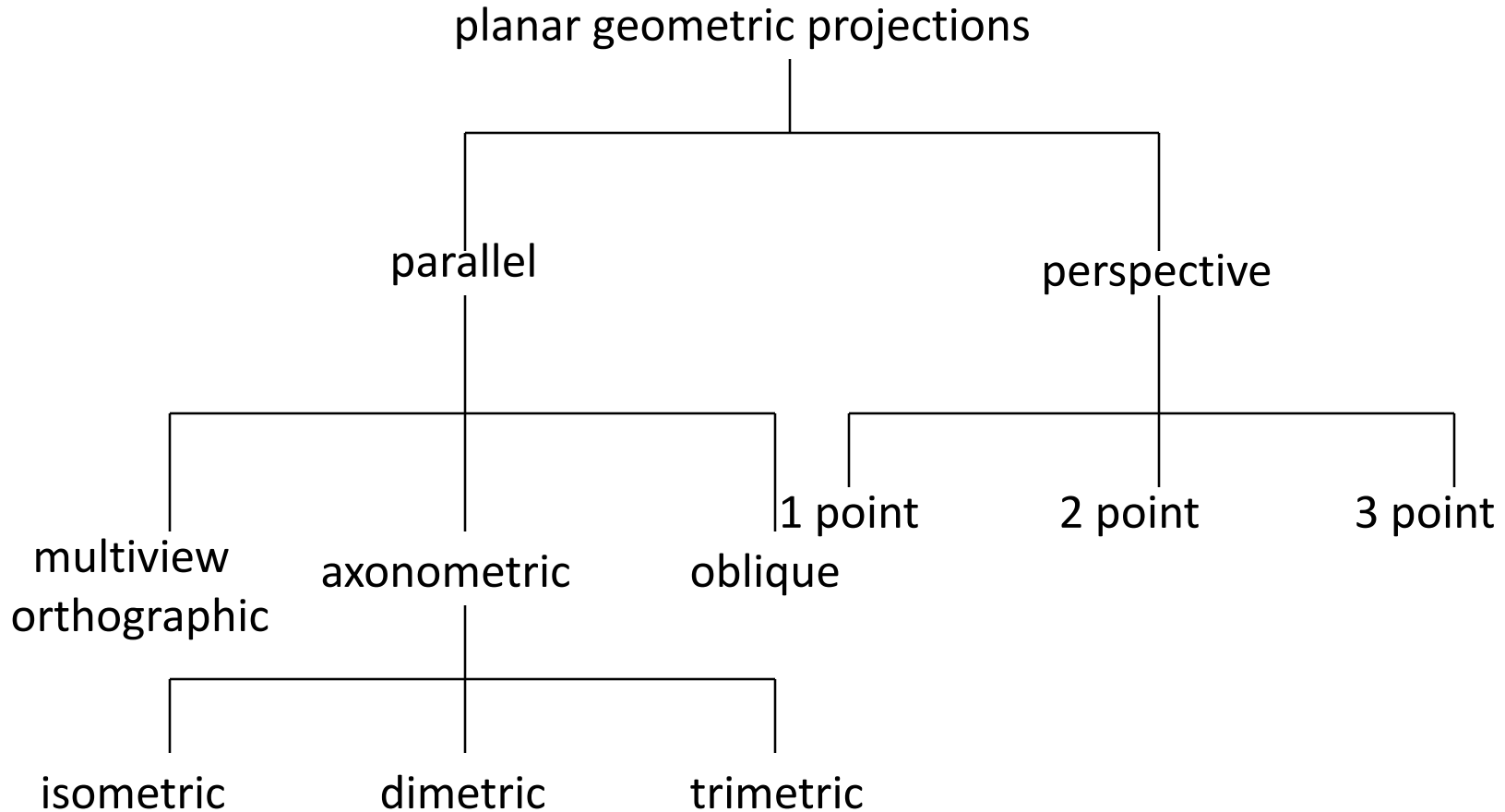


Three-point perspective

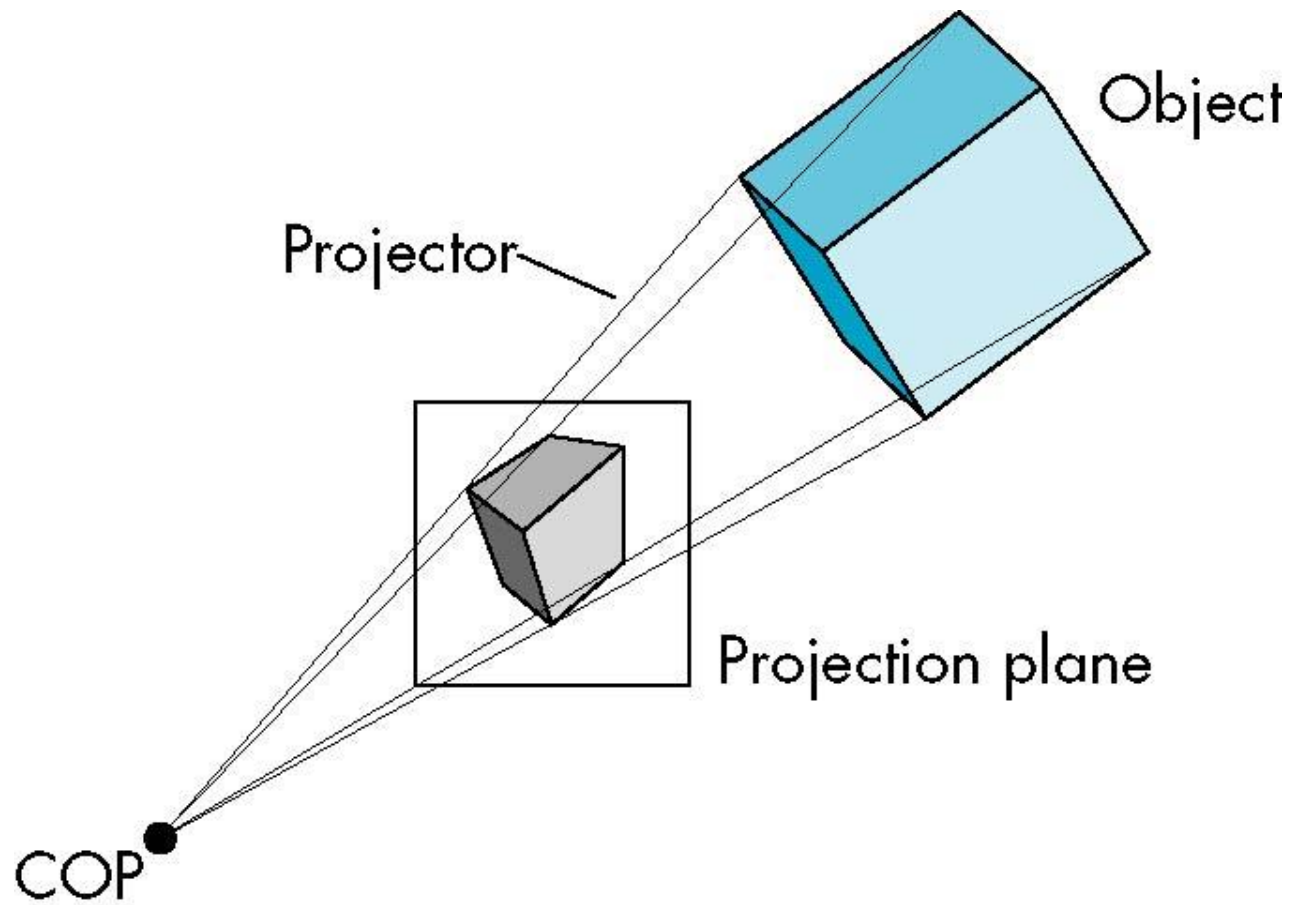
Perspective vs. Parallel

- ▶ Classical viewing developed different techniques for drawing each type of projection
- ▶ Mathematically parallel viewing is the limit of perspective viewing
- ▶ Computer graphics treats all projections the same and implements them with a single pipeline

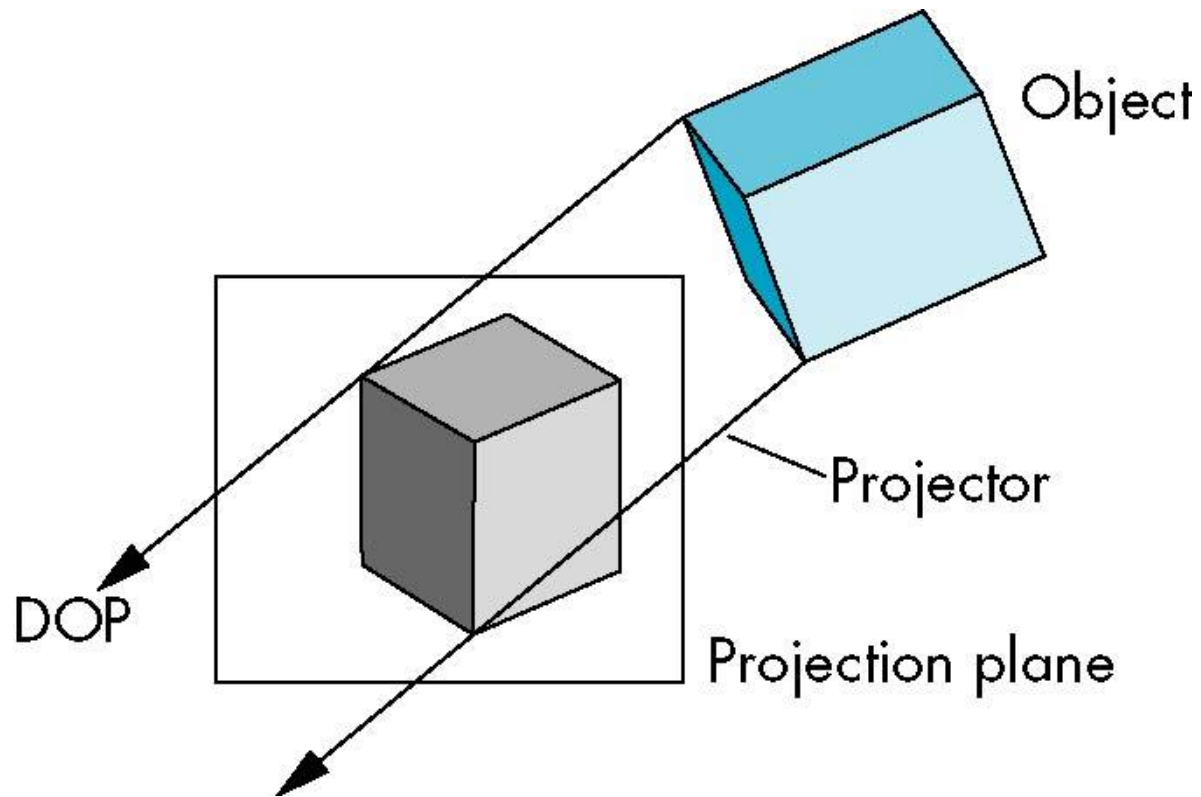
Taxonomy of Planar Geometric Projections



Perspective Projection

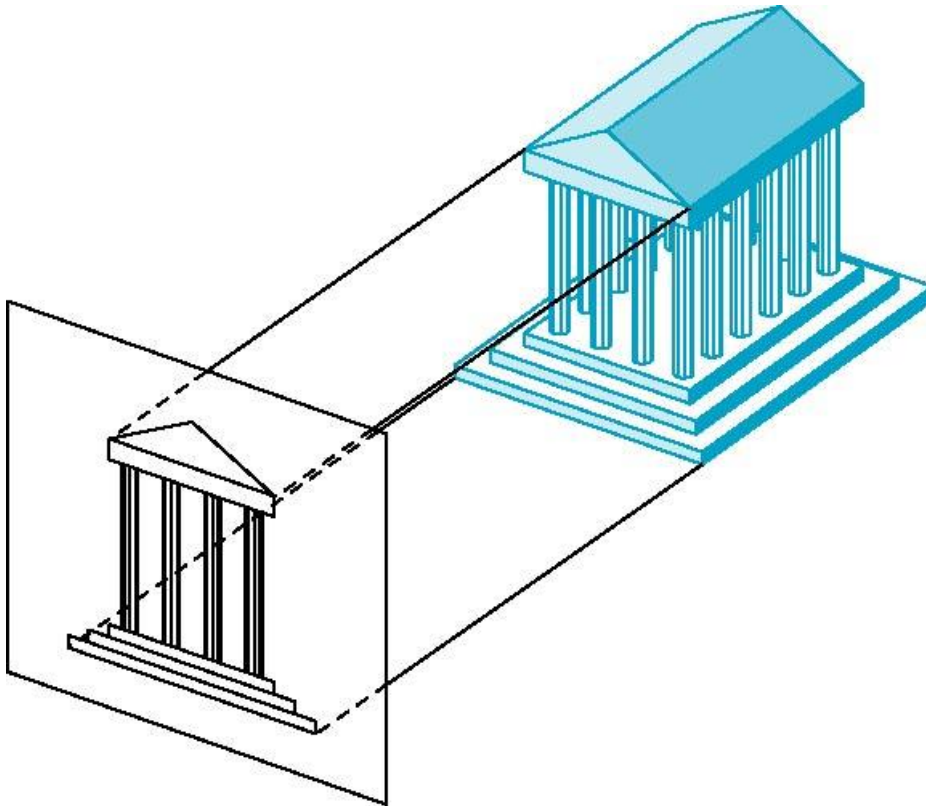


Parallel Projection



Orthographic Projection

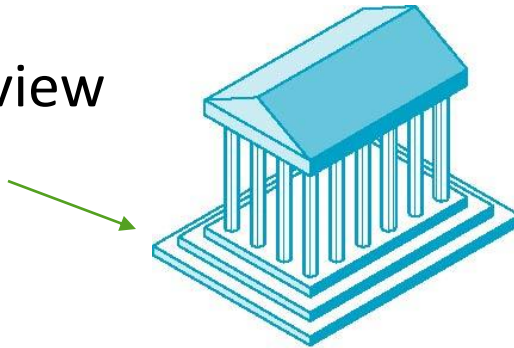
- Projectors are orthogonal to projection surface



Multi-view Orthographic Projection

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

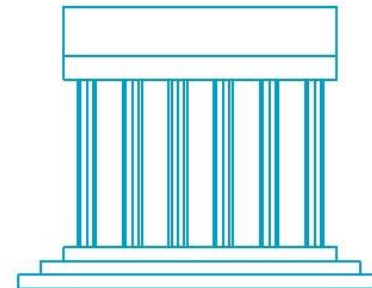
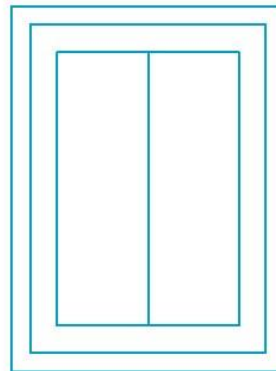
isometric (not multiview orthographic view)



front

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

top



side

Advantages and Disadvantages

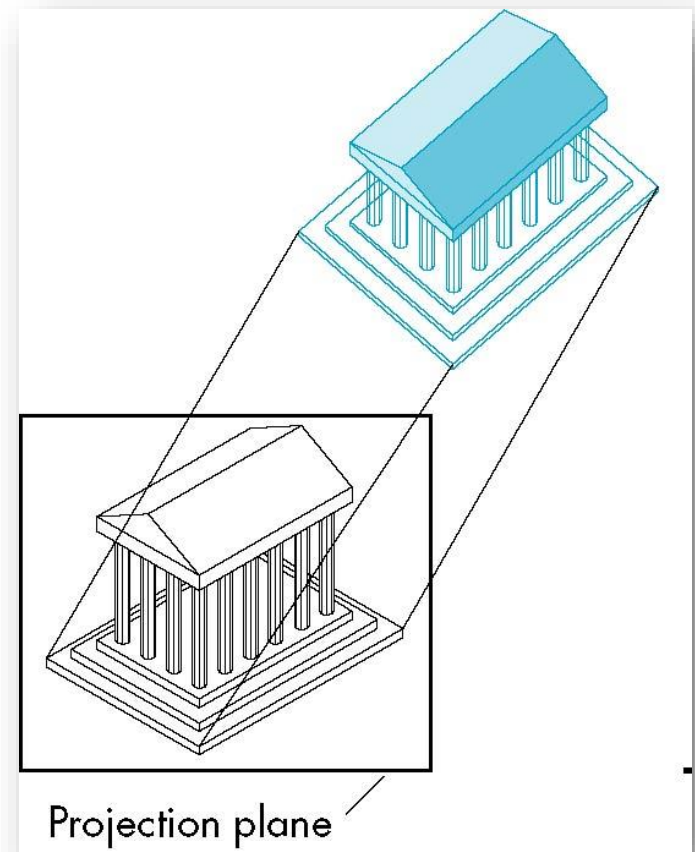
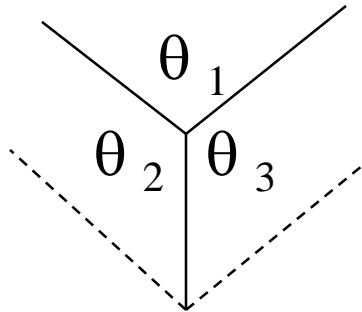
- ▶ Preserves both distances and angles
 - ▶ Shapes preserved
 - ▶ Can be used for measurements
 - ▶ Building plans
 - ▶ Manuals
- ▶ Cannot see what object really looks like because many surfaces hidden from view
 - ▶ Often we add the isometric

Axonometric Projections

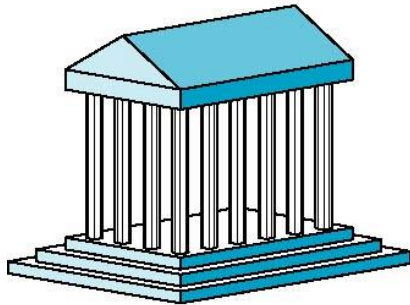
- Allow projection plane to move relative to object

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

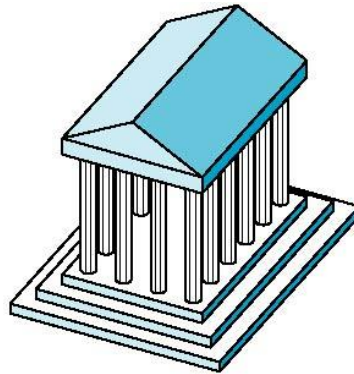
none: trimetric
two: dimetric
three: isometric



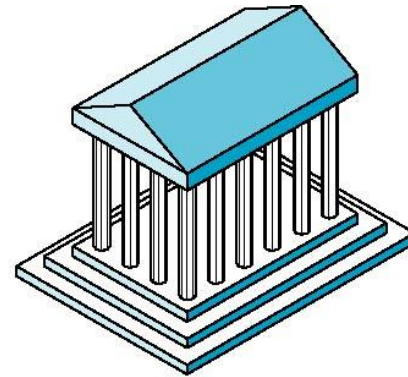
Types of Axonometric Projections



Dimetric



Trimetric



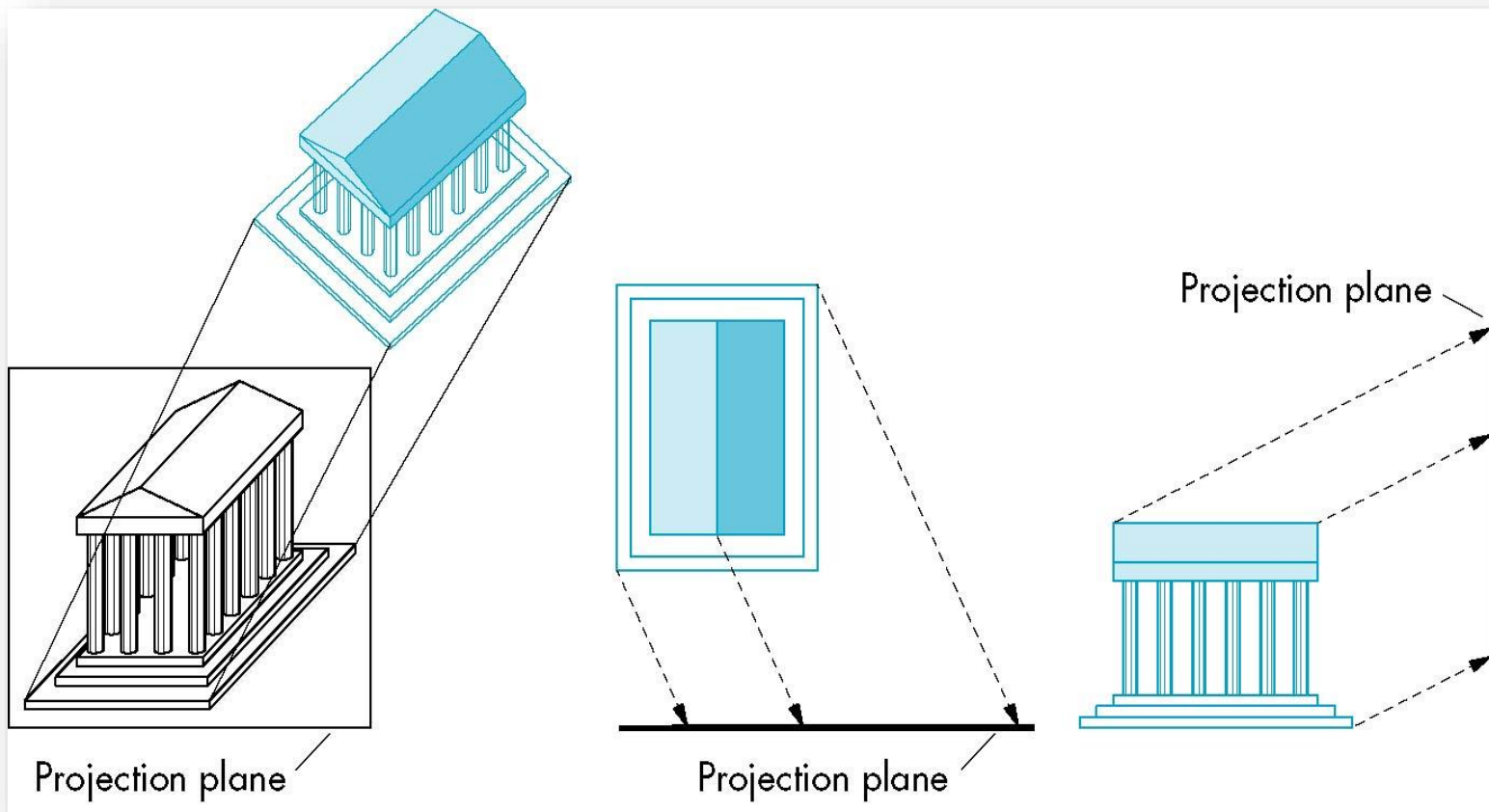
Isometric

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



Advantages and Disadvantages

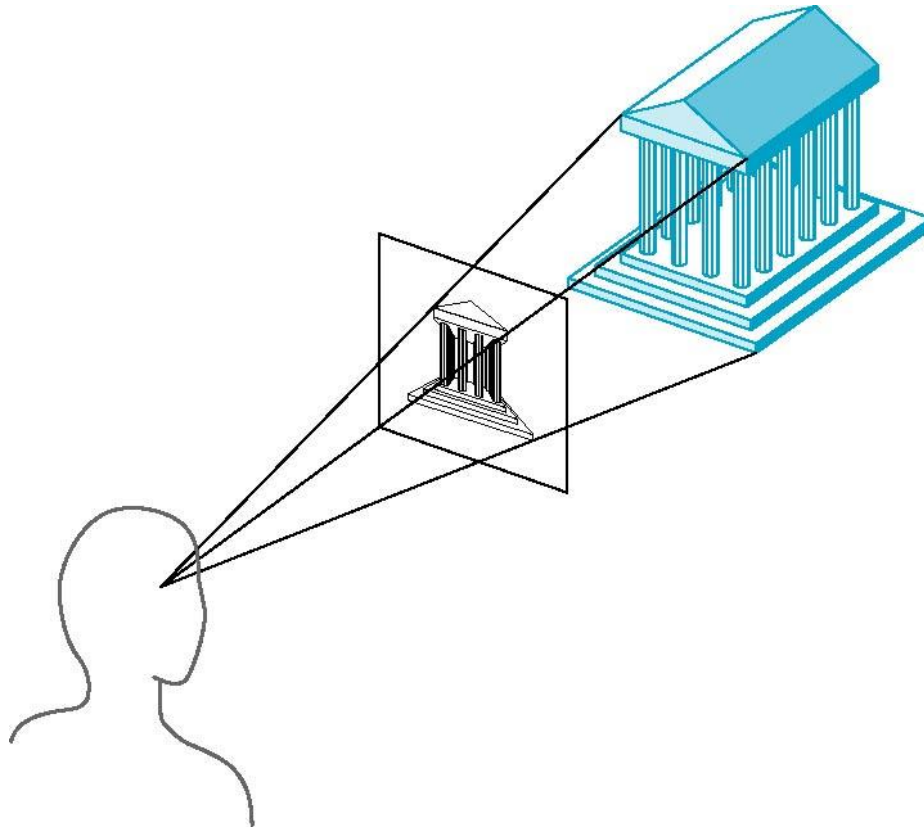
- ▶ Can pick the angles to emphasize a particular face
 - ▶ Architecture: plan oblique, elevation oblique
- ▶ Angles in faces parallel to projection plane are preserved while we can still see “around” side



- ▶ In physical world, cannot create with simple camera; possible with bellows camera or special lens (architectural)

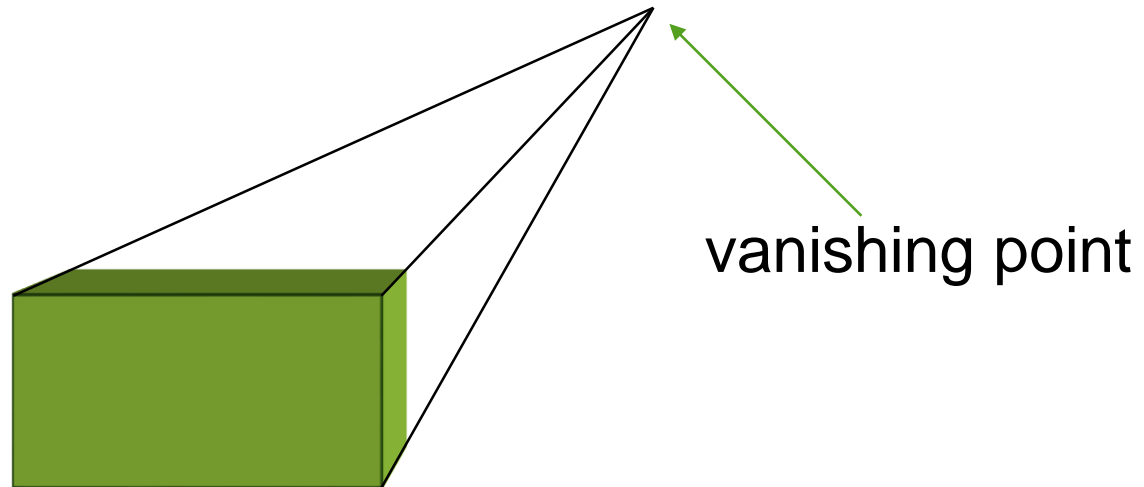
Perspective Projection

- Projectors coverage at center of projection



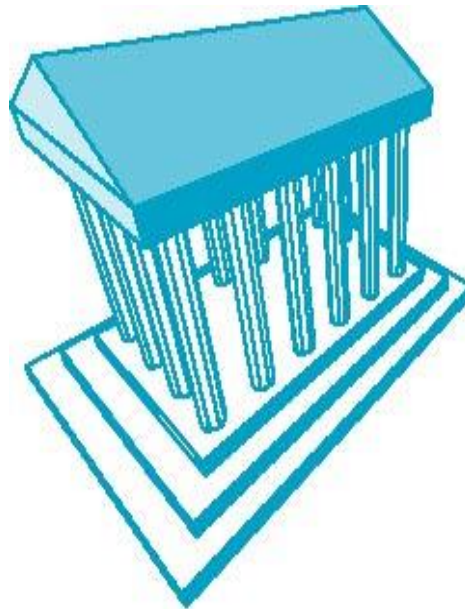
Vanishing Points

- ▶ 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 projection plane
- ▶ Three vanishing points for cube



Two-Point Perspective

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



One-Point Perspective

- ▶ One principal face parallel to projection plane
- ▶ One vanishing point for cube



Advantages and Disadvantages

- ▶ Diminution:
 - ▶ Objects further 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

Computer Viewing

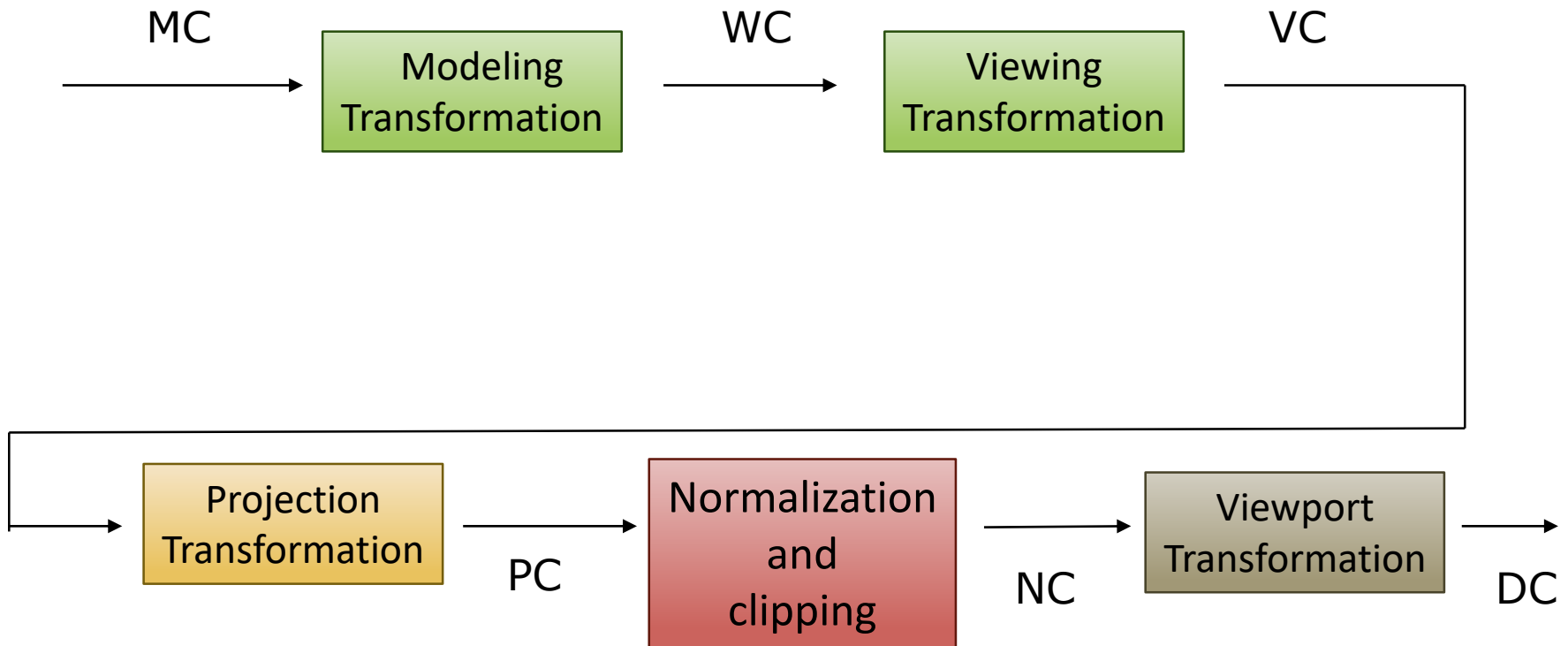
Computer Viewing

- ▶ 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*
 - ▶ Clipping
 - ▶ Setting the *view volume*

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

Pipeline View



Let's skip the clipping details temporarily !

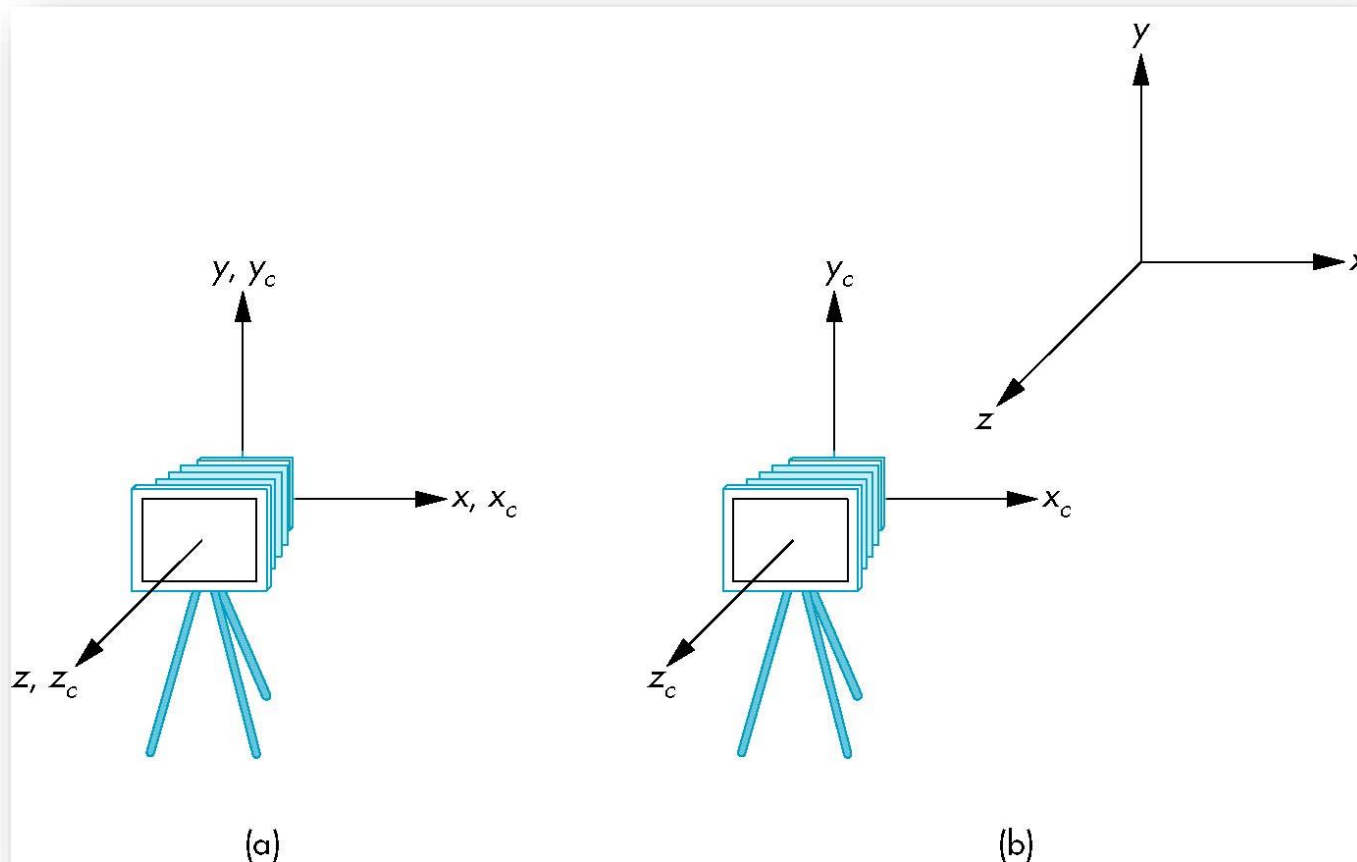
Moving the Camera Frame

- ▶ 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

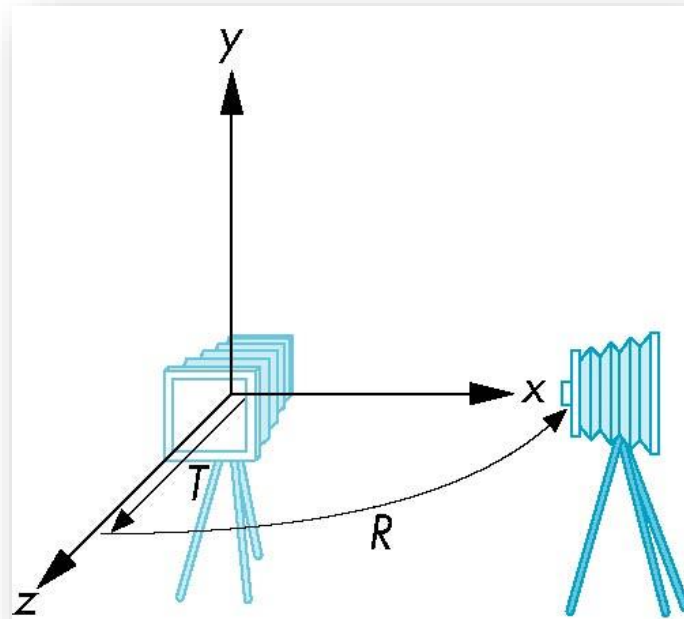
default frames

frames after translation by $-d$
 $d > 0$



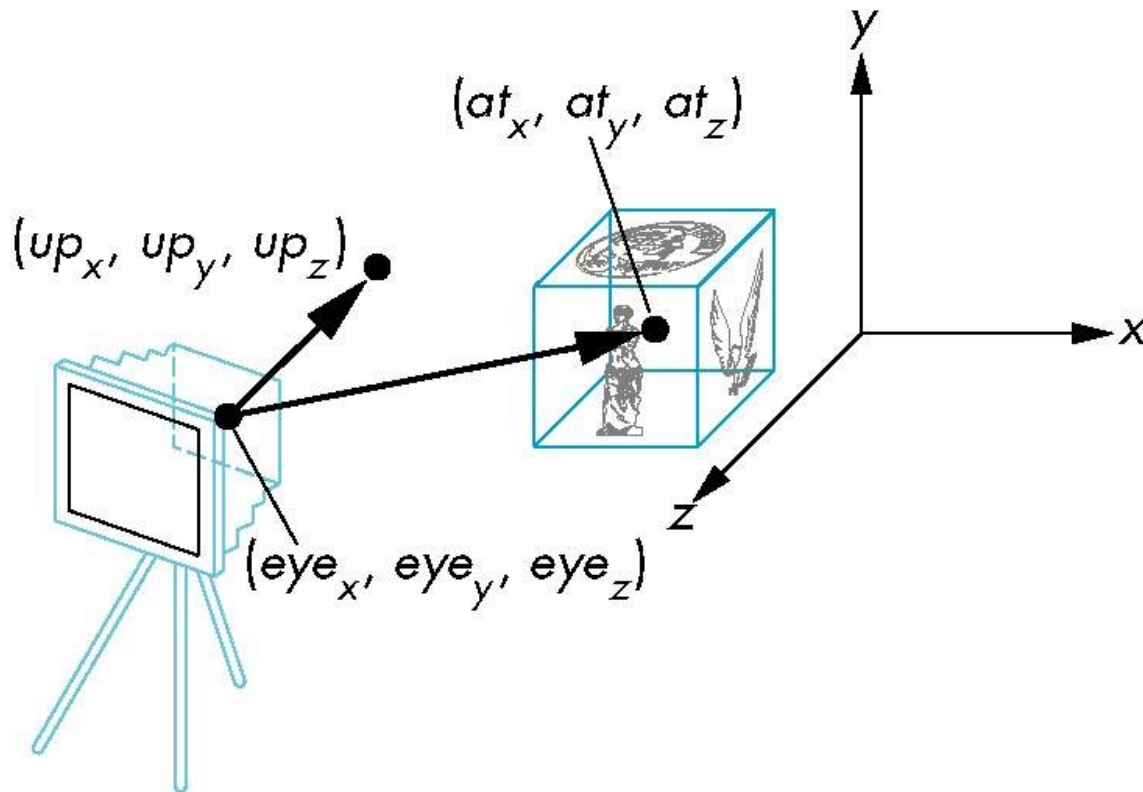
Moving the Camera

- ▶ 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
 - ▶ Model-view matrix $C = TR$

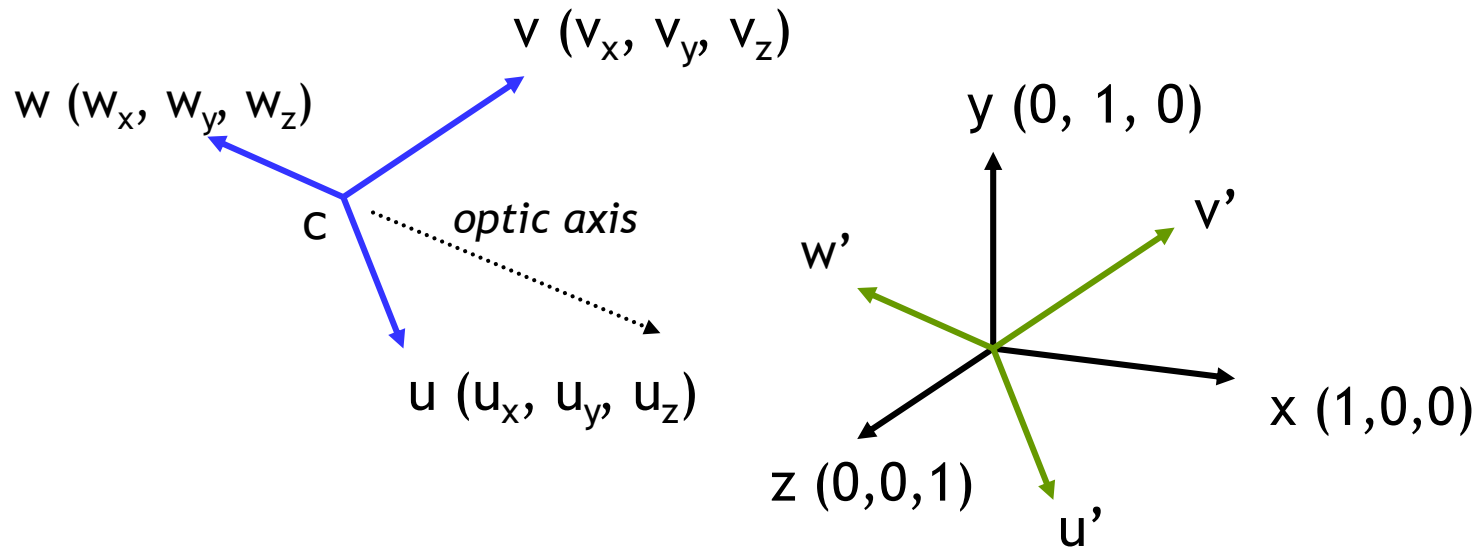


gluLookAt

- `gluLookAt(eyex, eyey, eyez, atx, aty, atz, upx, upy, upz)`



By Coordinate Transformations



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

$$\begin{bmatrix} x'_{vc} \\ y'_{vc} \\ z'_{vc} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & -c_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{vc} \\ y_{vc} \\ z_{vc} \\ 1 \end{bmatrix}$$

Projections and Normalization

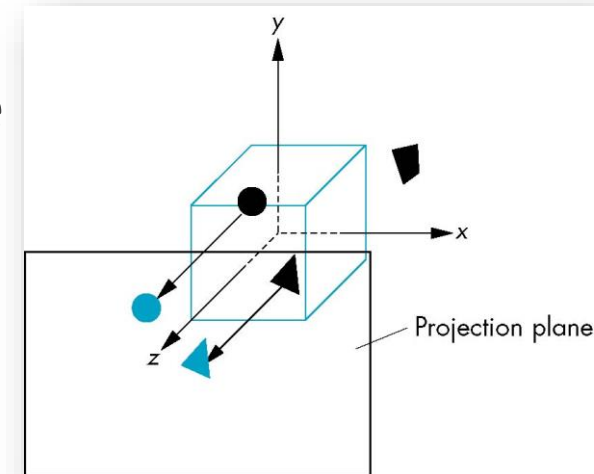
- ▶ The default projection in the eye (camera) frame is orthogonal

- ▶ For points within the default view volume

- ▶ $x_p = x$

- ▶ $y_p = y$

- ▶ $z_p = 0$



- ▶ Most graphics systems use view normalization
 - ▶ All other views are converted to the default view by transformations that determine the projection matrix
 - ▶ Allows use of the same pipeline for all views

Homogeneous Coordinate Representation

default orthographic projection

► $x_p = x$

► $y_p = y$

► $z_p = 0$

► $w_p = 1$

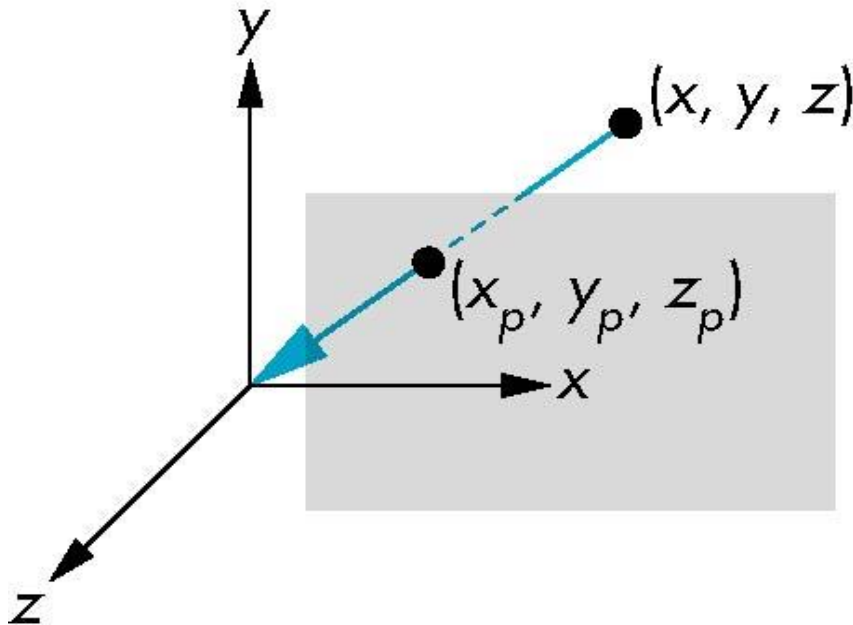
$$\mathbf{p}_p = \mathbf{M}\mathbf{p}$$

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

In practice, we can let $\mathbf{M} = \mathbf{I}$ and set the z term to zero later

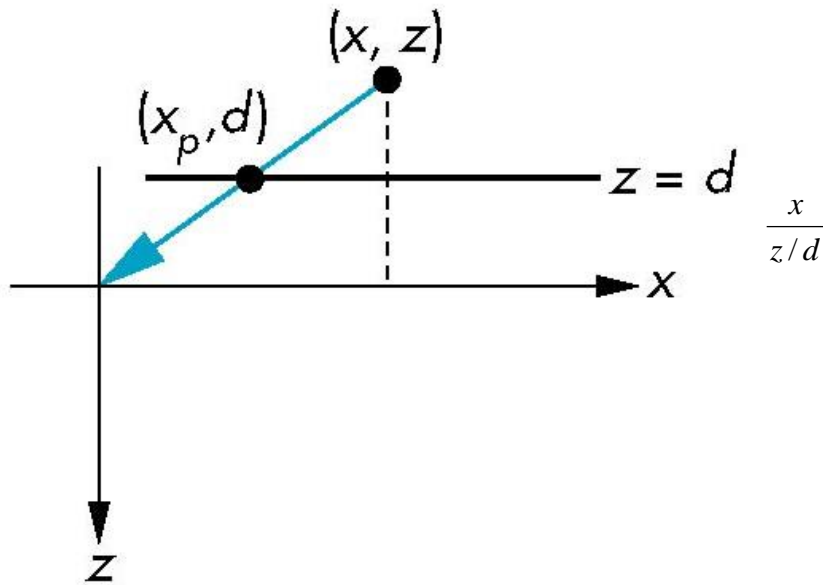
Simple Perspective

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

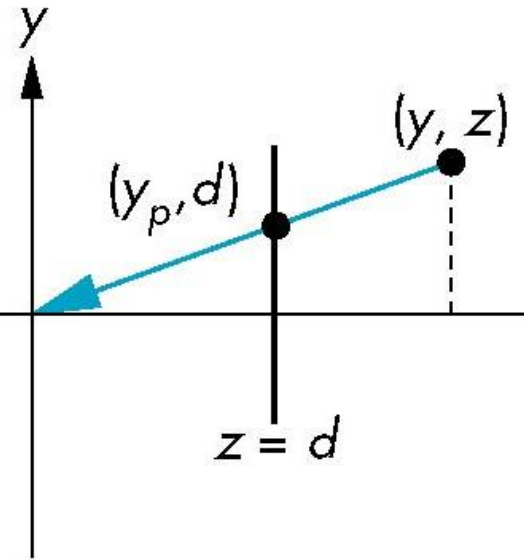


Perspective Equations

Top view



Side view



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

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

$$z_p = d$$

Homogeneous Coordinate Form

consider $\mathbf{q} = \mathbf{M}\mathbf{p}$ where

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

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \Rightarrow \mathbf{q} = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix}$$

Perspective Division

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

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

the desired perspective equations

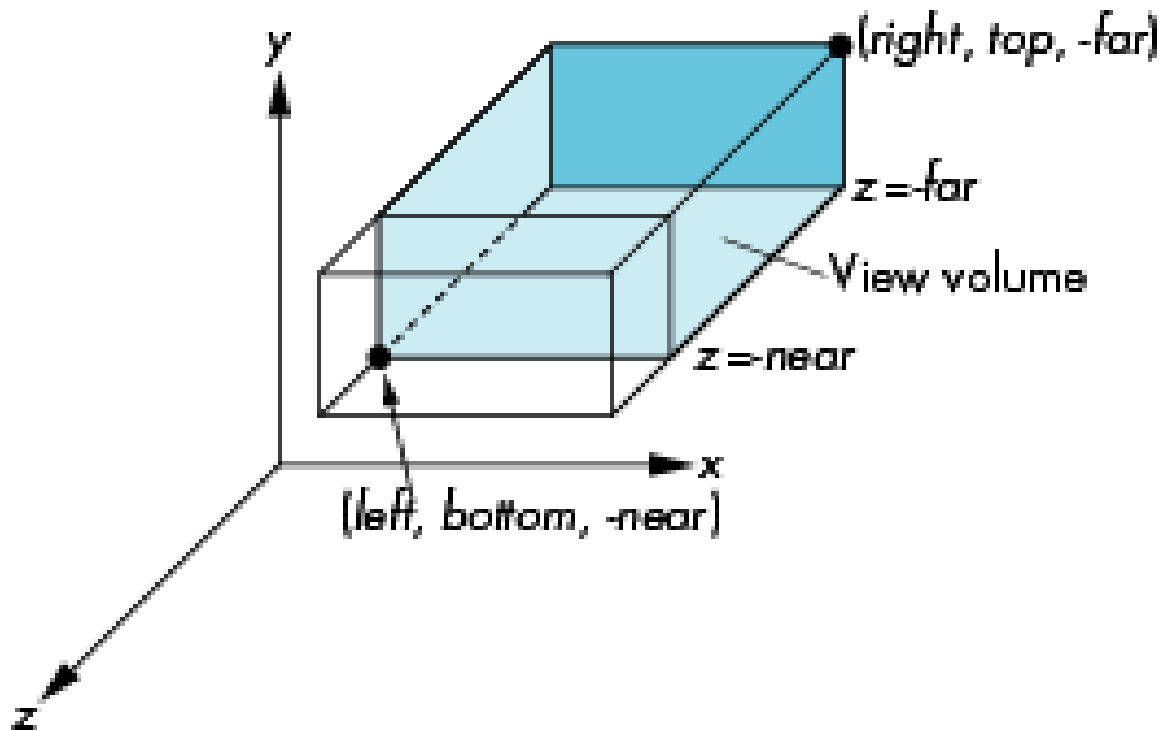
Taking Clipping into Account

- ▶ 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

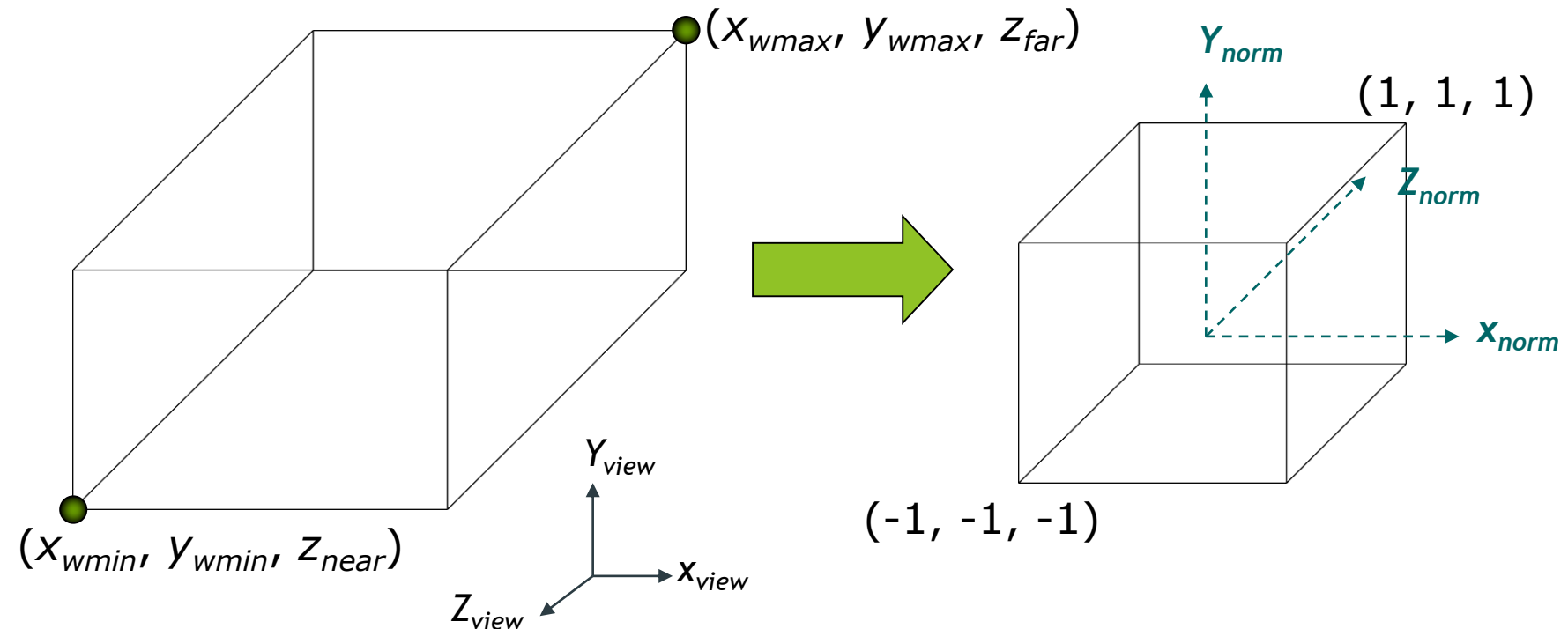
`Ortho (left, right, bottom, top, near, far)`



Orthogonal Normalization

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

normalization \Rightarrow find transformation to convert specified clipping volume to default



Orthogonal Matrix

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

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

Final Projection

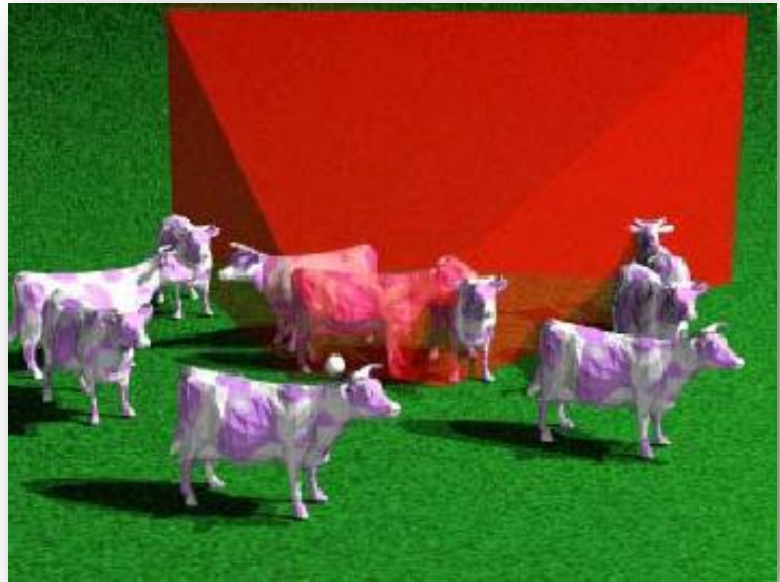
- ▶ Set $z=0$
- ▶ Equivalent to the homogeneous coordinate transformation

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

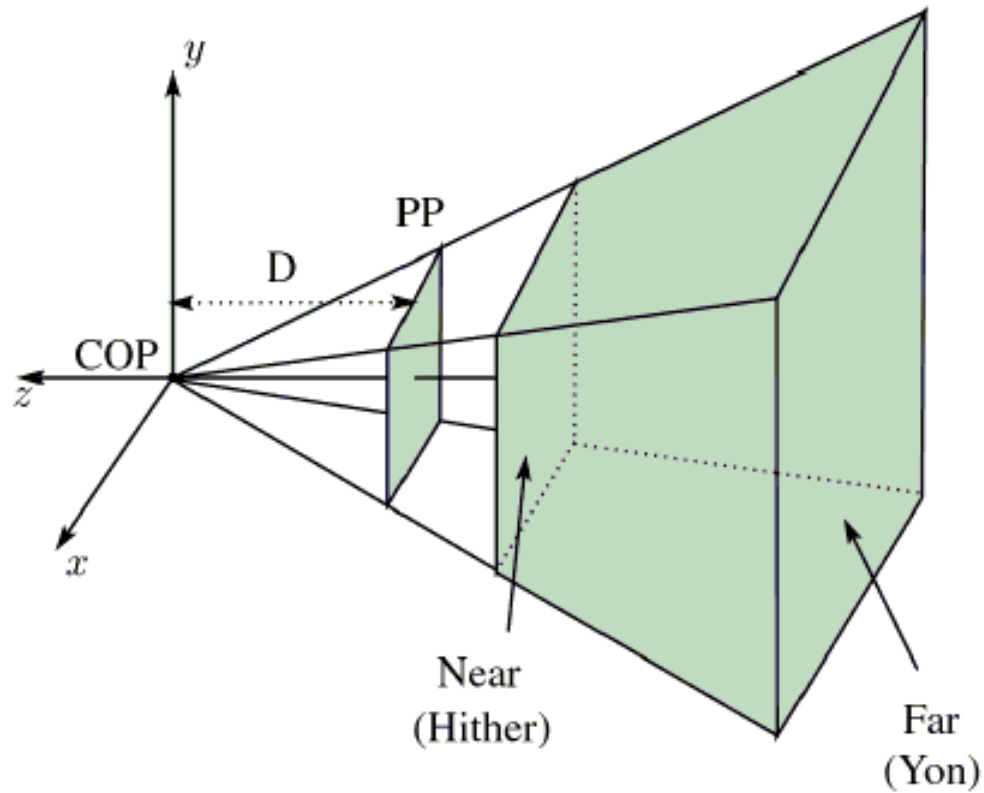
- ▶ Hence, general orthogonal projection in 4D is

$$\mathbf{P} = \mathbf{M}_{\text{orth}} \mathbf{S} \mathbf{T}$$

Clipping for Perspective Views



Perspective Viewing Volume



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*.

Normalization

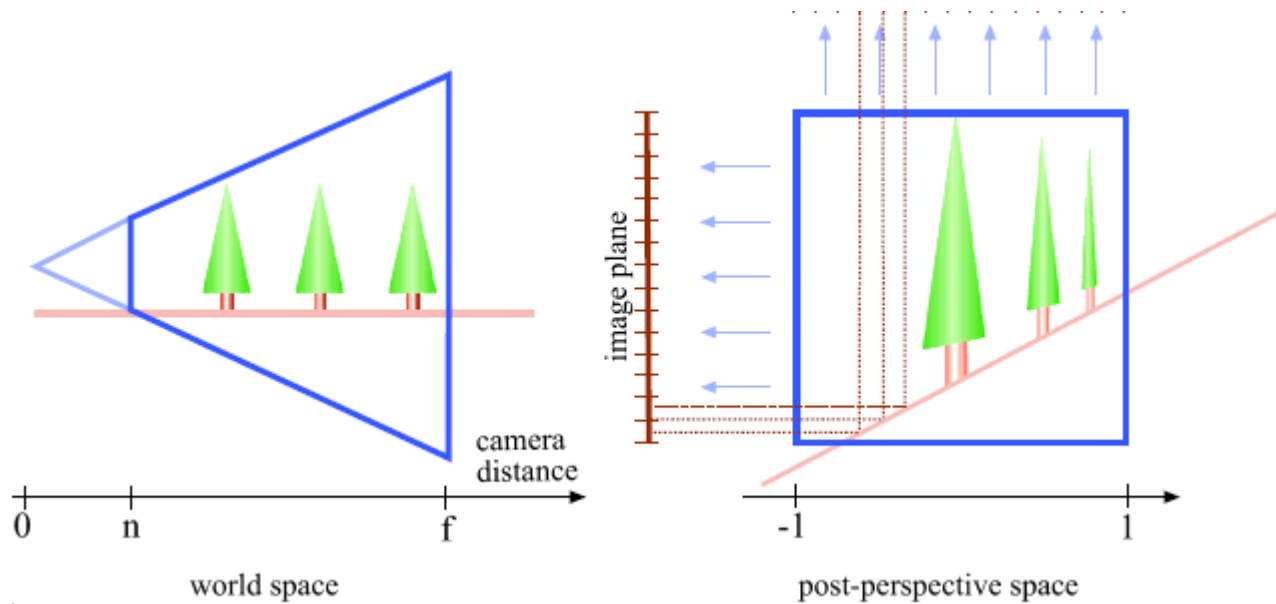
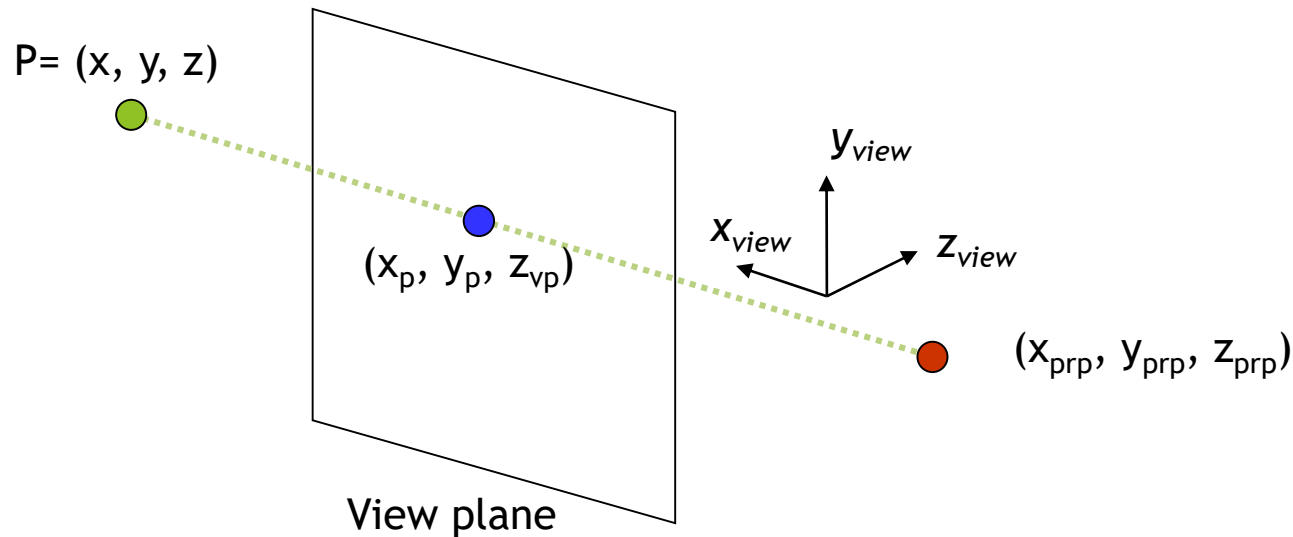


Fig. from: M. Stamminger, G. Drettakis, Perspective Shadow Maps, Proc. ACM SIGGRAPH 2002.

Perspective-Projection Trans.



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

$$y_p = (1-u)y + uy_{prp} \quad u = 0 \sim 1$$

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

$$x_p = x \left(\frac{z_{prp} - z_{vp}}{z_{prp} - z} \right) + x_{prp} \left(\frac{z_{vp} - z}{z_{prp} - z} \right)$$

$$y_p = y \left(\frac{z_{prp} - z_{vp}}{z_{prp} - z} \right) + y_{prp} \left(\frac{z_{vp} - z}{z_{prp} - z} \right)$$

$$x_p = x \left(\frac{-z_{near}}{-z} \right)$$

$$y_p = y \left(\frac{-z_{near}}{-z} \right)$$

Perspective-Projection Trans.

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

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

$$x_p = x \left(\frac{-z_{near}}{-z} \right)$$

$$y_p = y \left(\frac{-z_{near}}{-z} \right)$$

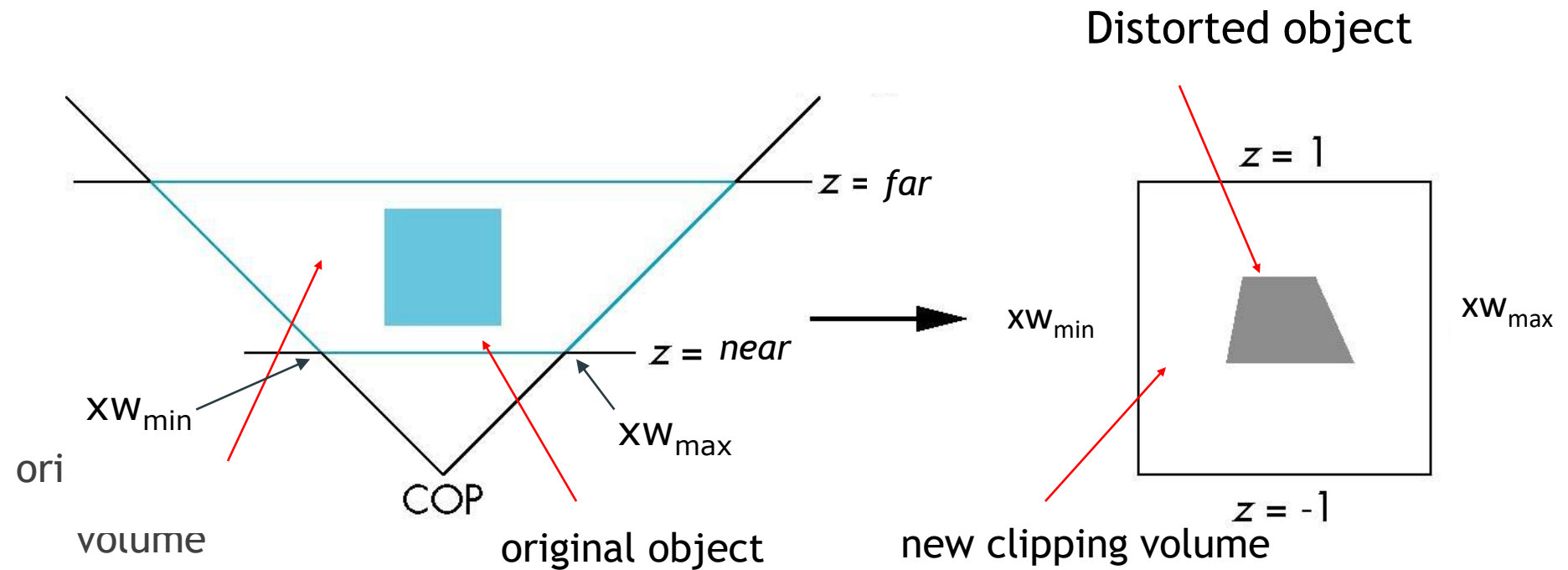
$$z_p = \frac{s_z z + t_z}{-z} = - \left(s_z + \frac{t_z}{z} \right)$$

To make $-1 \leq z_p \leq 1$

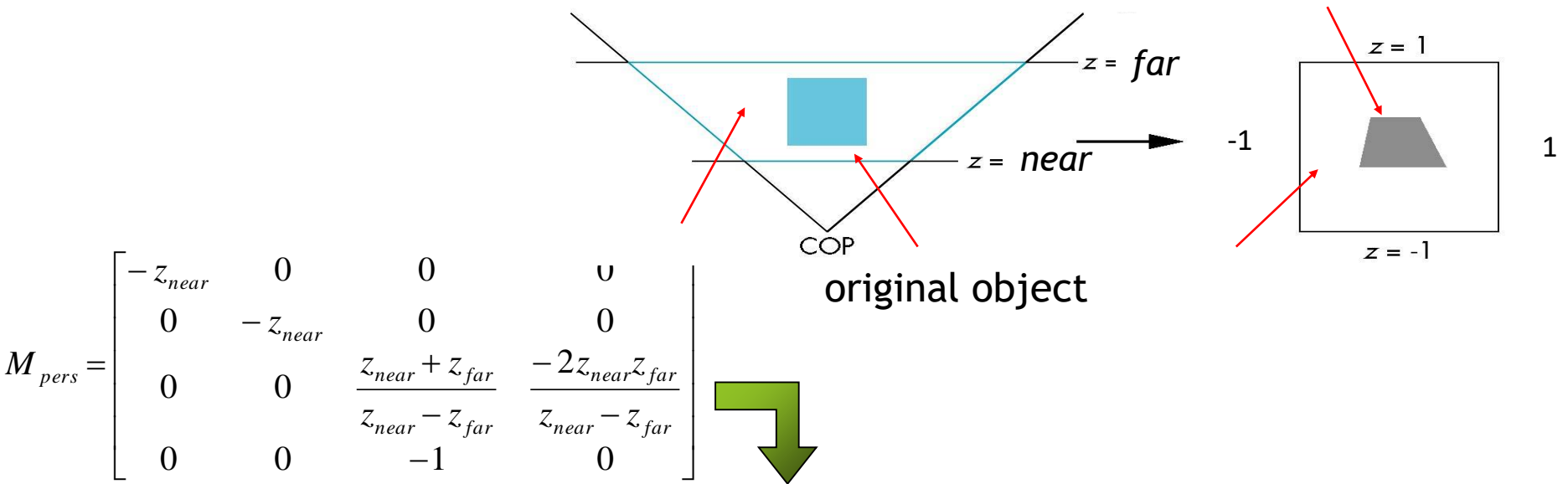
$$s_z = \frac{z_{near} + z_{far}}{z_{near} - z_{far}}$$

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

Perspective-Projection Trans.



Further Normalization



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

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

Normalization and Hidden-Surface Removal

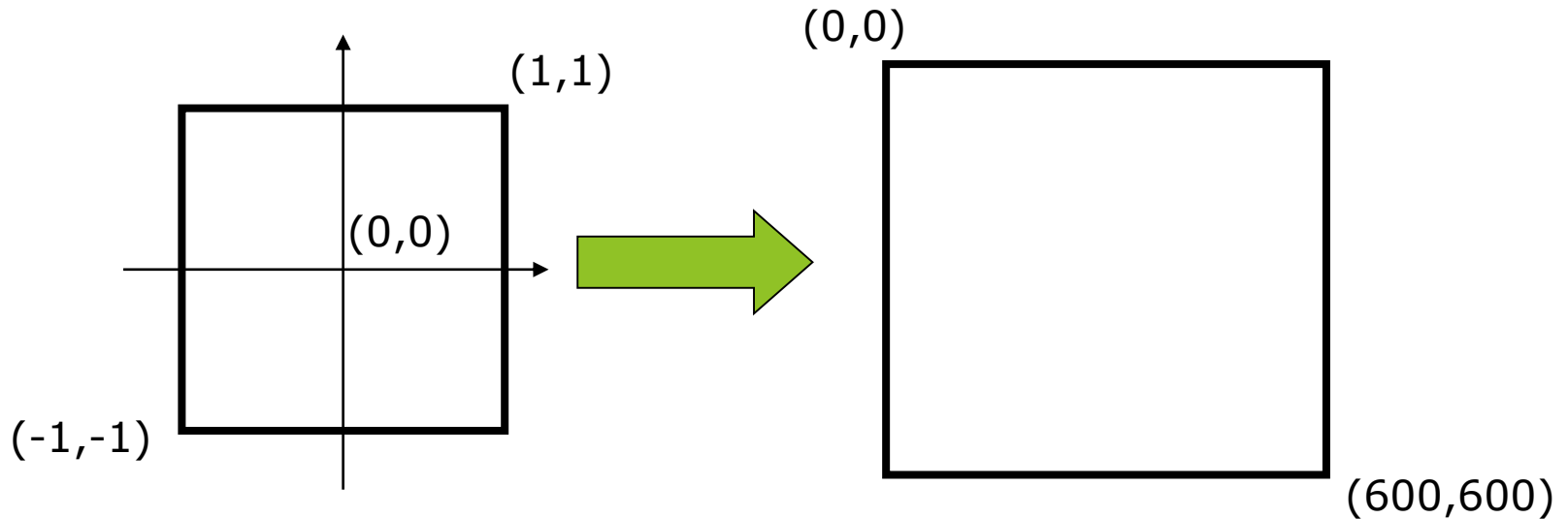
- ▶ if $z_1 > z_2$ in the original clipping volume then for the transformed points $z_1' < z_2'$
- ▶ Hidden surface removal works if we first apply the normalization transformation
- ▶ However, the formula $z'' = -(s_z + t_z/z)$ implies that the distances are distorted by the normalization which can cause **numerical problems** especially if the near distance is small

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”.*

Viewport Transformation

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

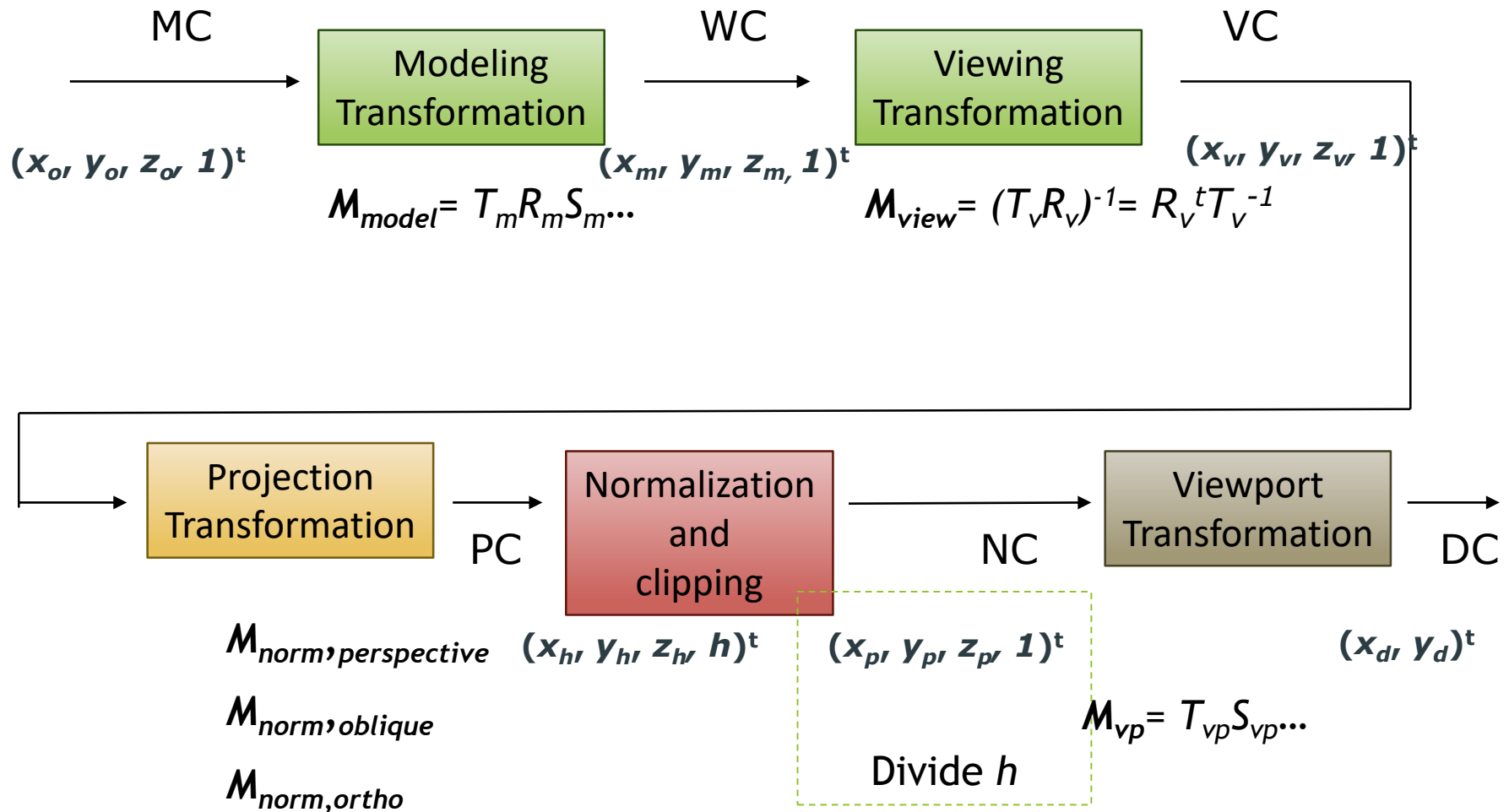


By 2D scaling and translation

Viewing in 3D

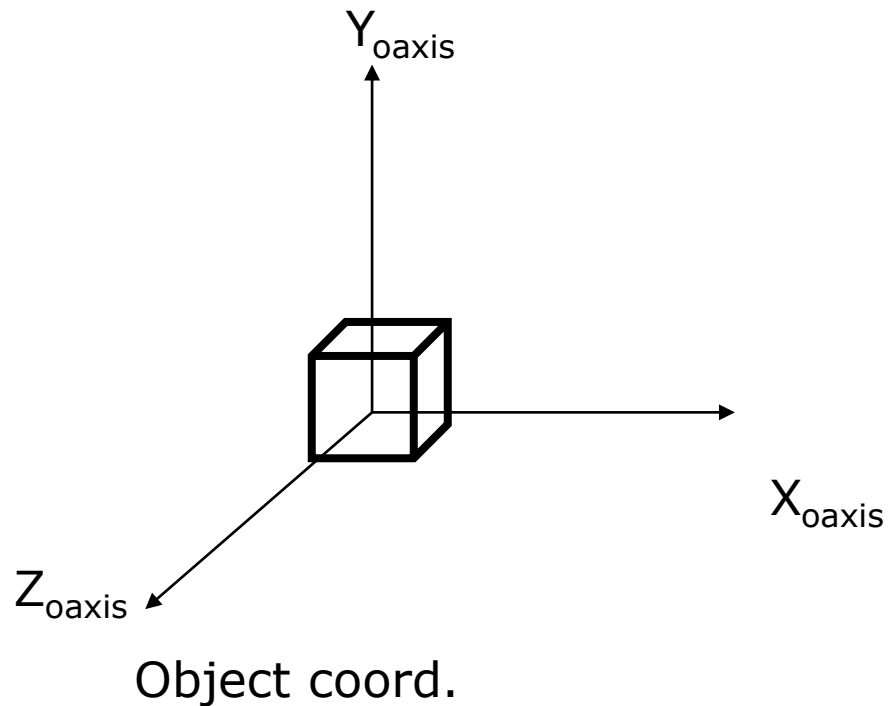
(Summary and Example)

Pipeline View



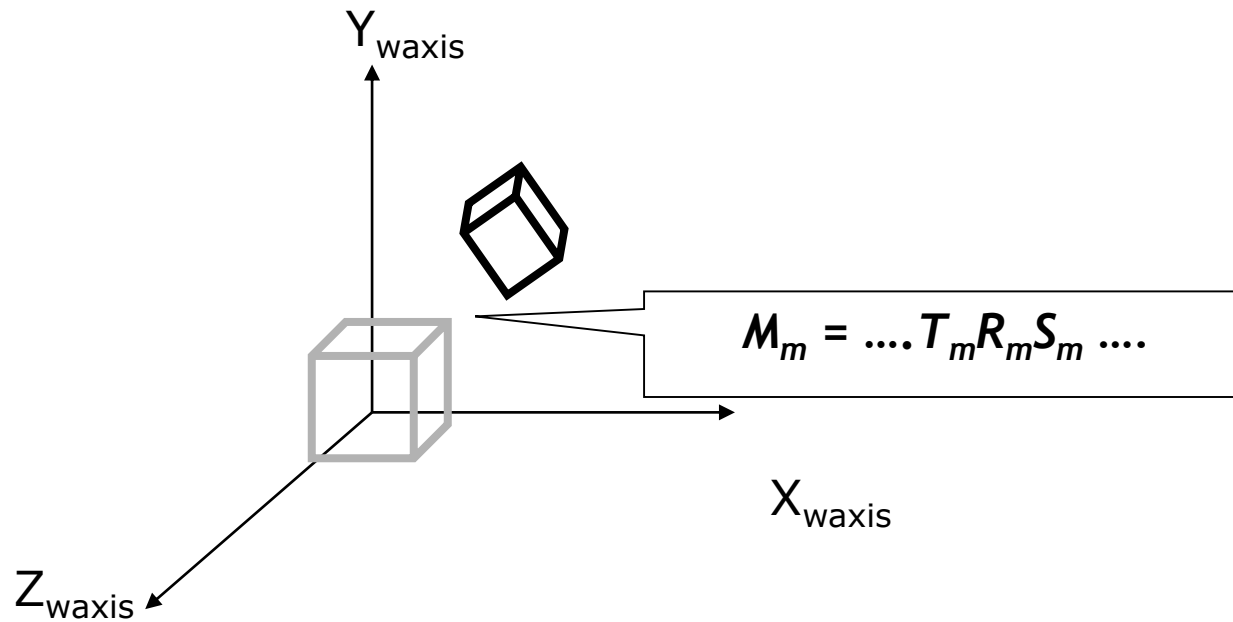
Loading an Object

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



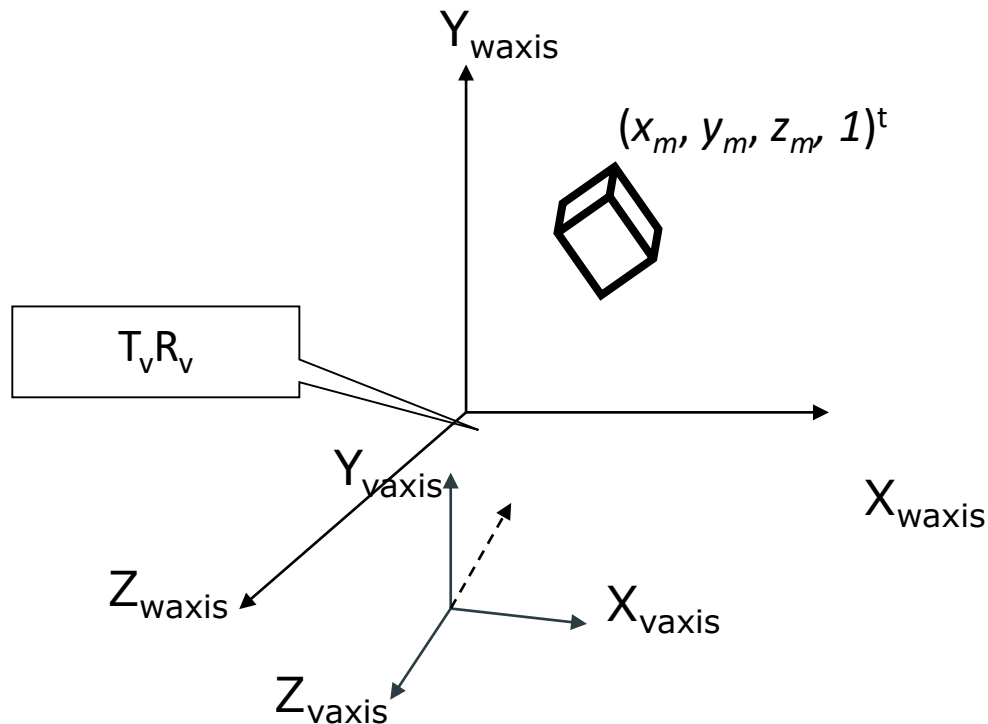
Modeling Transformation

► $(x_m, y_m, z_m, 1)^t = M_m(x_o, y_o, z_o, 1)^t$
where $M_m = \dots T_m R_m S_m \dots$



Put a Virtual Camera

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

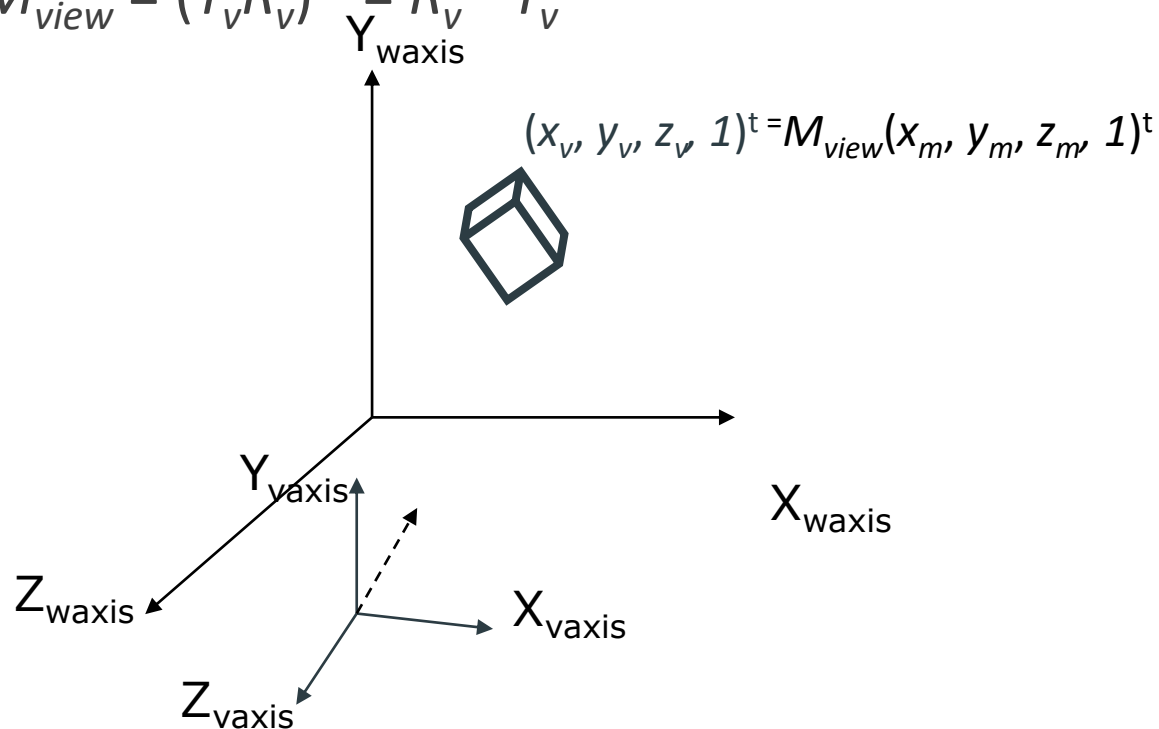


Virtual Camera's Coordinate

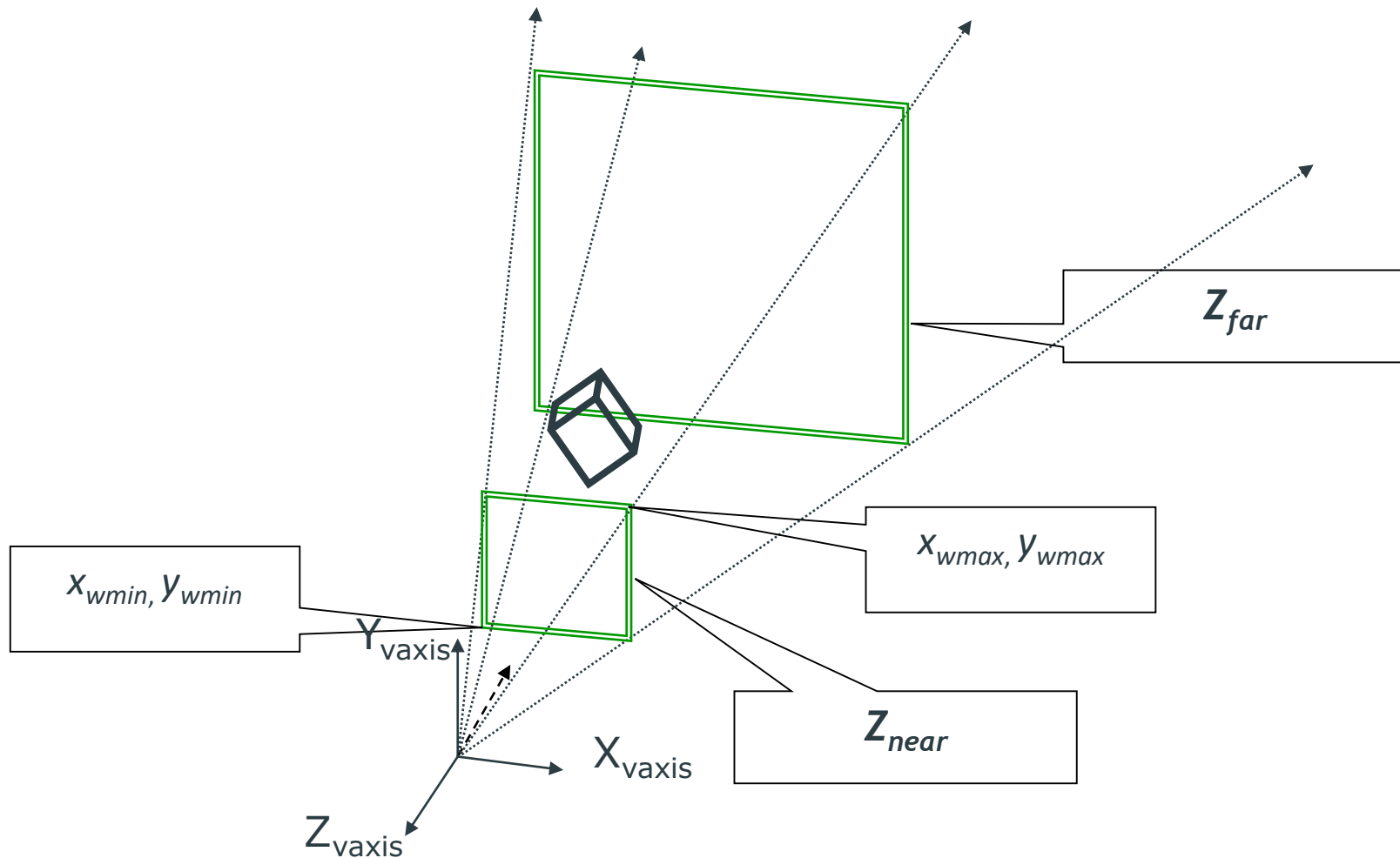
- Change the object's coordinate

- $(x_v, y_v, z_v, 1)^t = M_{view} (x_m, y_m, z_m, 1)^t$

- $M_{view} = (T_v R_v)^{-1} = R_v^{-1} T_v^{-1}$



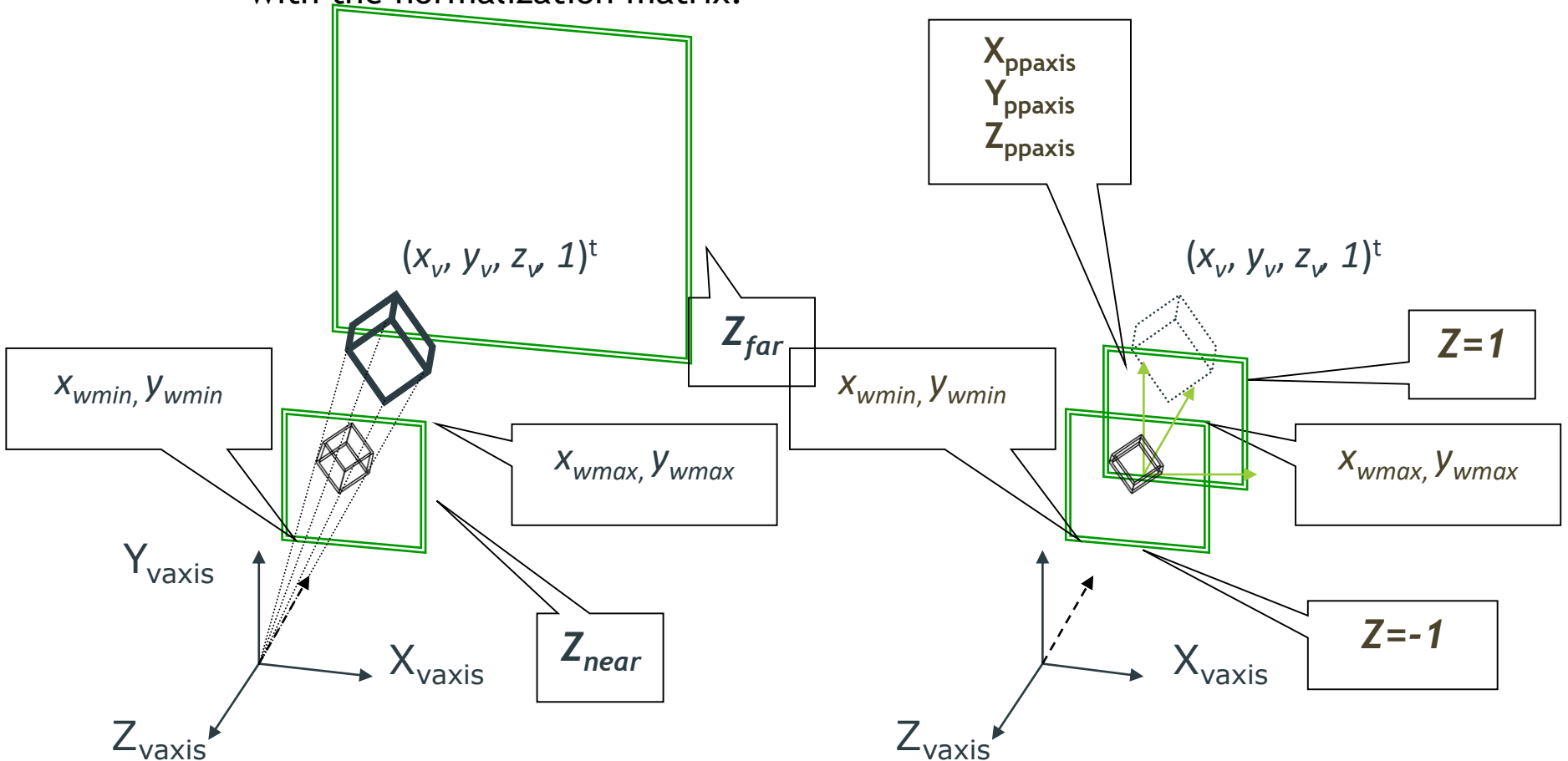
Virtual Camera's Coordinate



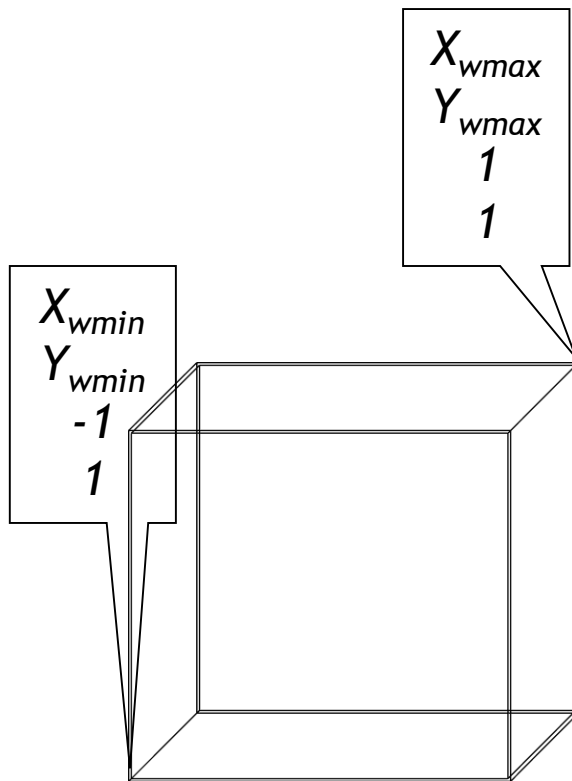
Perspective Proj.

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

This matrix is usually combined with the normalization matrix.

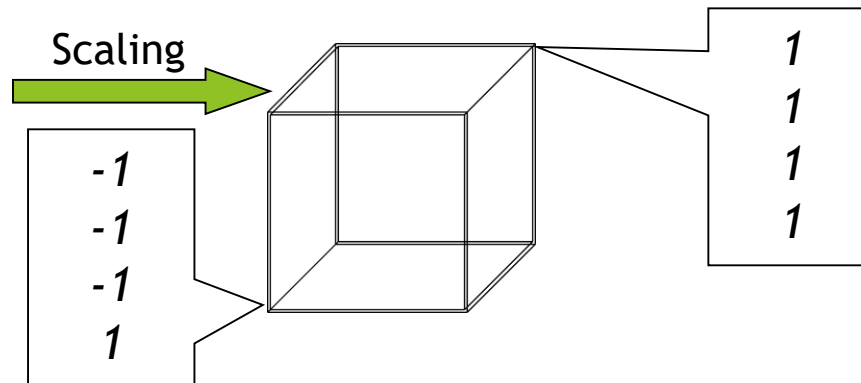


Projection + Normalization



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

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

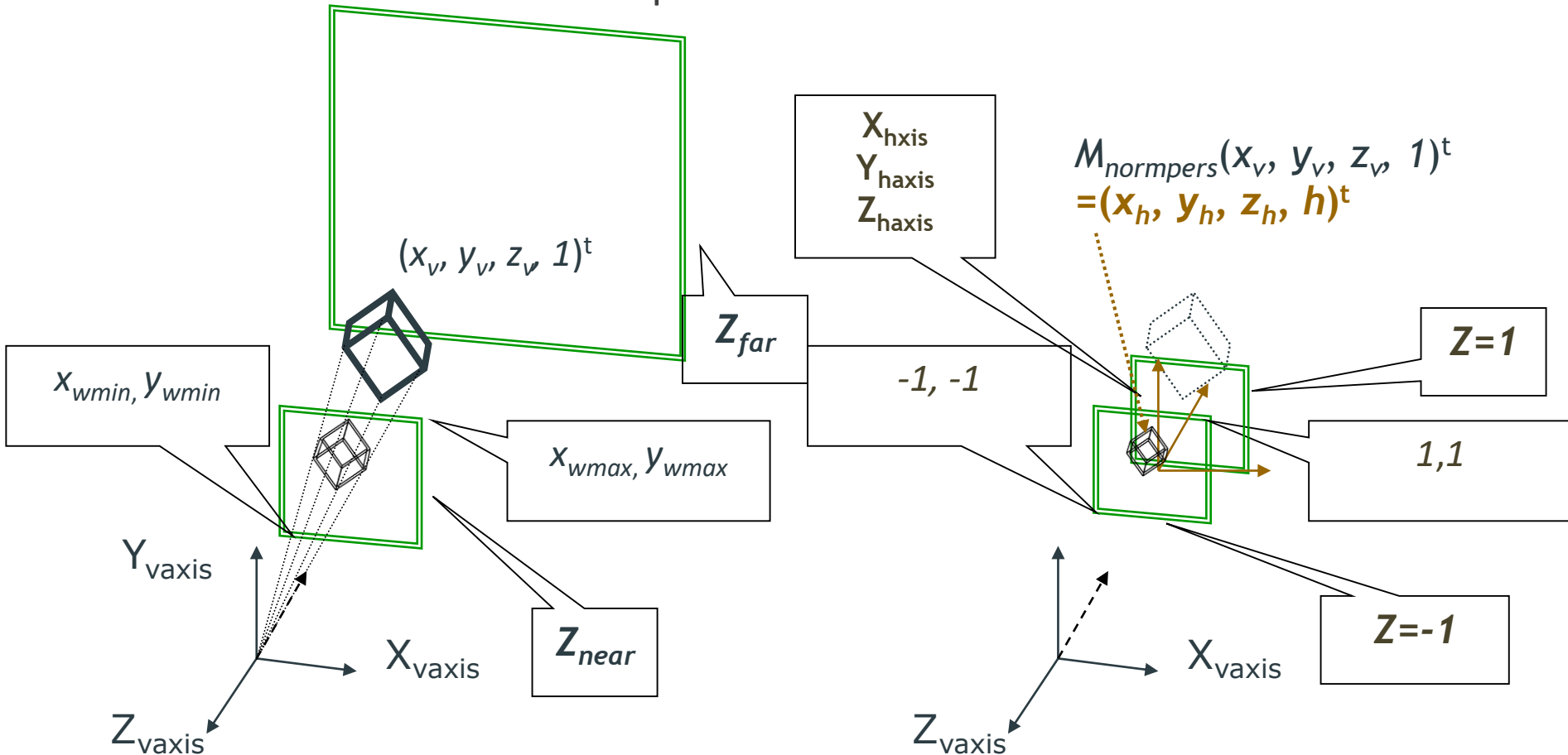


Projection+ Normalization

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

► $(x_h, y_h, z_h, h)^t = M_{normpers}(x_v, y_v, z_v, 1)^t$

► Don't divide h at this step.



Clipping

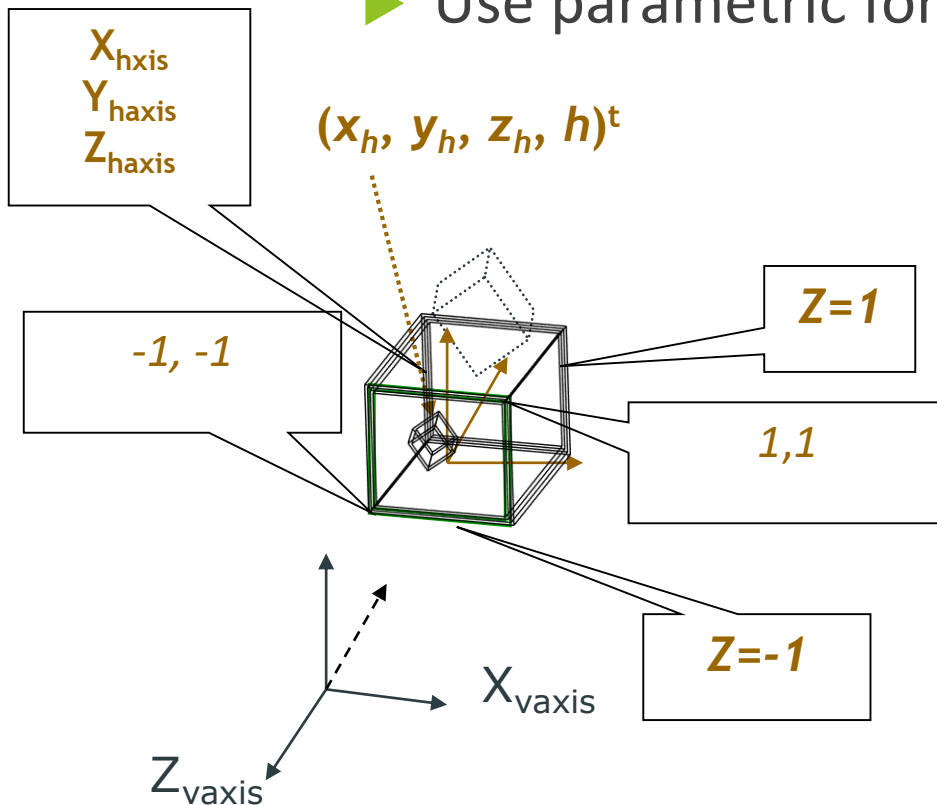
- ▶ Perform clipping with $(x_h, y_h, z_h, h)^t$
- ▶ Avoid unnecessary division $-h \leq x_h \leq h, -h \leq y_h \leq h, -h \leq z_h \leq h$
- ▶ Use parametric forms for intersection

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

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

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

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



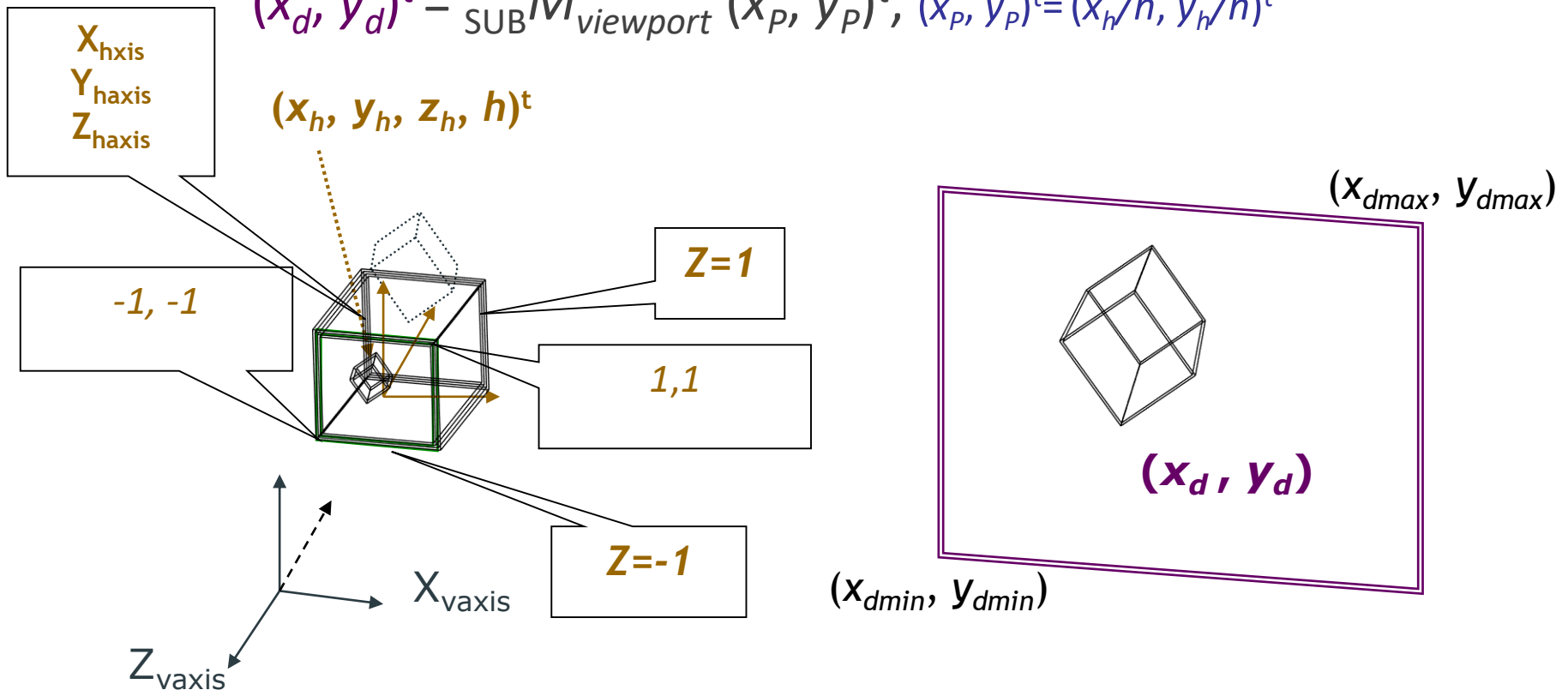
Viewport Transformation

$$M_{viewport} = \begin{bmatrix} \frac{x_{dmax} - x_{dmin}}{2} & 0 & 0 & \frac{x_{dmax} + x_{dmin}}{2} \\ 0 & \frac{y_{dmax} - y_{dmin}}{2} & 0 & \frac{y_{dmax} + y_{dmin}}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

OR

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



Rasterization

- Line drawing or polygon filling with

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

