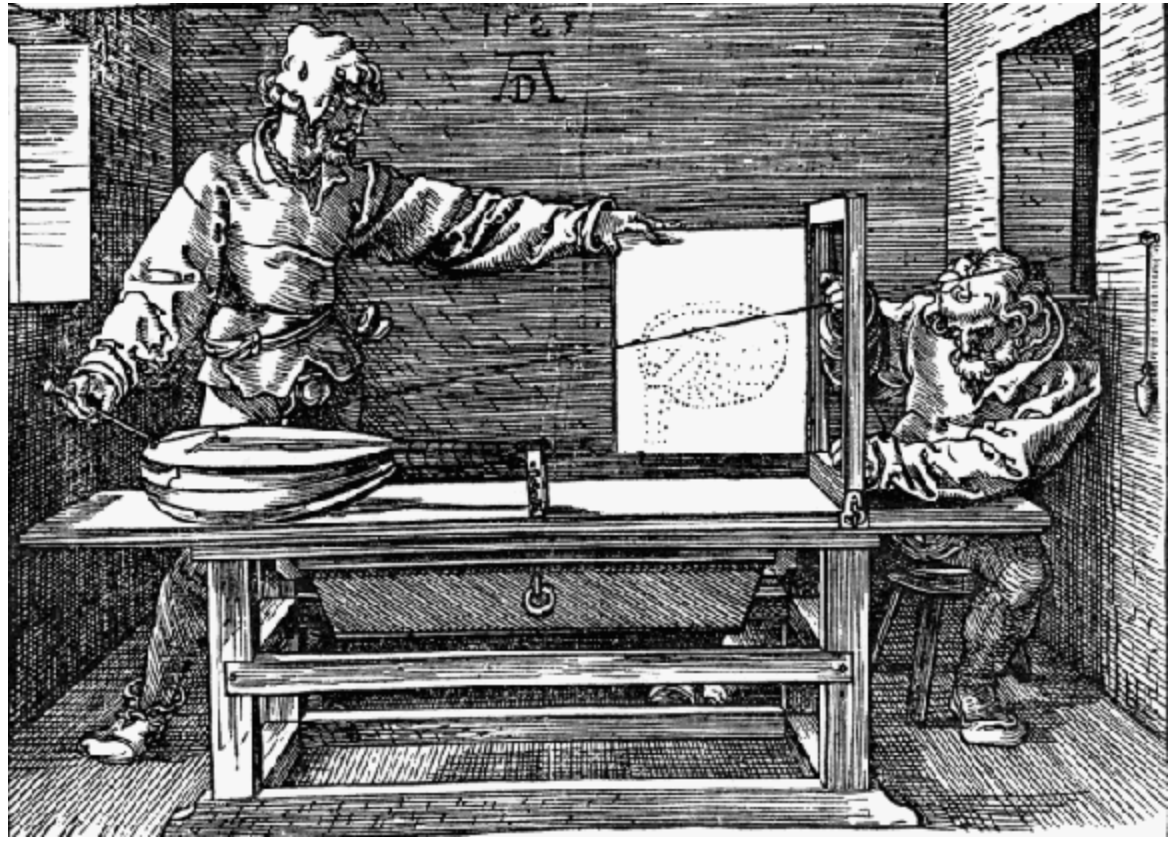
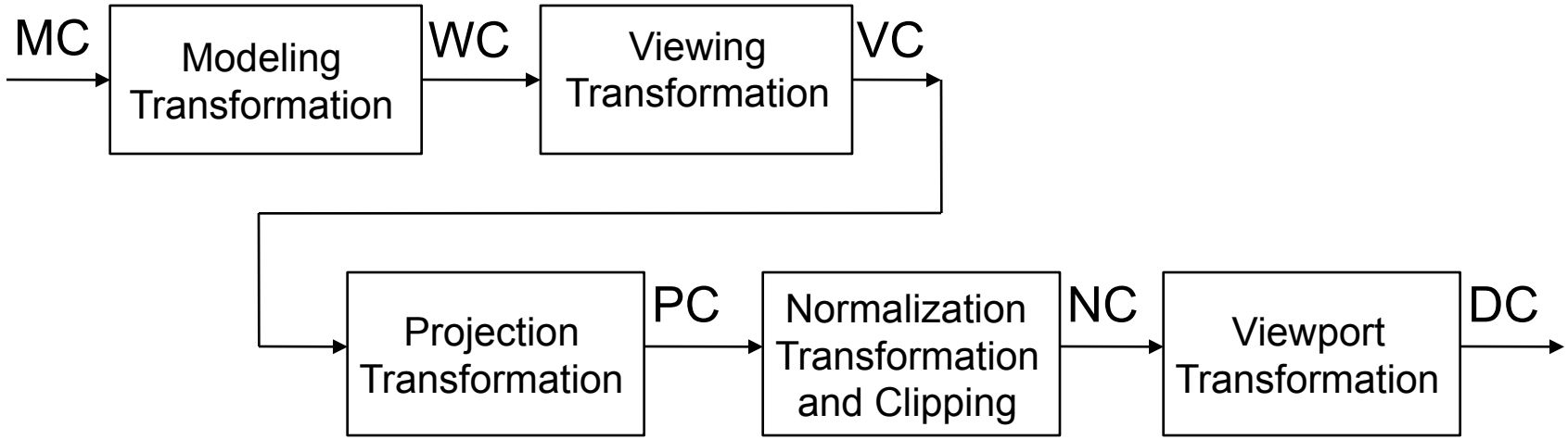
3D Viewing



3D Viewing

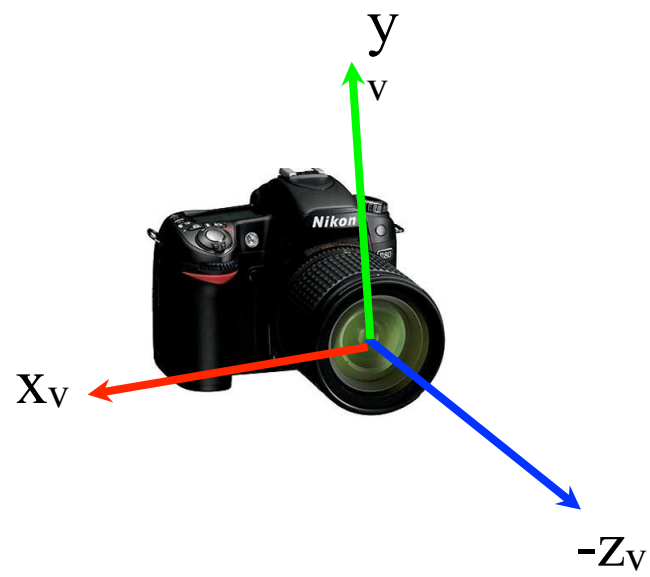
Daniel Keren

3D Viewing Pipeline



- Viewing transformation used to transform to "camera" coordinate
- Projection transformation usually transforms to 3D perspective view volume
- Clipping done in normalized coordinate (3D) for same reason as 2D

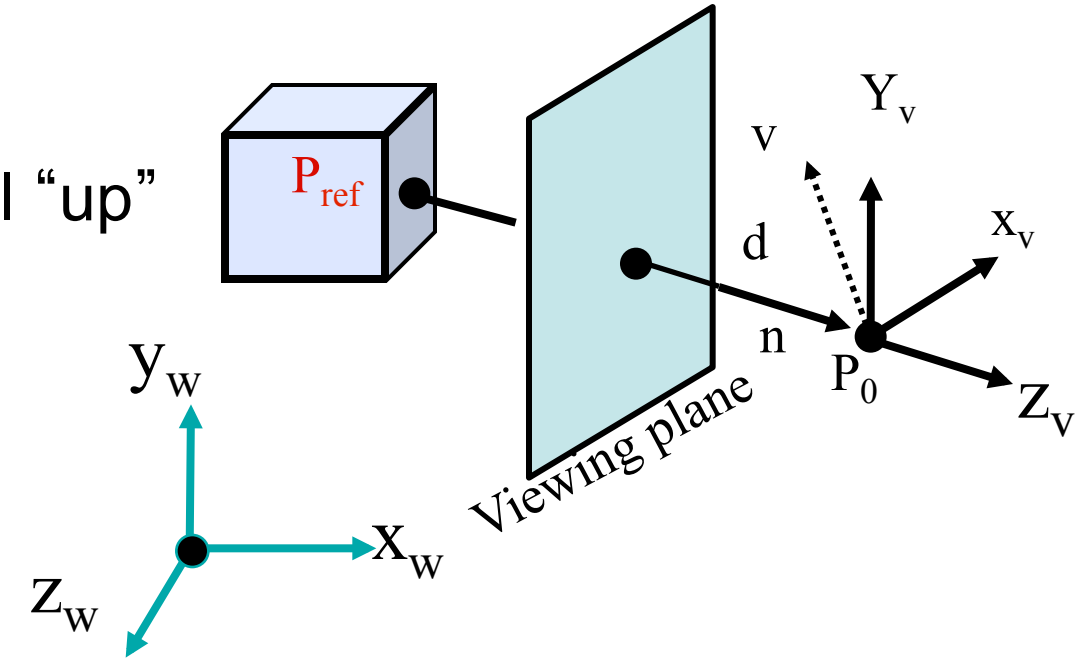
Viewing Coordinates



- Define position and orientation of the camera
- Orientation is represented as three orthogonal vectors - (u,v,n) or (x_v,y_v,z_v)
- The viewer is looking down the -n (or $-z_v$) axis
- This allows the camera coordinate system to be right-handed

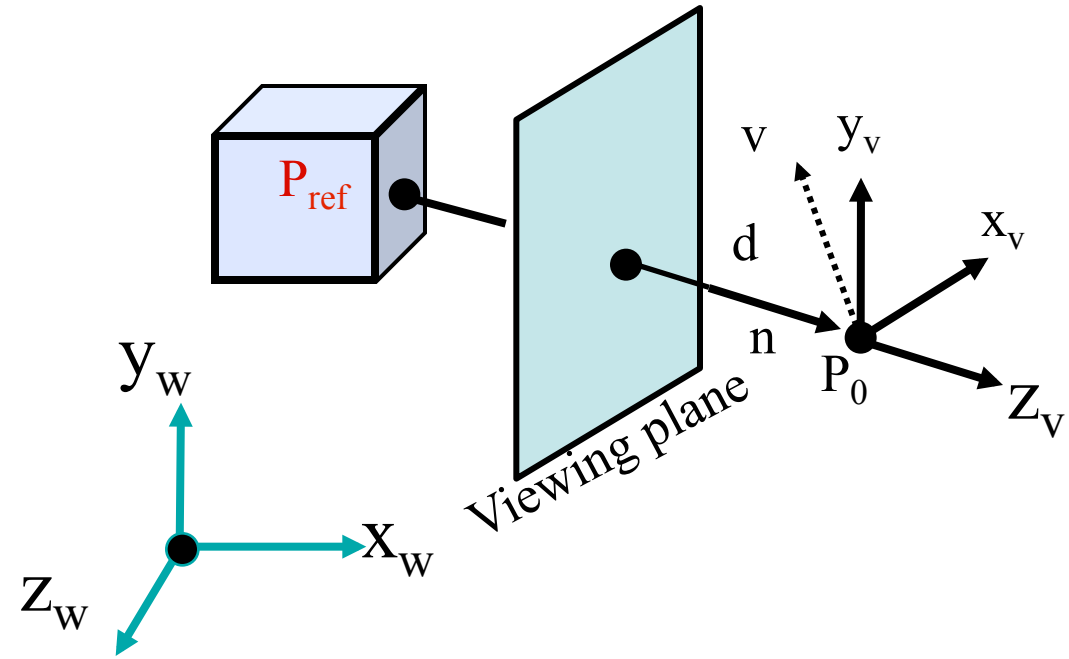
Viewing coordinate system

- View point, also called eye position or camera position: P_0
- P_{ref} gives the "look-at" point and used to define view-plane normal vector n (direction of $-z_v$)
- Viewing plane (or projection plane) perpendicular to z_v
- View-up vector v gives general "up" direction (usually (0,1,0))



View Vectors

$$z_v = n = \frac{P_0 - P_{ref}}{\left|P_0 - P_{ref}\right|}; \quad x_v = u = \frac{v \times z_v}{\left|v \times z_v\right|}; \quad y_v = z_v \times x_v;$$



Viewing Transformation

– Transforms from world to viewing coordinate

- Translate so that viewing coordinate origin is coincident with world coordinate
- Rotate so that the axes coincide
- M transforms points from world coordinates (x,y,z) to camera coordinates (u,v,n)

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

OpenGL Viewing

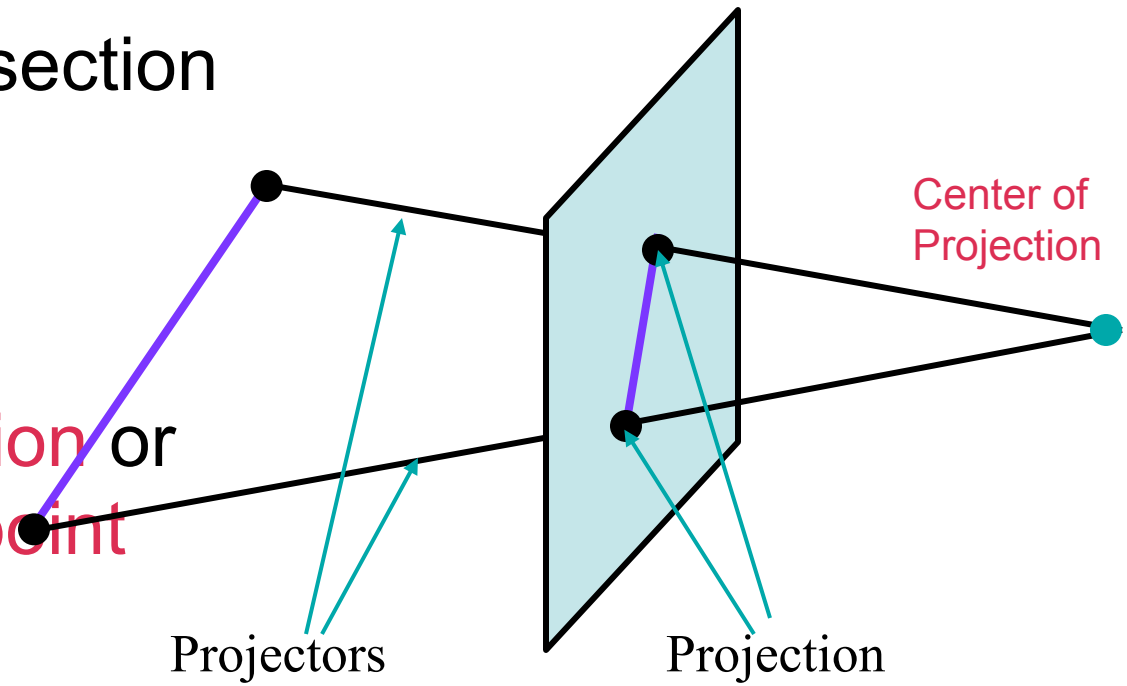
glMatrixMode(GL_MODELVIEW);

gluLookAt(eye_x,eye_y,eye_z, $center_x$,$center_y$,$center_z$, up_x,up_y,up_z);

- eye=> eye position
- center=> center of the thing you are looking at
- up=>approximate up vector

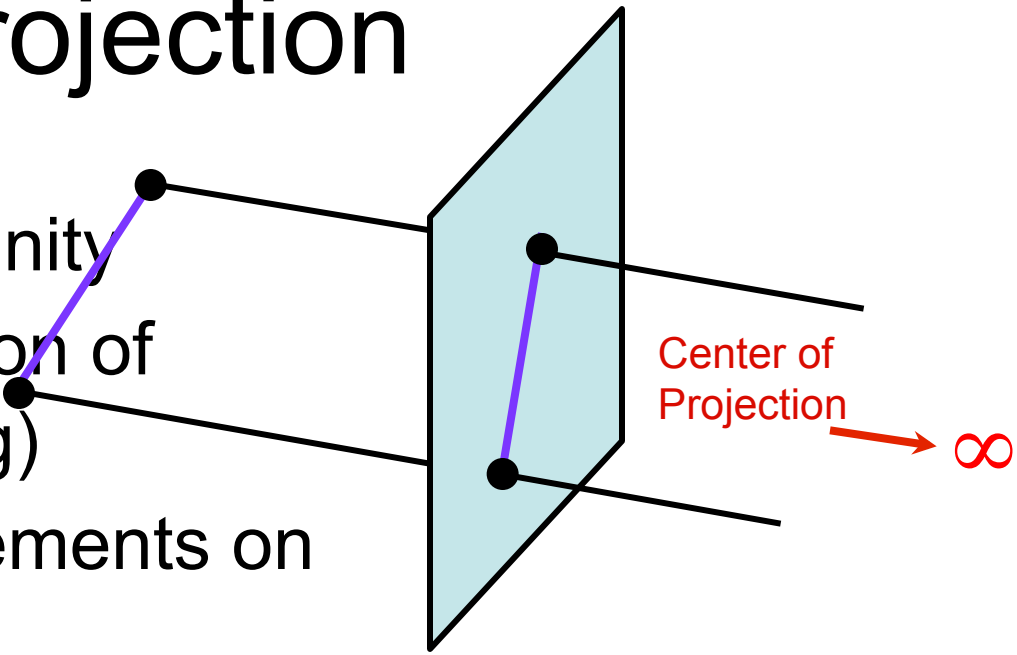
Projections

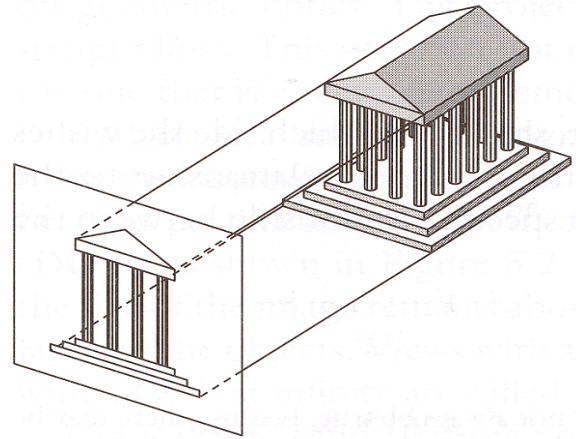
- Going from 3D to 2D requires projection
- Projectors are vectors from center of projection to a point in the 3D world
- Projection is the intersection of a projector with the viewing plane
- Projectors emanate from center of projection or projection reference point



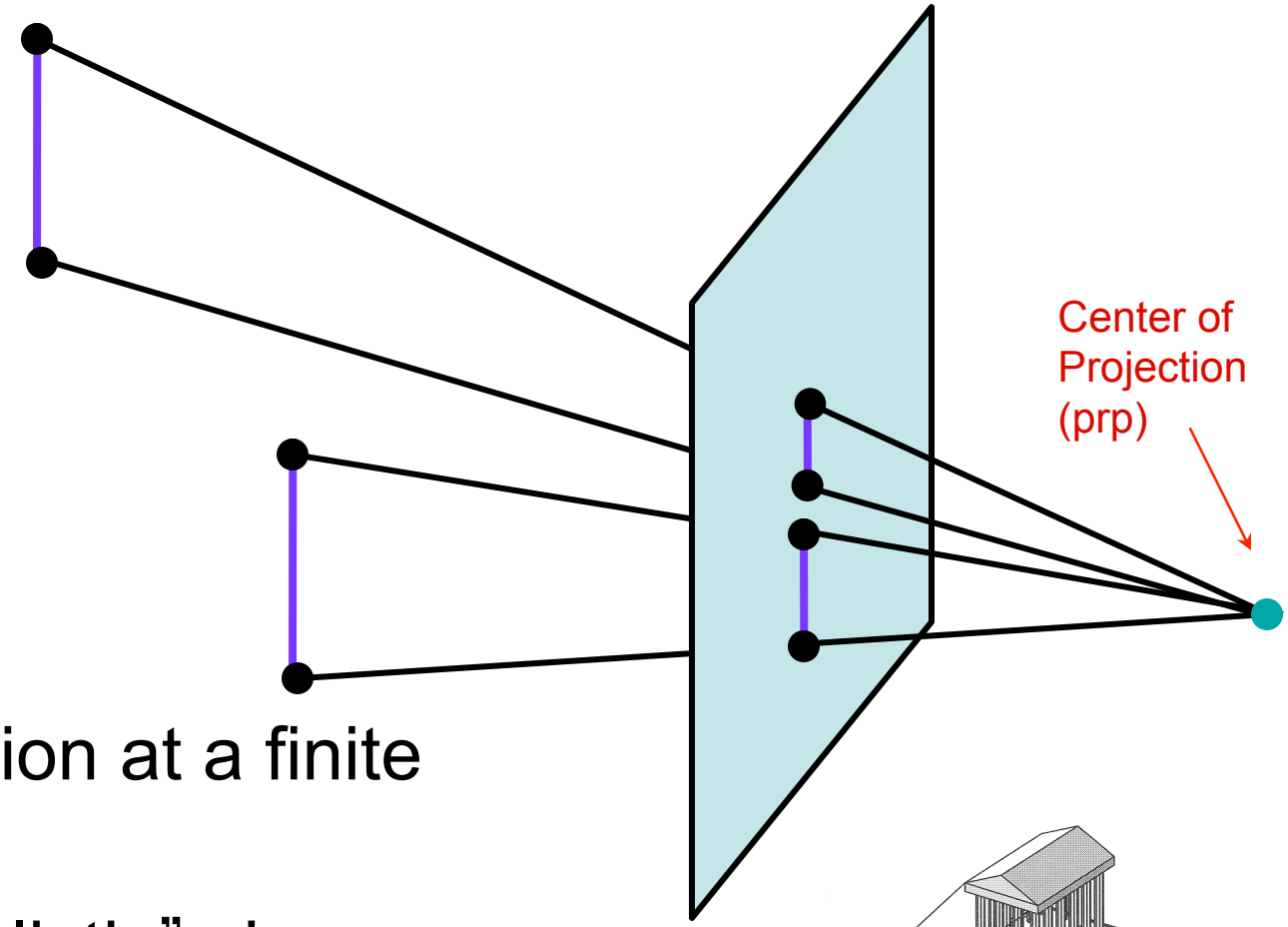
Parallel projection

- Center of projection at infinity
- Preserve relative proportion of objects (no foreshortening)
- Can make direct measurements on lengths
- Used for computer aided drafting and design in architecture and engineering

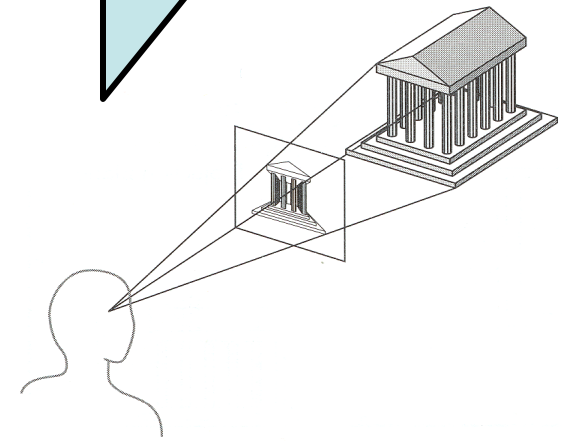




Perspective Projection

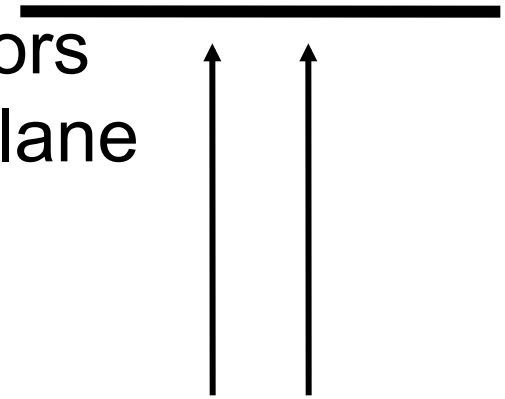


- Center of projection at a finite distance
- Give a more “realistic” view
- Perspective foreshortening

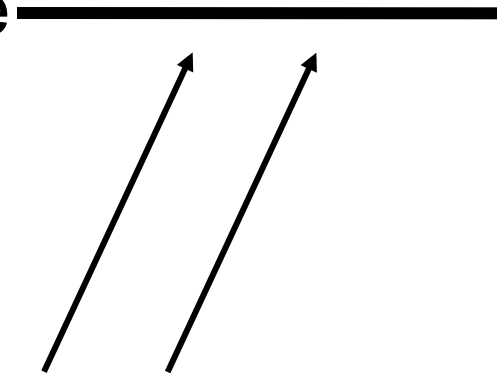


Parallel Projection

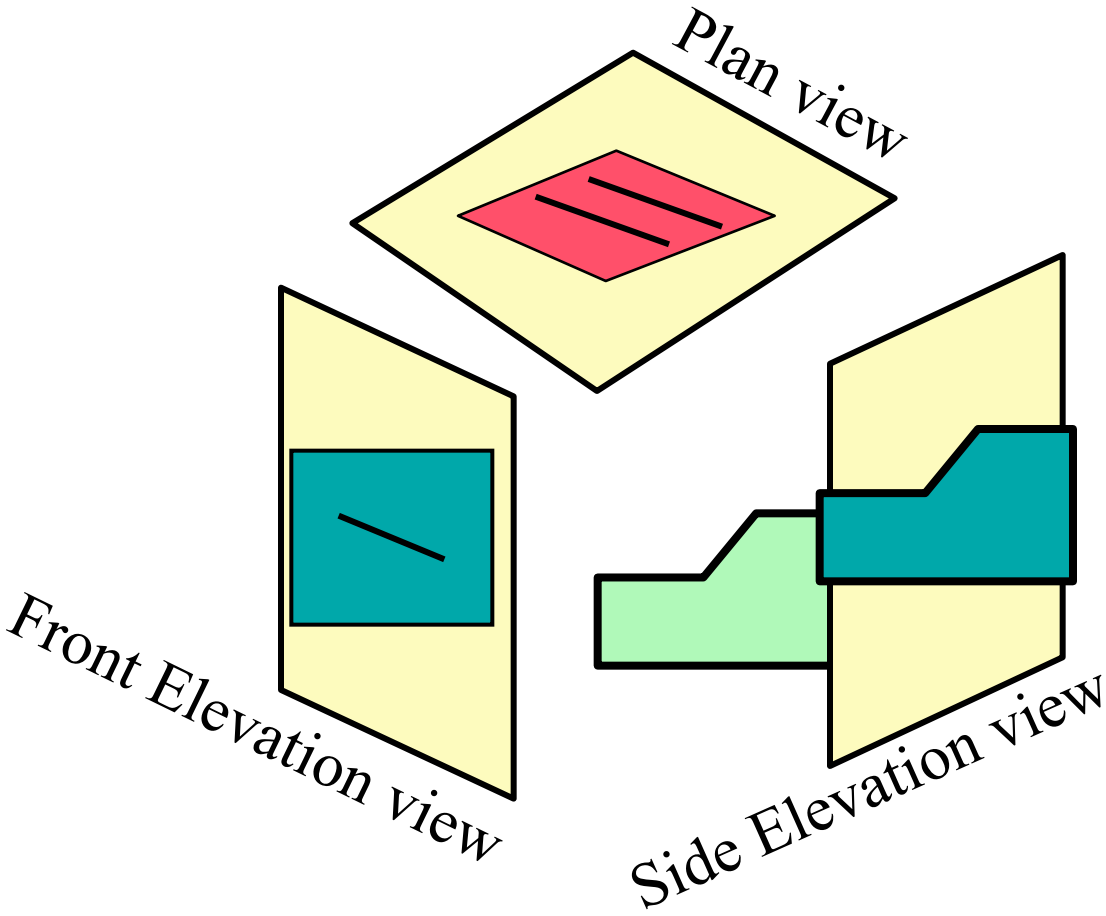
- Orthogonal or Orthographic: Projectors are perpendicular to the projection plane

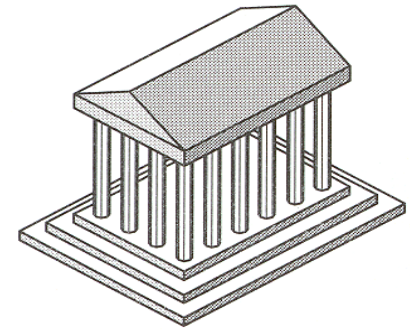


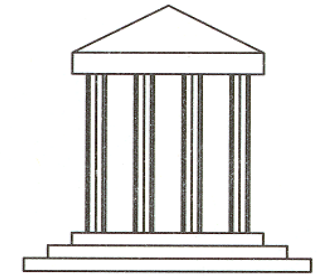
- Oblique: Projectors are not necessarily perpendicular to the projection plane



Orthogonal Projection

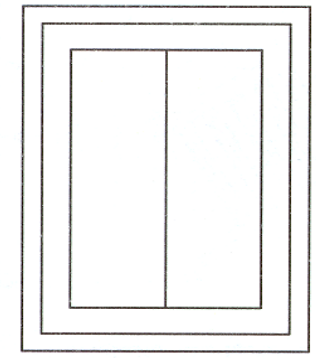




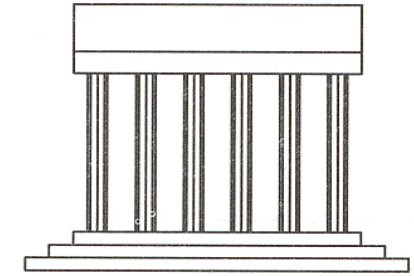


front elevation

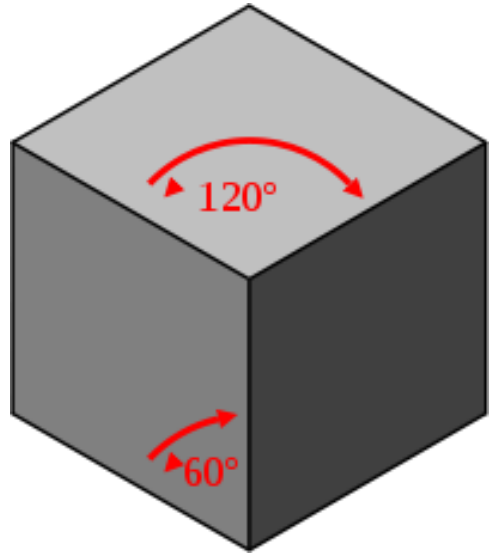
plan view



side elevation



– Axonometric orthogonal projection
  - Displays more than one face of the object
– Isometric projection
  - Projection plane intersects the object coordinate axes at the same distance from the origin
  - All three principle axes foreshortened the same amount



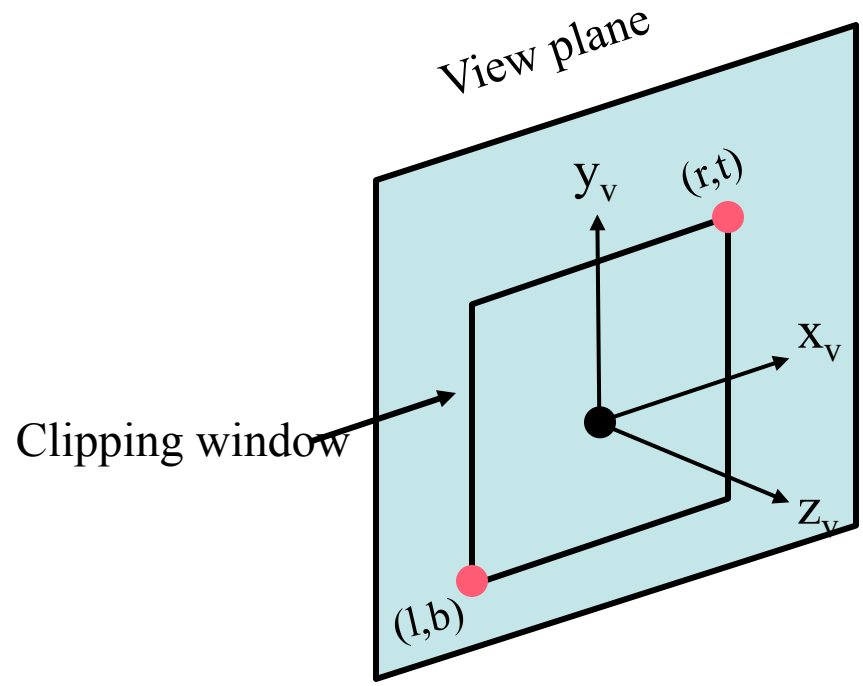
Simple orthogonal projection

– Just project onto the z=0 plane

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

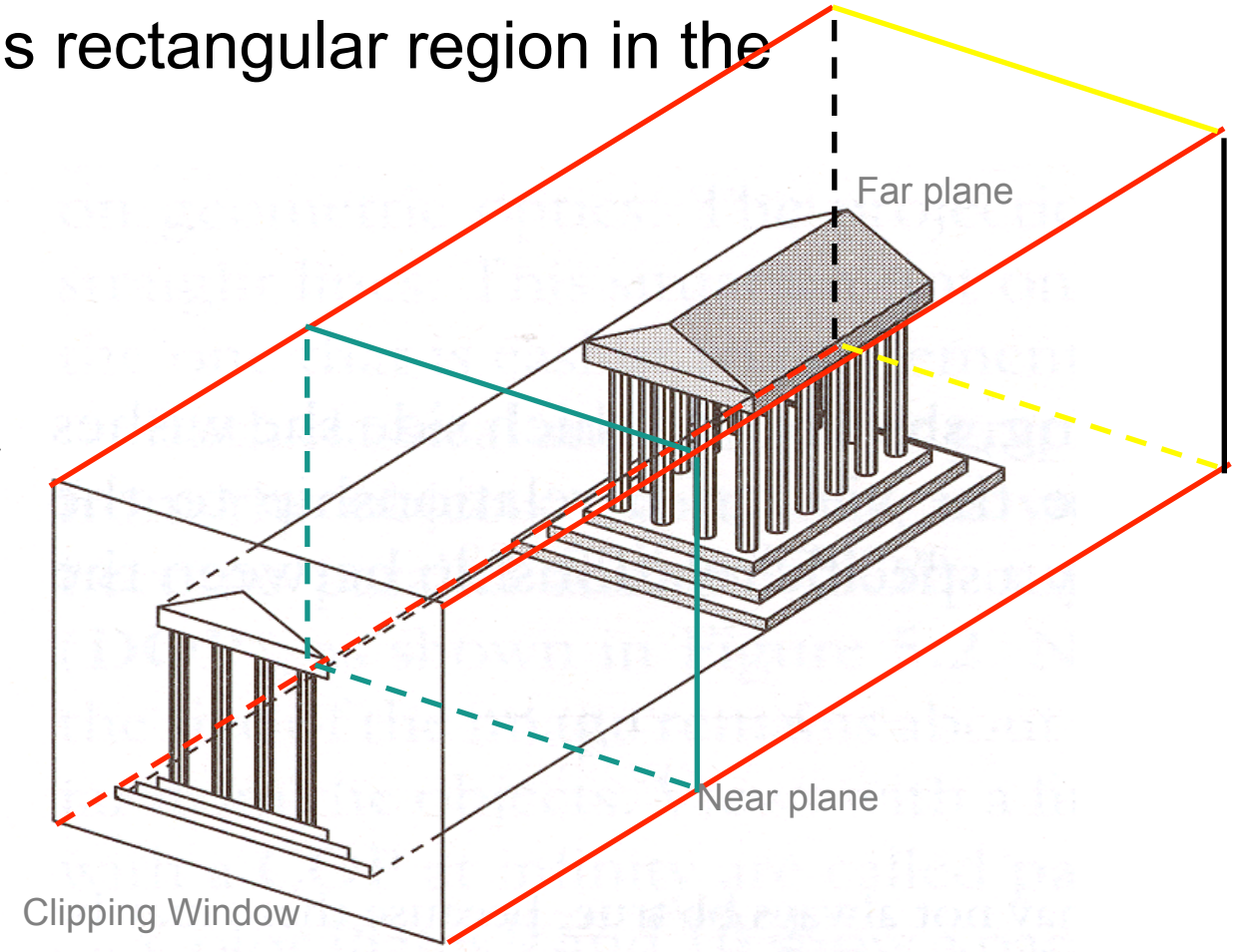
Clipping window

- Defined on the view plane
- Left bottom, right top



Orthogonal Projection View Volume

- Clipping window given as rectangular region in the view plane
- Near (or front) clipping plane and far (or back) clipping planes along with the clipping window defines view volume
- View volume is rectangular parallelepiped region
- Anything inside visible

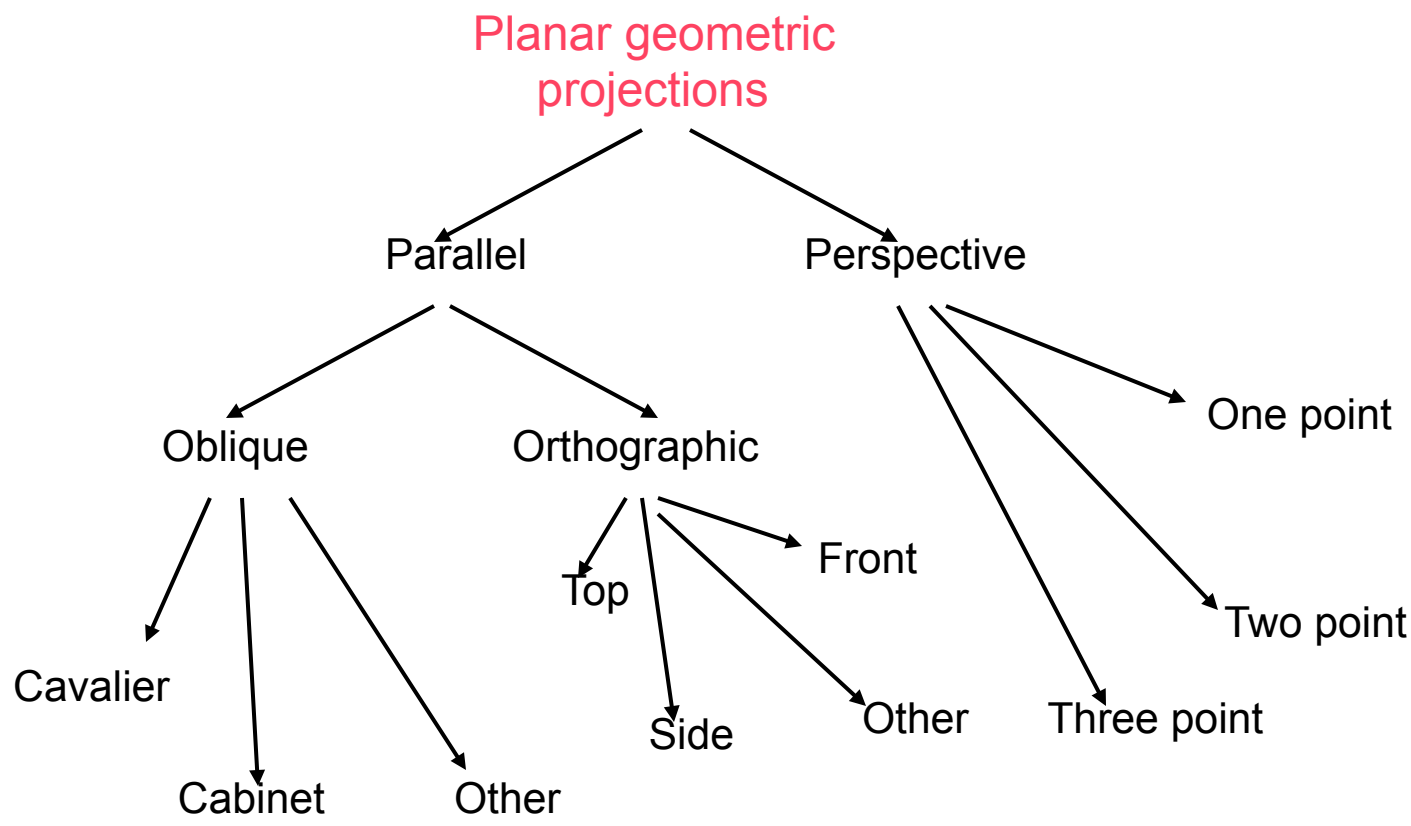


Normalized Orthogonal Projection View Volume

- Like 2D normalized view volume
- Device independent, simplifies clipping
- Unit cube (0,0,0)..(1,1,1) or (-1,-1,-1)..(1,1,1)
- Often left-handed coordinate system
  - y goes up, x to the right, z into the world
- Similar to 2D case but -z to make left-handed:

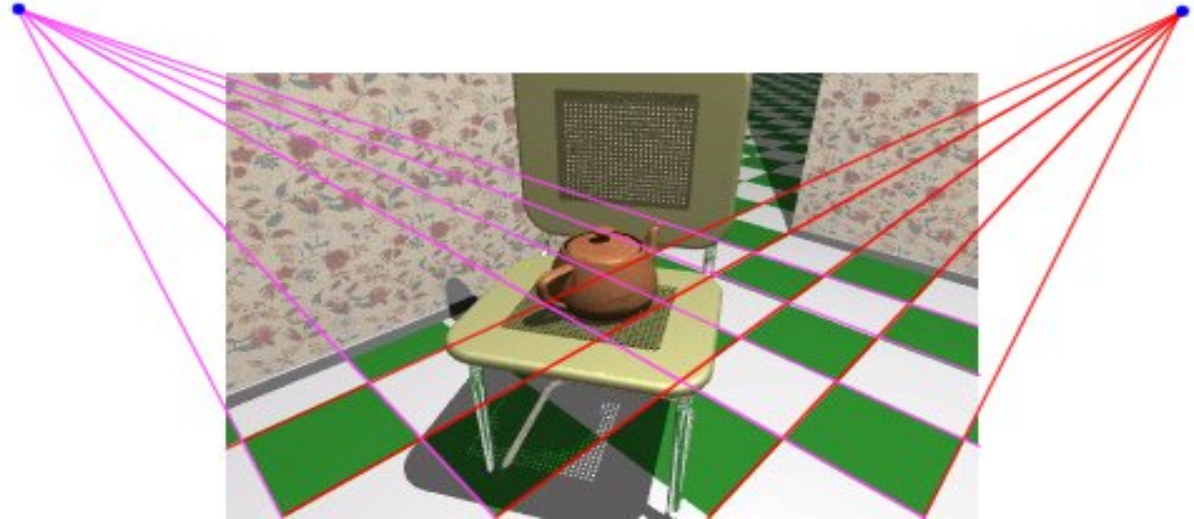
$$\begin{bmatrix} \frac{2}{xw_{max} - xw_{min}} & 0 & 0 & 0 \\ 0 & \frac{2}{yw_{max} - yw_{min}} & 0 & 0 \\ 0 & 0 & \frac{-2}{zw_{max} - zw_{min}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\frac{xw_{max} + xw_{min}}{2} \\ 0 & 1 & 0 & -\frac{yw_{max} + yw_{min}}{2} \\ 0 & 0 & 1 & -\frac{zw_{max} + zw_{min}}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Taxonomy of Planar Geometric Projections



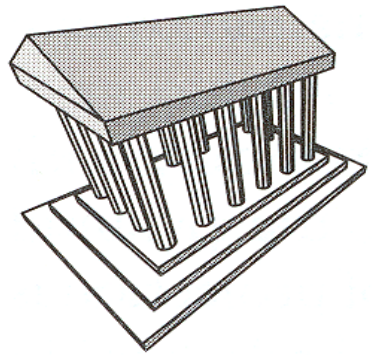
Perspective Projection

- Center of projection is finite distance from view plane
- Size of object inversely proportional to distance to center of projection
- Parallel lines not perpendicular to image plane converge to a “vanishing point”

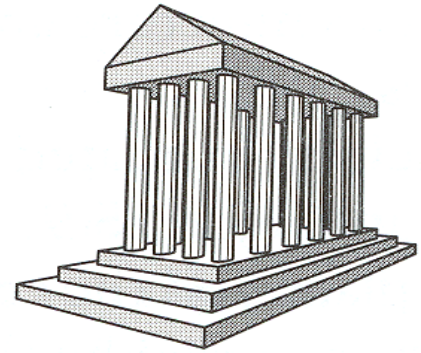


Number of vanishing points

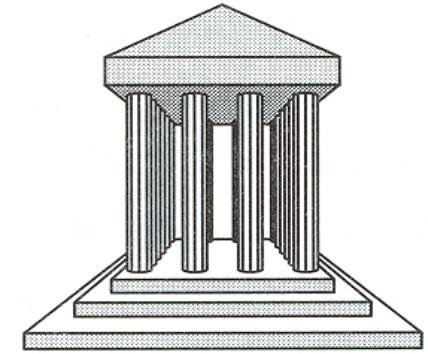
- In general, infinite number of vanishing points but only up to three principal vanishing points corresponding to the three principal axes
  - Principal axes corresponds to the axes used to define the object
  - For architectural scenes, they align with walls
- Number of principal vanishing points equal to the number of principal axes intersected by view plane



3-Point
Perspective



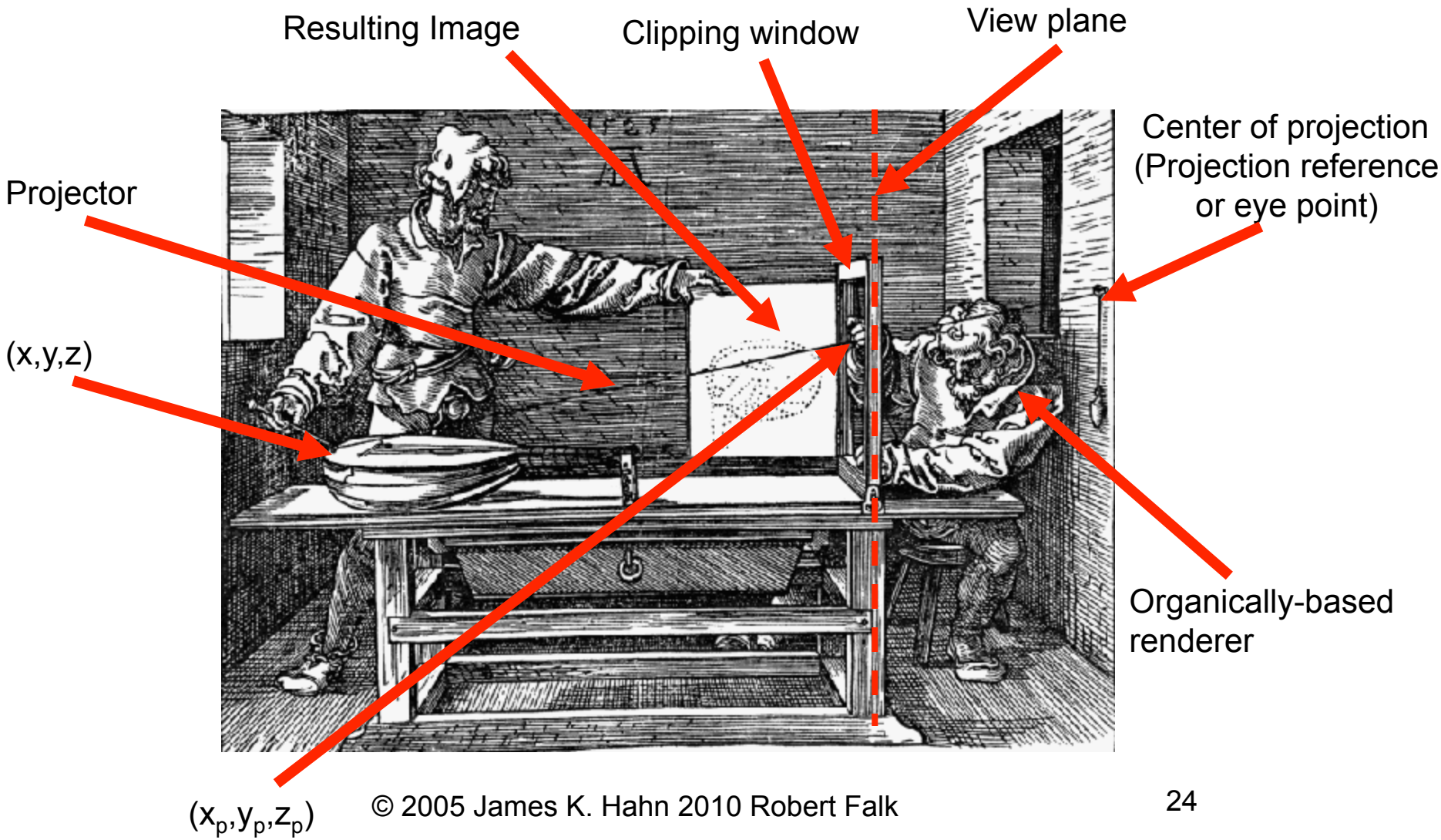
2-Point
Perspective



1-Point
Perspective

One point perspective

(Oblique) Perspective projection transformation

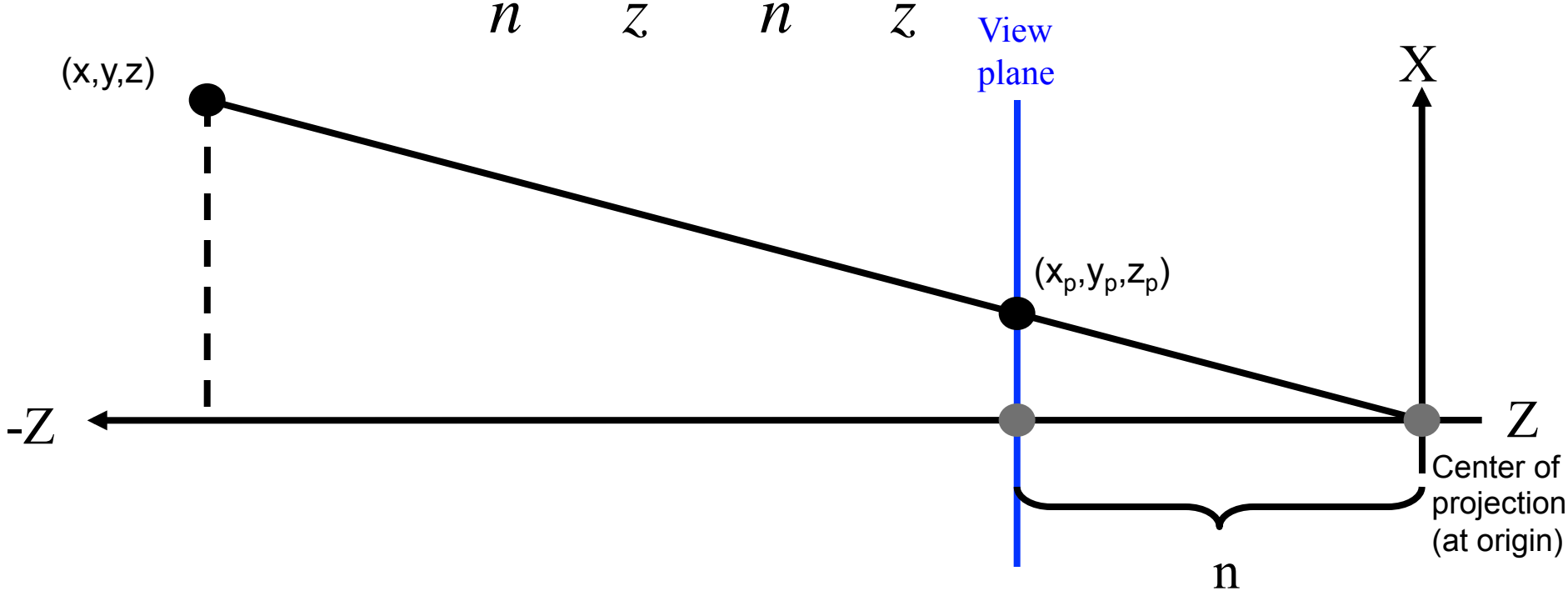


Computer version

- Camera at origin of viewing coordinate system defined by center of projection (or projection reference point)
- Looking along negative z axis toward the center of interest (or look-at point)
- Up vector aligned with the y axis
- Clipping window defined on the view plane

– Similar triangles:

$$\frac{x_p}{n} = \frac{x}{z}, \quad \frac{y_p}{n} = \frac{y}{z}$$



$$x_p = \frac{nx}{z}, \quad y_p = \frac{ny}{z}, \quad z = ?$$

– Corresponding matrix

$$M_{pers} = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & s_z & t_z \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} nx & ny & zs_z + t_z & -z \end{bmatrix}$$

– homogeneous coordinate will perform perspective divide

– but we still have to map z to -1..1

- Need to solve for s_z and t_z that preserve z and map it to (-1..1)
- And we know that at z-near z'=-1 and at z-far, z'=1

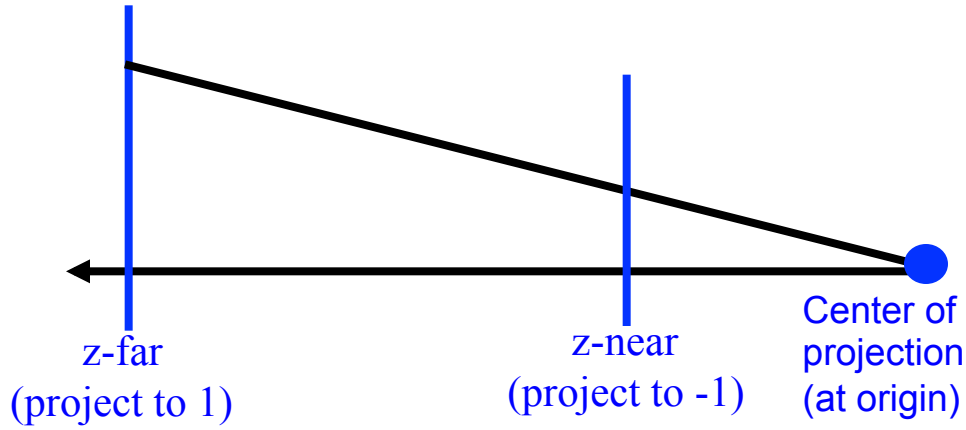
$$M_{pers} = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & s_z & t_z \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} nx & ny & zs_z + t_z & -z \end{bmatrix}$$

$$z' = \frac{zs_z + t_z}{-z}$$

$$z' = -1,\ z = -n: \quad -1 = \frac{-ns_z + t_z}{n}$$

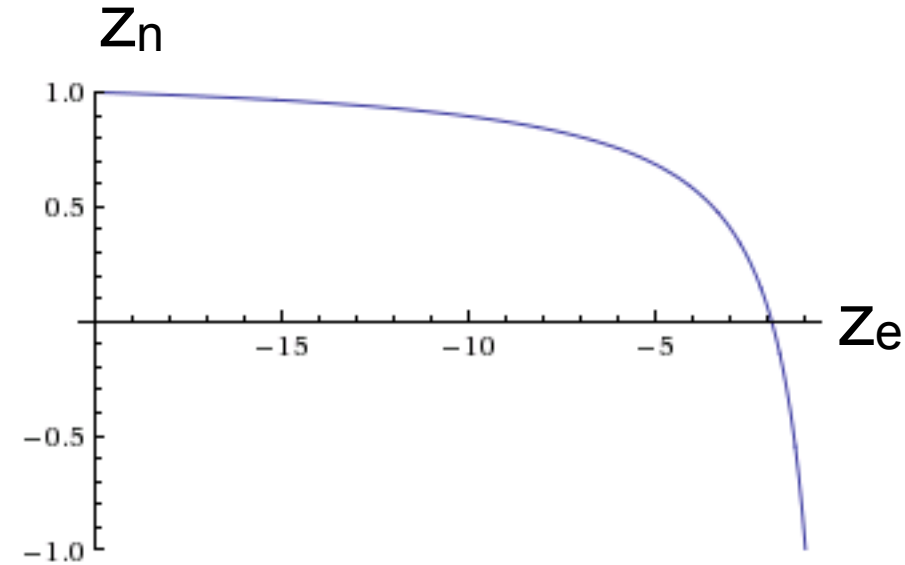
$$z' = 1, \quad z = -f: \quad 1 = \frac{-fs_z + t_z}{f}$$

$$t_z = \frac{2nf}{n-f}, \quad s_z = \frac{n+f}{n-f}$$



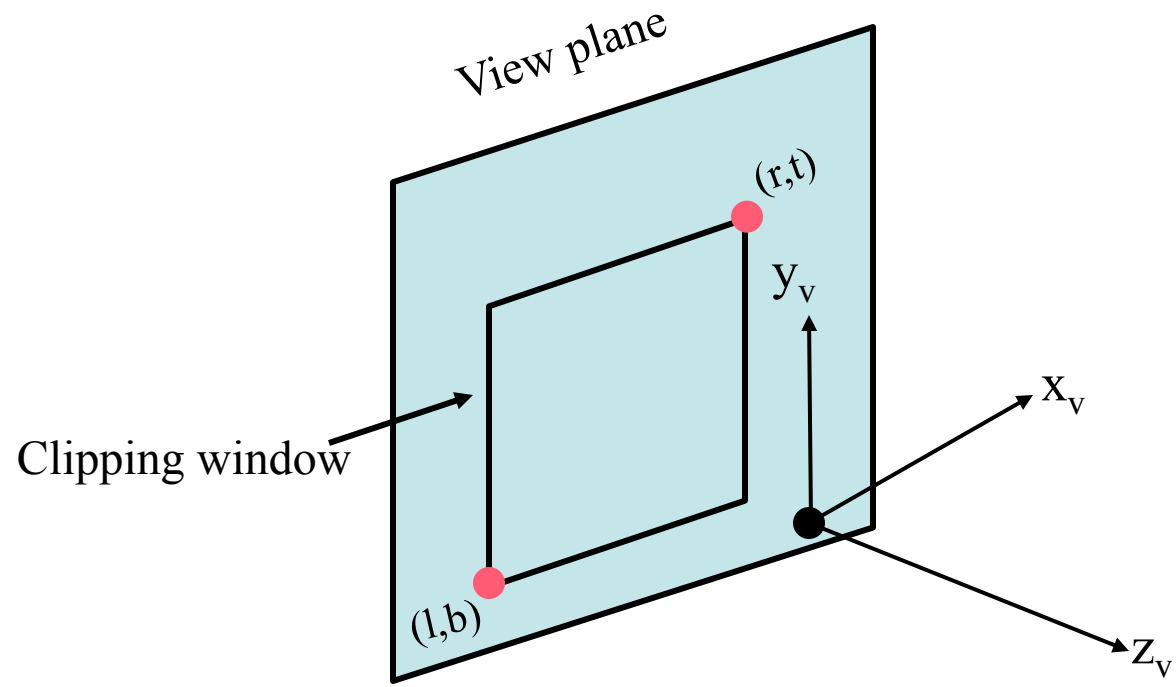
$$M_{pers} = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{2nf}{n-f} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ z_e \\ 1 \end{bmatrix} = \begin{bmatrix} x_e n & y_e n & \frac{z_e(n+f)}{n-f} + \frac{2nf}{(n-f)} & -z_e \end{bmatrix}$$

$$\left(\frac{x_e n}{-z_e}, \frac{y_e n}{-z_e}, \frac{f+n}{f-n} + \frac{2fn}{z_e(f-n)} \right)$$



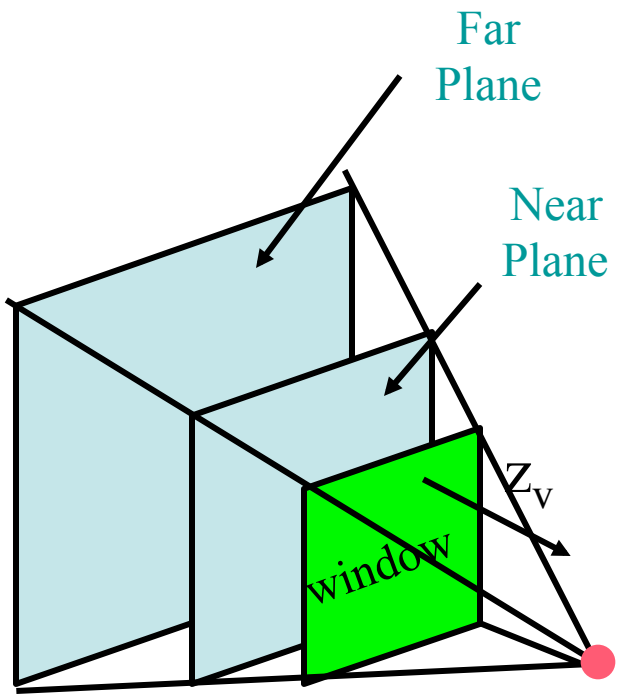
Clipping Window

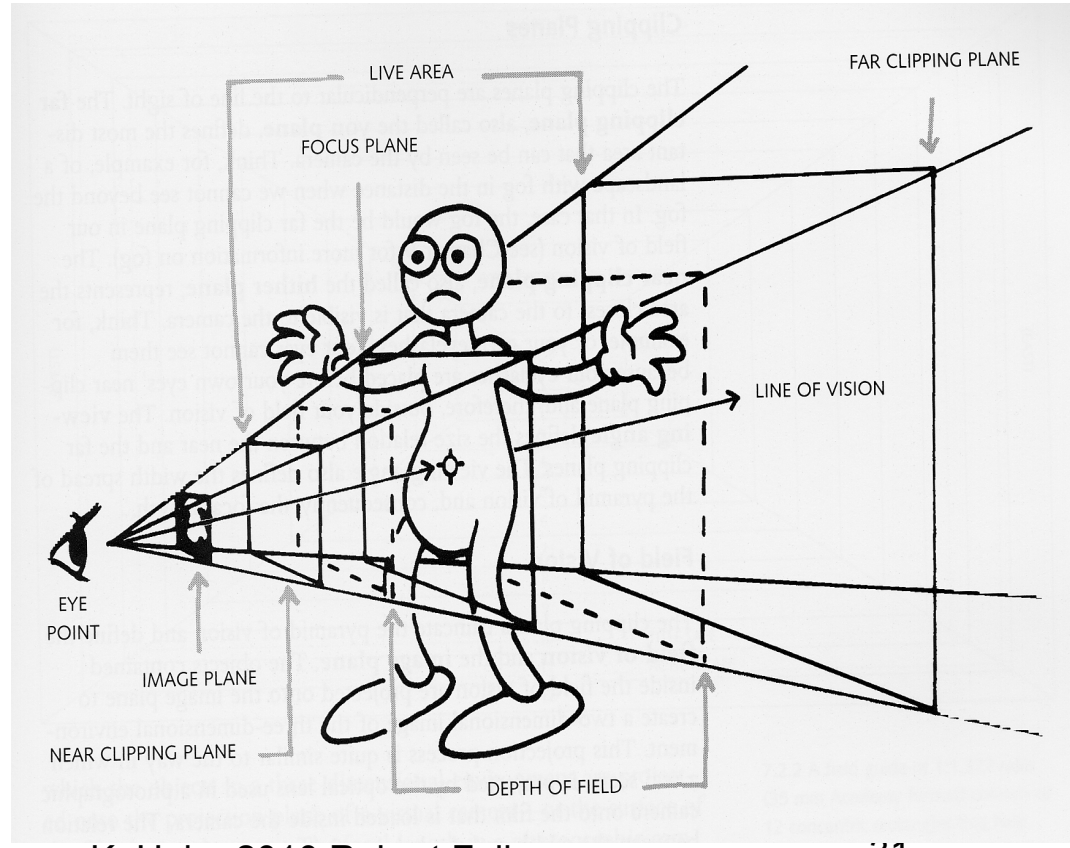
– Defined in view plane



View Volume

- View frustum defined by far and near clipping planes, clipping window, plus center of projection
- What is inside the view volume is seen and projected





OpenGL Viewing

- glFrustum: Specify left, right, top, bottom, near, far
- Viewing position at origin, near and far are positive values
- left, right, top, and bottom are dimensions of near clipping plane

Scale to normalized device coords: (-1..1), (-1..1), (-1..1):

$$s_x = \frac{2}{r-l}$$

$$s_y = \frac{2}{t-b}$$

If right != -left and/or top != -bottom we have shearing:

$$\begin{bmatrix} 0 \\ 0 \\ n \\ 1 \end{bmatrix} = \mathbf{M}_{shear} \cdot \begin{bmatrix} \frac{l+r}{2} \\ \frac{b+t}{2} \\ n \\ 1 \end{bmatrix}$$

$$sh_{zx} = -\frac{l+r}{2}$$

$$sh_{zy} = -\frac{t+b}{2}$$

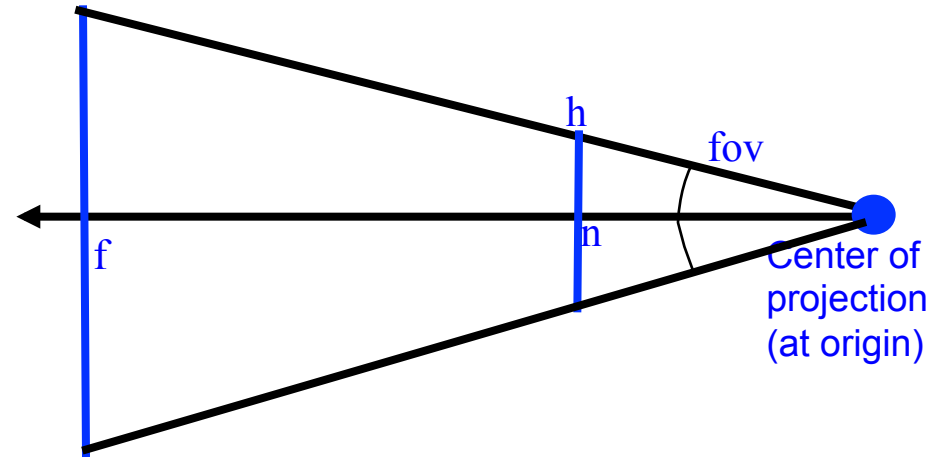
glFrustum

Multiply projection matrix by scaling and shearing in x and y to get:

glFrustum(l,r,t,b,n,f) =

$$M_{pers} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{2nf}{n-f} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

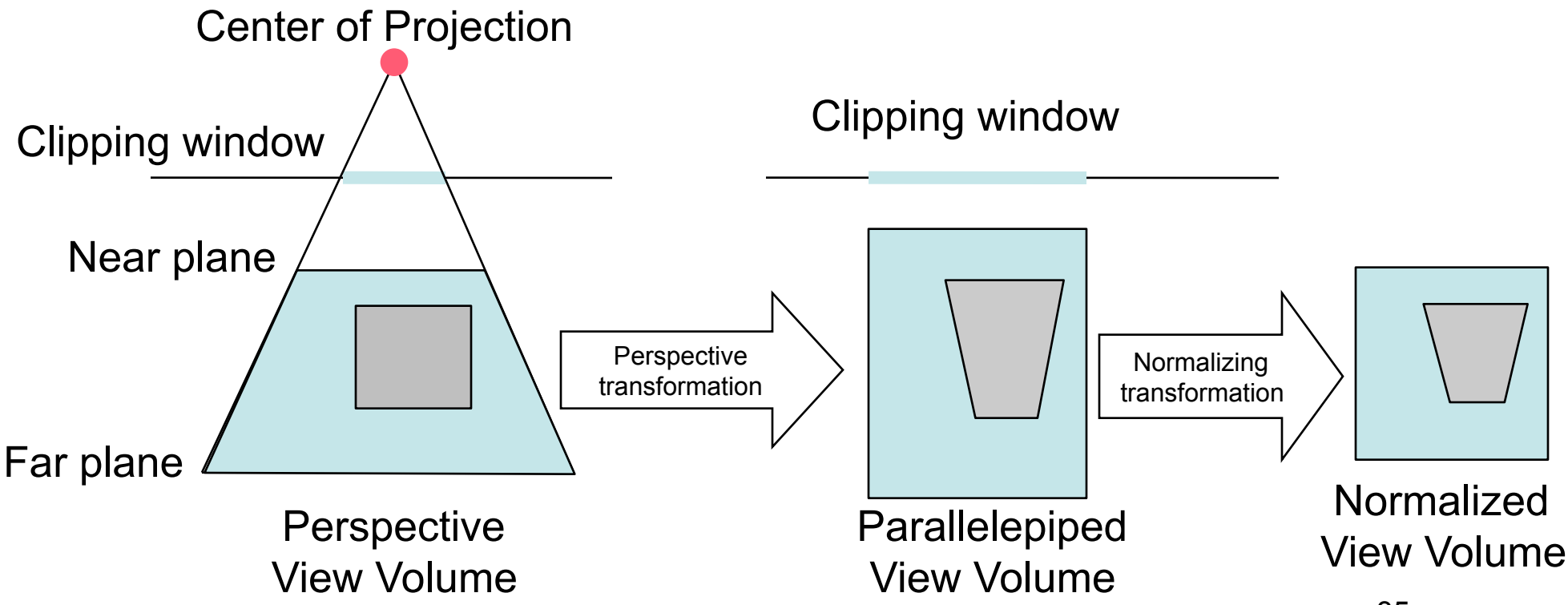
- gluPerspective(fov, aspect, n,f);
- aspect = width/height
- field of view and aspect ratio determine size of near clipping window



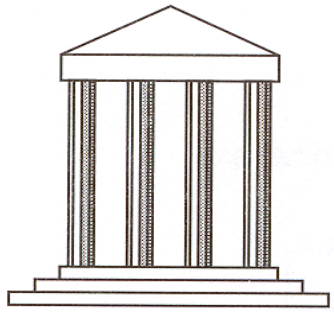
$$M_{pers} = \begin{bmatrix} \frac{t}{aspect} & 0 & 0 & 0 \\ 0 & t & 0 & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{2nf}{n-f} \\ 0 & 0 & -1 & 0 \end{bmatrix}, \quad t = \cot\left(\frac{fov}{2}\right) = \frac{n}{\left(\frac{h}{2}\right)} = \frac{2n}{h} = \frac{2n}{t-b}$$

Effect of perspective projection transformation on view volume

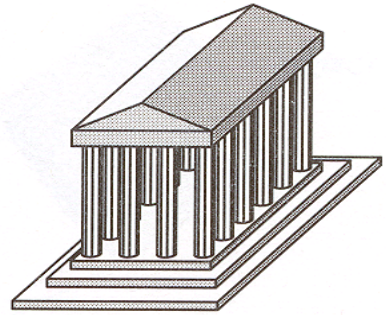
– M_{pers} takes the perspective view volume and converts into the parallel view volume



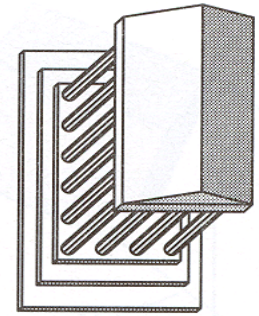
Classic projections



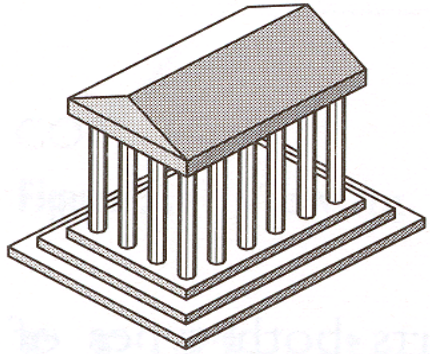
Front elevation



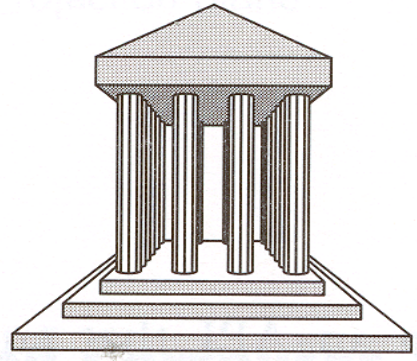
Elevation oblique



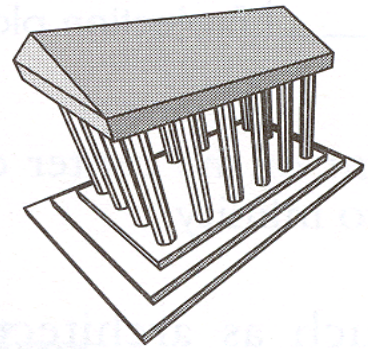
Plan oblique



Isometric



One-point perspective



Three-point perspective

Next: Illumination Models