3D Viewing

CS 465 Lecture 10

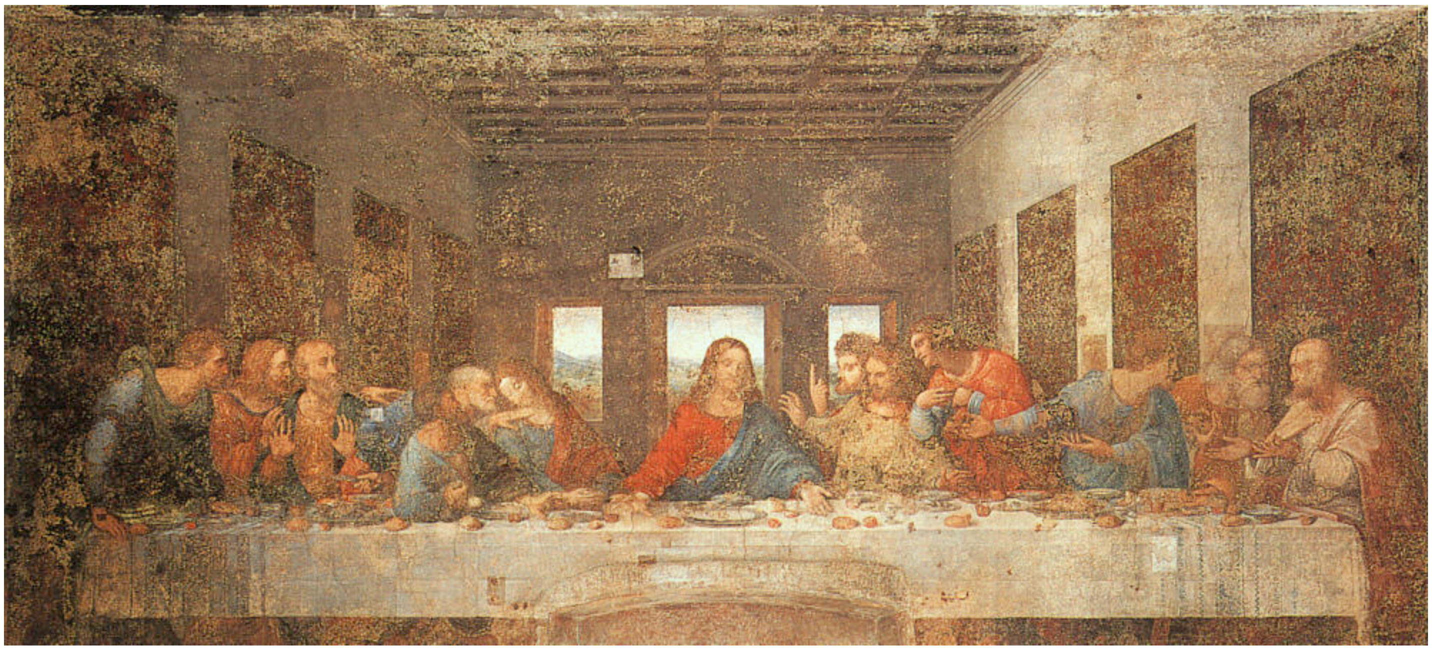
History of projection

- Ancient times: Greeks wrote about laws of perspective
- Renaissance: perspective is adopted by artists

Duccio c. 1308

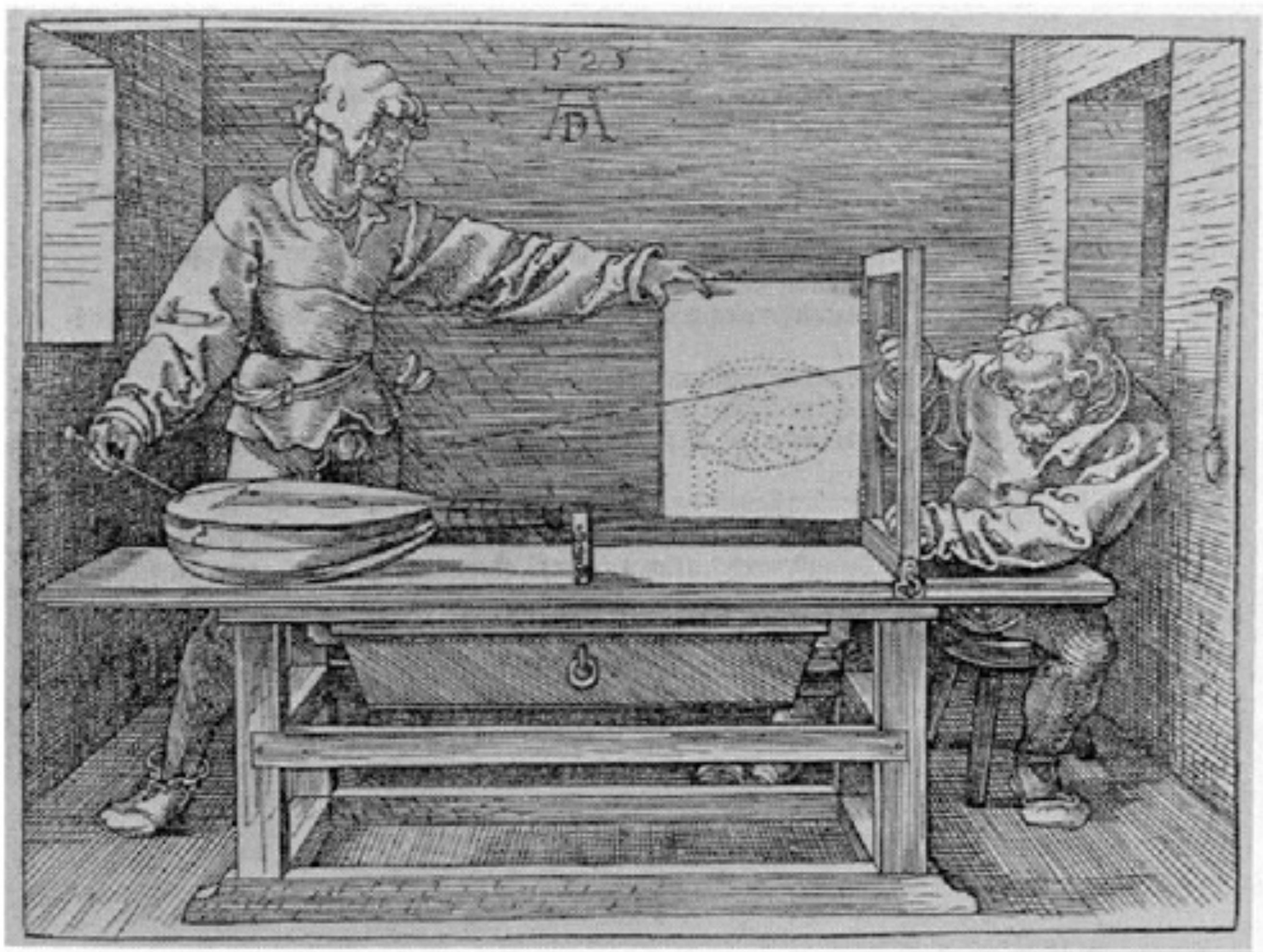
History of projection

- Later Renaissance: perspective formalized precisely



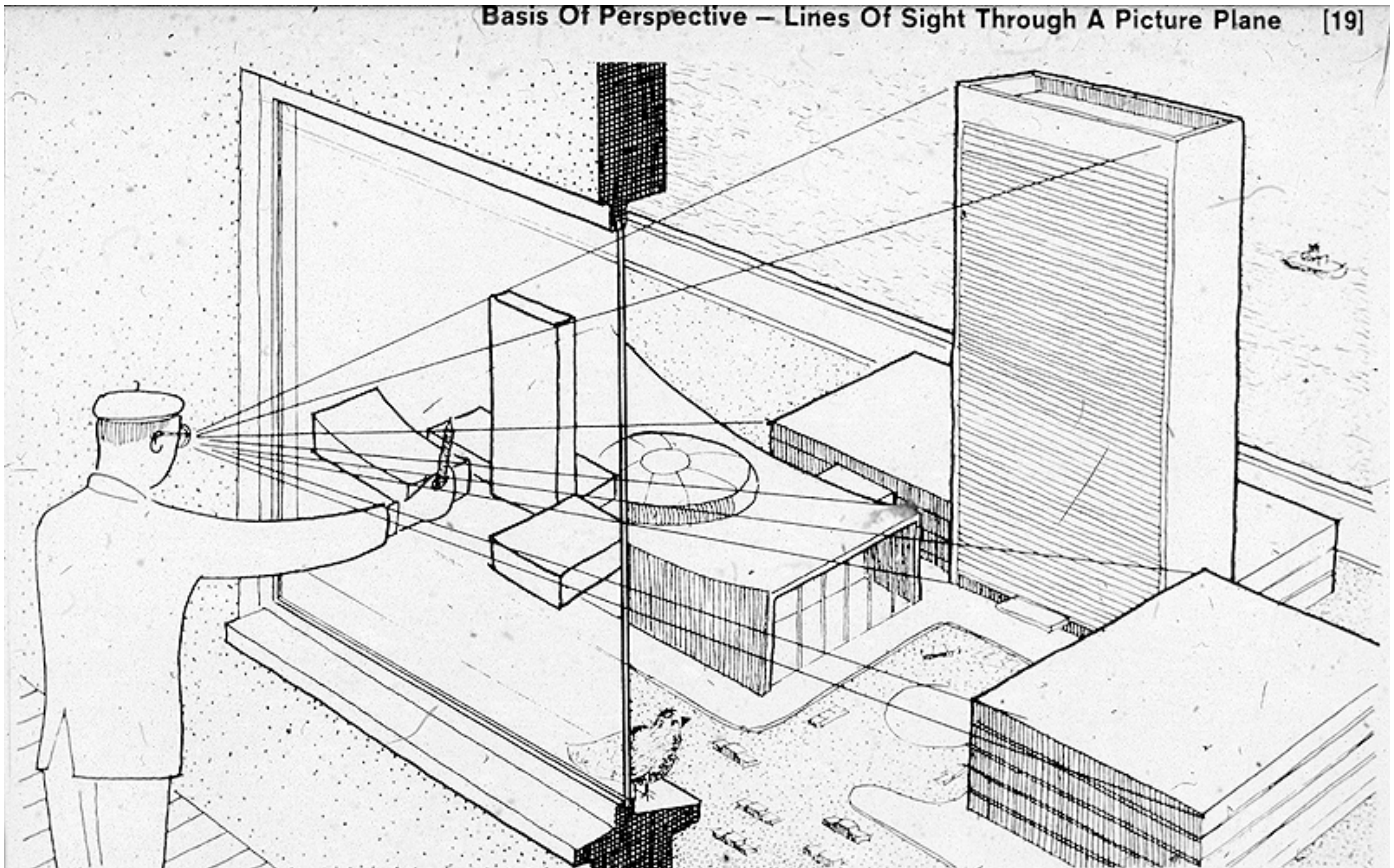
da Vinci c. 1498

Plane projection in drawing



[Carlbom & Paciorek 78]

Plane projection in drawing

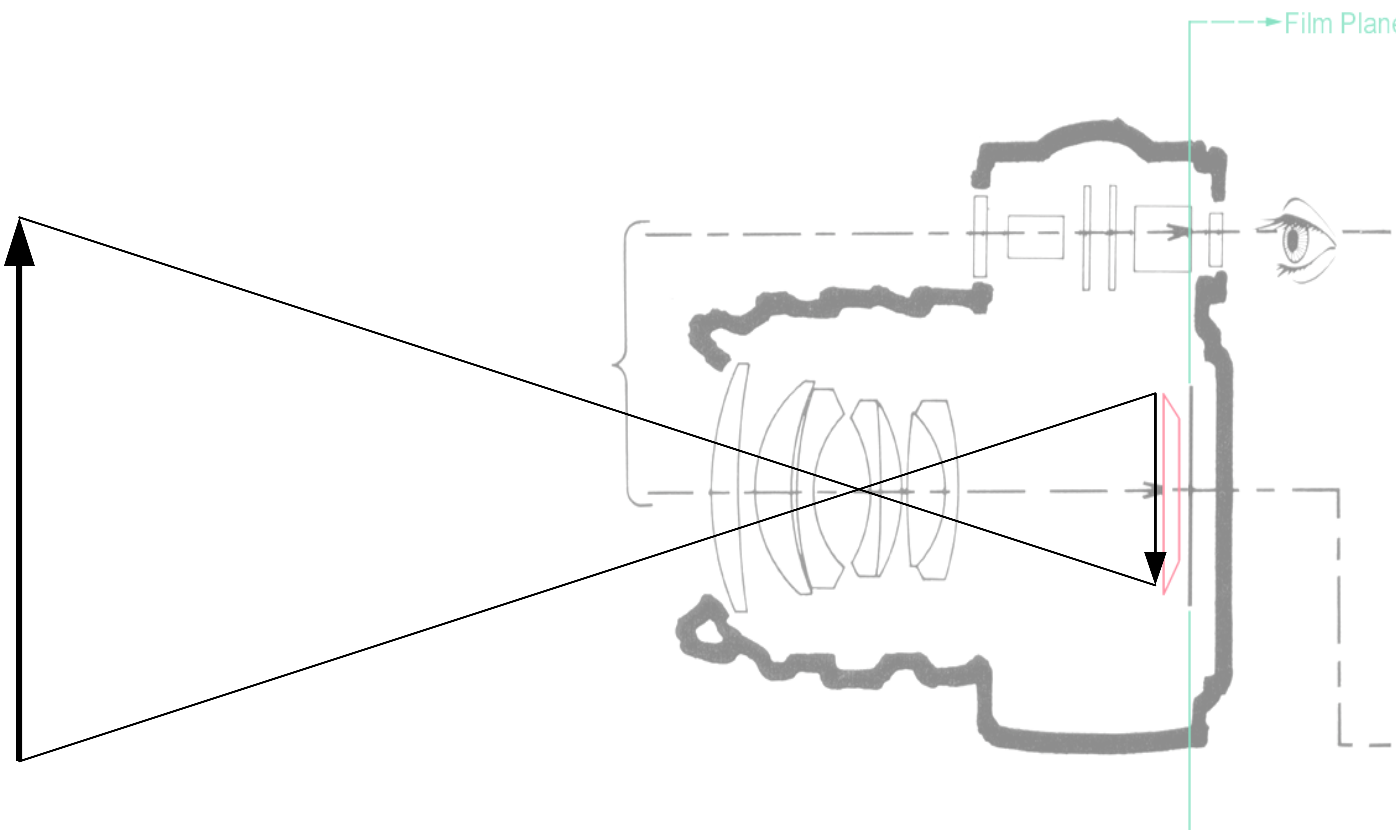


The concept of the picture plane may be better understood by looking through a window or other transparent plane from a fixed viewpoint. Your lines of sight, the multitude of straight lines leading from your eye to the subject, will all intersect this plane. Therefore, if you were to reach out with a grease pencil and draw the image of the subject on this plane you would be "tracing out" the infinite number of points of intersection of sight rays and plane. The result would be that you would have "transferred" a real three-dimensional object to a two-dimensional plane.

[CS 417 Spring 2002]

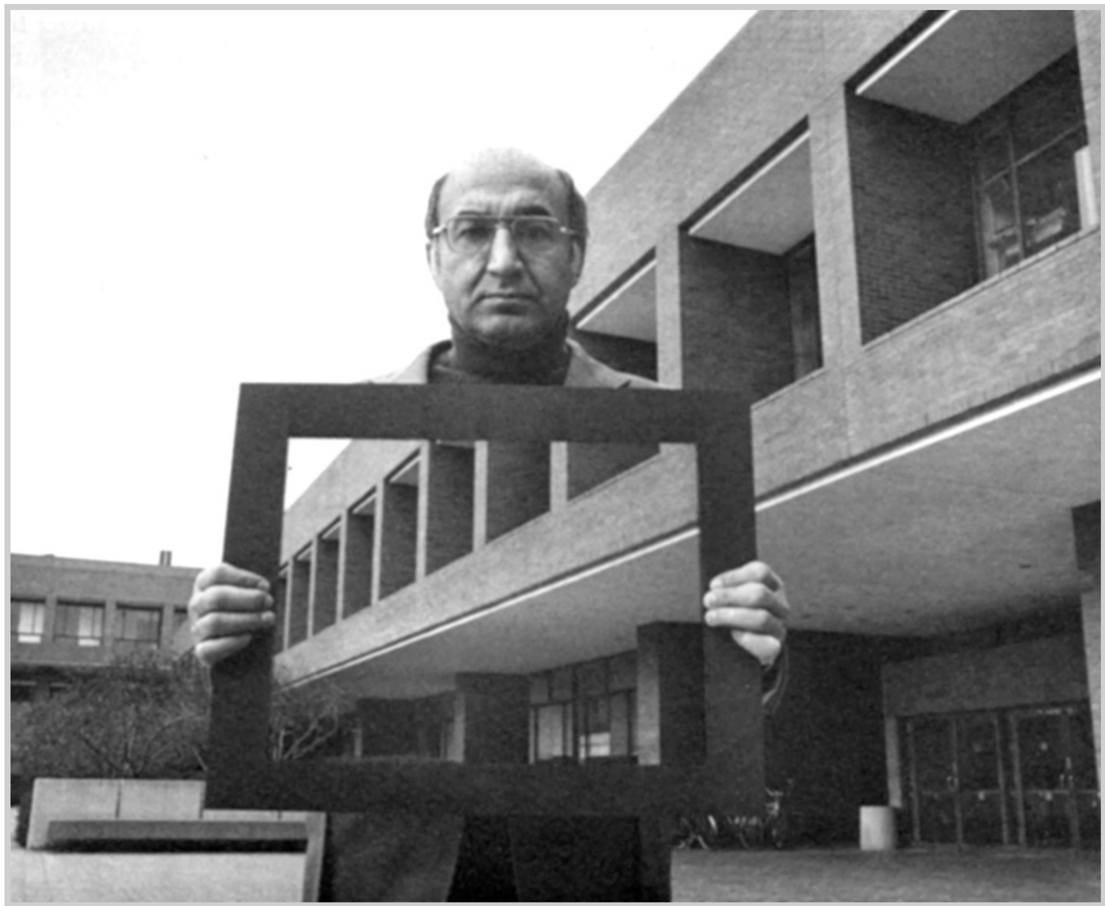
Plane projection in photography

- This is another model for what we are doing
  - applies more directly in realistic rendering



[CS 417 Spring 2002]

Plane projection in photography



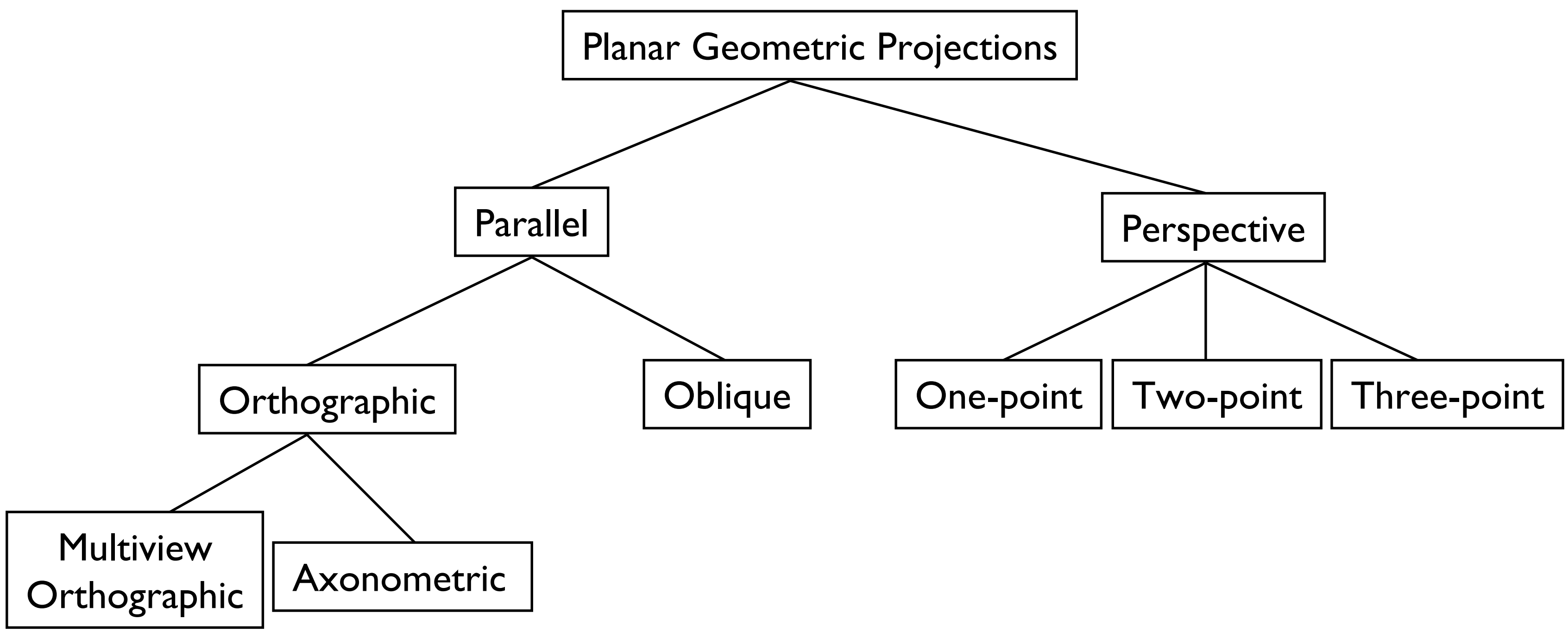
[Richard Zakia]

Ray generation vs. projection

- Viewing in ray tracing
  - start with image point
  - compute ray that projects to that point
  - do this using geometry
- Viewing by projection
  - start with 3D point
  - compute image point that it projects to
  - do this using transforms
- Inverse processes
  - ray gen. computes the preimage of projection

Classical projections

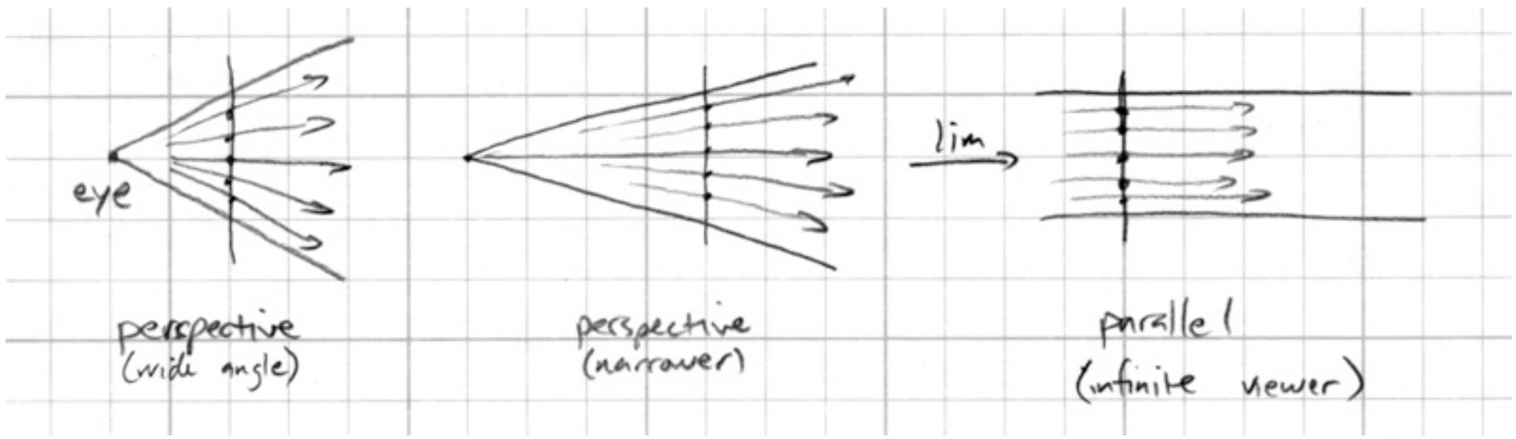
- Emphasis on cube-like objects
  – traditional in mechanical and architectural drawing



[after Carlbom & Paciorek 78]

Parallel projection

- Viewing rays are parallel rather than diverging
  - like a perspective camera that's far away



Multiview orthographic

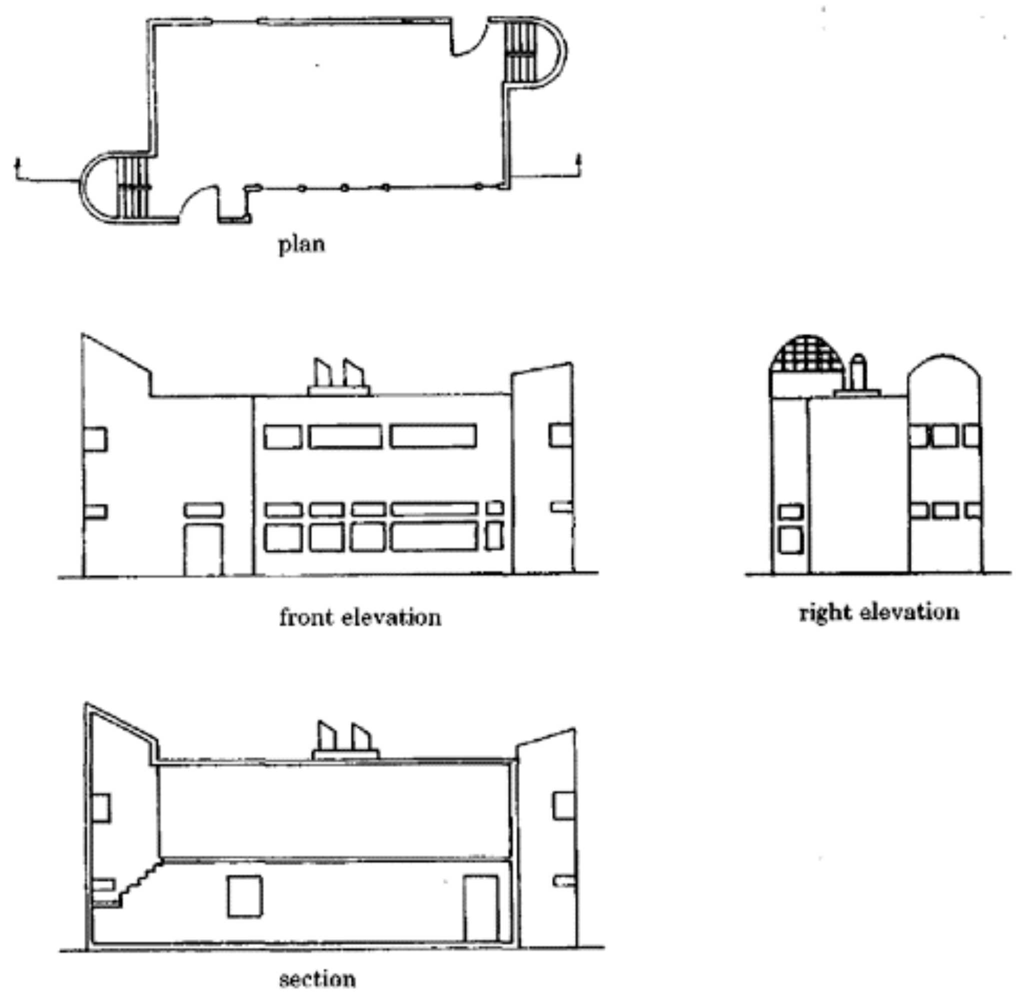
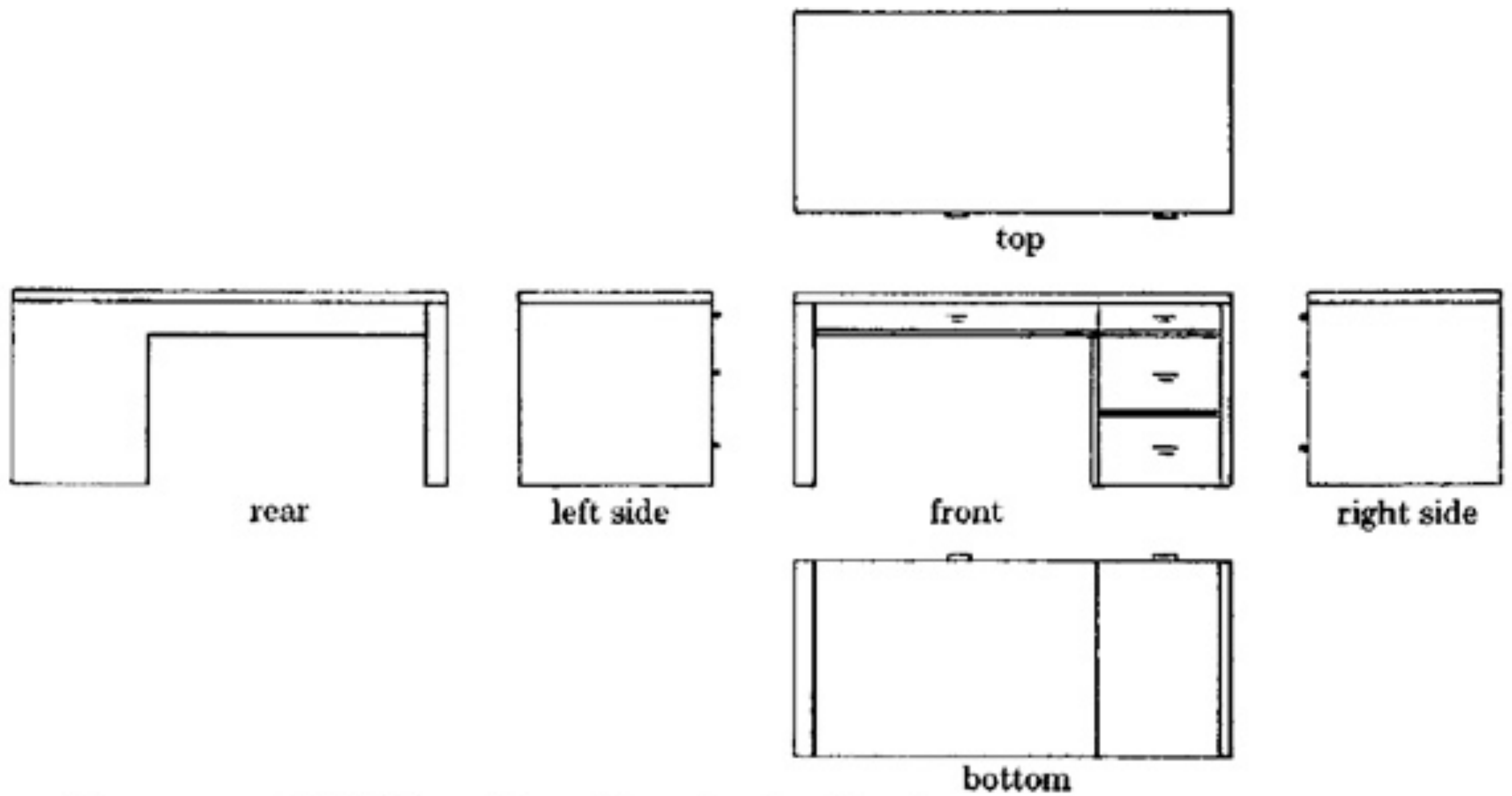


FIGURE 2-1. Multiview orthographic projection: plan, elevations, and section of a building.

[Carlbom & Paciorek 78]

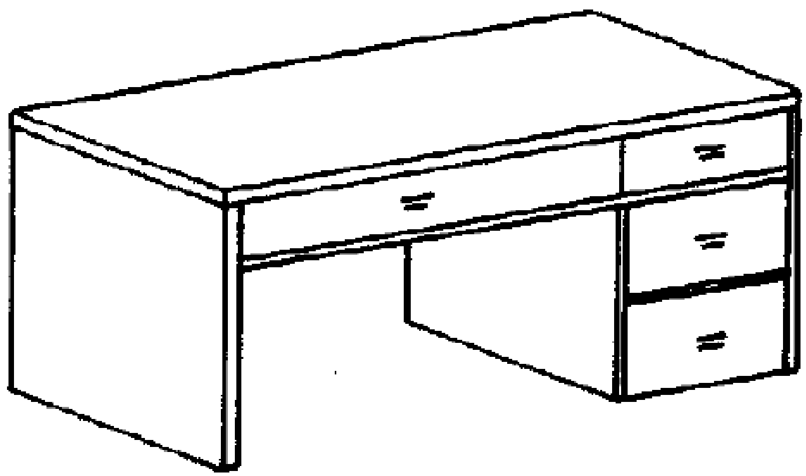
Multiview orthographic

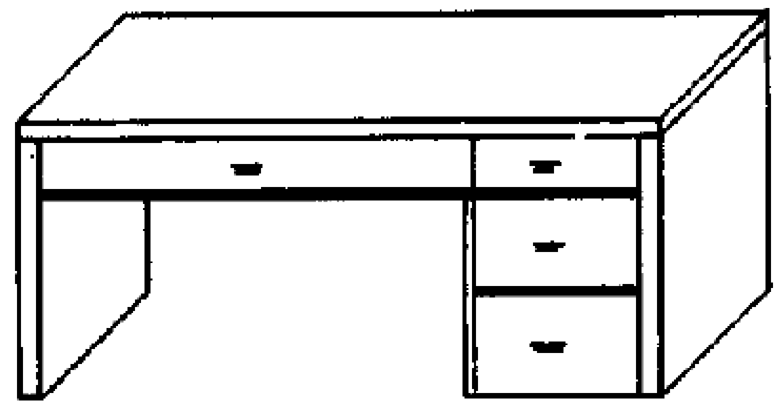


[Carlbom & Paciorek 78]

– projection plane parallel to a coordinate plane
– projection direction perpendicular to projection plane

Off-axis parallel





[Carlbom & Paciorek 78]

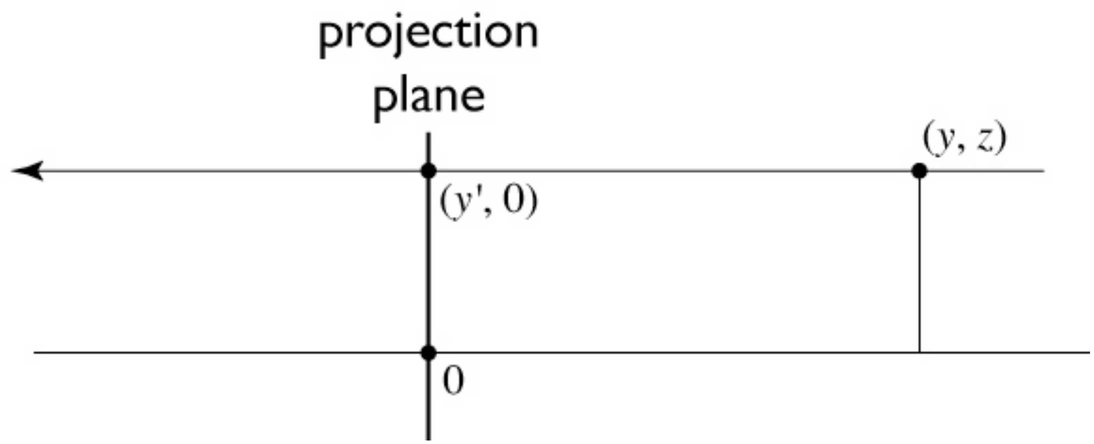
axonometric: projection plane perpendicular to projection direction but not parallel to coordinate planes

oblique: projection plane parallel to a coordinate plane but not perpendicular to projection direction.

Mathematics of projection

- Assume eye point at 0 and plane perpendicular to z
- Parallel case
  - a simple projection: just toss out z
- Perspective case: scale diminishes with z
  - and increases with d

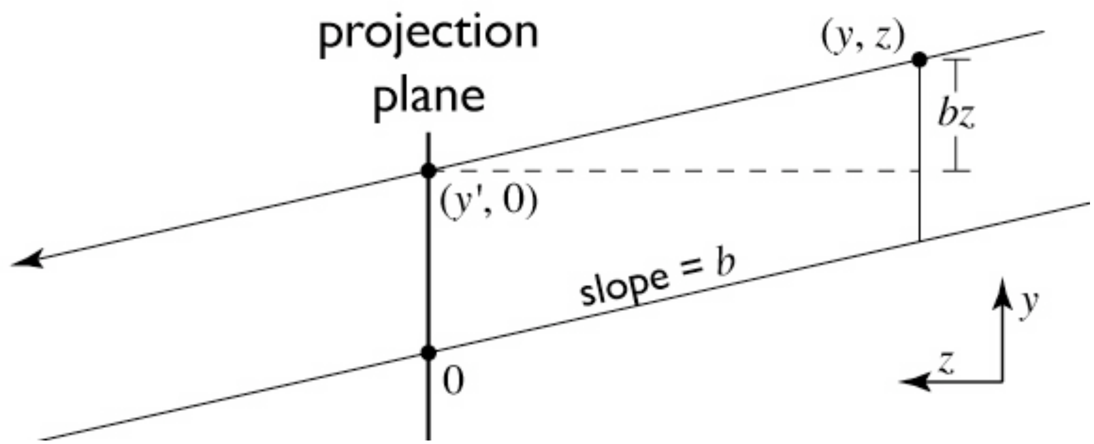
Parallel projection: orthographic



to implement orthographic, just toss out z:

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

Parallel projection: oblique



to implement oblique, shear then toss out *z*:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x + az \\ y + bz \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

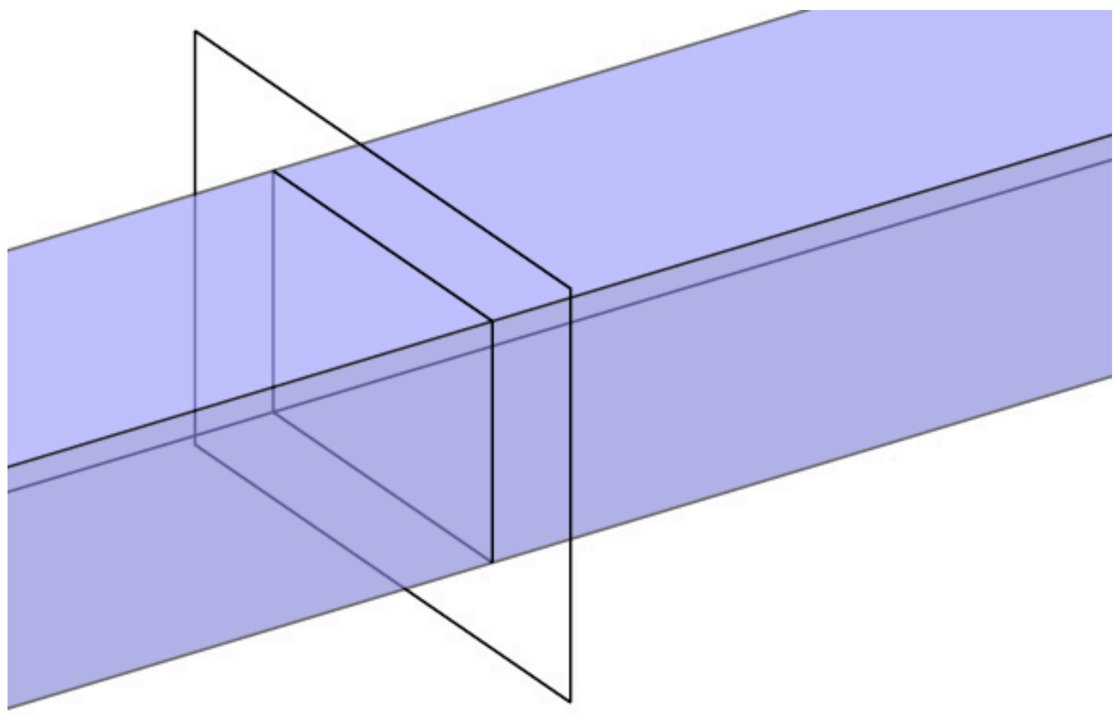
View volumes

- The volume of space that ends up in the image

$$V = \{\mathbf{p} \,|\, P\mathbf{p} \in R\}$$

  – *P* is the projection matrix; *R* is the image rectangle

View volume: orthographic

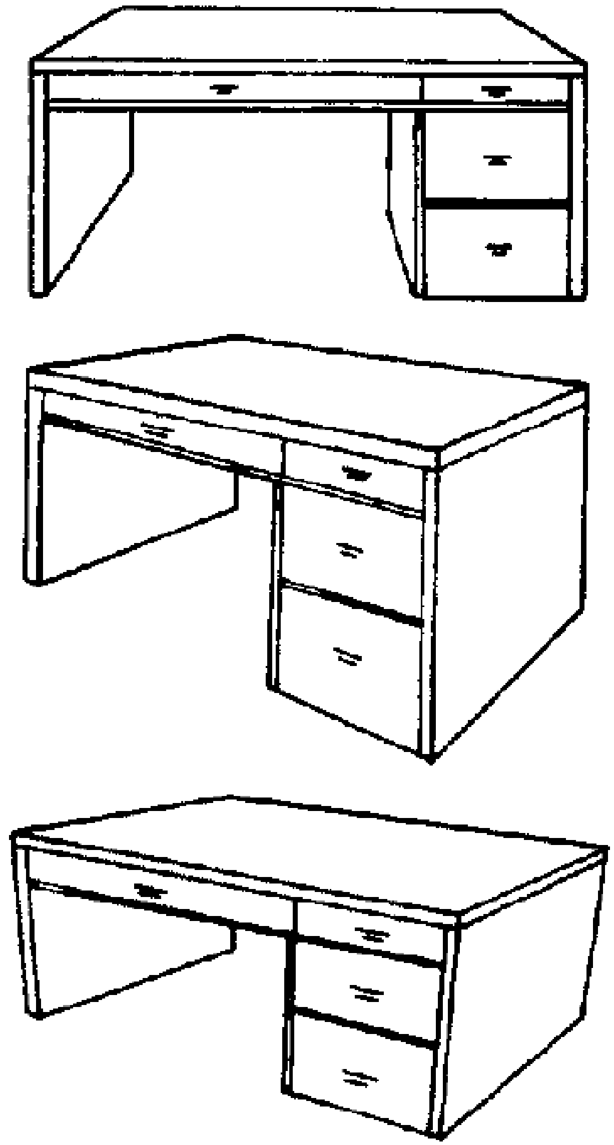


Perspective

one-point: projection plane parallel to a coordinate plane (to two coordinate axes)

two-point: projection plane parallel to one coordinate axis

three-point: projection plane not parallel to a coordinate axis



one-point

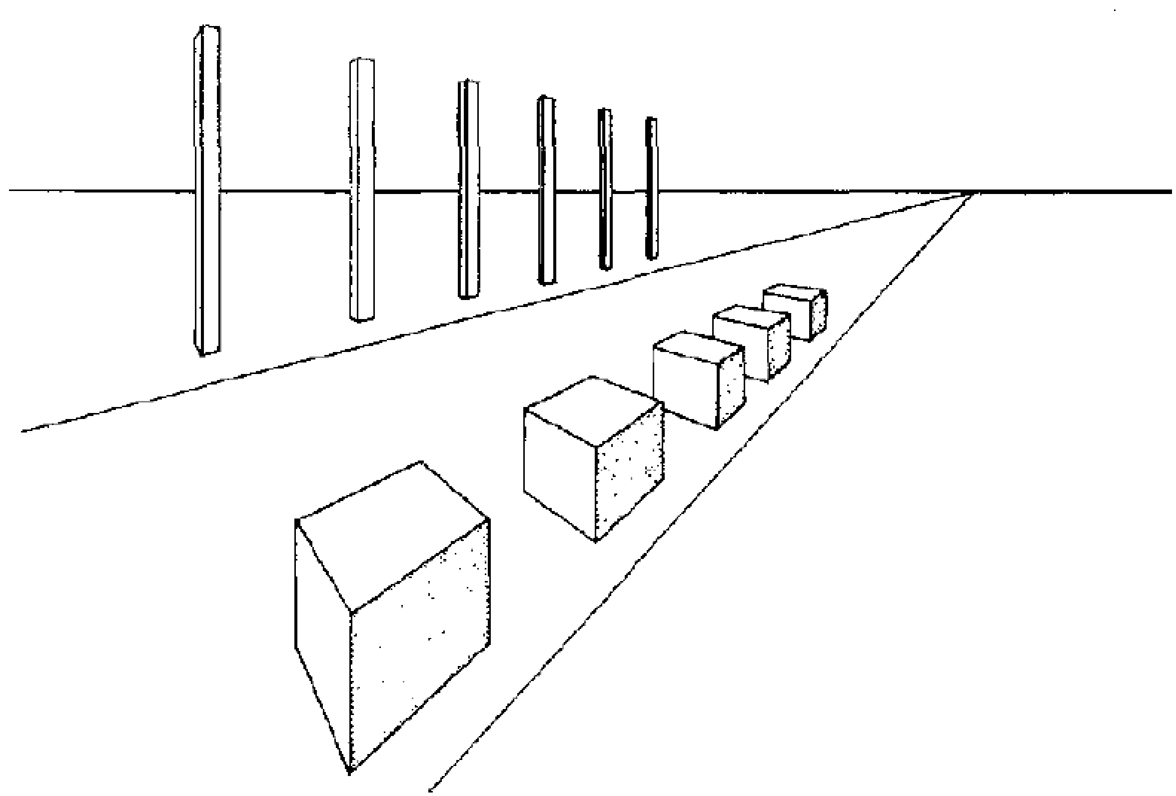
two-point

three-point

[Carlbom & Paciorek 78]

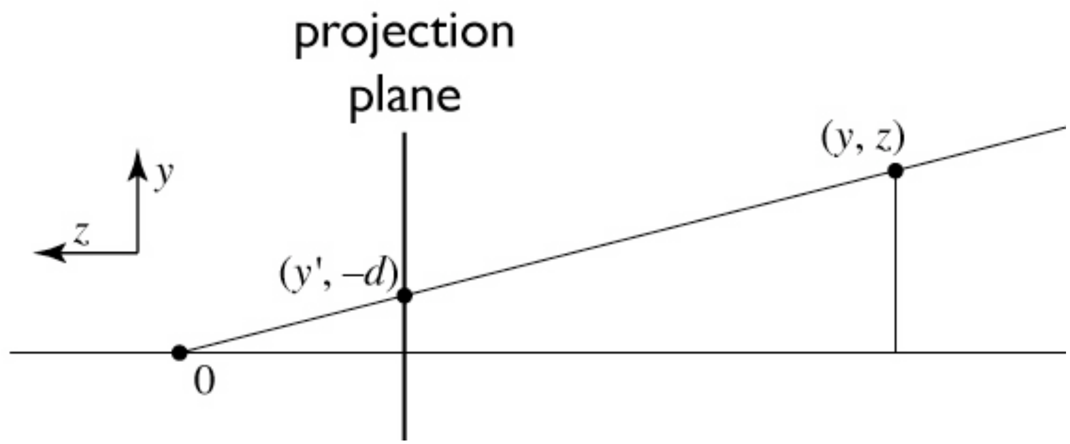
Perspective distortions

- Lengths, length ratios



[Carlbom & Paciorek 78]

Perspective projection



similar triangles:

$$\frac{y'}{d} = \frac{y}{-z}$$

$$y' = -dy/z$$

Homogeneous coordinates revisited

- Perspective requires division
  - that is not part of affine transformations
  - in affine, parallel lines stay parallel
    - therefore not vanishing point
    - therefore no rays converging on viewpoint
- "True" purpose of homogeneous coords: projection

Homogeneous coordinates revisited

- Introduced $w = 1$ coordinate as a placeholder

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

  – used as a convenience for unifying translation with linear

- Can also allow arbitrary *w*

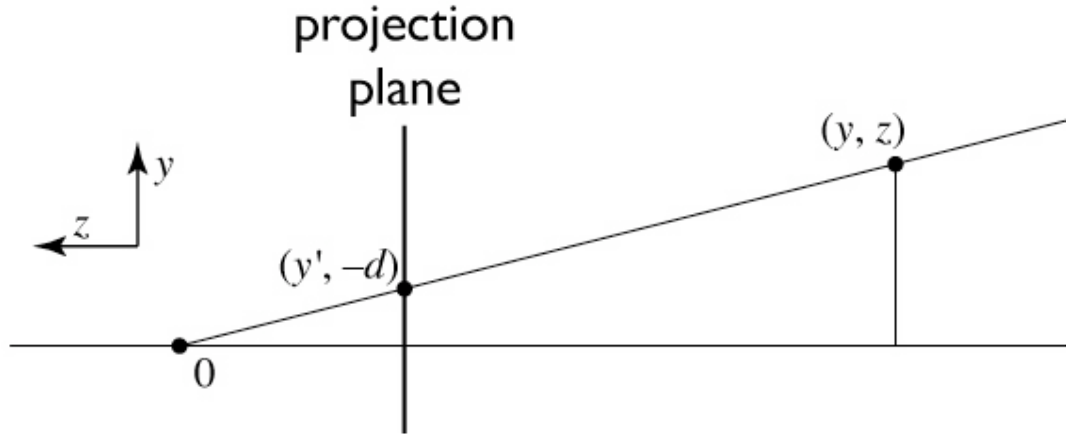
$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \sim \begin{bmatrix} wx \\ wy \\ wz \\ w \end{bmatrix}$$

Implications of *w*

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \sim \begin{bmatrix} wx \\ wy \\ wz \\ w \end{bmatrix}$$

- All scalar multiples of a 4-vector are equivalent
- When *w* is not zero, can divide by *w*
  - therefore these points represent “normal” affine points
- When *w* is zero, it’s a point at infinity
  - this is the point where parallel lines intersect
  - can also think of it as the vanishing point
- Digression on projective space

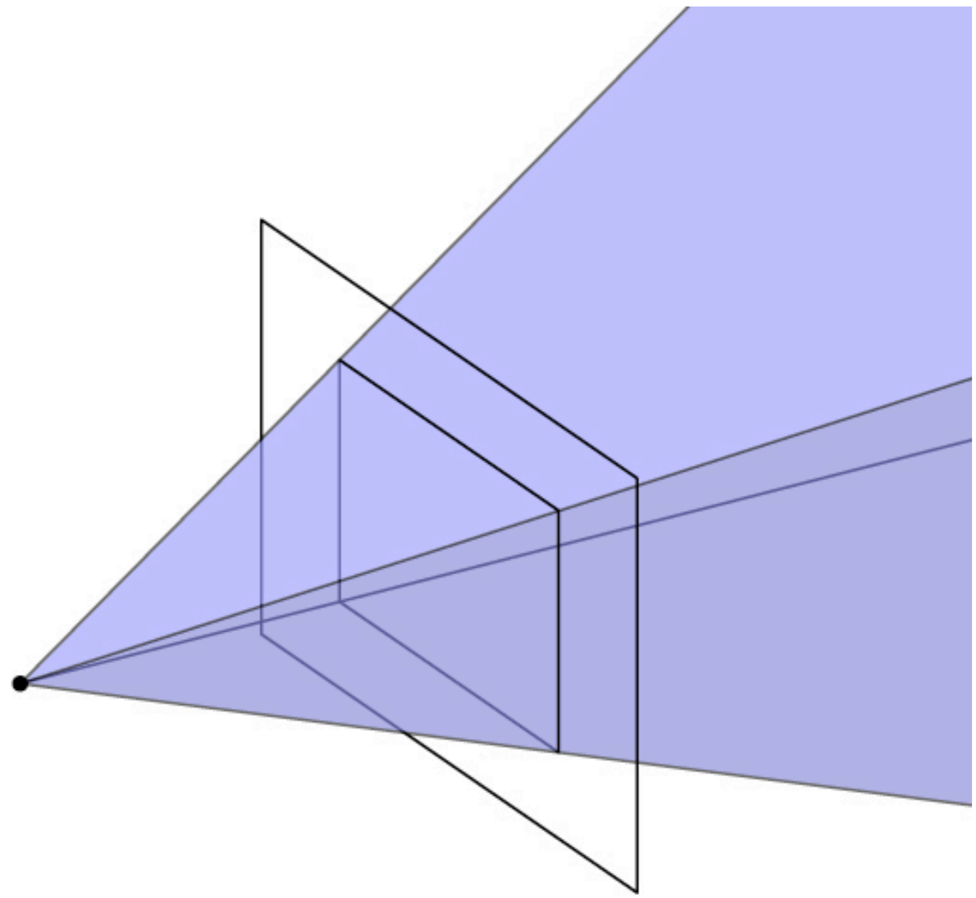
Perspective projection



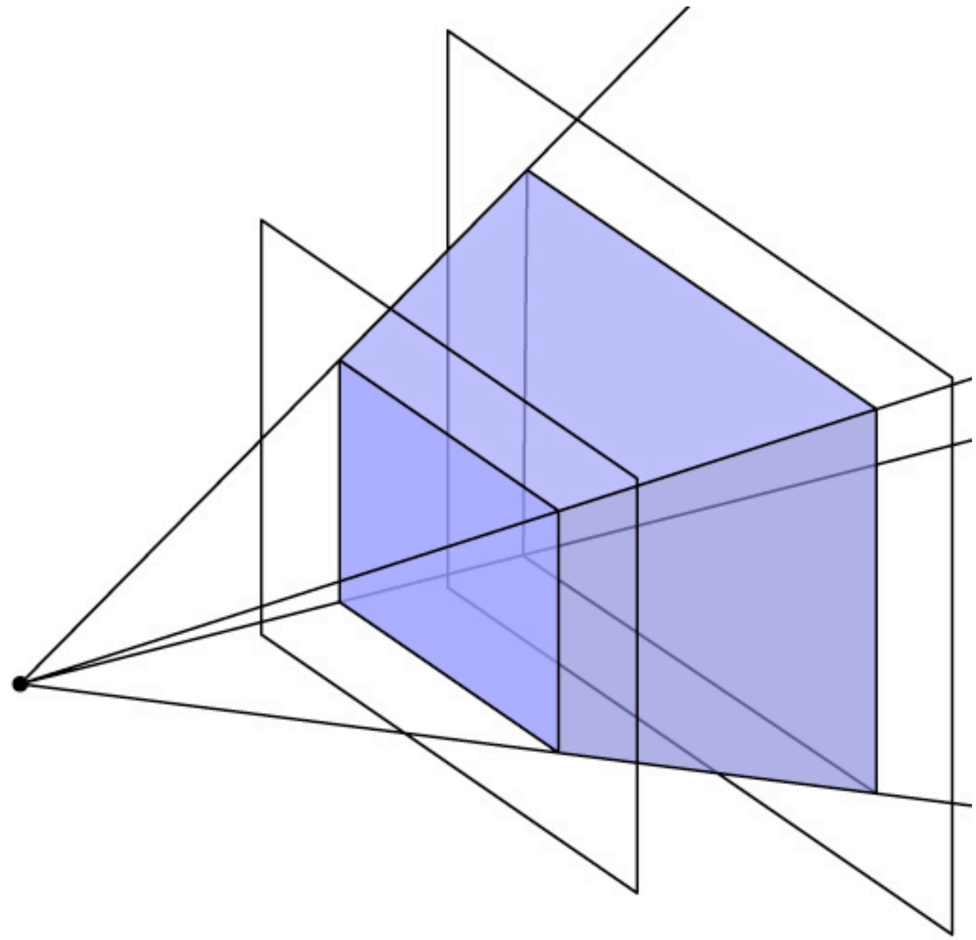
to implement perspective, just move z to w:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -dx/z \\ -dy/z \\ 1 \end{bmatrix} \sim \begin{bmatrix} dx \\ dy \\ -z \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

View volume: perspective



View volume: perspective (clipped)



Clipping planes

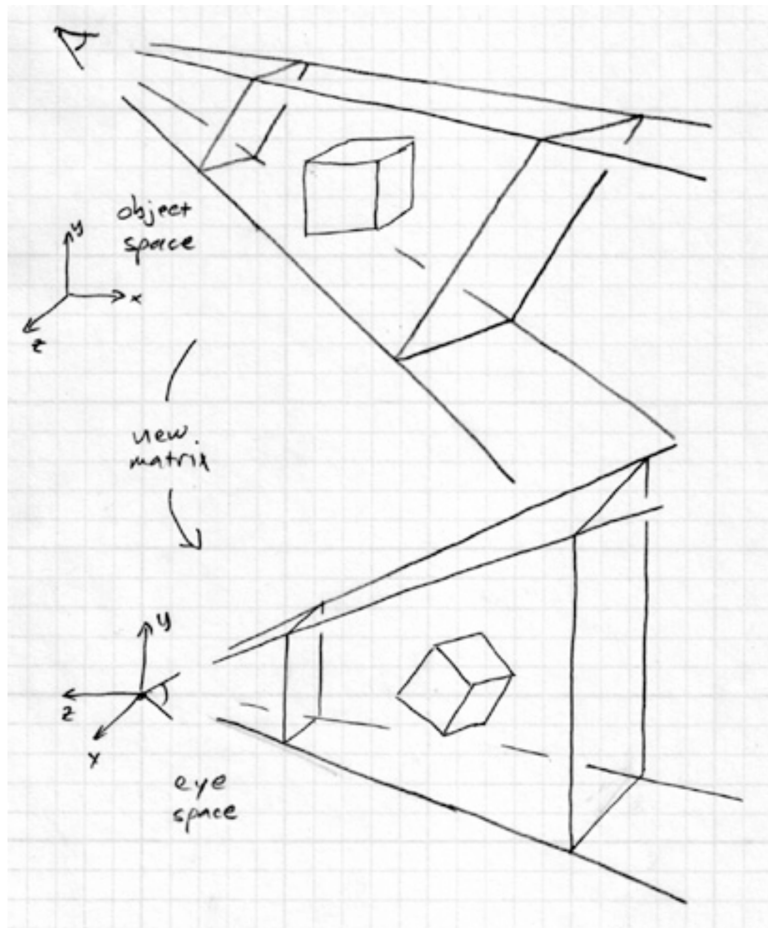
- In object-order systems we always use at least two *clipping planes* that further constrain the view volume
  - near plane: parallel to view plane; things between it and the viewpoint will not be rendered
  - far plane: also parallel; things behind it will not be rendered
- These planes are:
  - partly to remove unnecessary stuff (e.g. behind the camera)
  - but really to constrain the range of depths

    (we'll see why later)

Viewing in 3D

- The application also chooses a camera pose (position and orientation)
  - this defines a coordinate frame for the camera
  - transform geometry into that frame for rendering
  - *viewing matrix* is the c.-to-b. transform of the camera frame
  - the resulting coordinates are *eye coordinates*
  - we can now assume that the camera is in standard pose

Viewing transformation



the view matrix rewrites all coordinates in eye space

Projection transformation

- With geometry in eye space, projection is simple:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -dx/z \\ -dy/z \\ 1 \end{bmatrix} \sim \begin{bmatrix} dx \\ dy \\ -z \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- To enable hidden surface removal, want to keep a pseudo-depth *z'* that increases with *z*:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \sim \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ -z \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

(recall this means "is a scalar multiple of")

Projection transformation

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \sim \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ -z \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Just like *x'* and *y'* run from –1 to 1, we'd like *z'* to run from –1 to 1

$$\tilde{z}(z) = az + b$$

$$z'(n) = -1 \Rightarrow \tilde{z}(n) = n$$

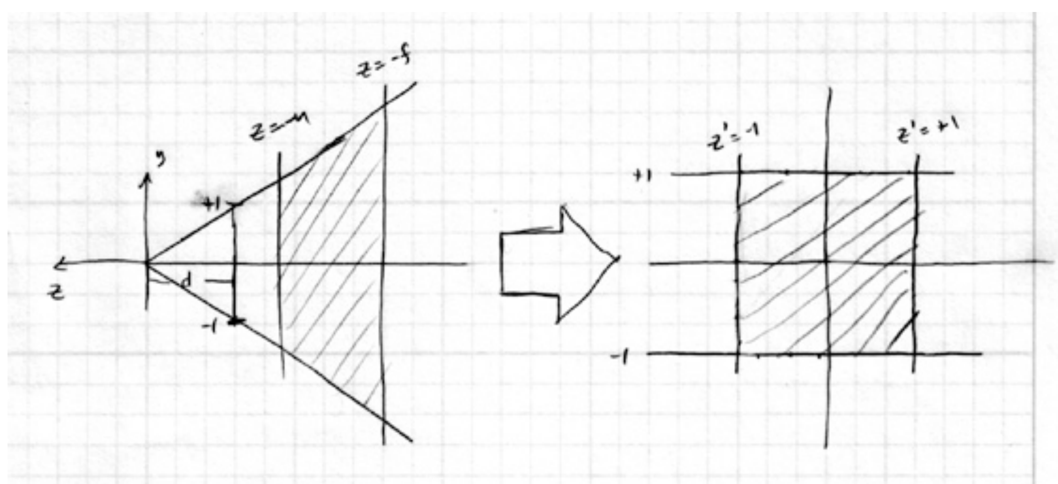
$$z'(f) = 1 \Rightarrow \tilde{z}(f) = -f$$

  – solving for *a* and *b* leads to $a = \frac{n+f}{n-f}; b = 2\frac{nf}{f-n}$

Projection transformation

- Thus the projection matrix for projection plane distance *d* and near and far distances *n* and *f* is:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \sim \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ -z \end{bmatrix} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & \frac{n+f}{n-f} & 2\frac{nf}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



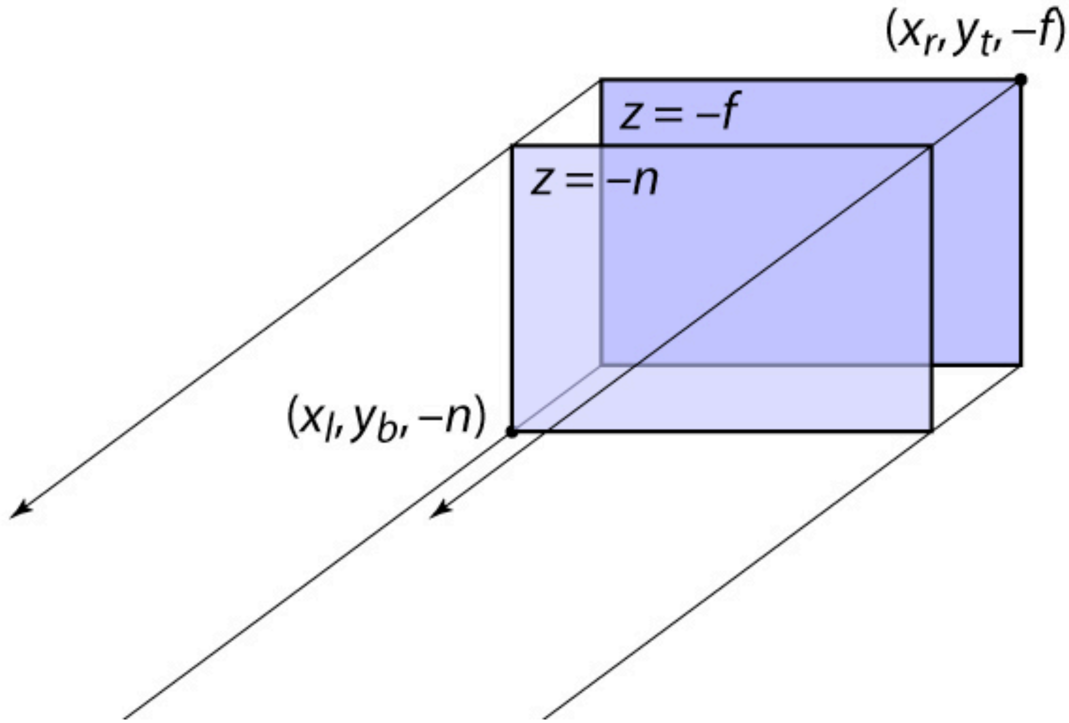
Projection transformation

- Projection matrix maps from eye space to *clip space*
- In this space, the two-unit cube $[-1, 1]^3$ contains exactly what needs to be drawn
- It's called "clip" coordinates because everything outside of this box is clipped out of the view
  - this can be done at this point, geometrically
  - or it can be done implicitly later on by careful rasterization

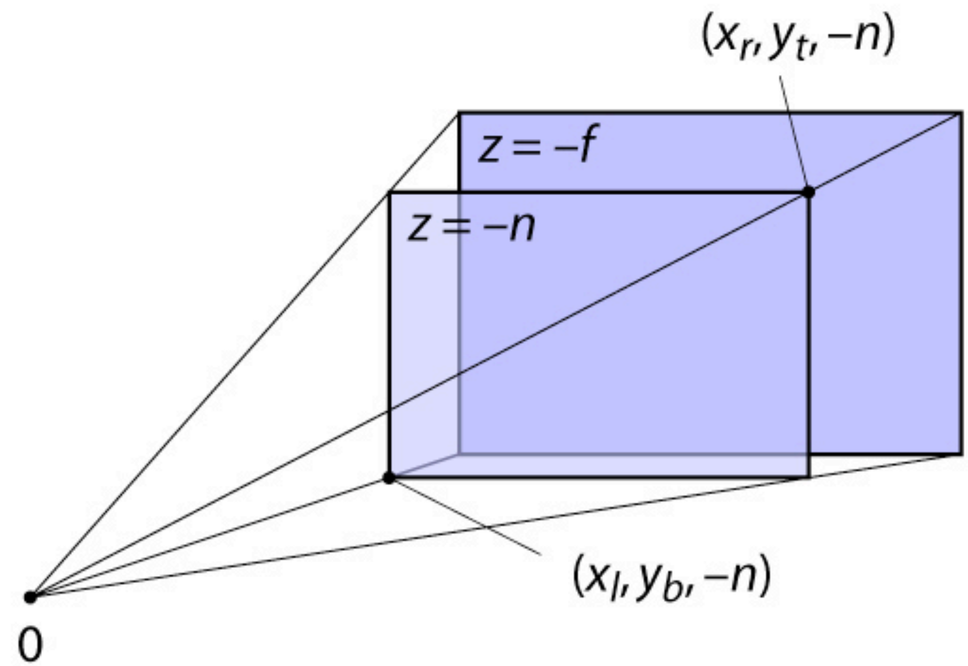
Viewport transformation

- A simple bookkeeping step to scale image
  - clip volume was a simple cube
  - rasterizer needs input in pixel coords
  - therefore scale and translate to map the [–1, 1] box to the desired rectangle in window coordinates, or *screen space*
- Also shift *z'* to the desired range
  - usually that range is [0, 1] so that it can be represented by a fixed-point fraction
- Homogeneous divide usually happens here

View frustum: orthographic



View frustum: perspective



Vertex processing: spaces

- Standard sequence of transforms