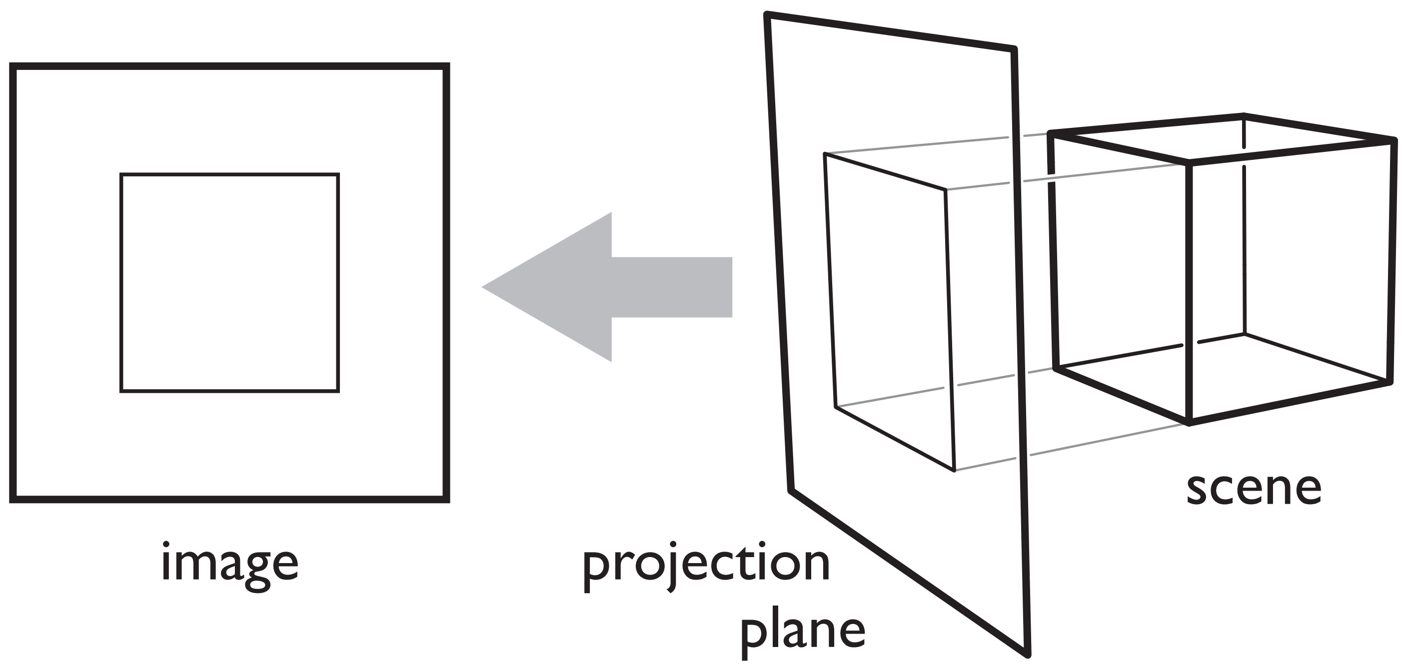
Perspective

CS 4620 Lecture 11

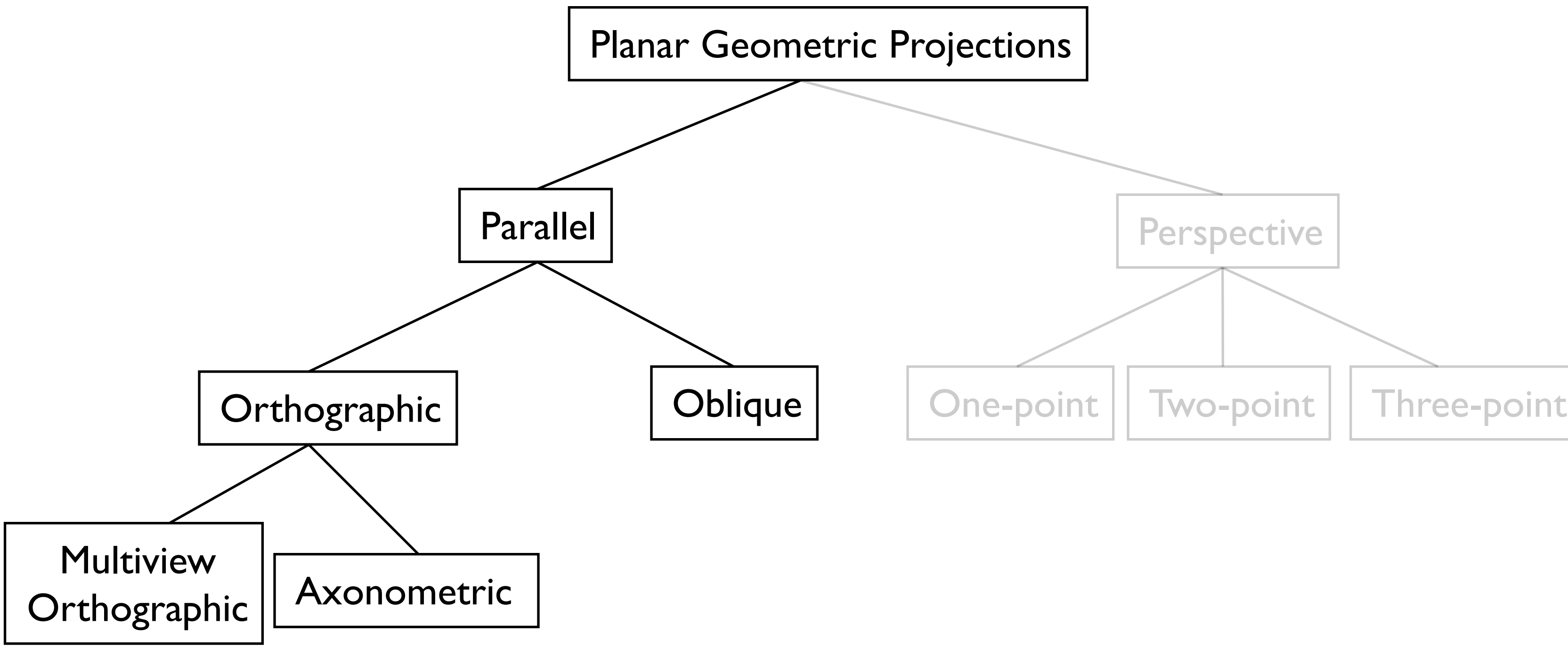
Parallel projection

- To render an image of a 3D scene, we *project* it onto a plane
- Simplest kind of projection is *parallel projection*



Classical projections—parallel

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



[after Carlbom & Paciorek 78]

Orthographic

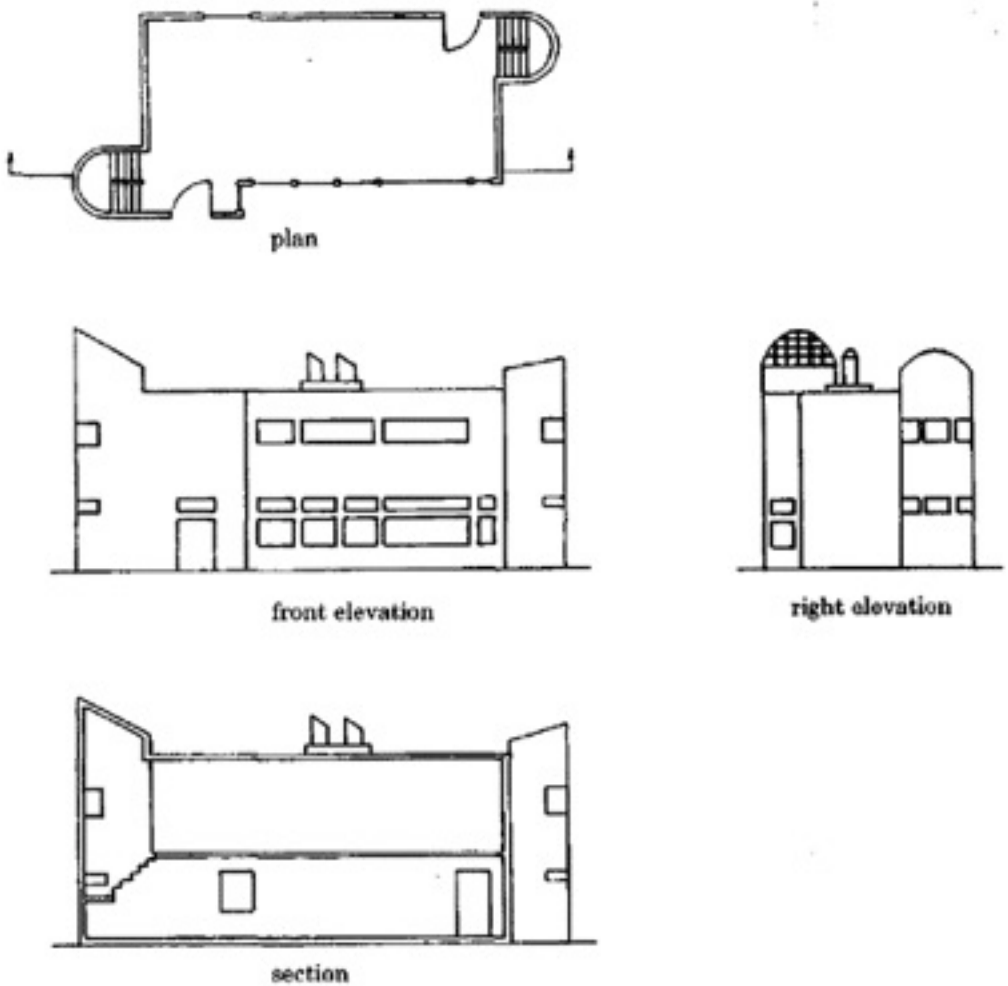
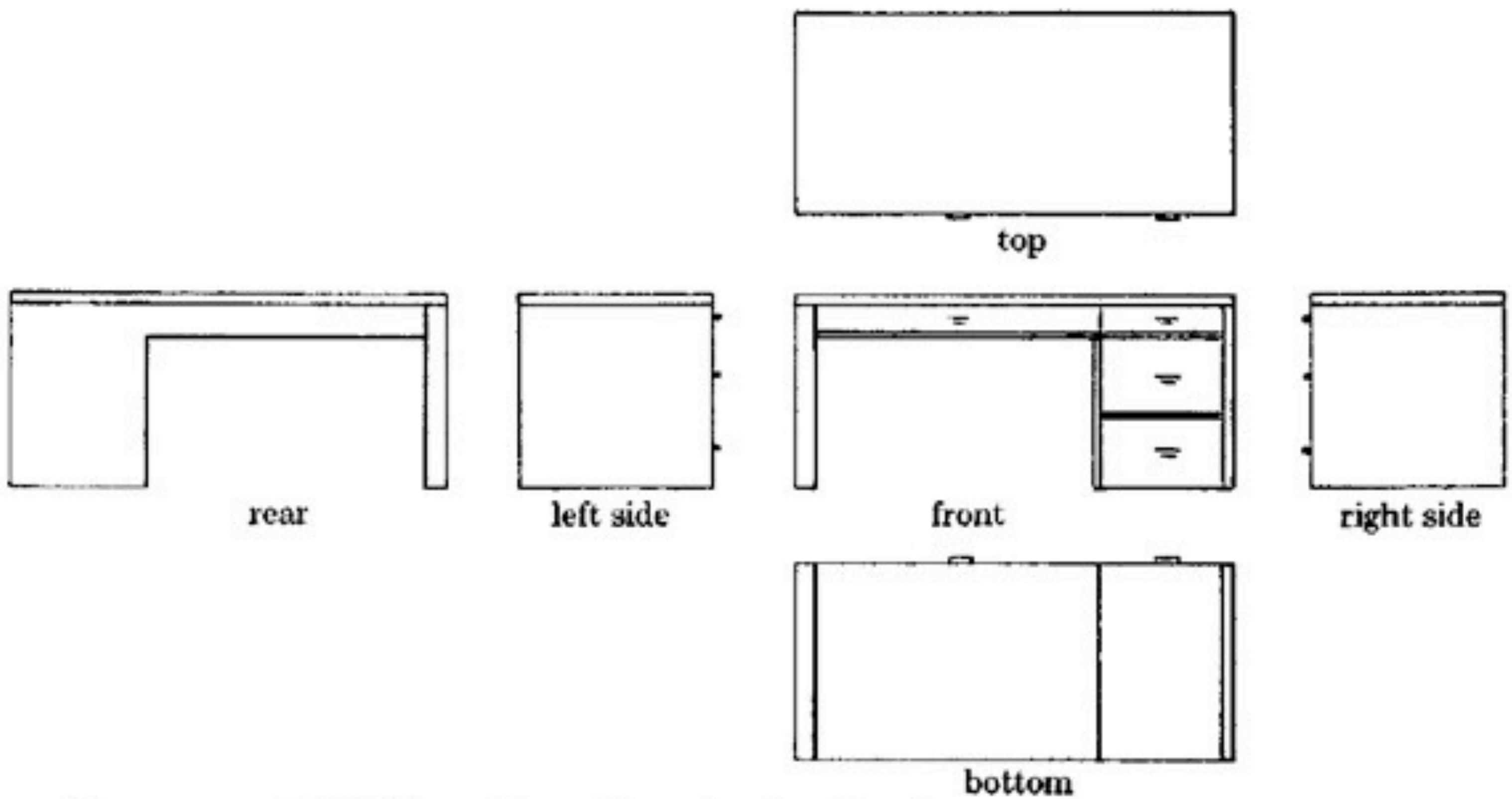


FIGURE 2-1. Multiview orthographic projection: plan, elevations, and section of a building.

[Carlbom & Paciorek 78]

Orthographic

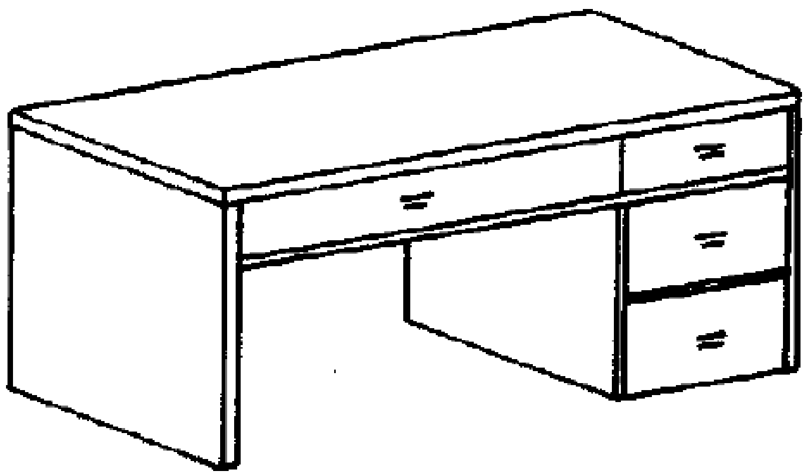


[Carlbom & Paciorek 78]

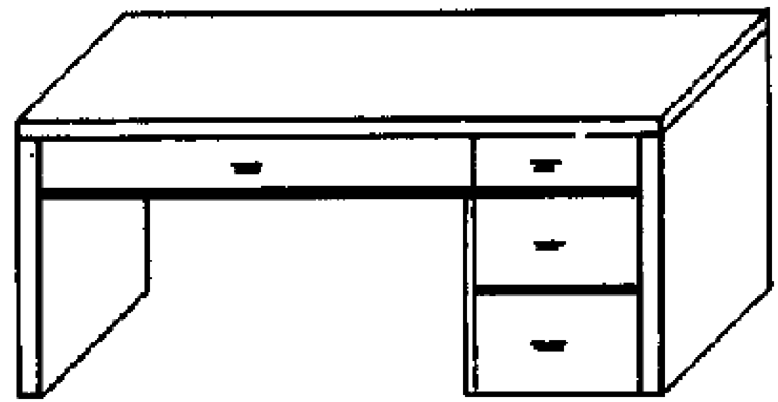
– projection plane parallel to a coordinate plane

– projection direction perpendicular to projection plane

Off-axis parallel



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

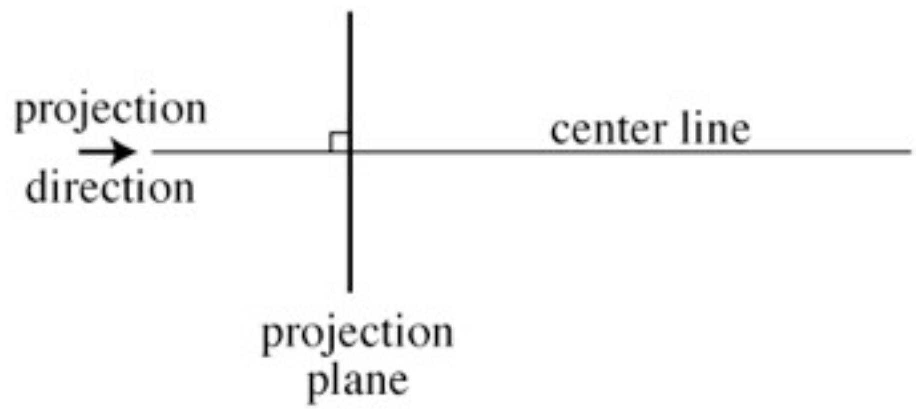


oblique: projection plane parallel to a coordinate plane but not perpendicular to projection direction.

[Carlbom & Paciorek 78]

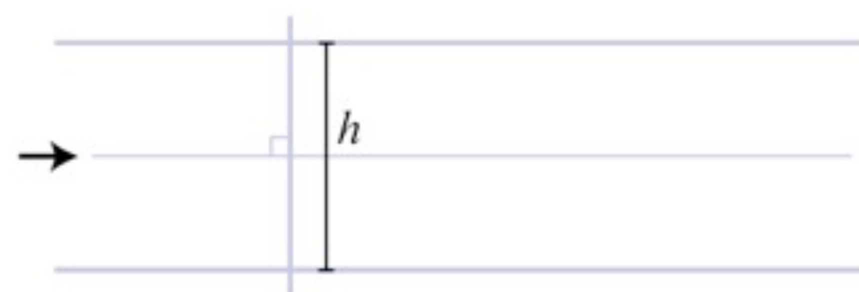
"Orthographic" projection

- In graphics usually we lump axonometric with orthographic
  - projection plane perpendicular to projection direction
  - image height determines size of objects in image



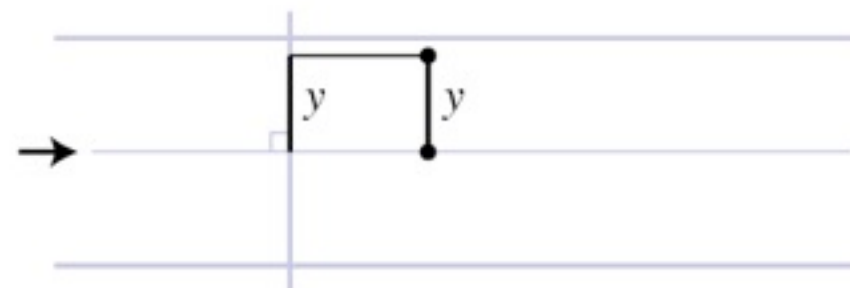
"Orthographic" projection

- In graphics usually we lump axonometric with orthographic
  - projection plane perpendicular to projection direction
  - image height determines size of objects in image



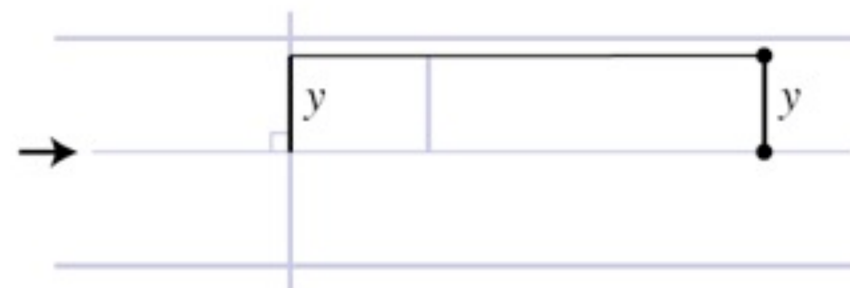
“Orthographic” projection

- In graphics usually we lump axonometric with orthographic
  - projection plane perpendicular to projection direction
  - image height determines size of objects in image



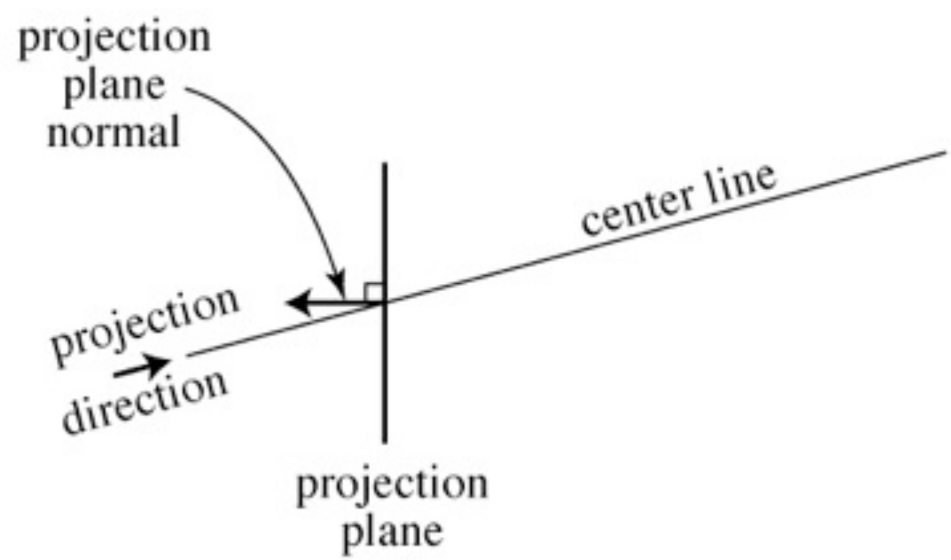
"Orthographic" projection

- In graphics usually we lump axonometric with orthographic
  - projection plane perpendicular to projection direction
  - image height determines size of objects in image



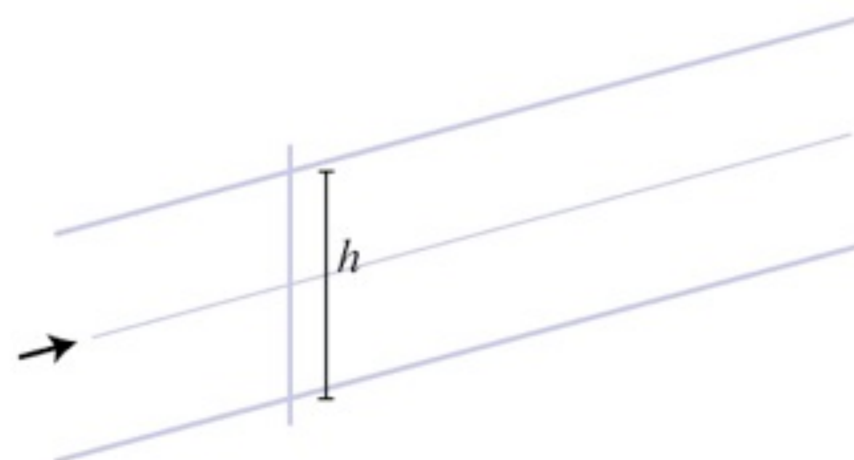
Oblique projection

- View direction no longer coincides with projection plane normal (one more parameter)
  - objects at different distances still same size
  - objects are shifted in the image depending on their depth



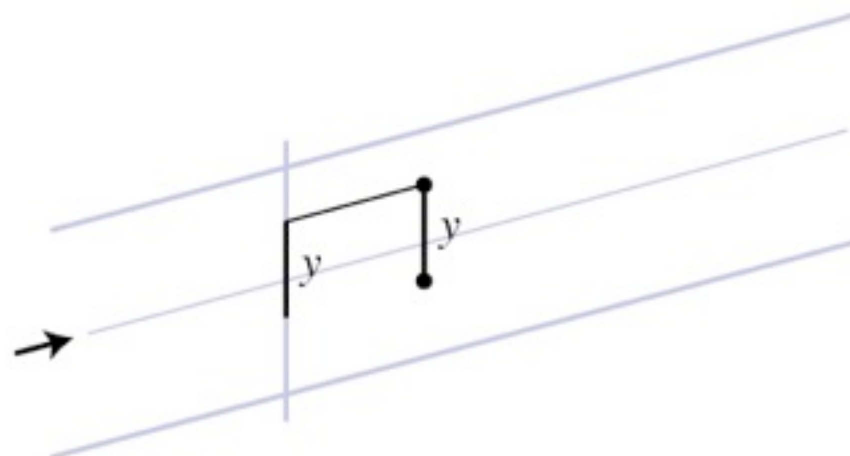
Oblique projection

- View direction no longer coincides with projection plane normal (one more parameter)
  - objects at different distances still same size
  - objects are shifted in the image depending on their depth



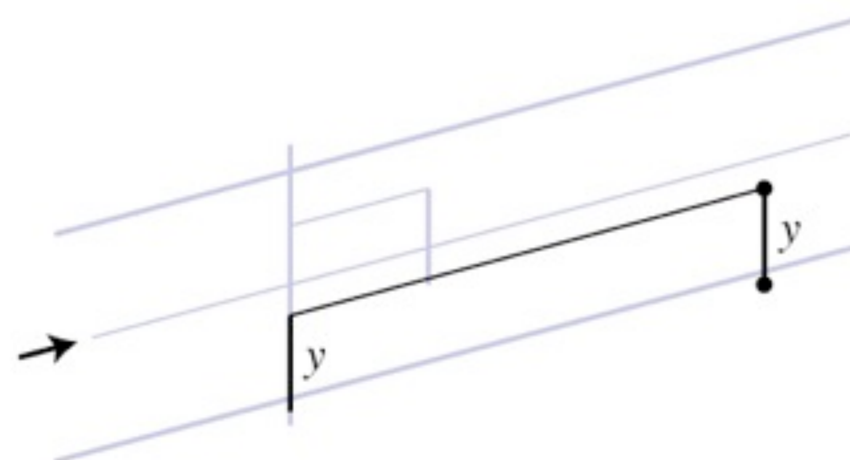
Oblique projection

- View direction no longer coincides with projection plane normal (one more parameter)
  - objects at different distances still same size
  - objects are shifted in the image depending on their depth



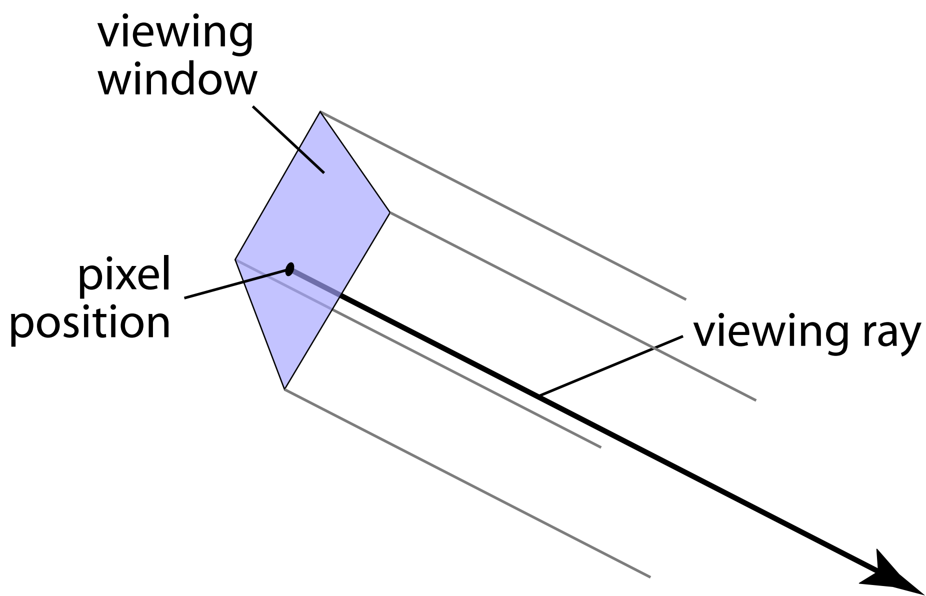
Oblique projection

- View direction no longer coincides with projection plane normal (one more parameter)
  - objects at different distances still same size
  - objects are shifted in the image depending on their depth



Generating eye rays—orthographic

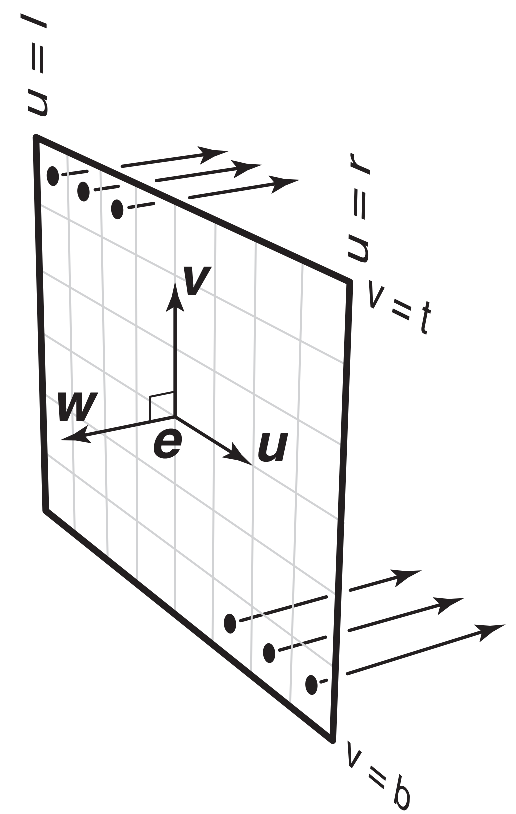
- Ray origin (varying): pixel position on viewing window
- Ray direction (constant): view direction



  – but where exactly is the view rectangle?

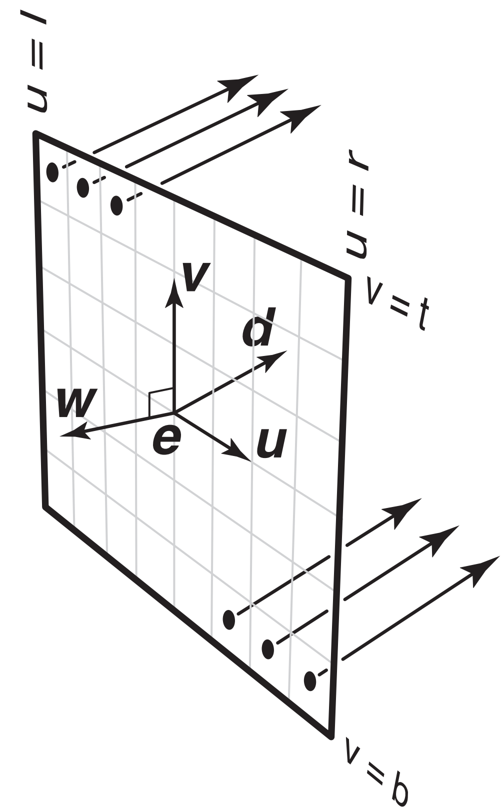
Generating eye rays—parallel

- Positioning the view rectangle
  - establish three vectors to be *camera basis:* **u**, **v**, **w**
  - view rectangle is in **u**–**v** plane, specified by *l*, *r*, *t*, *b* (often *l* = –*r* and *b* = –*t*)
- Generating rays
  - for (*u*, *v*) in [*l*, *r*] × [*b*, *t*]
  - ray.origin = **e** + *u* **u** + *v* **v**
  - ray.direction = –**w**



Oblique parallel views

- View rectangle is the same
  - ray origins identical to orthographic
  - view direction **d** differs from –**w**
- Generating rays
  - for (u, v) in $[l, r] \times [b, t]$
  - ray.origin = **e** + u **u** + v **v**
  - ray.direction = **d**

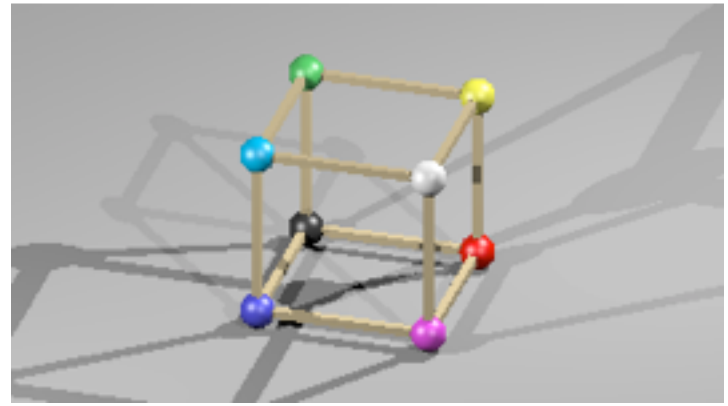


Establishing the camera basis

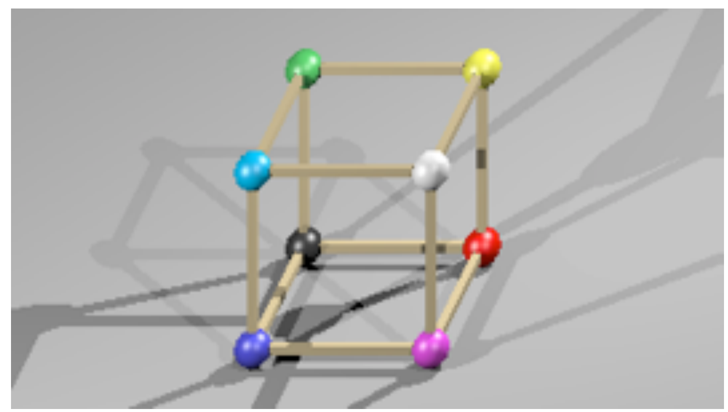
- Could require user to provide **e**, **u**, **v**, and **w**
  - but this is error prone and unintuitive
- Instead, calculate basis from things the user cares about
  - viewpoint: where the camera is → **e**
  - view direction: which way the camera is looking → **d**
  - view plane normal (by default, same as view direction)
  - up vector: how the camera is oriented
- This is enough to calculate **u**, **v**, and **w**
  - set **w** parallel to v.p. normal, facing away from **d**
  - set **u** perpendicular to **w** and perpendicular to up-vector
  - set **v** perpendicular to **w** and **u** to form a right-handed ONB

Specifying views in a ray tracer

```
<camera type="ParallelCamera">
  <viewPoint>2.0 4.0 7.0</viewPoint>
  <viewDir>–2.0 –4.0 –7.0</viewDir>
  <viewUp>0.0 1.0 0.0</viewUp>
  <viewWidth>8.0</viewWidth>
  <viewHeight>4.5</viewHeight>
</camera>
```



```
<camera type="ParallelCamera">
  <viewPoint>2.0 4.0 7.0</viewPoint>
  <viewDir>–2.0 –4.0 –7.0</viewDir>
  <projNormal>0.0 0.0 1.0</projNormal>
  <viewUp>0.0 1.0 0.0</viewUp>
  <viewWidth>8.0</viewWidth>
  <viewHeight>4.5</viewHeight>
</camera>
```



History of projection

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

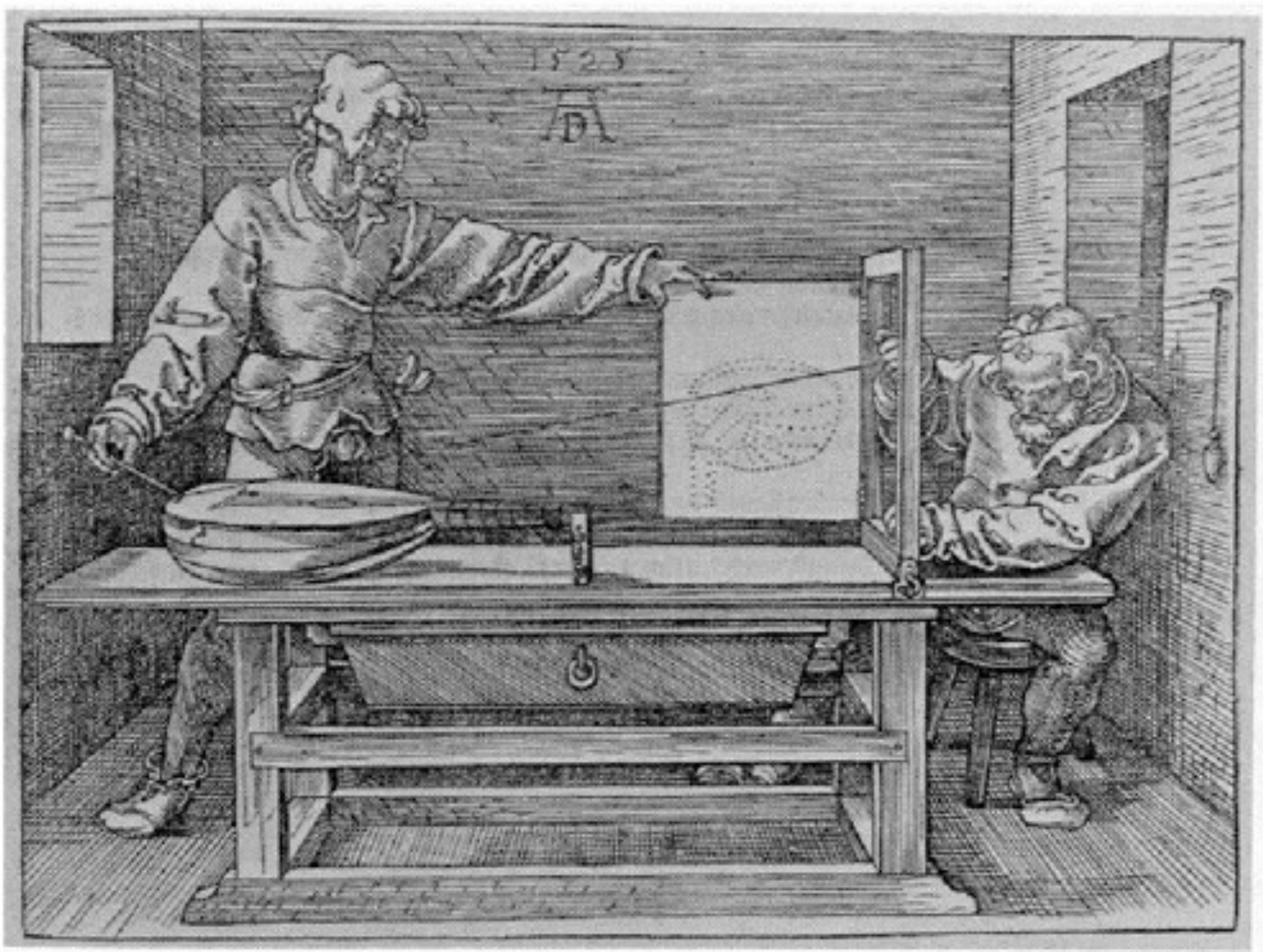
Duccio c. 1308

History of projection

- Later Renaissance: perspective formalized precisely

da Vinci c. 1498

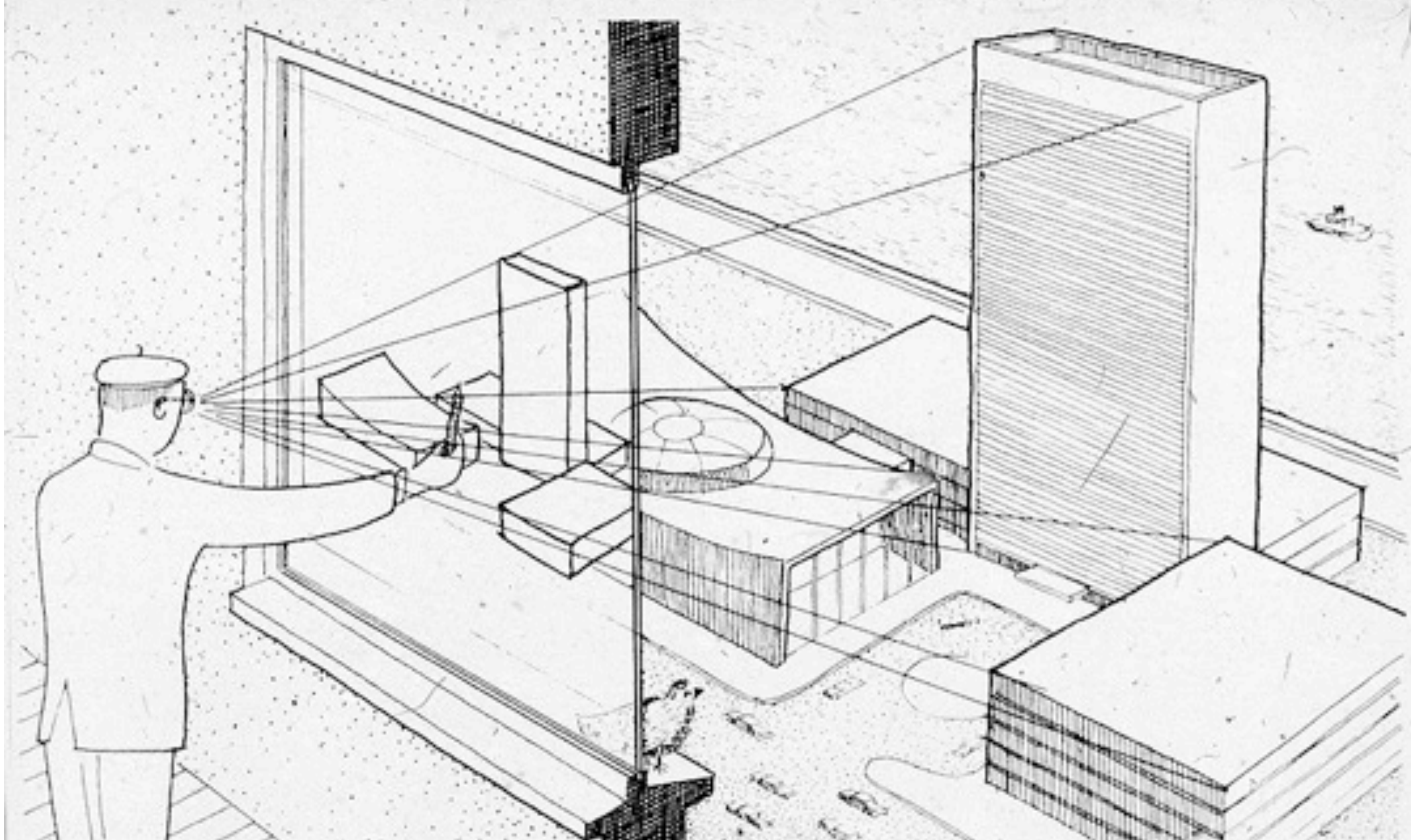
Plane projection in drawing



Albrecht Dürer

Plane projection in drawing

Basis Of Perspective — Lines Of Sight Through A Picture Plane [19]

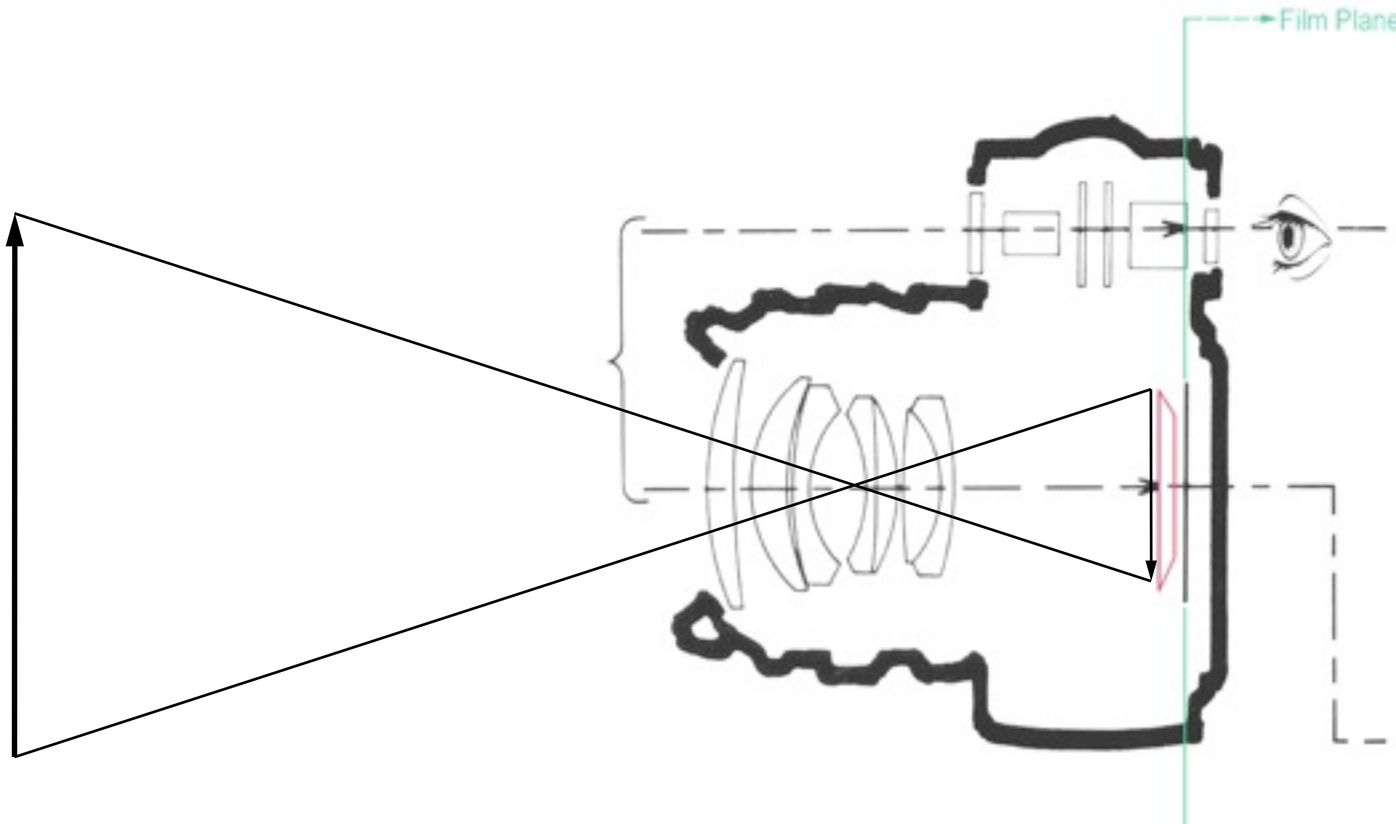


The concept of the picture plane may be better understood by looking through a window or other transparent plane from a fixed viewpoint. Your lines of sight, the multitude of straight lines leading from your eye to the subject, will all intersect this plane. Therefore, if you were to reach out with a 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.

source unknown

Plane projection in photography

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



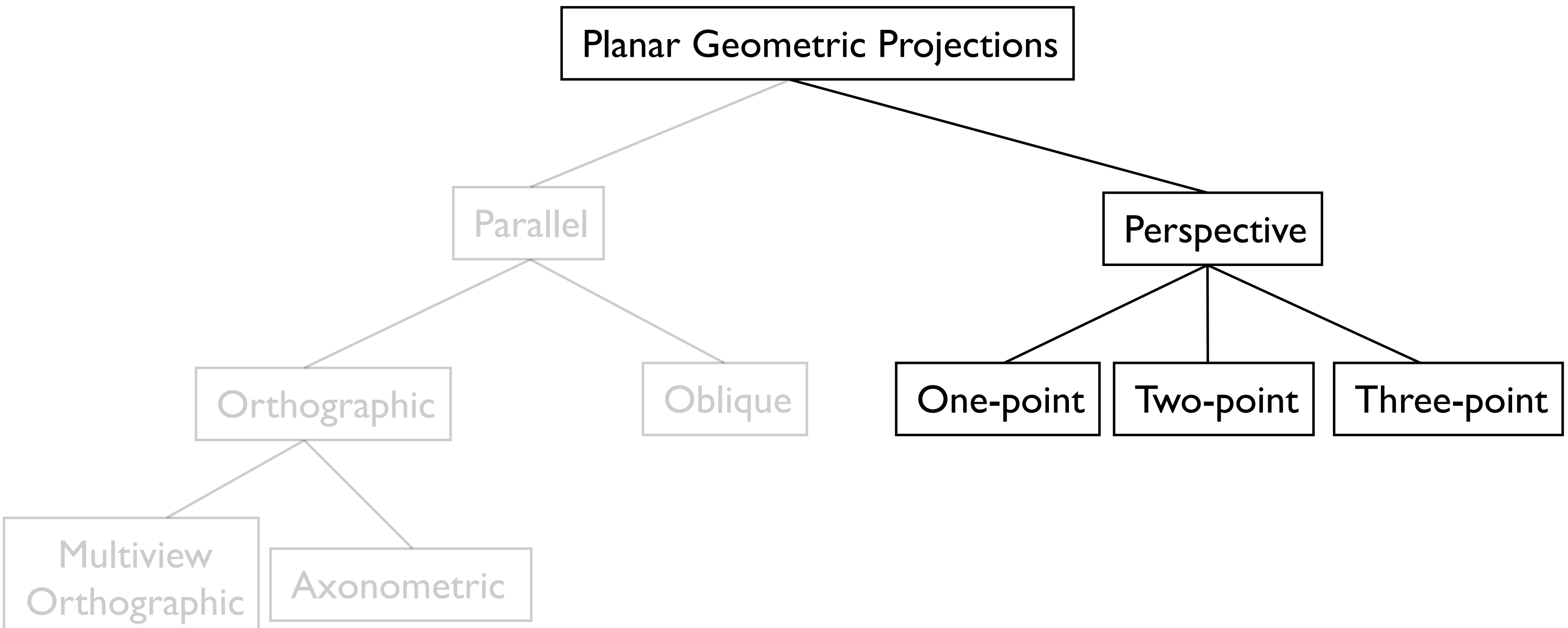
[Source unknown]

Plane projection in photography

[Richard Zakia]

Classical projections—perspective

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



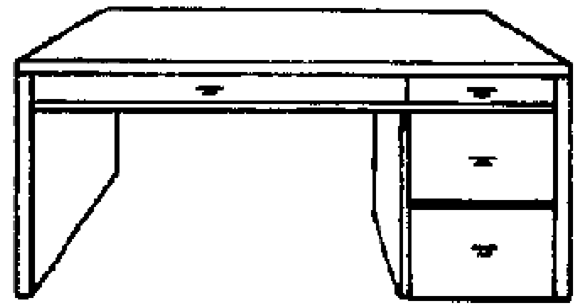
[after Carlbom & Paciorek 78]

Perspective

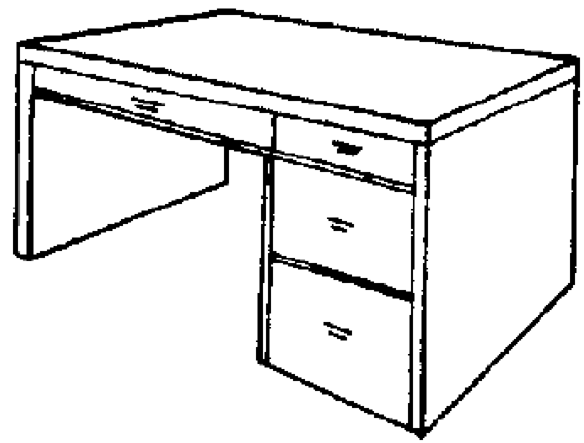
one-point: projection plane parallel to a coordinate plane (to two coordinate axes)

two-point: projection plane parallel to one coordinate axis

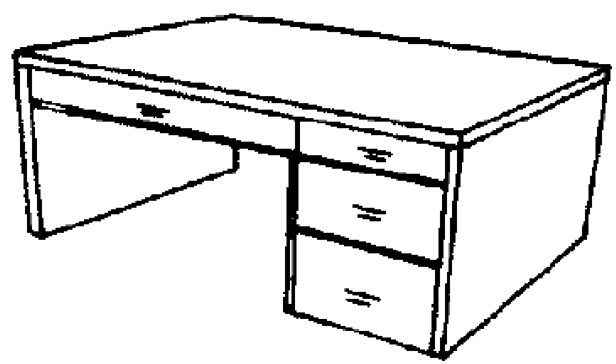
three-point: projection plane not parallel to a coordinate axis



one-point



two-point

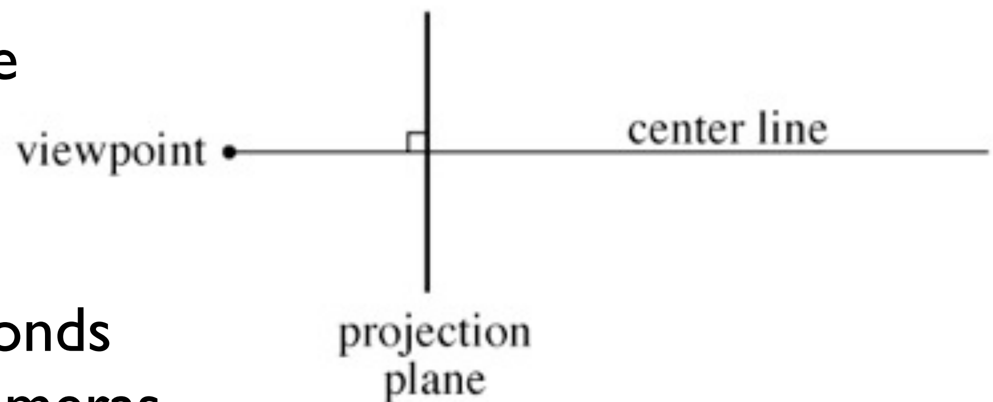


three-point

[Carlbom & Paciorek 78]

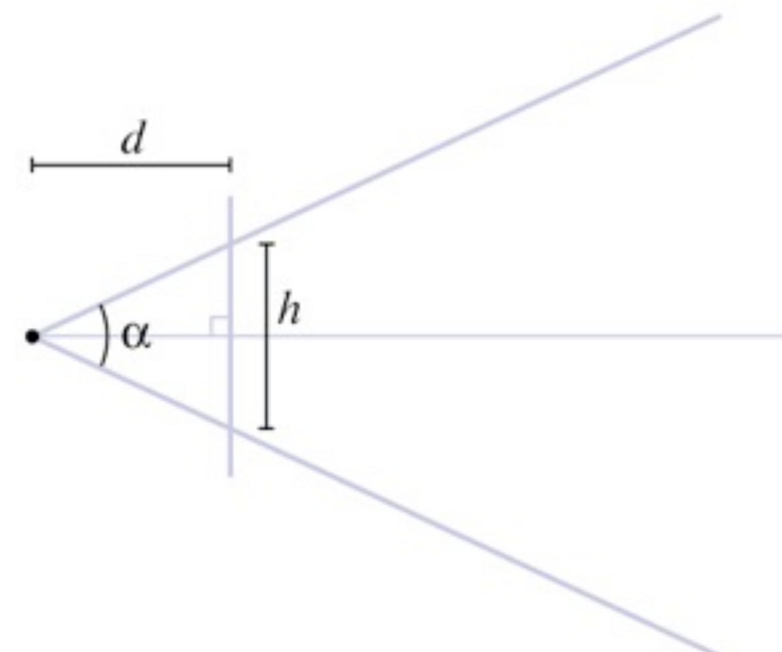
Perspective projection (normal)

- Perspective is projection by lines through a point; "normal" = plane perpendicular to view direction
  - magnification determined by:
    - image height
    - object depth
    - image plane distance
  - f.o.v. $\alpha = 2 \operatorname{atan}(h/(2d))$
  - $y' = d\, y / z$
  - "normal" case corresponds to common types of cameras



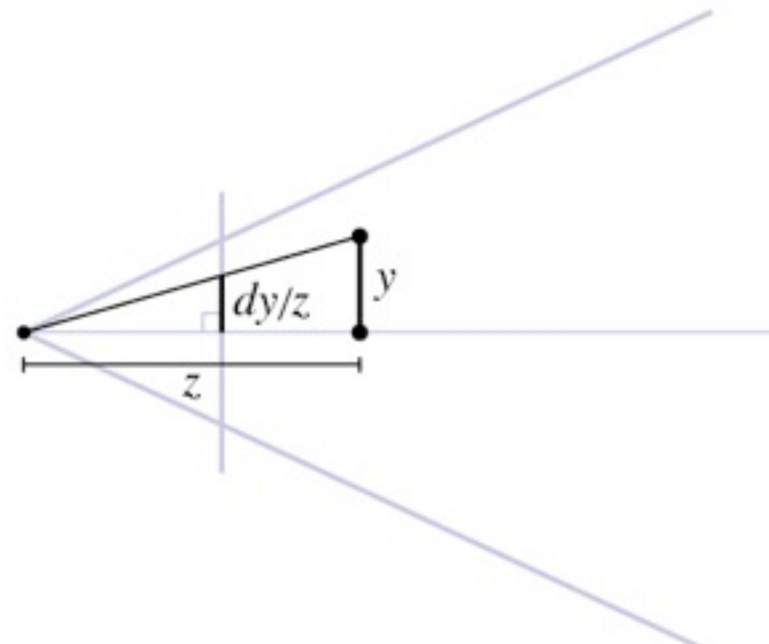
Perspective projection (normal)

- Perspective is projection by lines through a point; "normal" = plane perpendicular to view direction
  - magnification determined by:
    - image height
    - object depth
    - image plane distance
  - f.o.v. $\alpha = 2\ \mathrm{atan}(h/(2d))$
  - $y' = d\,y / z$
  - "normal" case corresponds to common types of cameras



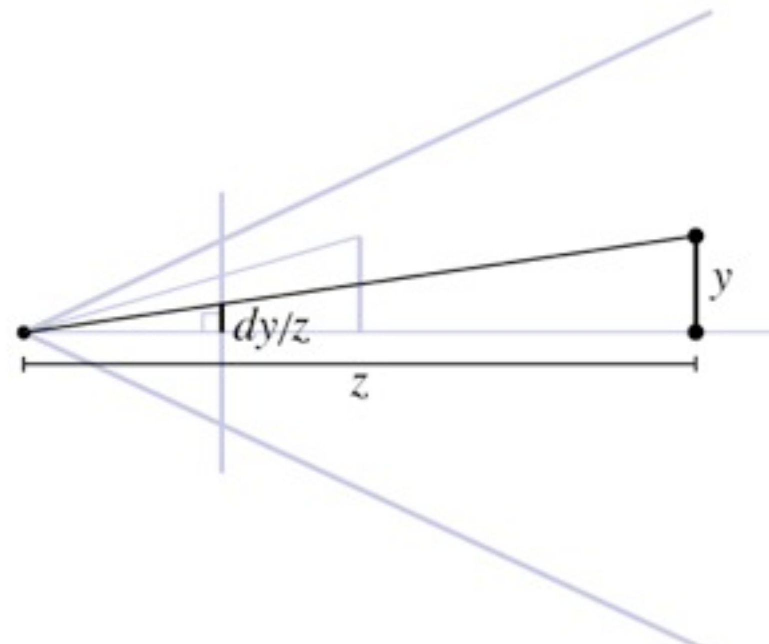
Perspective projection (normal)

- Perspective is projection by lines through a point; "normal" = plane perpendicular to view direction
  - magnification determined by:
    - image height
    - object depth
    - image plane distance
  - f.o.v. $\alpha = 2 \operatorname{atan}(h/(2d))$
  - $y' = d\, y / z$
  - "normal" case corresponds to common types of cameras



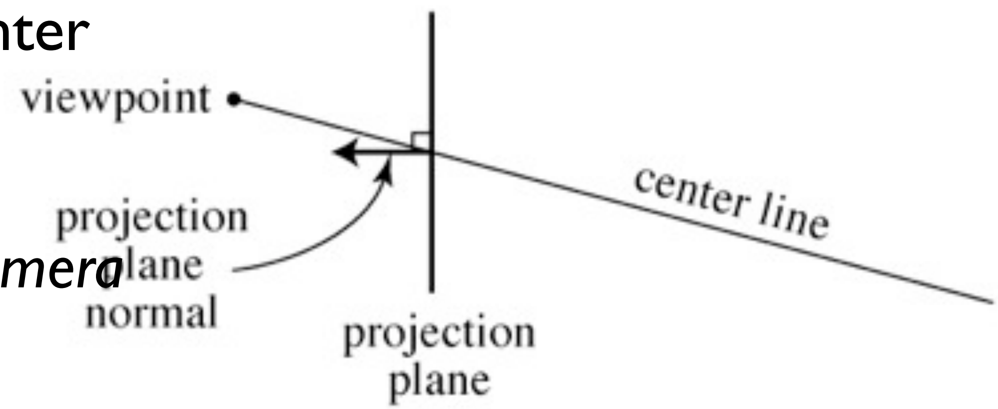
Perspective projection (normal)

- Perspective is projection by lines through a point; "normal" = plane perpendicular to view direction
  - magnification determined by:
    - image height
    - object depth
    - image plane distance
  - f.o.v. $\alpha = 2 \operatorname{atan}(h/(2d))$
  - $y' = d\, y / z$
  - "normal" case corresponds to common types of cameras



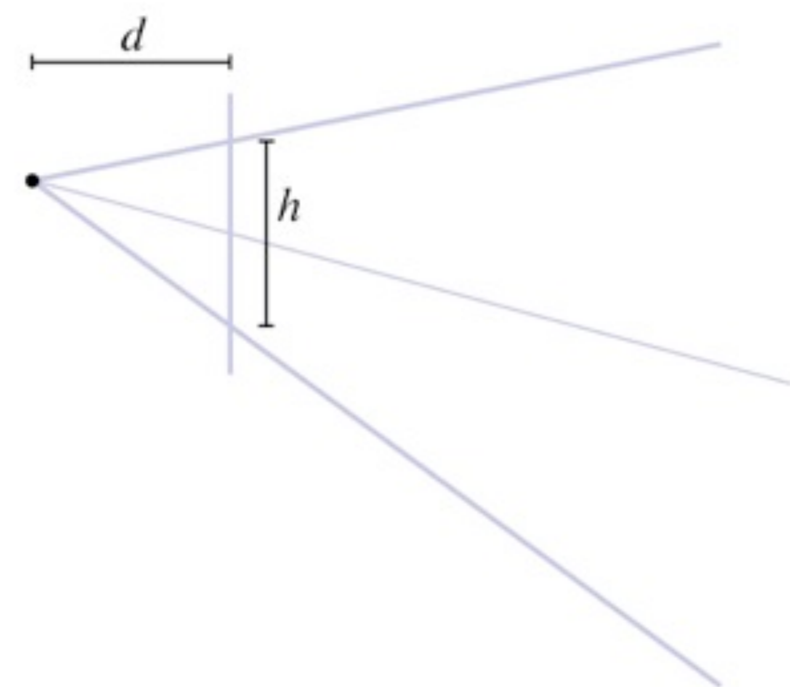
Shifted perspective projection

- Perspective but with projection plane not perpendicular to view direction
  - additional parameter: projection plane normal
  - exactly equivalent to cropping out an off-center rectangle from a larger "normal" perspective
  - corresponds to *view camera* in photography



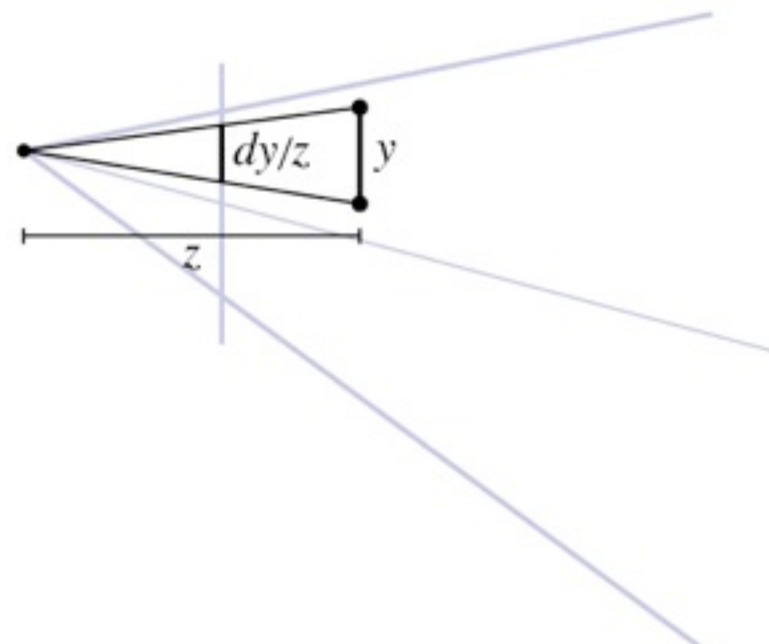
Shifted perspective projection

- Perspective but with projection plane not perpendicular to view direction
  - additional parameter: projection plane normal
  - exactly equivalent to cropping out an off-center rectangle from a larger “normal” perspective
  - corresponds to *view camera* in photography



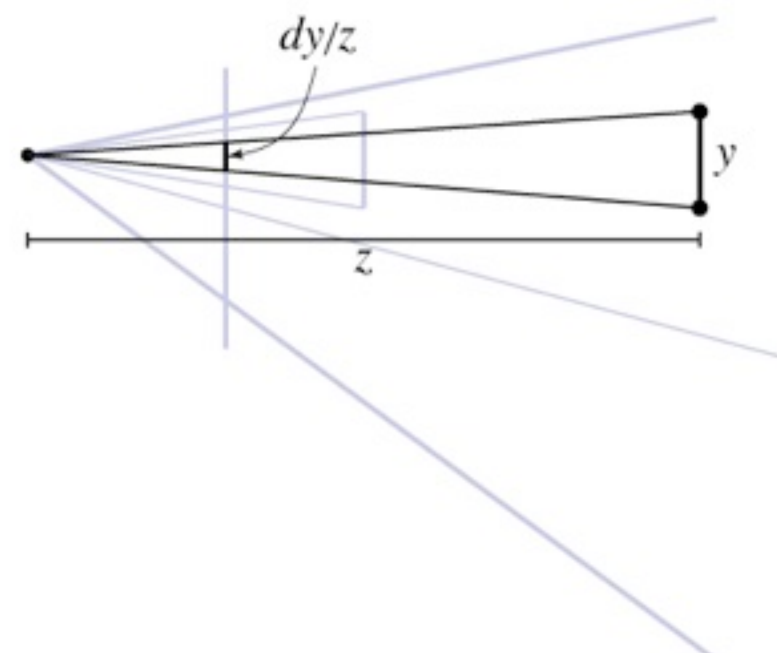
Shifted perspective projection

- Perspective but with projection plane not perpendicular to view direction
  - additional parameter: projection plane normal
  - exactly equivalent to cropping out an off-center rectangle from a larger "normal" perspective
  - corresponds to *view camera* in photography



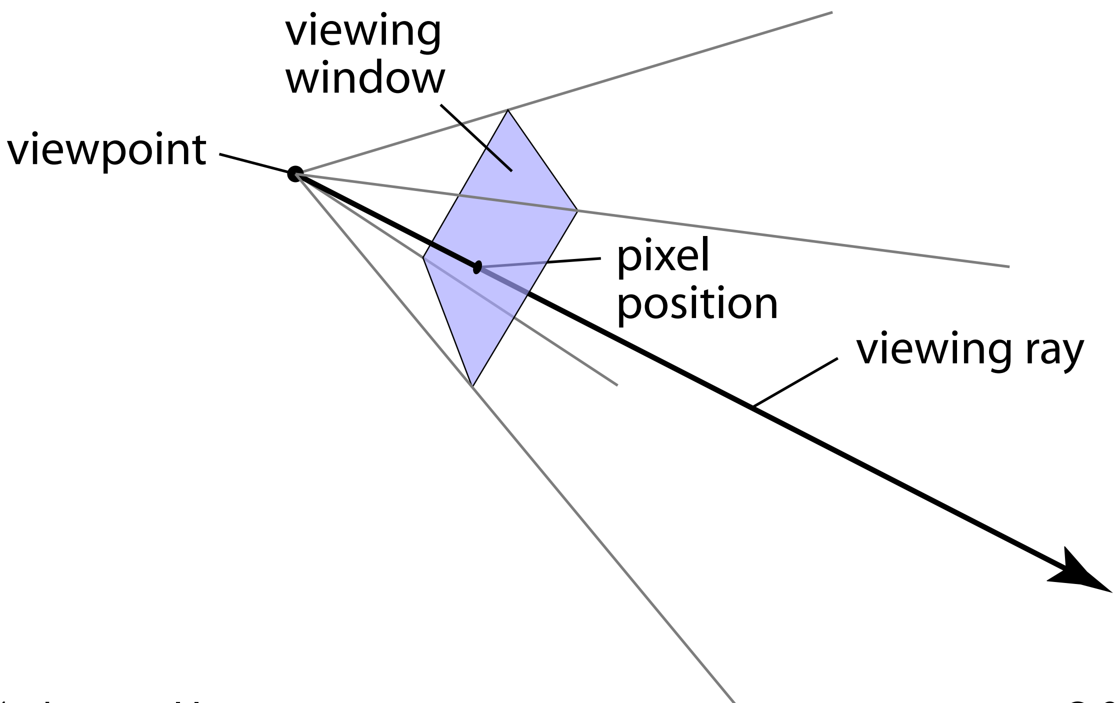
Shifted perspective projection

- Perspective but with projection plane not perpendicular to view direction
  - additional parameter: projection plane normal
  - exactly equivalent to cropping out an off-center rectangle from a larger "normal" perspective
  - corresponds to *view camera* in photography



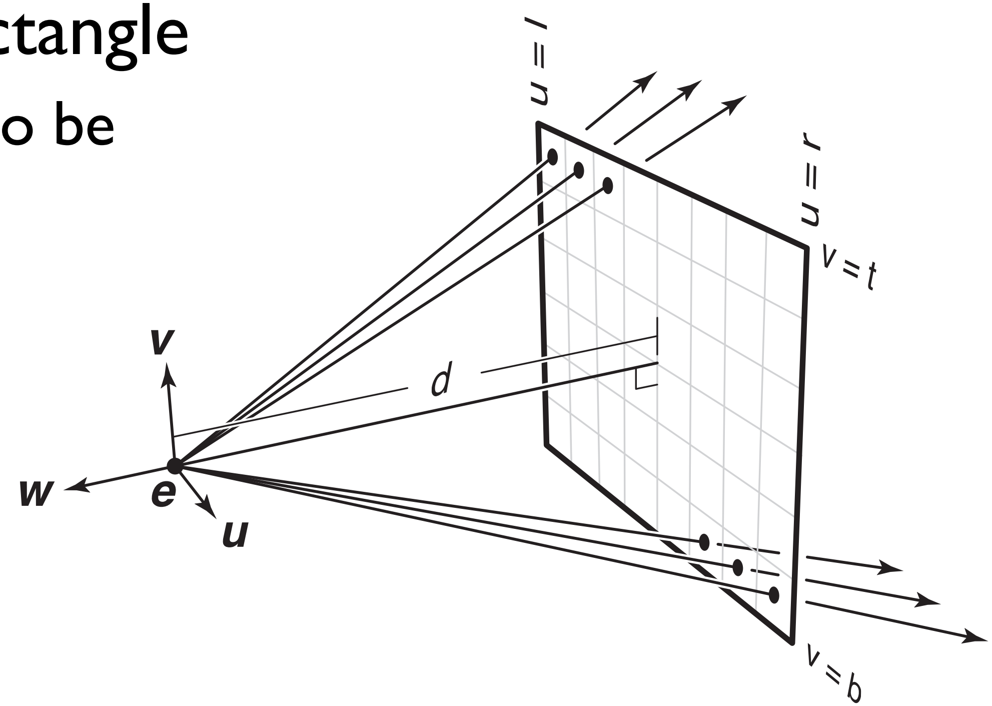
Generating eye rays—perspective

- Use window analogy directly
- Ray origin (constant): viewpoint
- Ray direction (varying): toward pixel position on viewing window



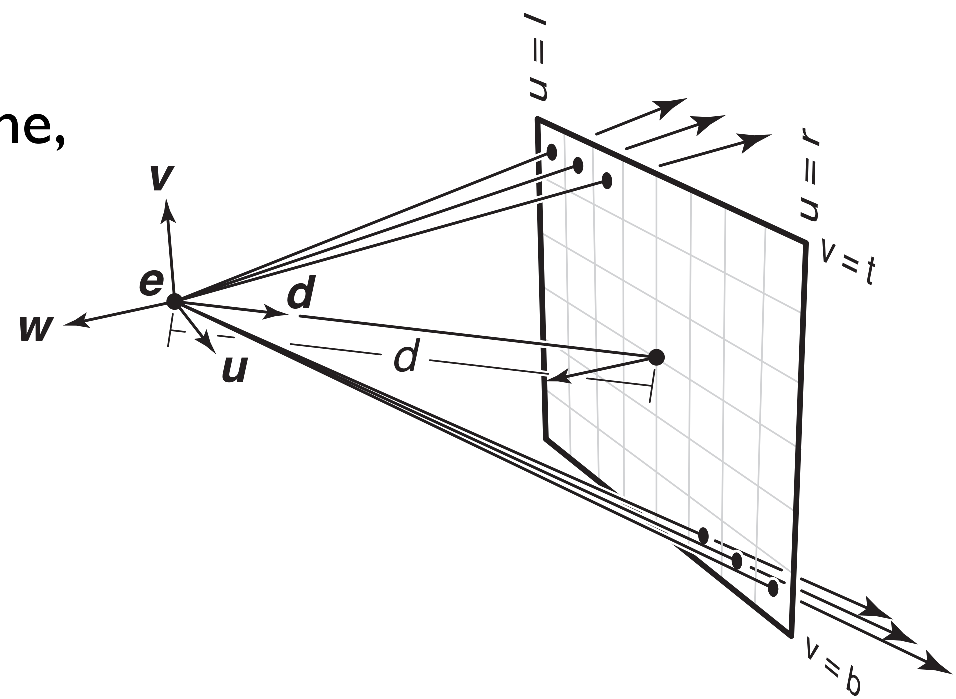
Generating eye rays—perspective

- Positioning the view rectangle
  - establish three vectors to be *camera basis:* **u**, **v**, **w**
  - view rectangle is parallel to **u**–**v** plane, at $w = -d$, specified by l, r, t, b
- Generating rays
  - for (u, v) in $[l, r] \times [b, t]$
  - ray.origin = **e**
  - ray.direction = $-d\,\mathbf{w} + u\,\mathbf{u} + v\,\mathbf{v}$



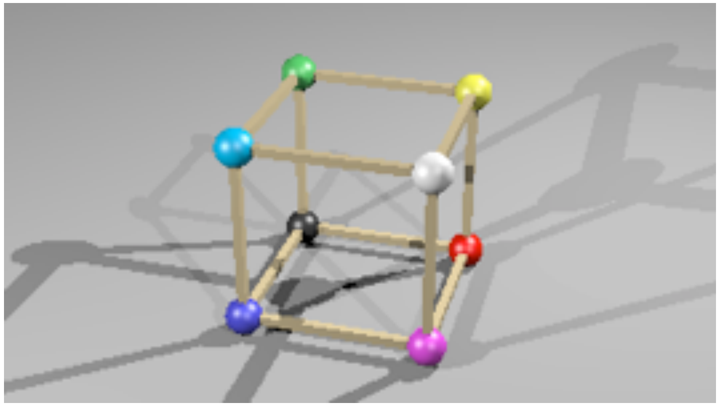
Oblique perspective views

- Positioning the view rectangle
  - establish three vectors to be *camera basis:* **u**, **v**, **w**
  - view rectangle is the same, but shifted so that the center is in the direction **d** from **e**
- Generating rays
  - for (u, v) in $[l, r] \times [b, t]$
  - ray.origin = **e**
  - ray.direction = $d\,\mathbf{d} + u\,\mathbf{u} + v\,\mathbf{v}$

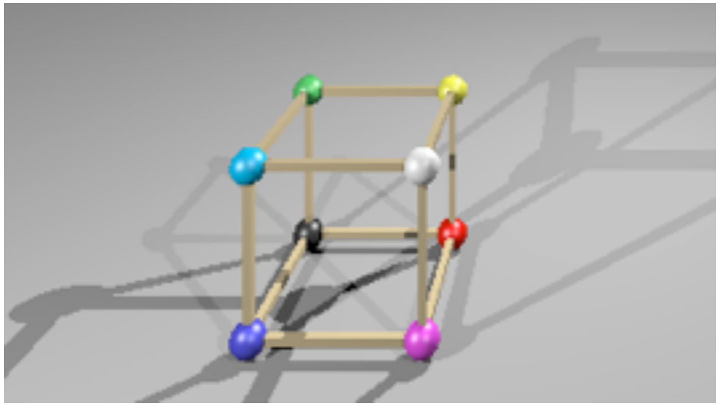


Perspective views in ray tracing

```
<camera type="PerspectiveCamera">
  <viewPoint>3.0 6.0 10.5</viewPoint>
  <viewDir>−3.0 −6.0 −10.5</viewDir>
  <viewUp>0.0 1.0 0.0</viewUp>
  <projDistance>13.0</projDistance>
  <viewWidth>8.0</viewWidth>
  <viewHeight>4.5</viewHeight>
</camera>
```



```
<camera type="PerspectiveCamera">
  <viewPoint>3.0 6.0 10.5</viewPoint>
  <viewDir>−3.0 −6.0 −10.5</viewDir>
  <projNormal>0.0 0.0 1.0</projNormal>
  <viewUp>0.0 1.0 0.0</viewUp>
  <projDistance>11.0</projDistance>
  <viewWidth>8.0</viewWidth>
  <viewHeight>4.5</viewHeight>
</camera>
```



Field of view (or f.o.v.)

- The angle between the rays corresponding to opposite edges of a perspective image
  - simpler to compute for "normal" perspective
  - have to decide to measure vert., horiz., or diag.
- In cameras, determined by focal length
  - confusing because of many image sizes
  - for 35mm format (36mm by 24mm image)
    - 18mm = 67° v.f.o.v. — super-wide angle
    - 28mm = 46° v.f.o.v. — wide angle
    - 50mm = 27° v.f.o.v. — "normal"
    - 100mm = 14° v.f.o.v. — narrow angle ("telephoto")

Field of view

- Determines “strength” of perspective effects

close viewpoint
wide angle
prominent foreshortening

far viewpoint
narrow angle
little foreshortening

[Ansel Adams]

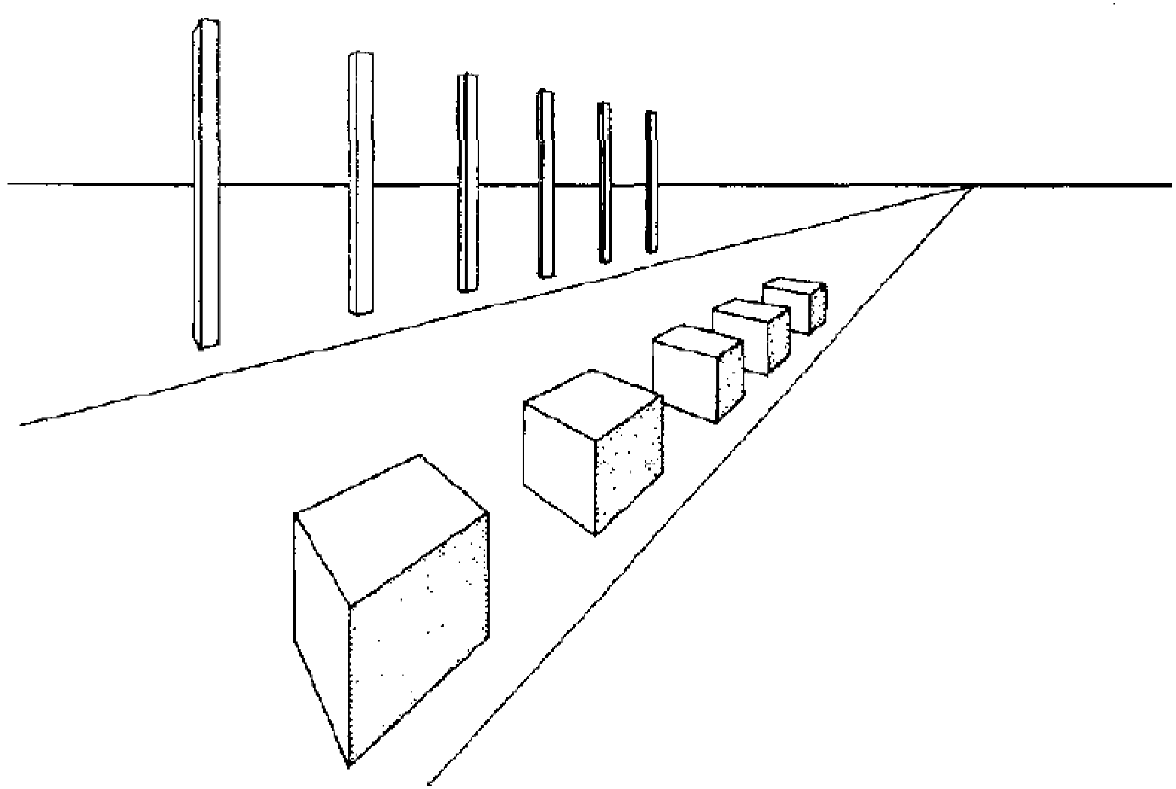
Choice of field of view

- In photography, wide angle lenses are specialty tools
  - "hard to work with"
  - easy to create weird-looking perspective effects
- In graphics, you can type in whatever f.o.v. you want
  - and people often type in big numbers!

[Ken Perlin]

Perspective distortions

- Lengths, length ratios



[Carlbom & Paciorek 78]

Why shifted perspective?

- Control convergence of parallel lines
- Standard example: architecture
  - buildings are taller than you, so you look up
  - top of building is farther away, so it looks smaller
- Solution: make projection plane parallel to facade
  - top of building is the same distance *from the projection plane*
- Same perspective effects can be achieved using post-processing
  - (though not the focus effects)
  - choice of *which* rays vs. arrangement of rays in image

[Philip Greenspun]

camera tilted up: converging vertical lines

[Philip Greenspun]

lens shifted up: parallel vertical lines