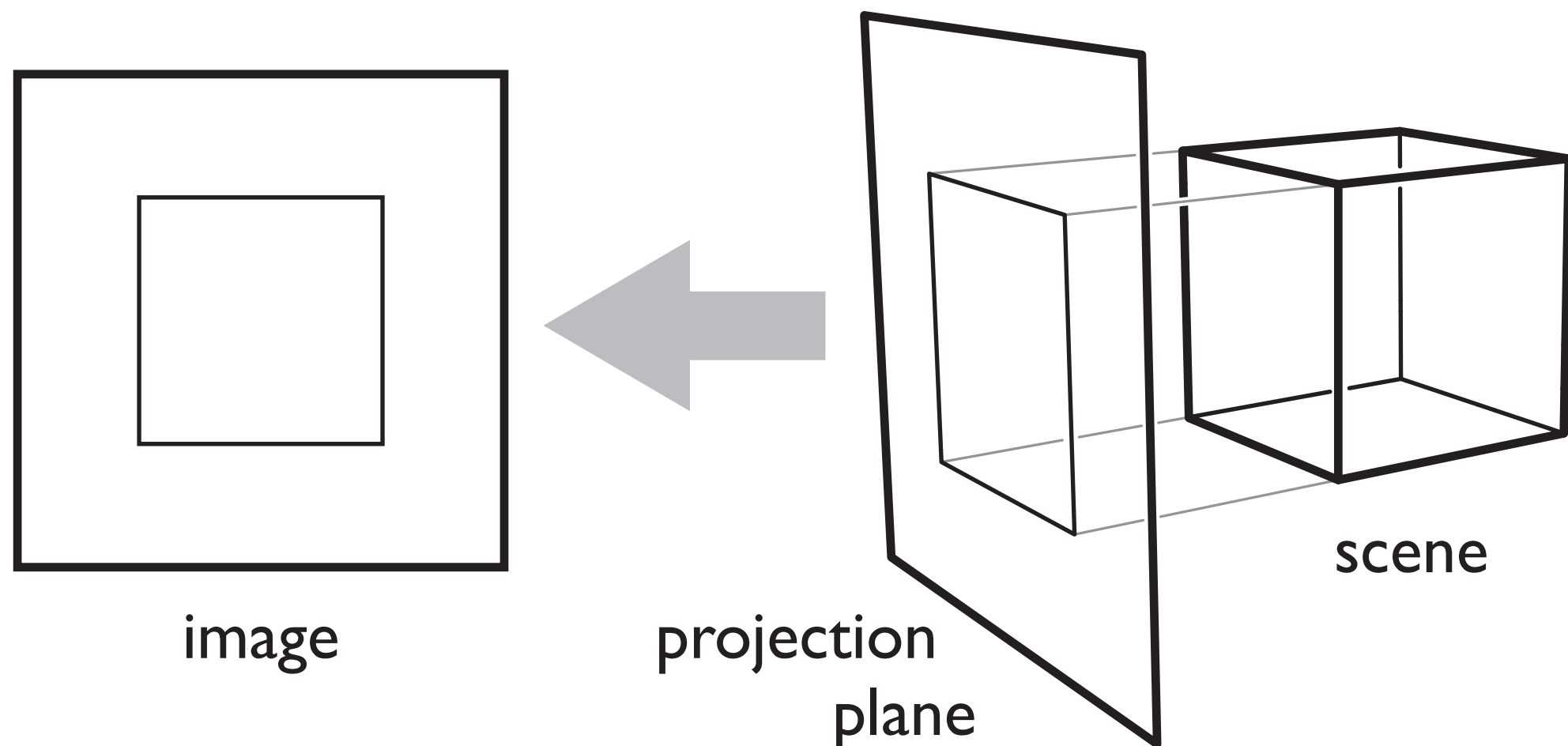


Perspective

CS 4620 Lecture 8

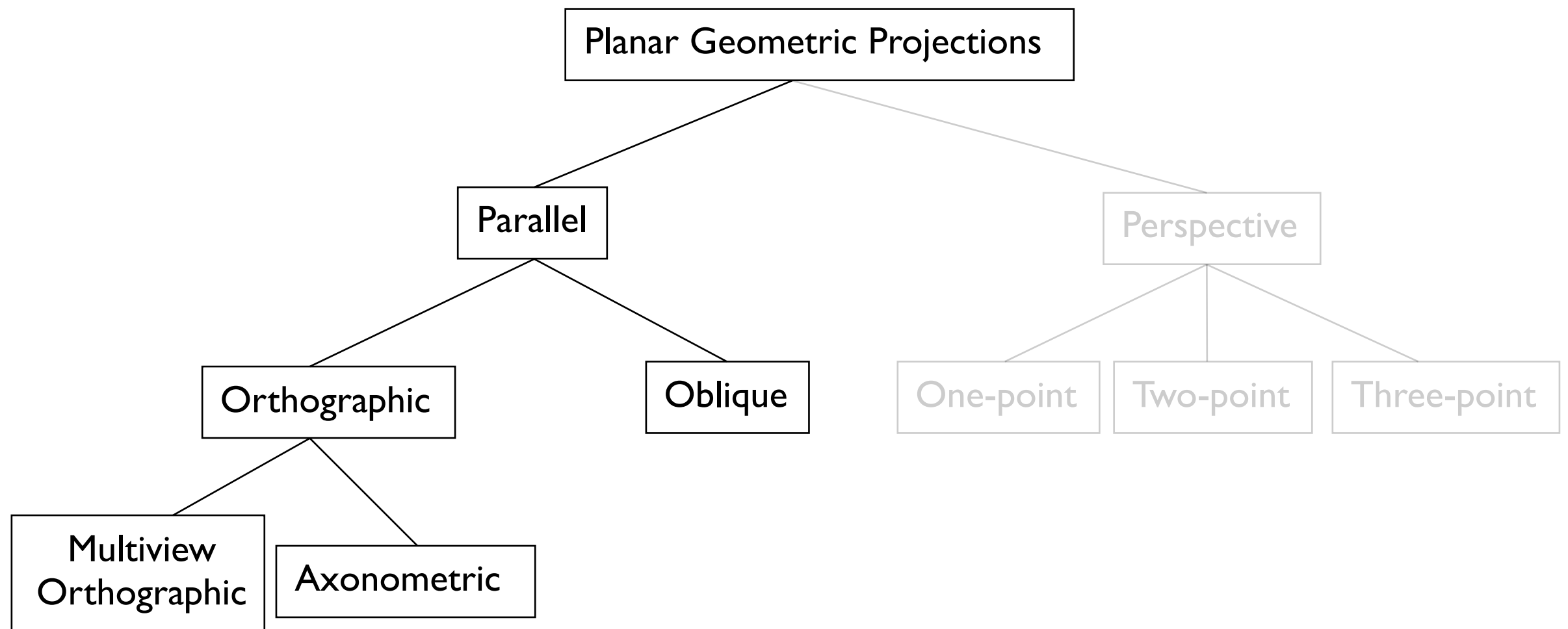
Parallel projection

- 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



Orthographic

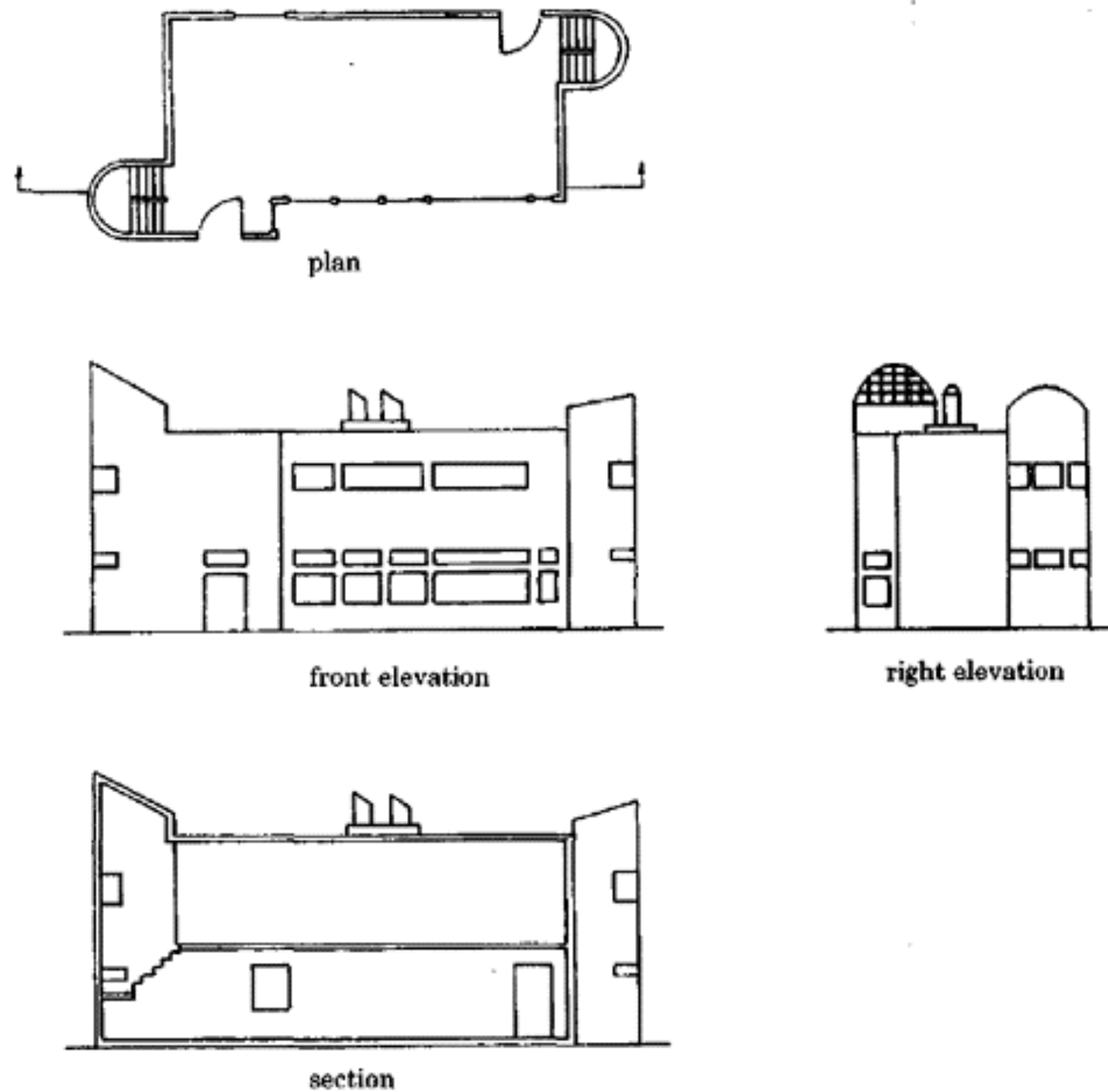
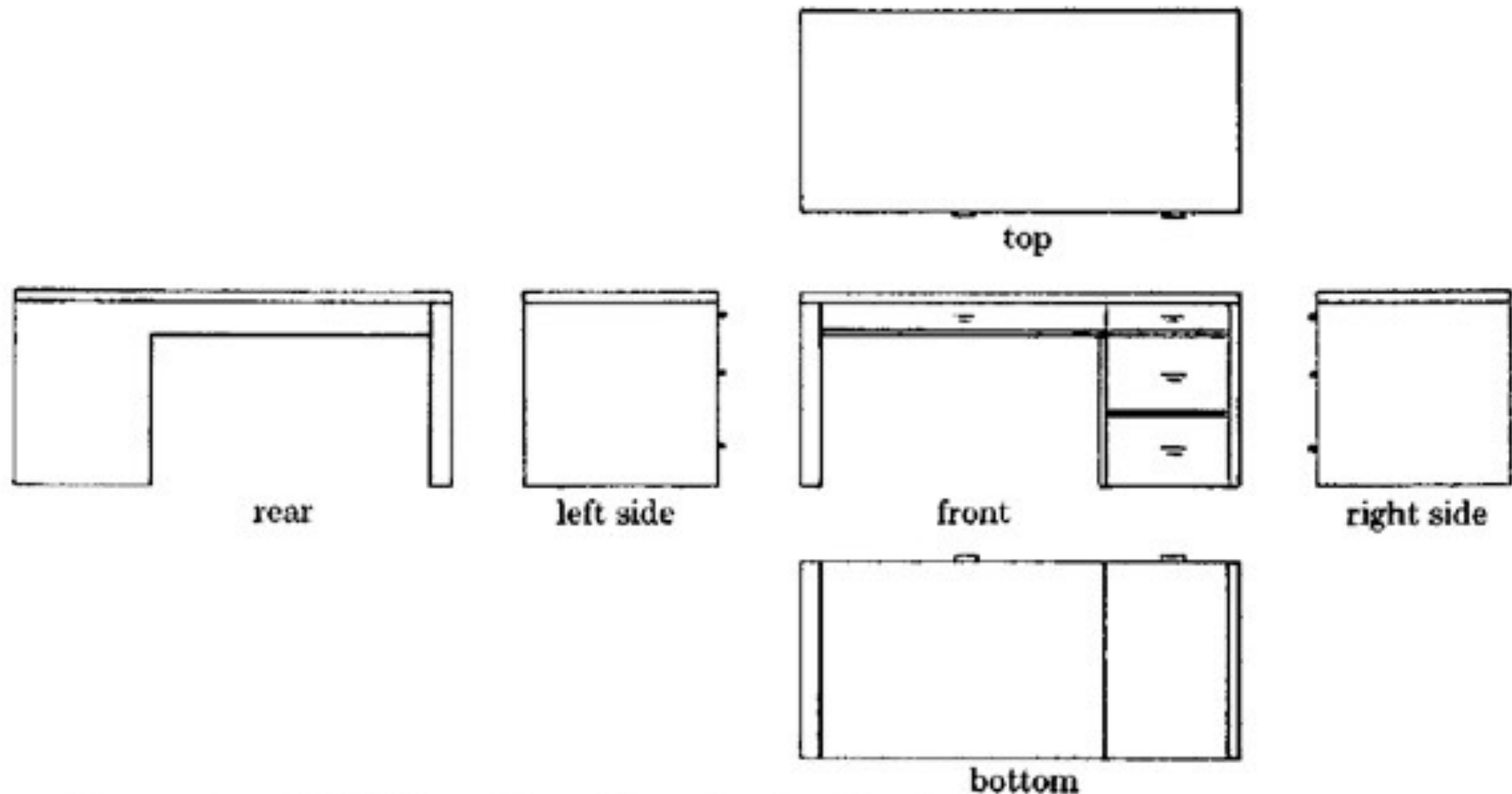


FIGURE 2-1. Multiview orthographic projection: plan, elevations, and section of a building.

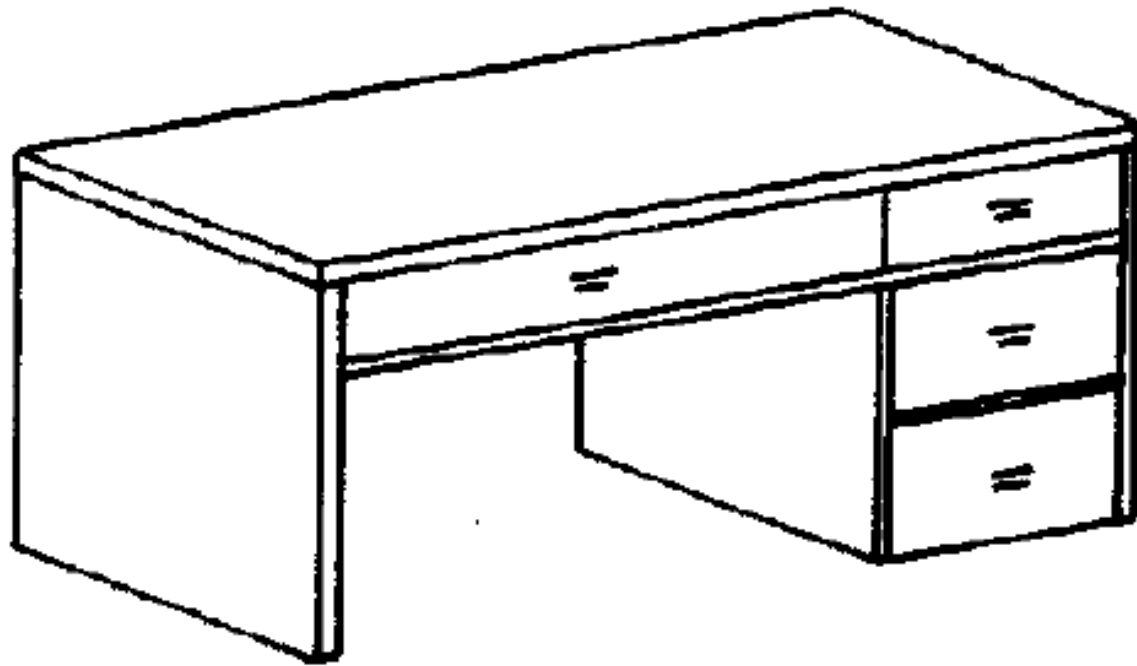
Orthographic



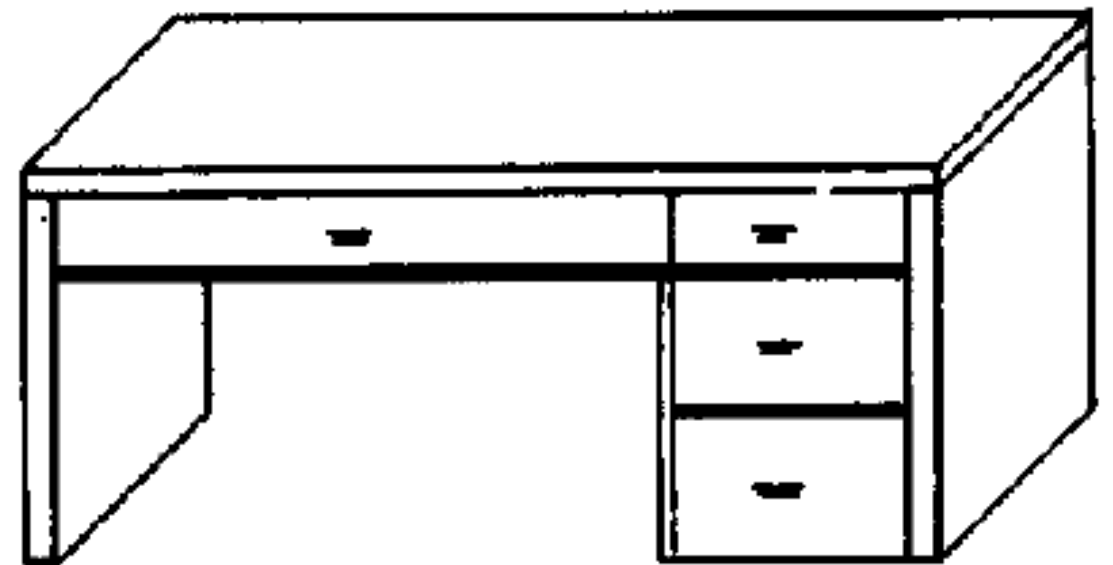
[Carlbom & Paciorek 78]

- 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

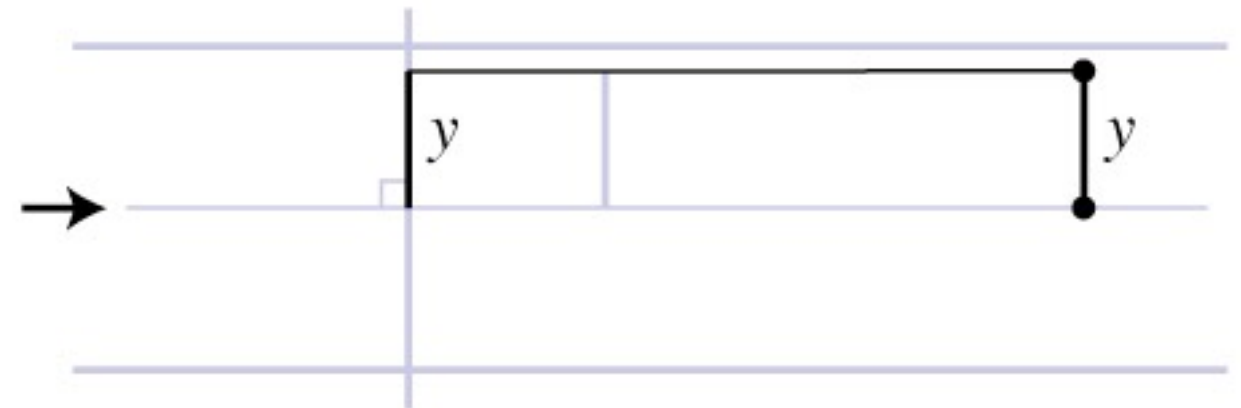


oblique: projection plane parallel to a coordinate plane but not perpendicular to projection direction.

[Carl bom & Paciorek 78]

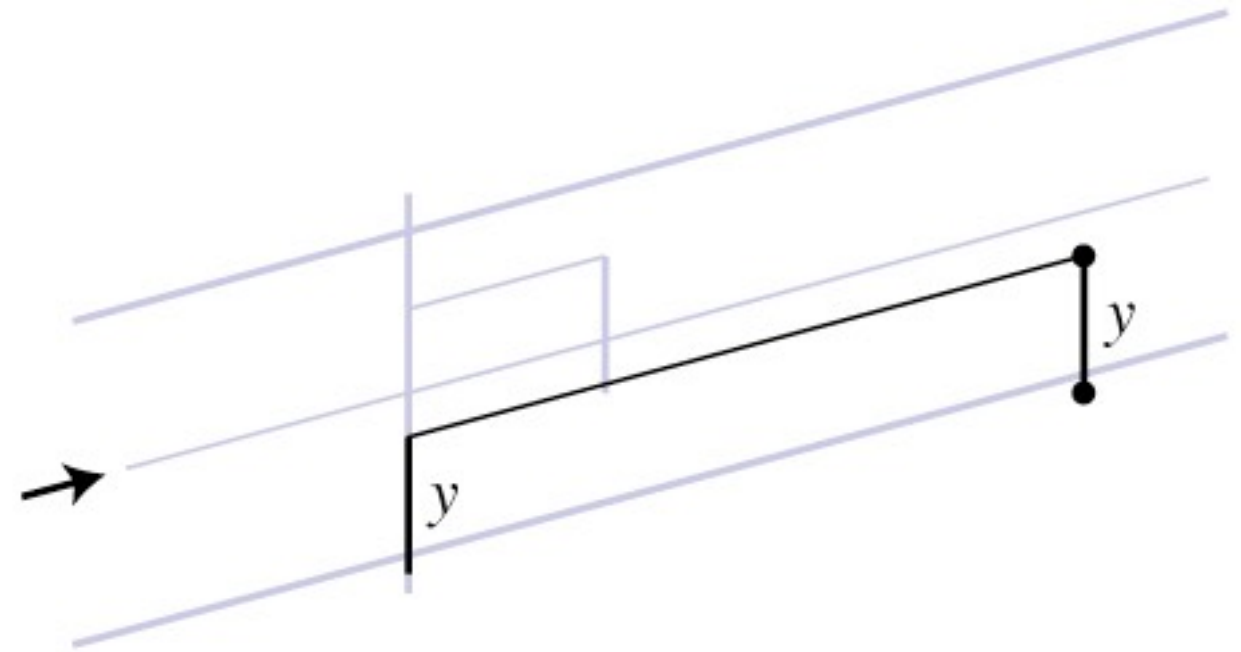
“Orthographic” projection

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



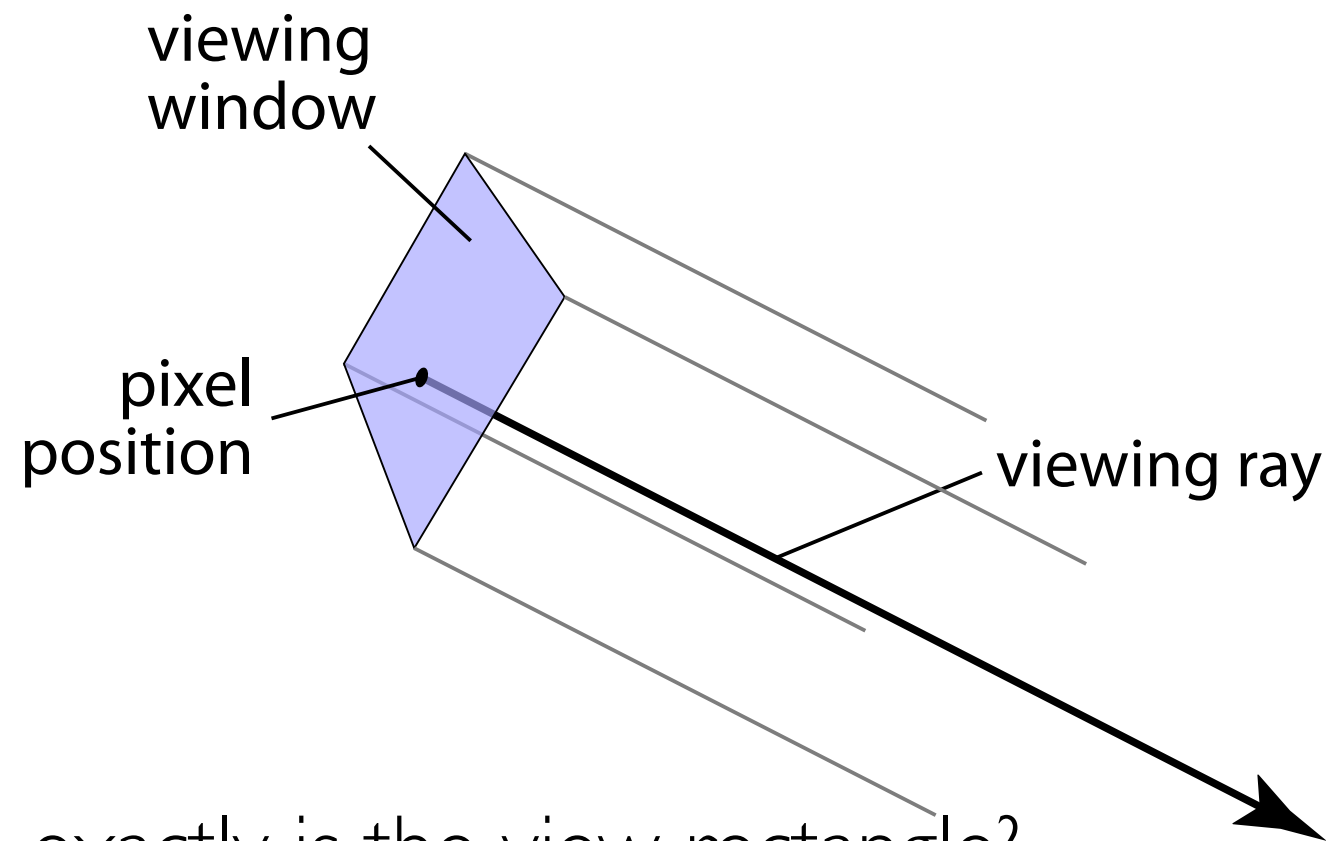
Oblique projection

- **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?

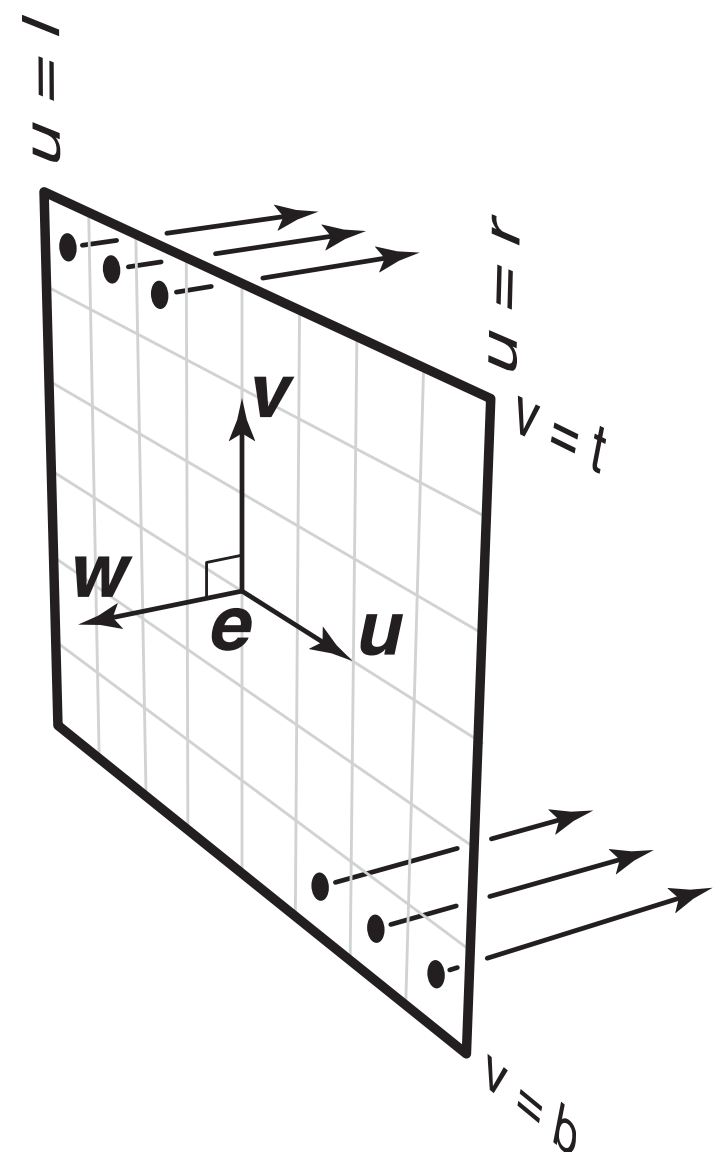
Generating eye rays—orthographic

- **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 $b = -t$)

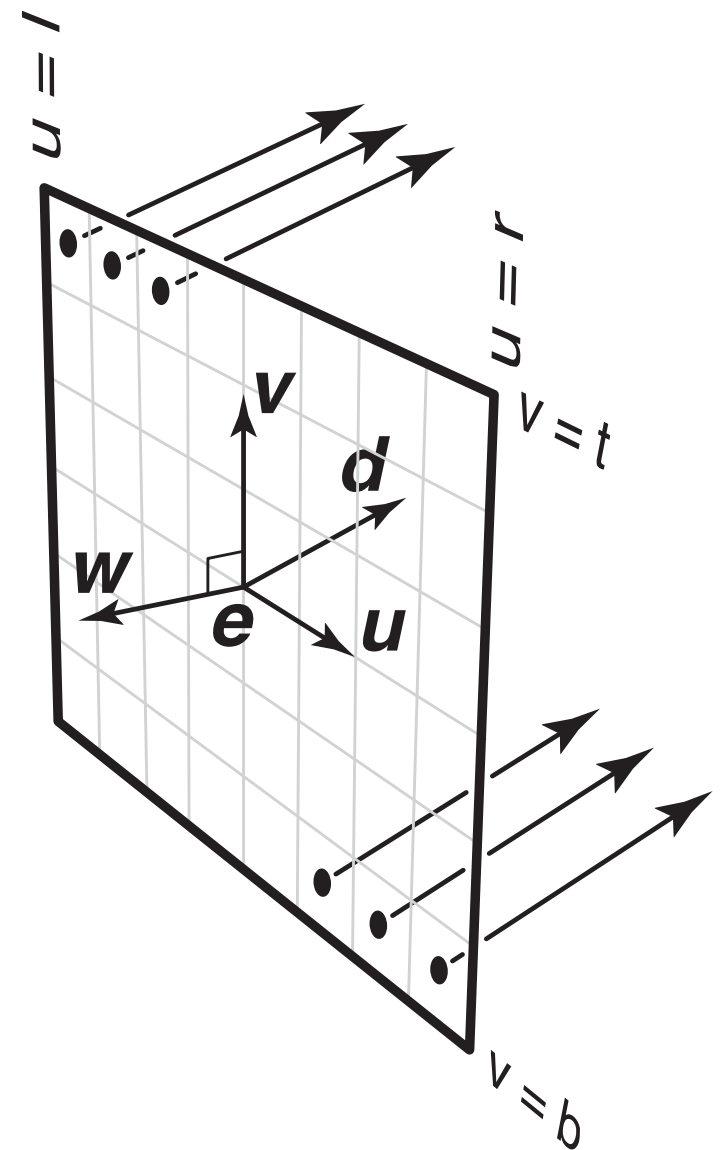
- **Generating rays**

- for (u, v) in $[l, r] \times [b, t]$
- $\text{ray.origin} = \mathbf{e} + u \mathbf{u} + v \mathbf{v}$
- $\text{ray.direction} = -\mathbf{w}$



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**

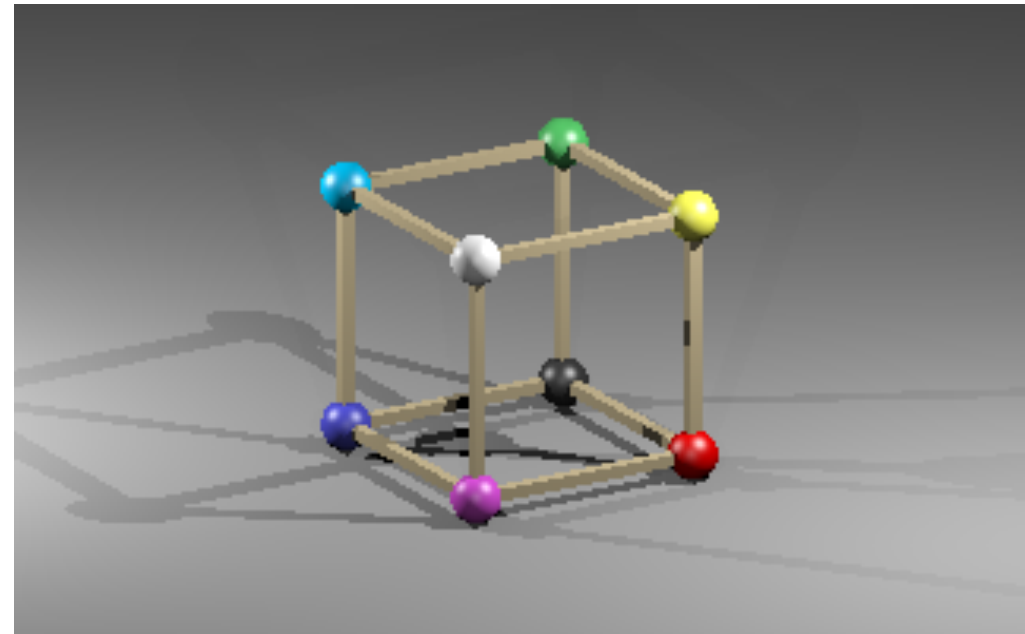


Establishing the camera basis

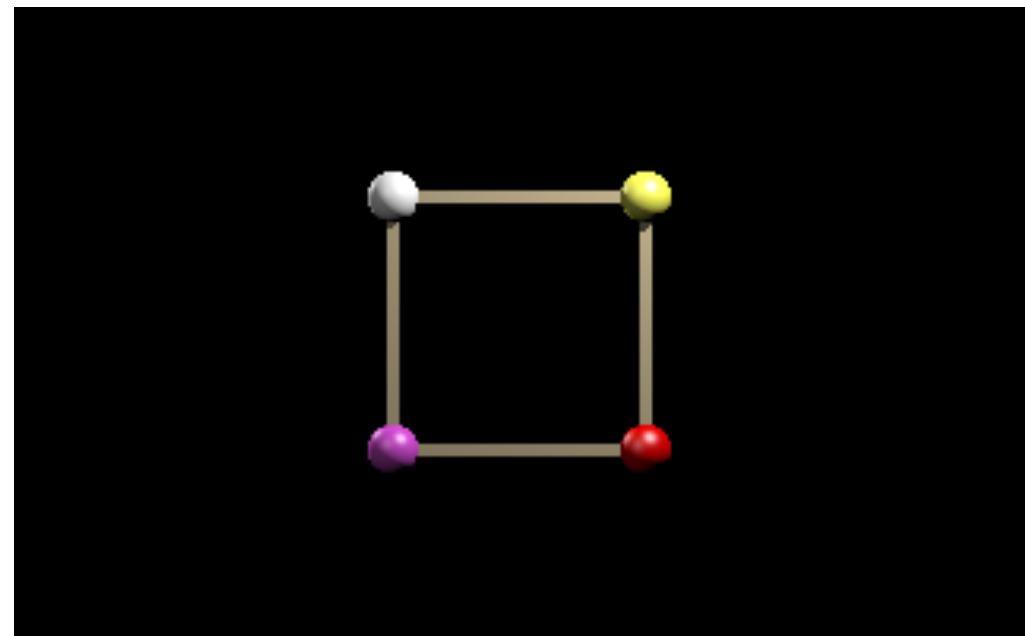
- **Could require user to provide \mathbf{e} , \mathbf{u} , \mathbf{v} , and \mathbf{w}**
 - but this is error prone and unintuitive
- **Instead, calculate basis from things the user cares about**
 - viewpoint: where the camera is $\rightarrow \mathbf{e}$
 - view direction: which way the camera is looking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

Orthographic views in Ray I

```
<camera type="OrthographicCamera">  
  <viewPoint>10 4.2 6</viewPoint>  
  <viewDir>-5 -2.1 -3</viewDir>  
  <viewUp>0 1 0</viewUp>  
  <viewWidth>8</viewWidth>  
  <viewHeight>5</viewHeight>  
</camera>
```



```
<camera type="OrthographicCamera">  
  <viewPoint>10 0 0</viewPoint>  
  <viewDir>-1 0 0</viewDir>  
  <viewUp>0 1 0</viewUp>  
  <viewWidth>8</viewWidth>  
  <viewHeight>5</viewHeight>  
</camera>
```



History of projection

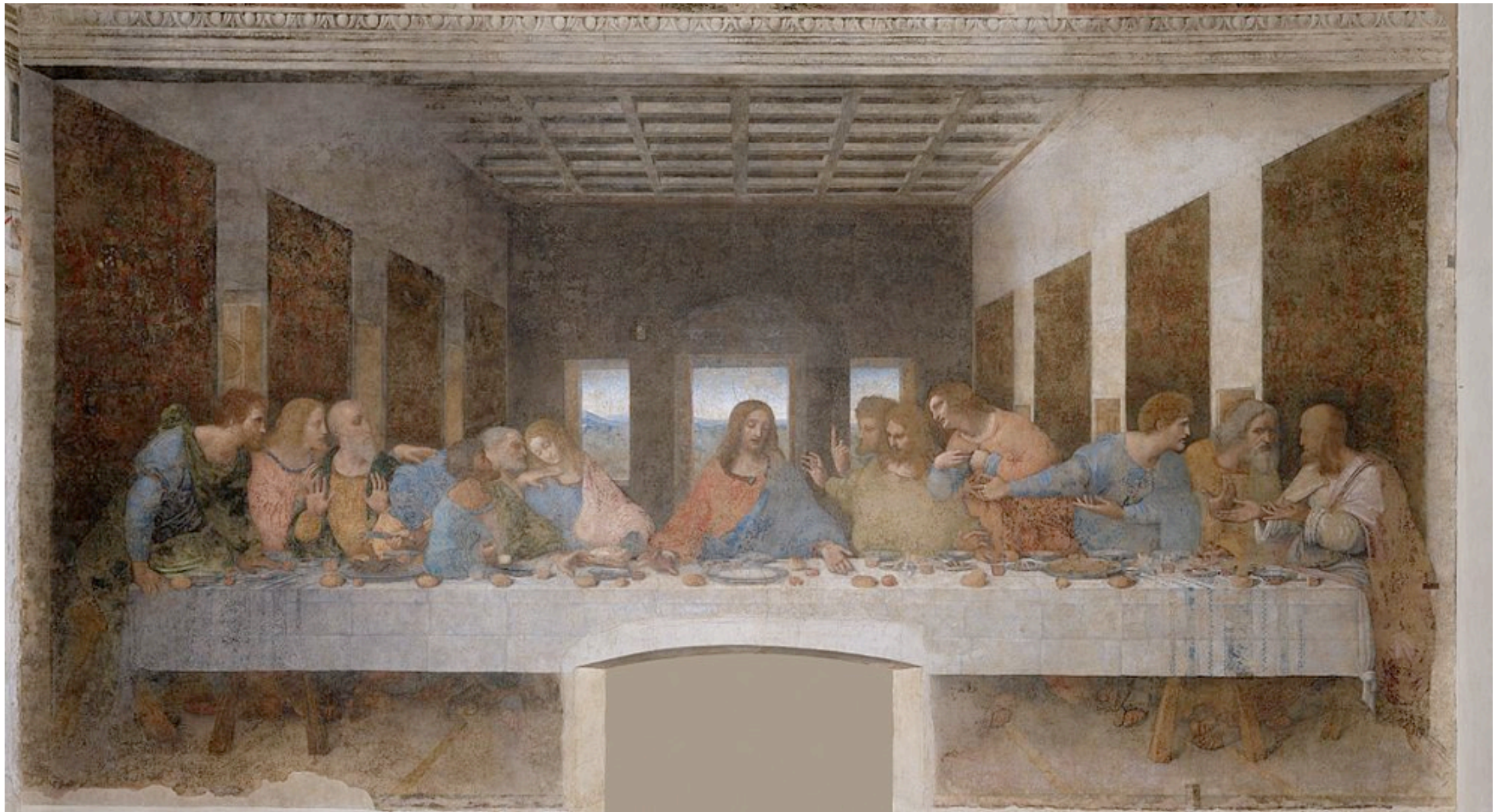
- **Ancient times: Greeks wrote about laws of perspective**
- **Renaissance: perspective is adopted by artists**



Duccio c. 1308

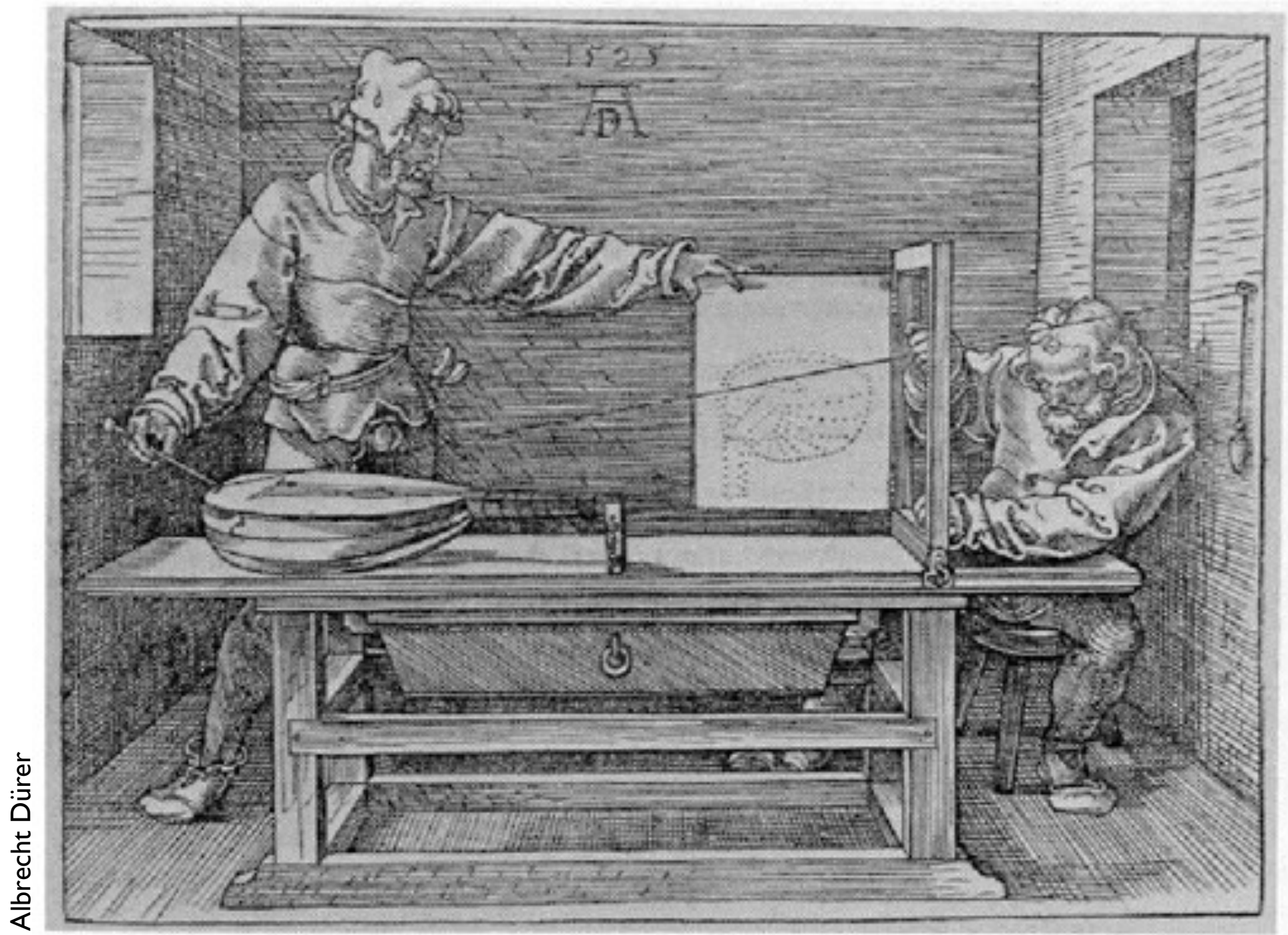
History of projection

- **Later Renaissance: perspective formalized precisely**



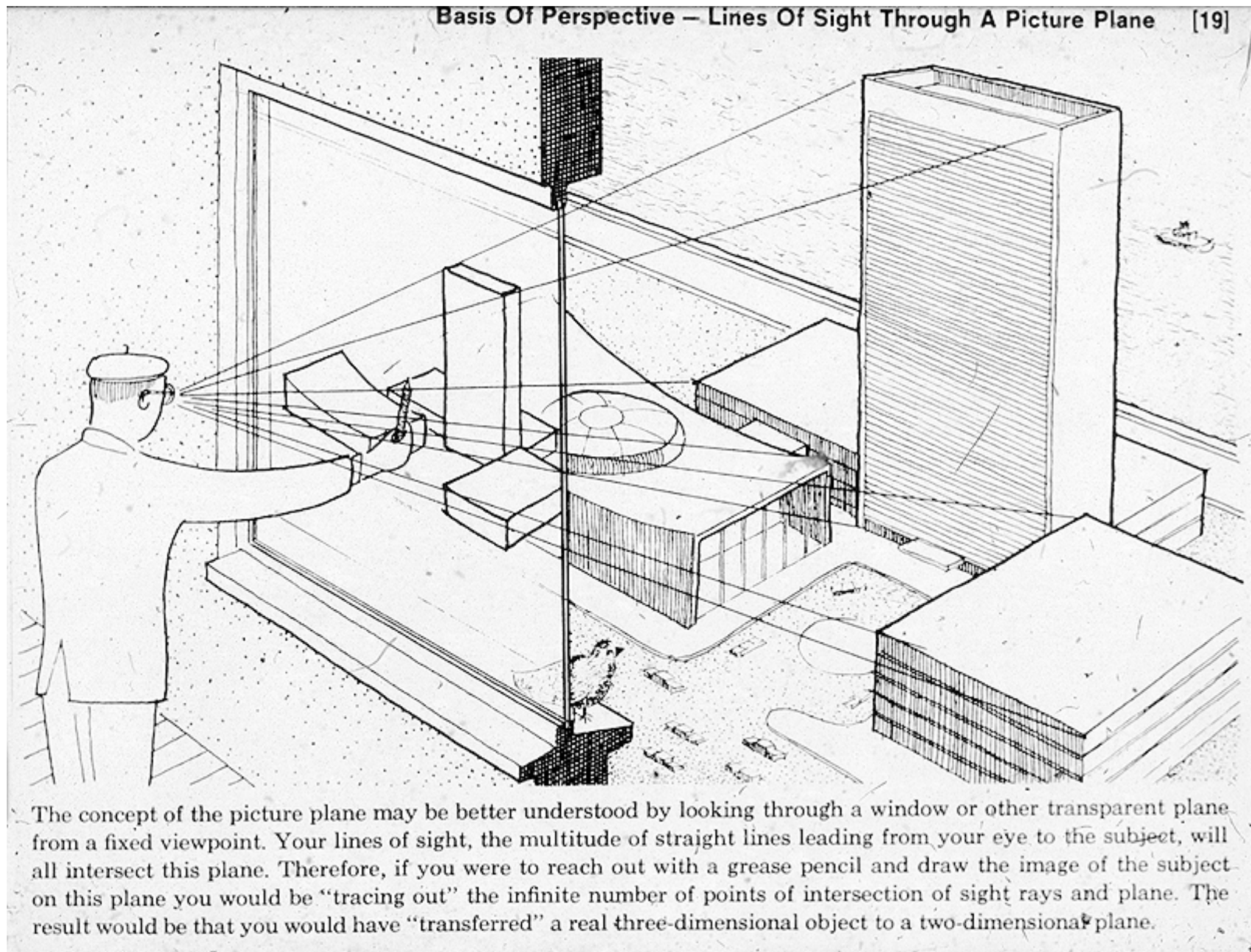
da Vinci c. 1498

Plane projection in drawing



Albrecht Dürer

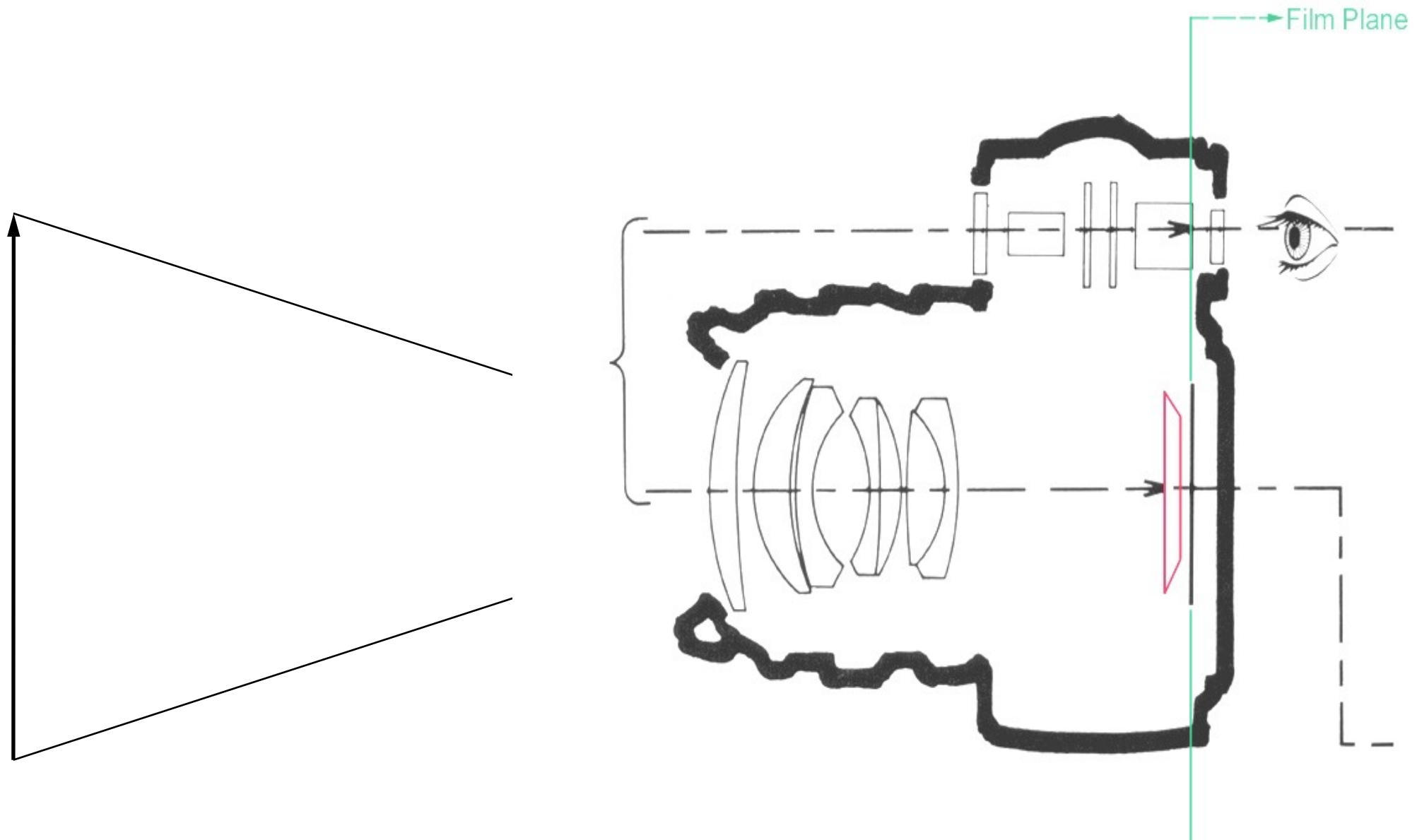
Plane projection in drawing



source unknown

Plane projection in photography

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



[Source unknown]

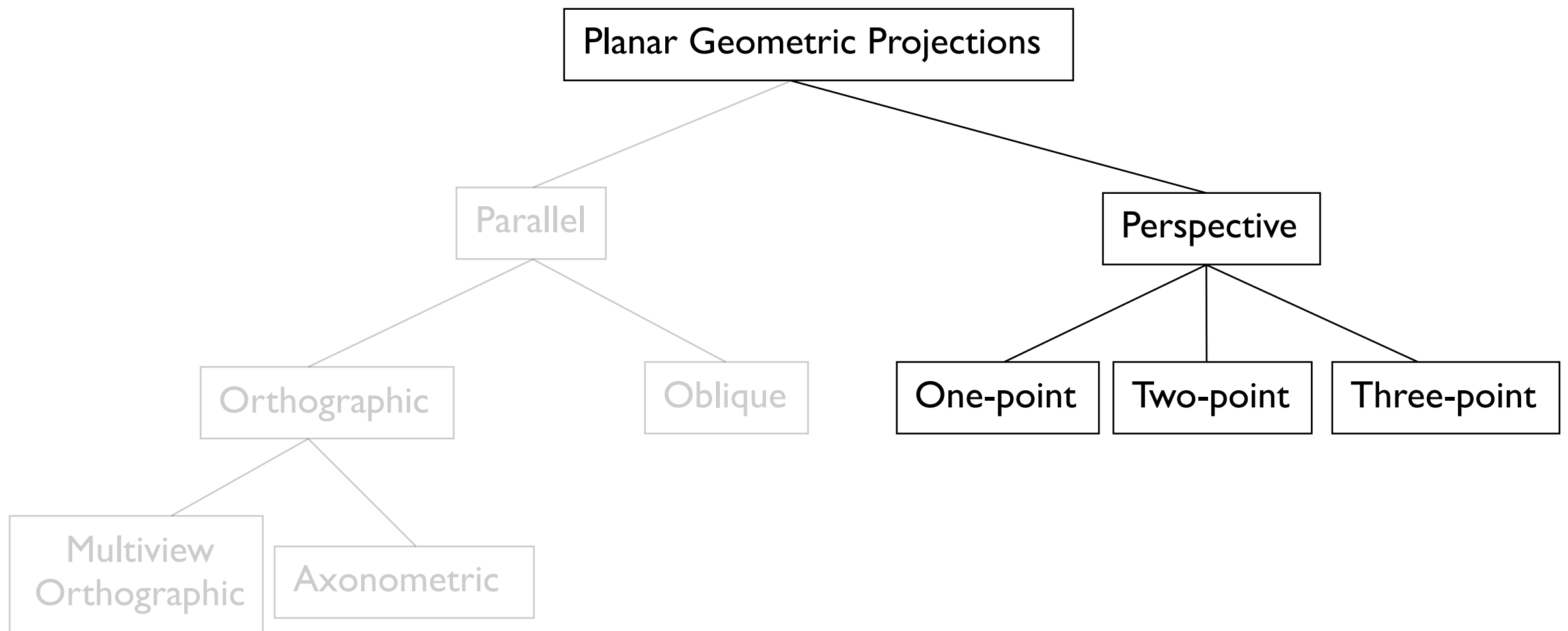
Plane projection in photography



[Richard Zakia]

Classical projections—perspective

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

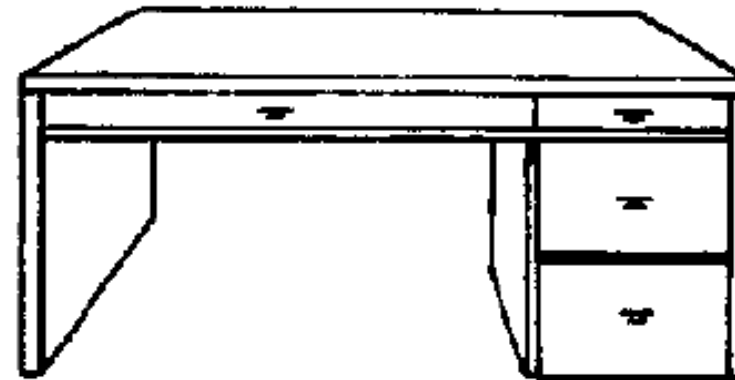


Perspective

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



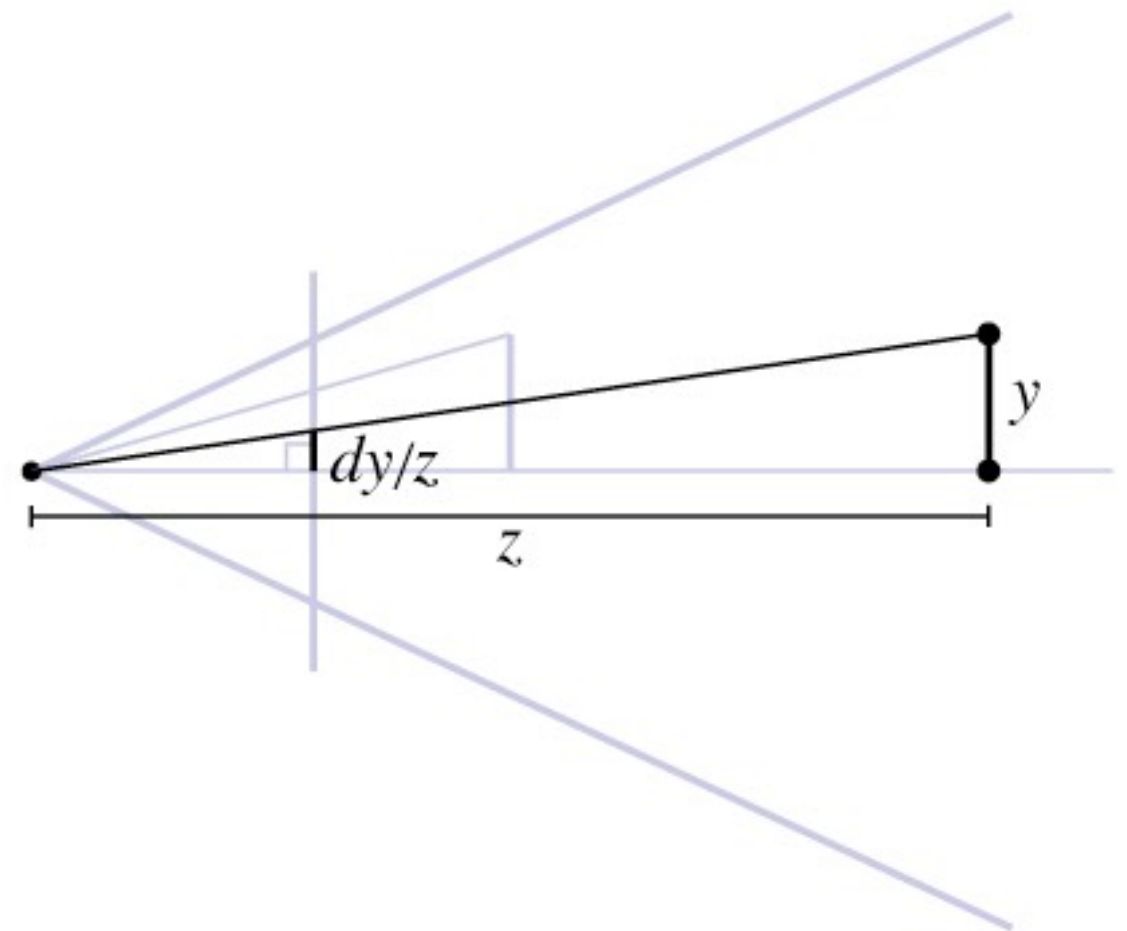
two-point



three-point

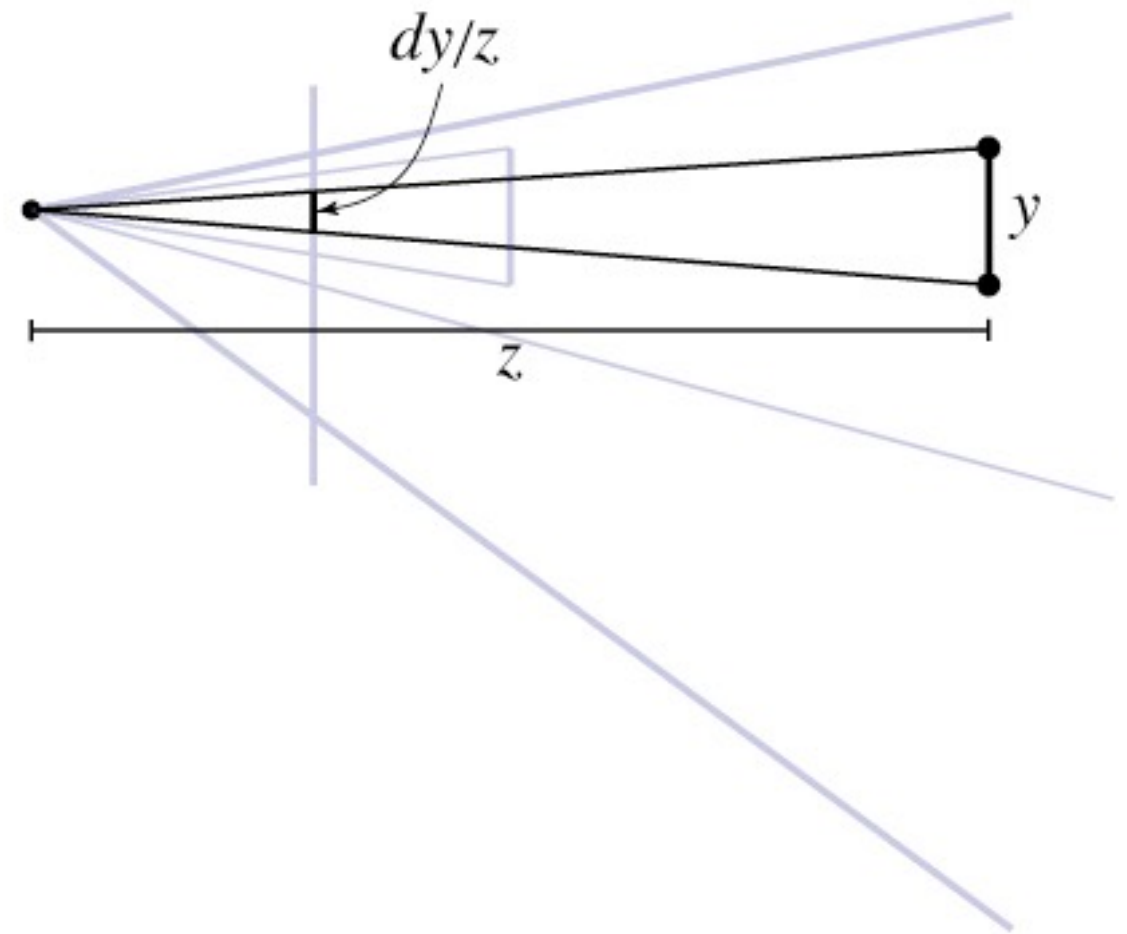
Perspective projection (normal)

- **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



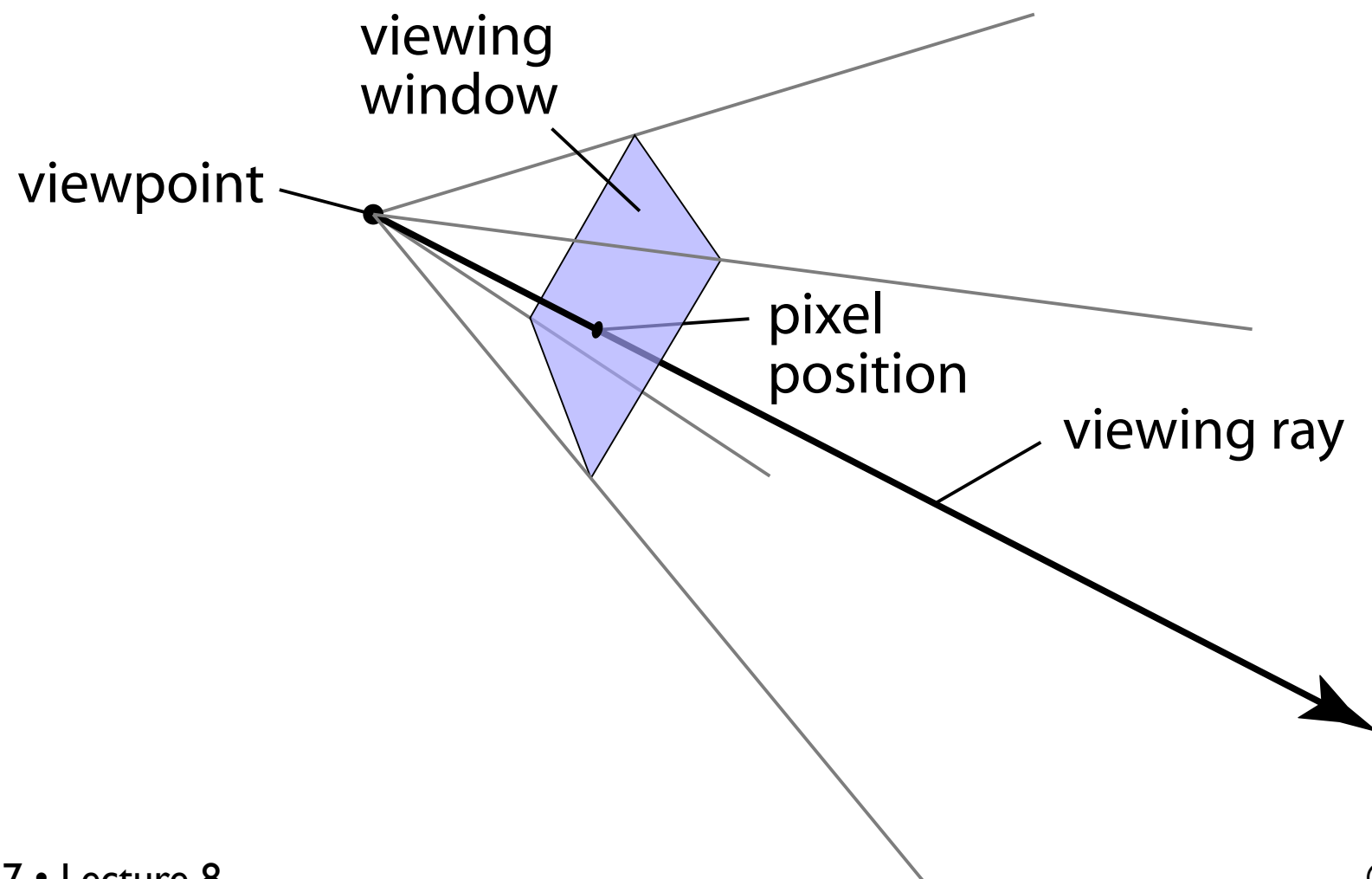
Shifted perspective projection

- **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



Generating eye rays—perspective

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



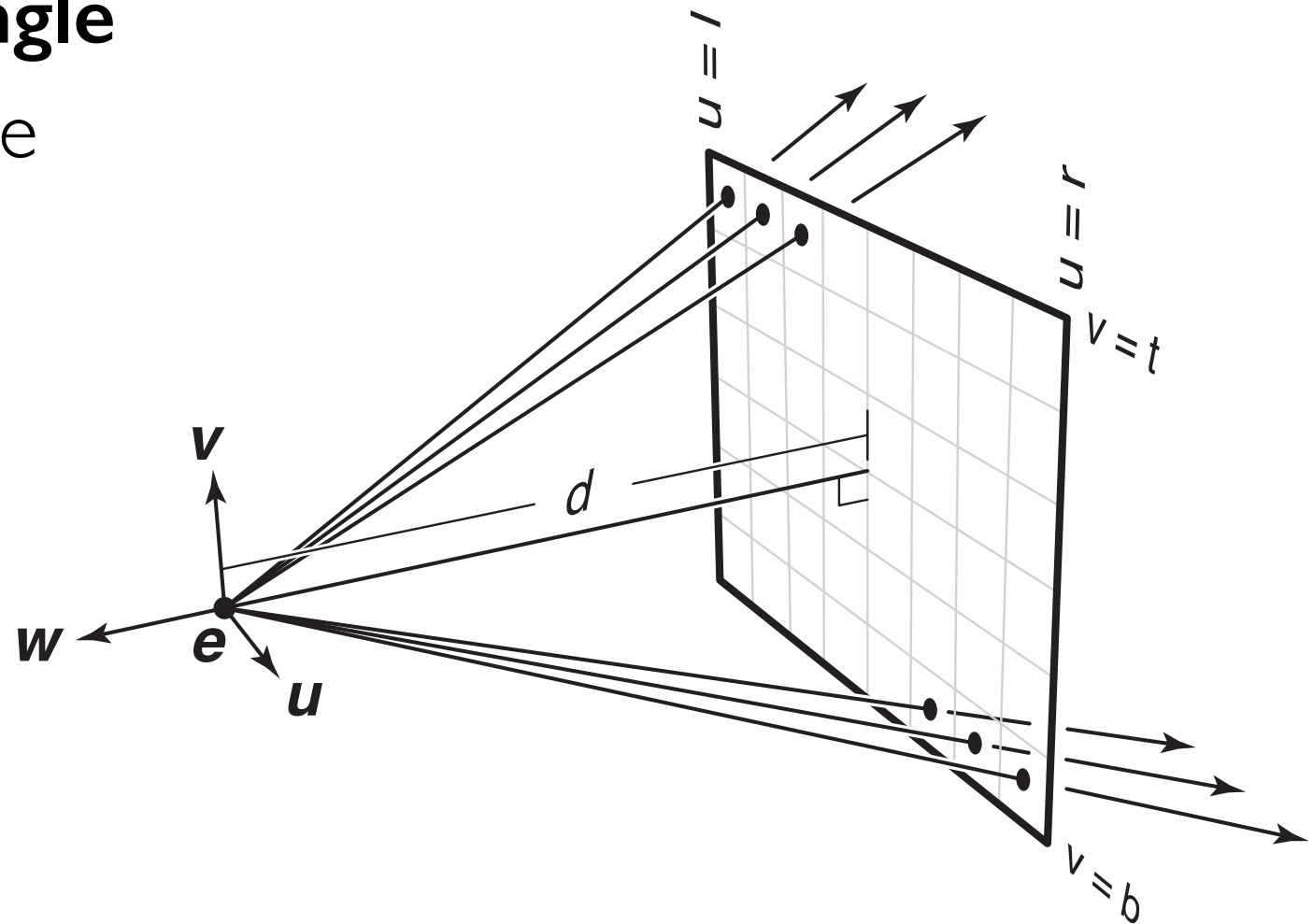
Generating eye rays—perspective

- **Positioning the view rectangle**

- establish three vectors to be *camera basis*: **\mathbf{u}** , **\mathbf{v}** , **\mathbf{w}**
- view rectangle is parallel to **\mathbf{u}** – **\mathbf{v}** plane, at $w = -d$, specified by l, r, t, b

- **Generating rays**

- for (u, v) in $[l, r] \times [b, t]$
- ray.origin = **\mathbf{e}**
- ray.direction = $-d \mathbf{w} + u \mathbf{u} + v \mathbf{v}$



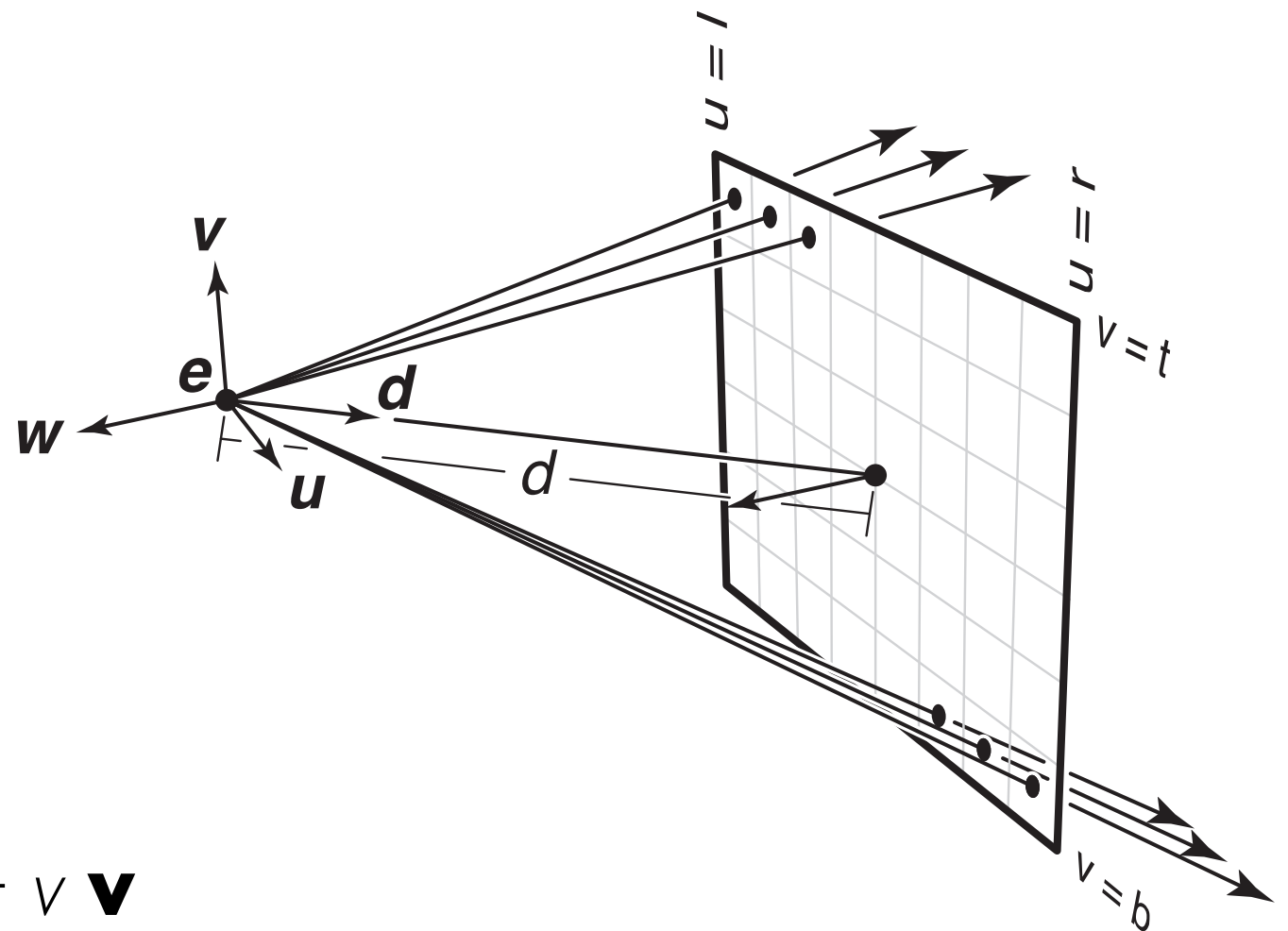
Oblique perspective views

- **Positioning the view rectangle**

- establish three vectors to be *camera basis*: **\mathbf{u}** , **\mathbf{v}** , **\mathbf{w}**
- view rectangle is the same, but shifted so that the center is in the direction **\mathbf{d}** from **\mathbf{e}**

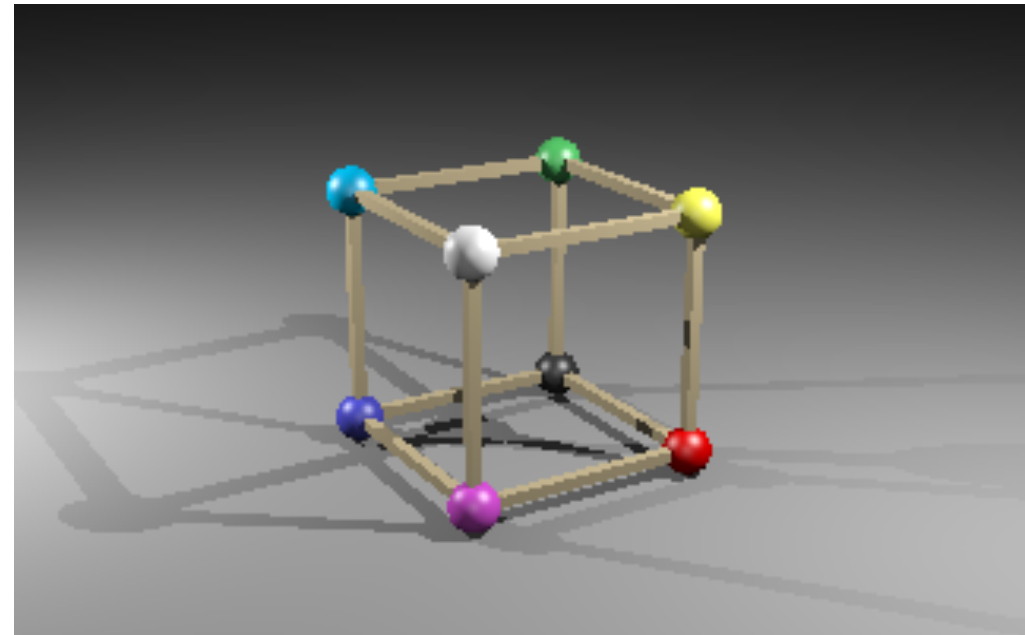
- **Generating rays**

- for (u, v) in $[l, r] \times [b, t]$
- ray.origin = **\mathbf{e}**
- ray.direction = $d \mathbf{d} + u \mathbf{u} + v \mathbf{v}$

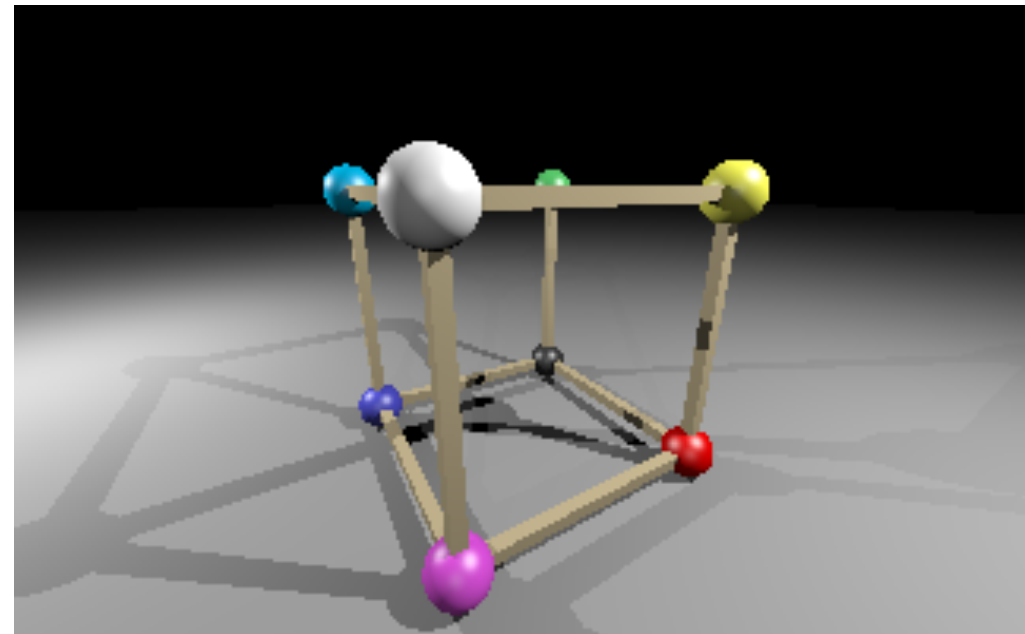


Perspective views in Ray I

```
<camera type="PerspectiveCamera">  
  <viewPoint>10 4.2 6</viewPoint>  
  <viewDir>-5 -2.1 -3</viewDir>  
  <viewUp>0 1 0</viewUp>  
  <projDistance>12</projDistance>  
  <viewWidth>8</viewWidth>  
  <viewHeight>5</viewHeight>  
</camera>
```



```
<camera type="PerspectiveCamera">  
  <viewPoint>2.5 1.05 1.5</viewPoint>  
  <viewDir>-5 -2.1 -3</viewDir>  
  <viewUp>0 1 0</viewUp>  
  <projDistance>3</projDistance>  
  <viewWidth>8</viewWidth>  
  <viewHeight>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

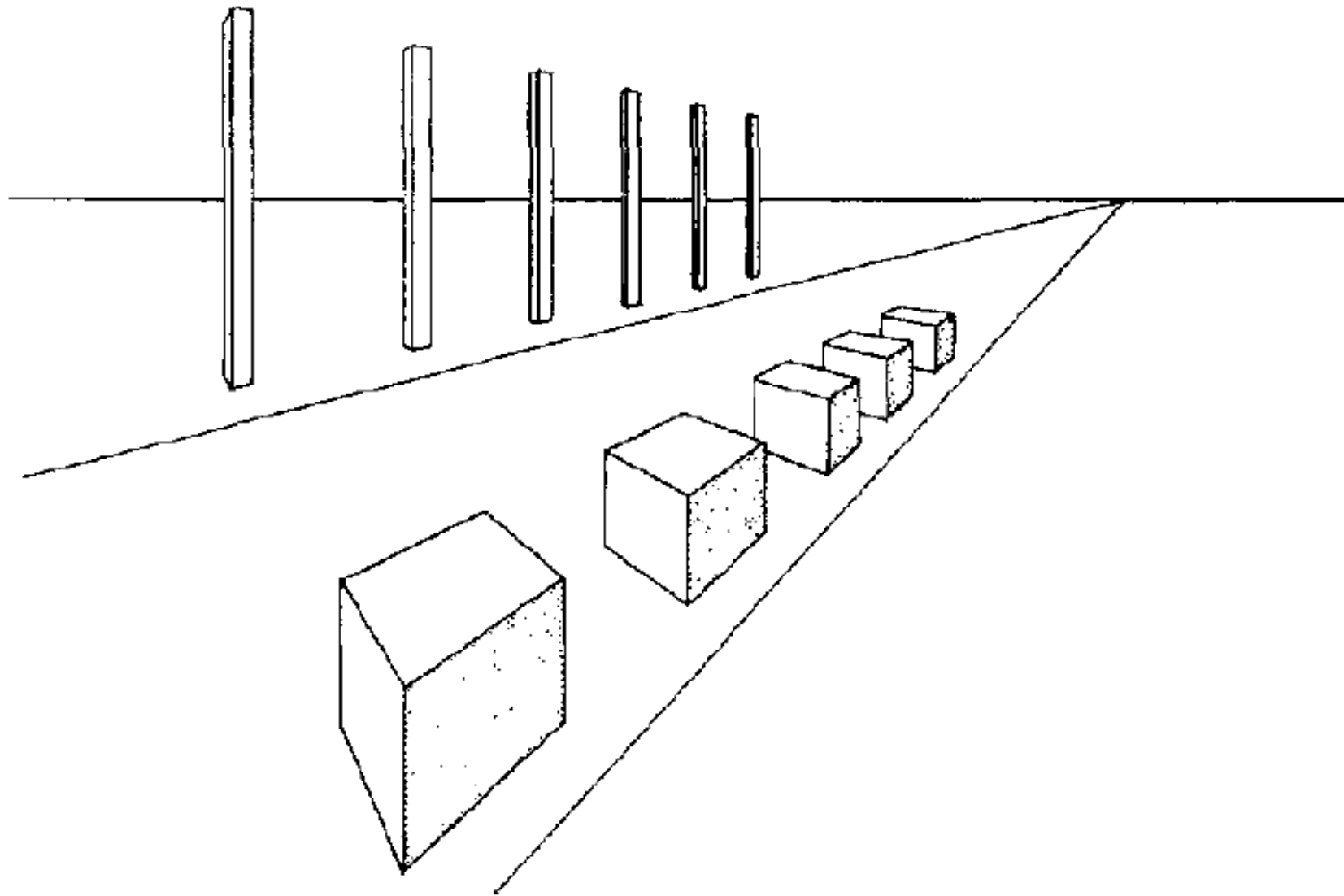
- **In photography, wide angle lenses are specialty tools**
 - “hard to work with”
 - easy to create weird-looking perspective effects
- **In graphics, you can type in whatever f.o.v. you want**
 - and people often type in big numbers!



[Ken Perlin]

Perspective distortions

- **Lengths, length ratios**



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



camera tilted up: converging vertical lines



lens shifted up: parallel vertical lines