3D Viewing

Canonical View Volume
Orthographic Projection
Perspective Projection

Shirley Chapter 7

Viewing and Projection

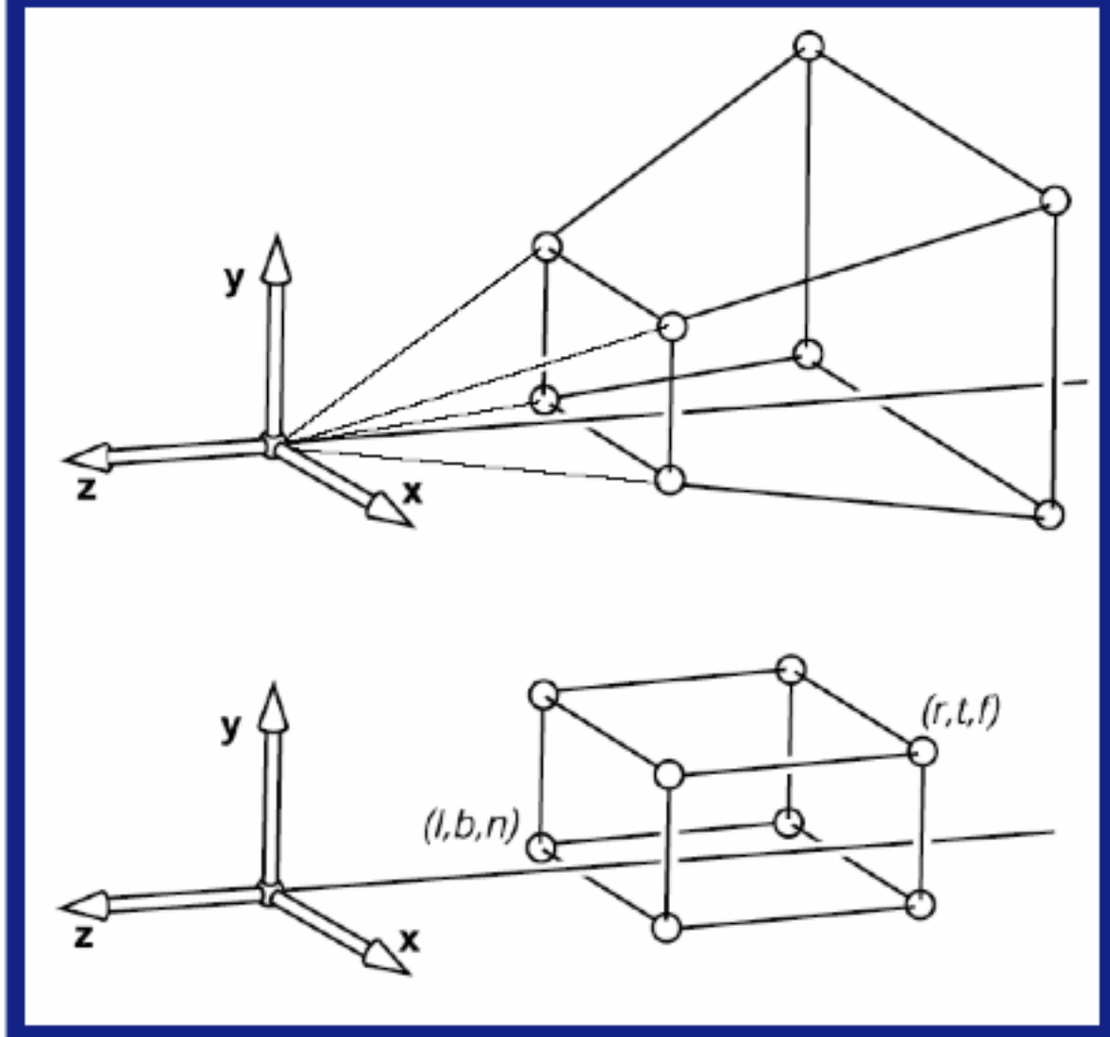
- Our eyes collapse 3-D world to 2-D retinal image (brain then has to reconstruct 3D)
- In CG, this process occurs by *projection*
- Projection has two parts:
  - *Viewing transformations:* camera position and direction
  - *Perspective/orthographic transformation:* reduces 3-D to 2-D
- Use homogeneous transformations (of course…)

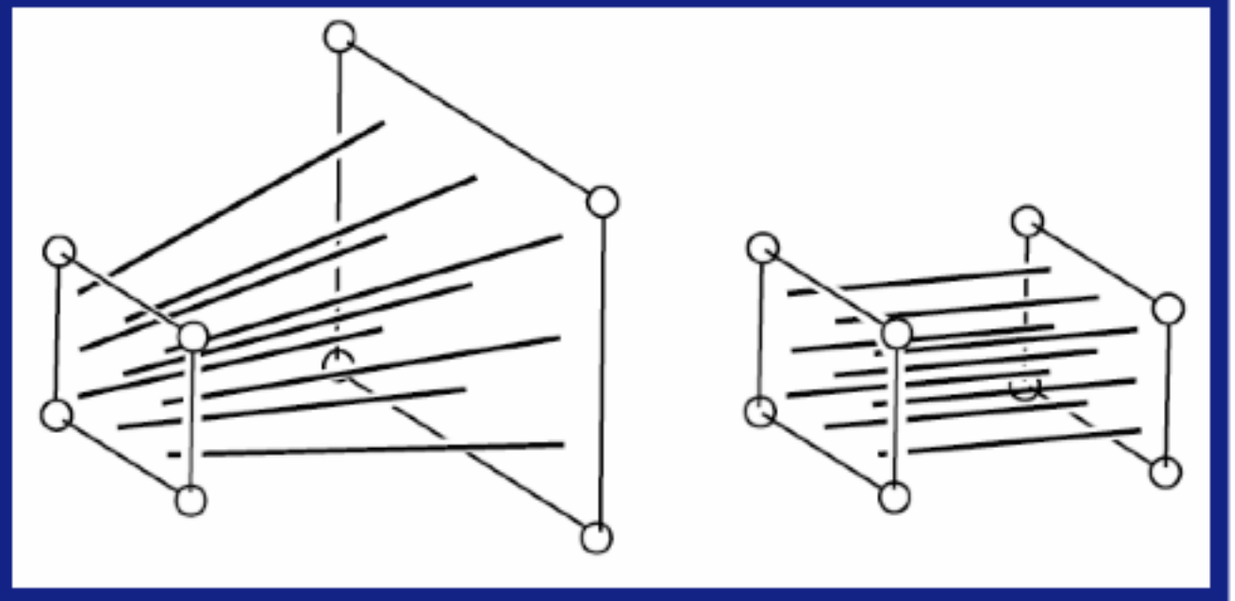
Getting Geometry on the Screen

Given geometry positioned in the world coordinate system, how do we get it onto the display?

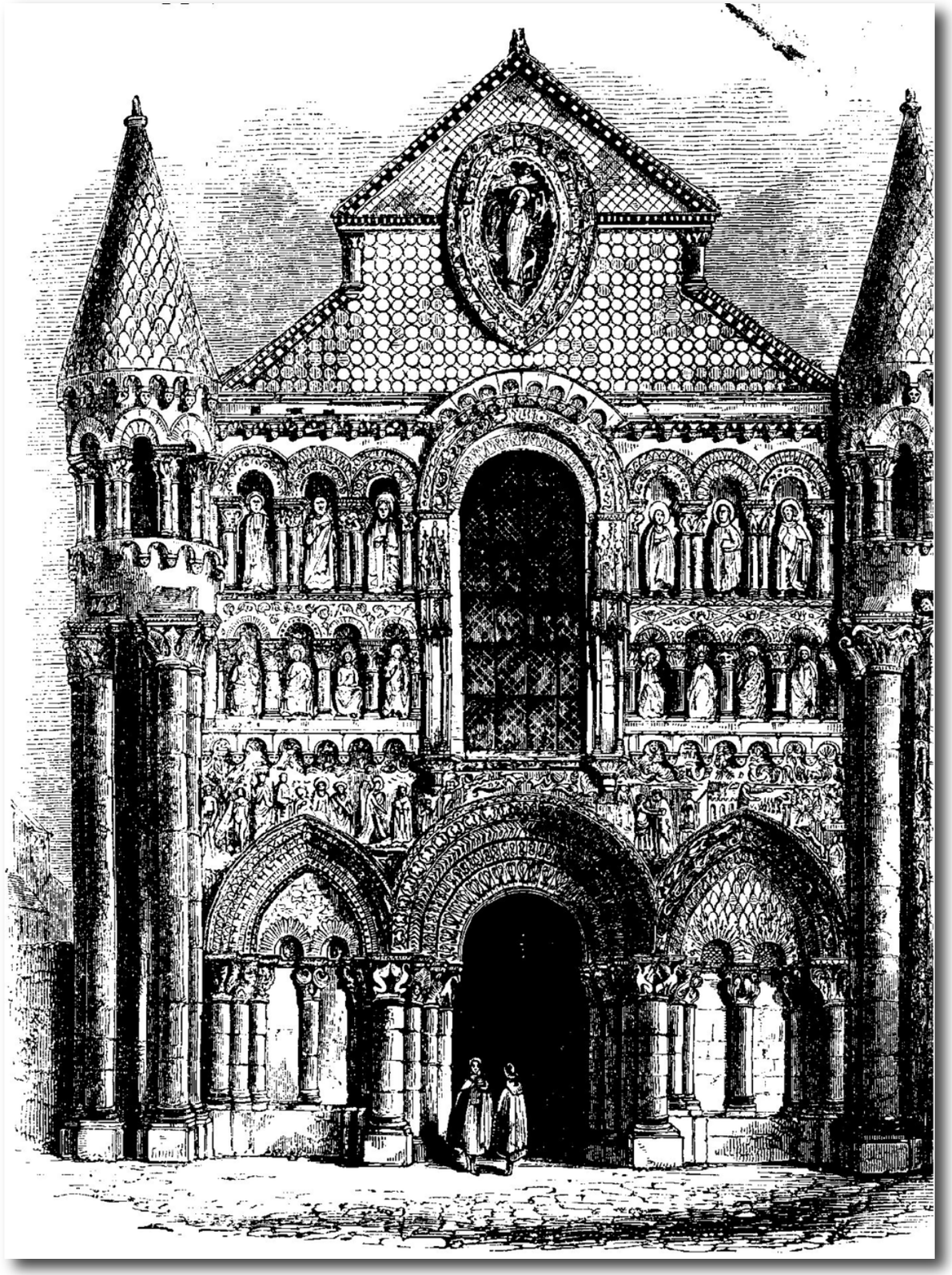
- Transform to camera coordinate system
- Transform (warp) into canonical view volume
- Clip
- Project to display coordinates
- Rasterize

Perspective and Orthographic Projection





Orthographic Projection



Viewing and Projection

Build this up in stages

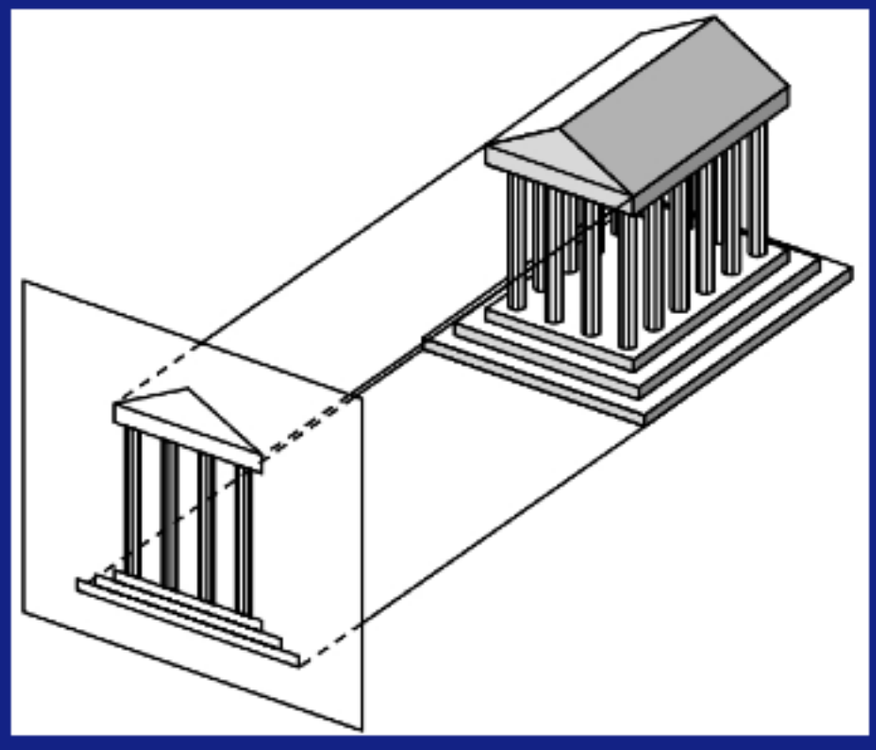
- Canonical view volume to screen
- Orthographic projection to canonical view volume
- Perspective projection to orthographic space

Orthographic Projection

the focal point is at infinity, the rays are parallel, and orthogonal to the image plane

good model for telephoto lens. No perspective effects.

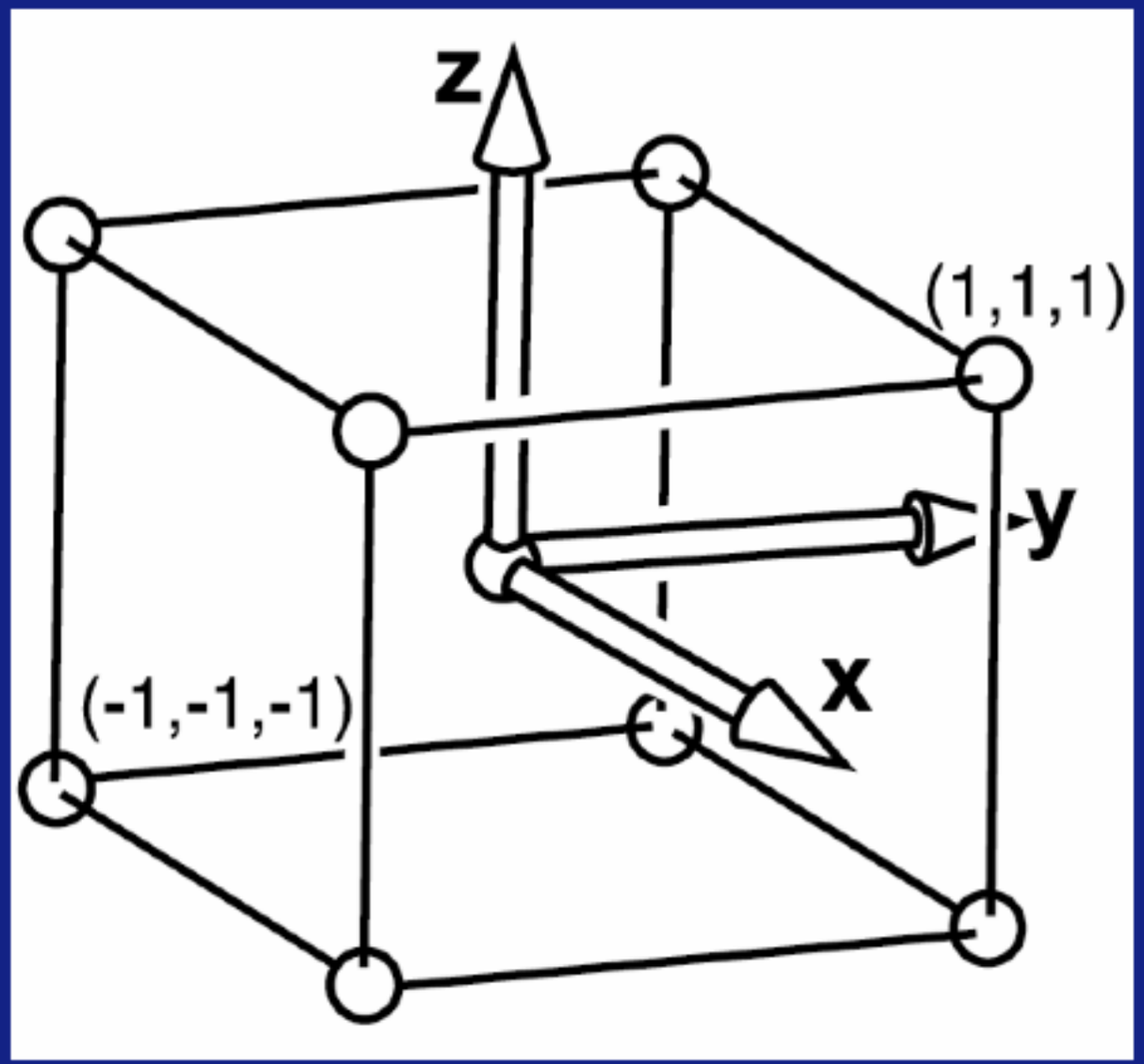
when *xy*-plane is the image plane (x,y,z) -> (x,y,0)
front orthographic view



Telephoto Lenses and Fashion Photography

http://farm4.static.flickr.com/3057/2555706112_20a3015ddb.jpg

Canonical View Volume

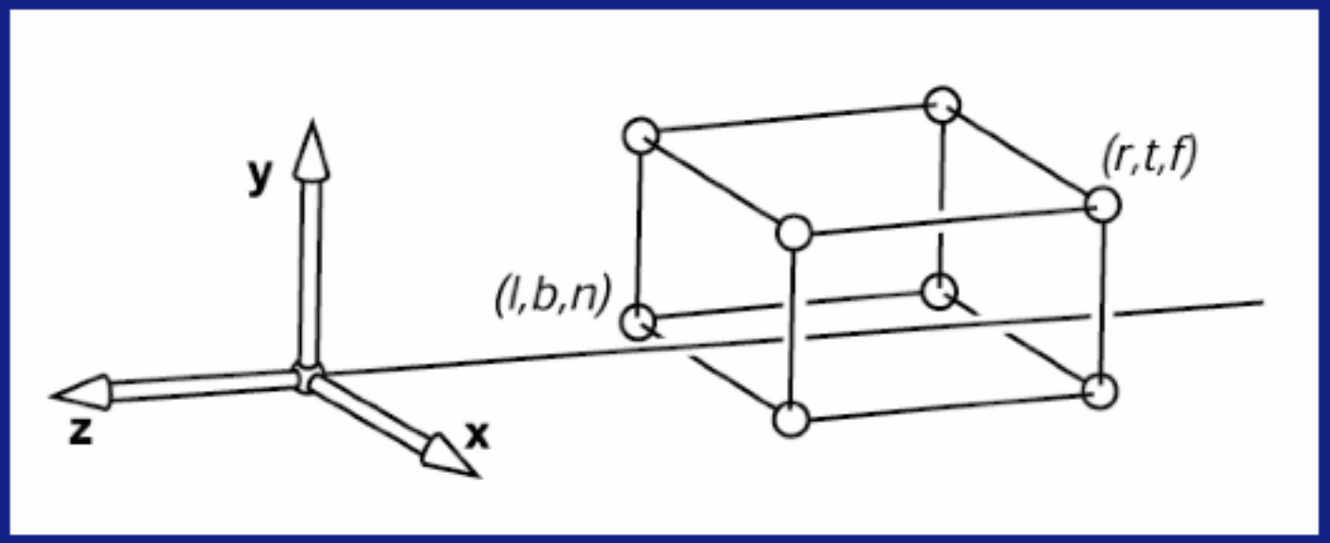


Why this shape?

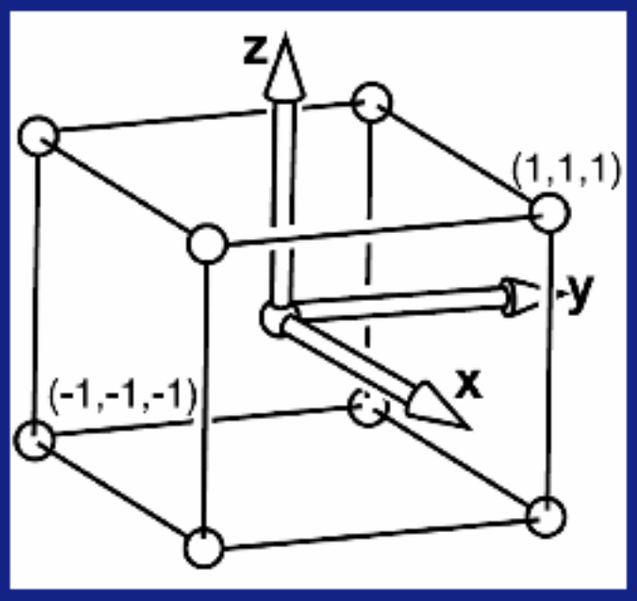
- Easy to clip to
- Trivial to project from 3D to 2D image plane

chalkboard

Orthographic Projection



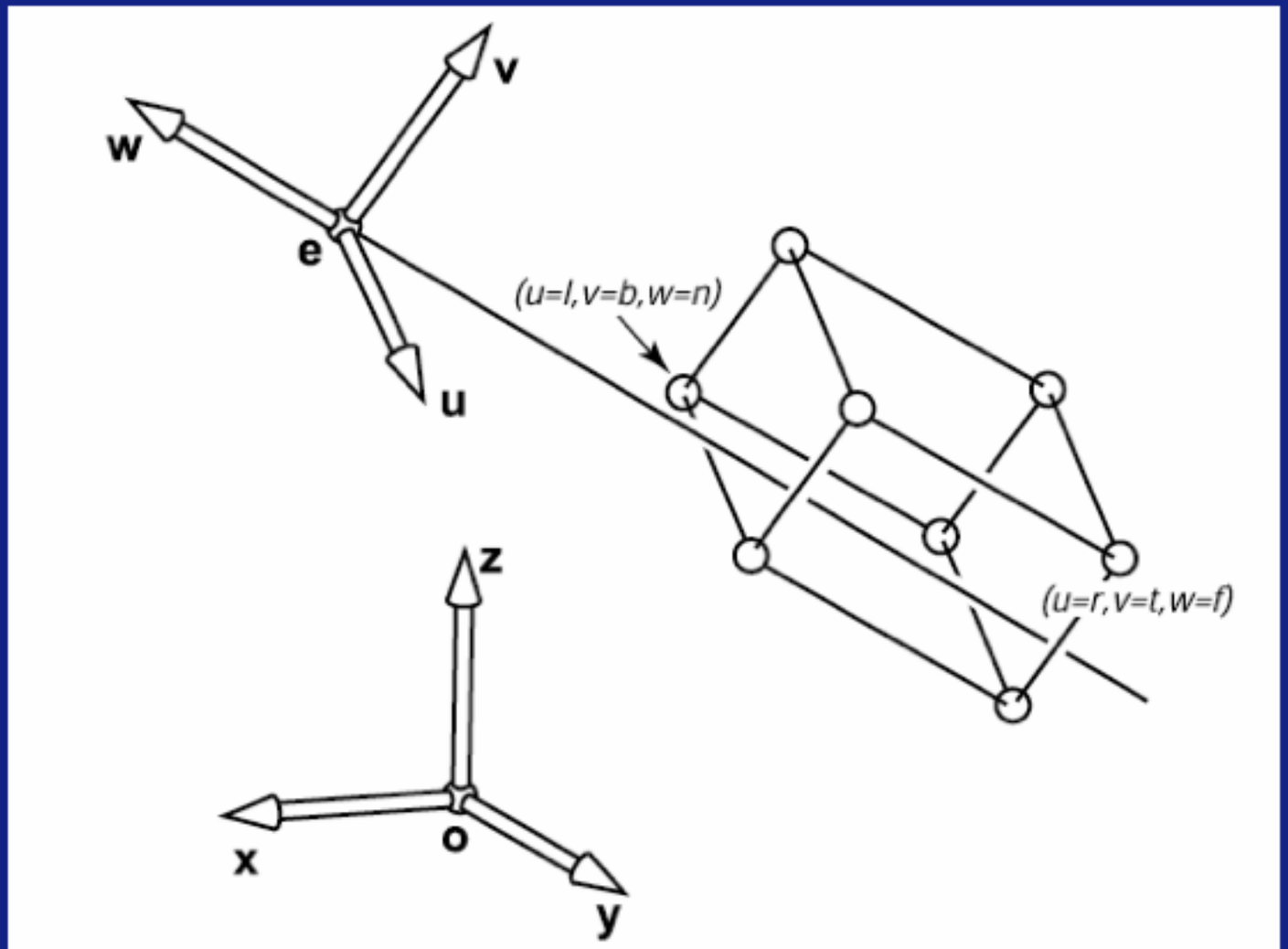
X=l left plane
X=r right plane
Y=b bottom plane
Y=t top plane
Z=n near plane
Z=f far plane

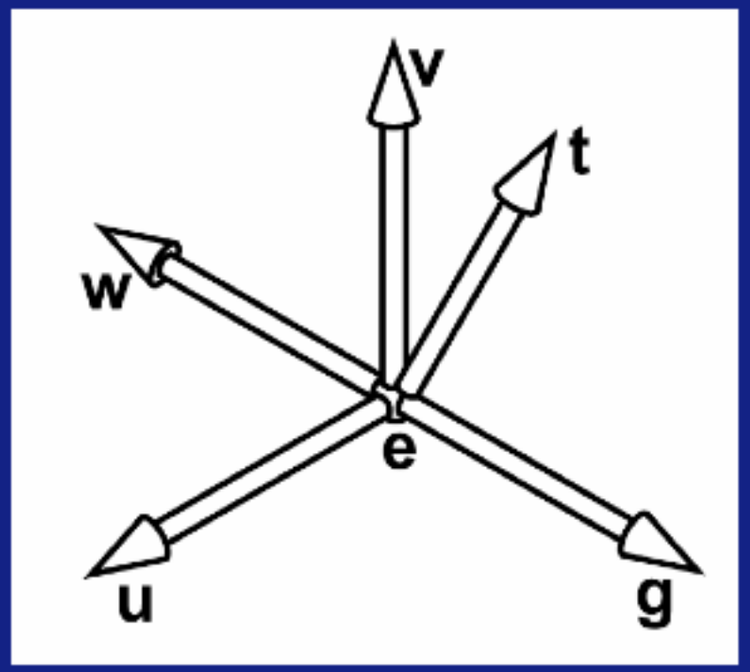


Why near plane? Prevent points behind the camera being seen
Why far plane? Allows *z* to be scaled to a limited fixed-point value (*z*-buffering)

chalkboard

Arbitrary View Positions

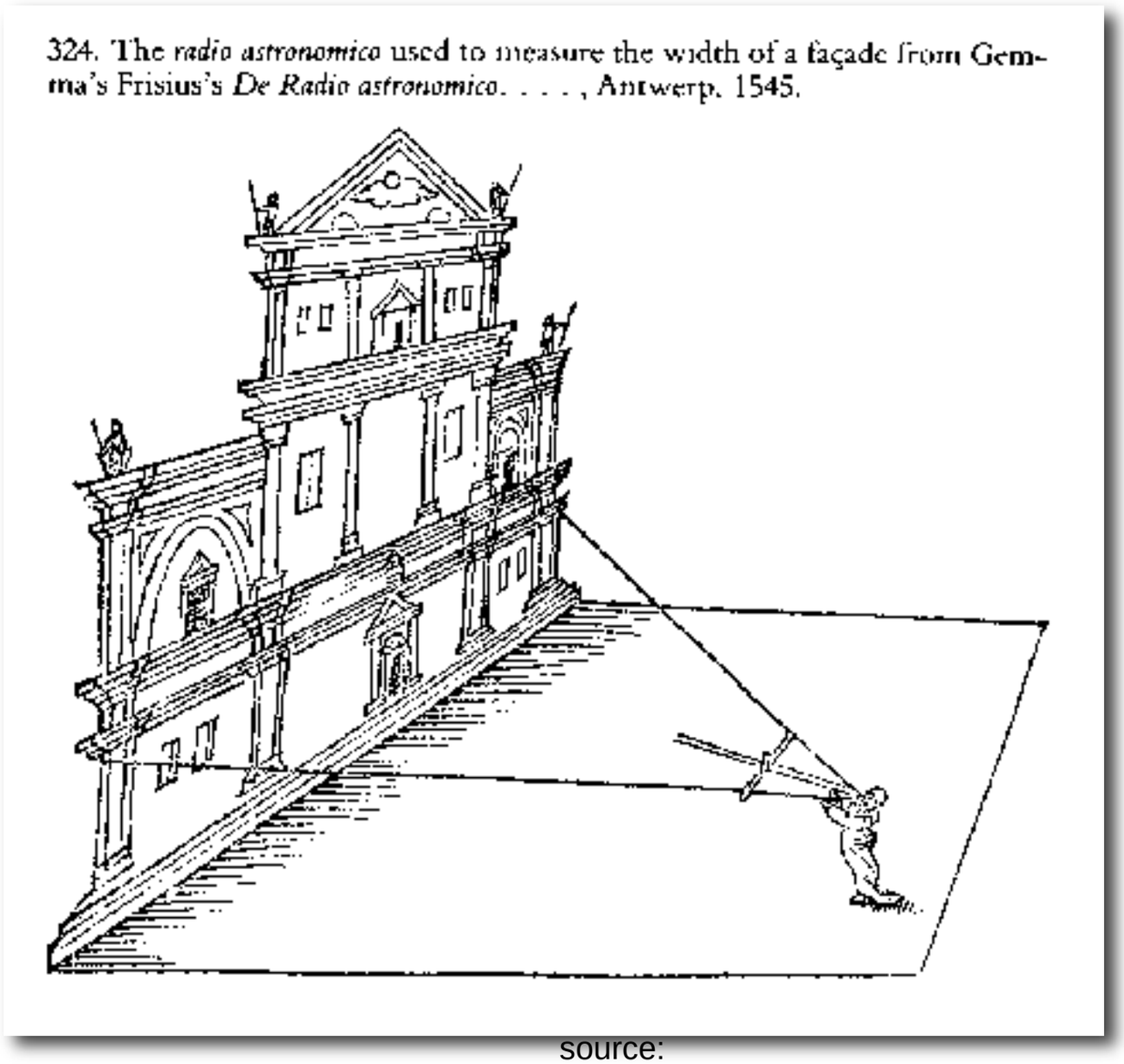




Eye position: e
Gaze direction: g
view-up vector: t

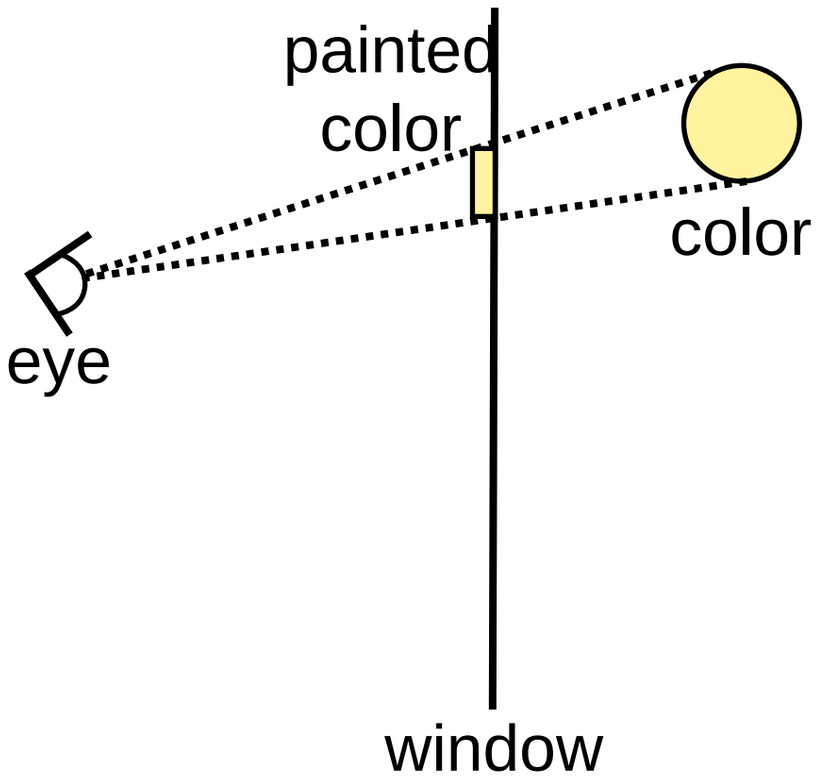
chalkboard

Perspective Projection



source:
http://www.dartmouth.edu/~matc/math5.geometry/unit15/Frisius.gif

The simplest way to look at perspective projection is as painting on a window....



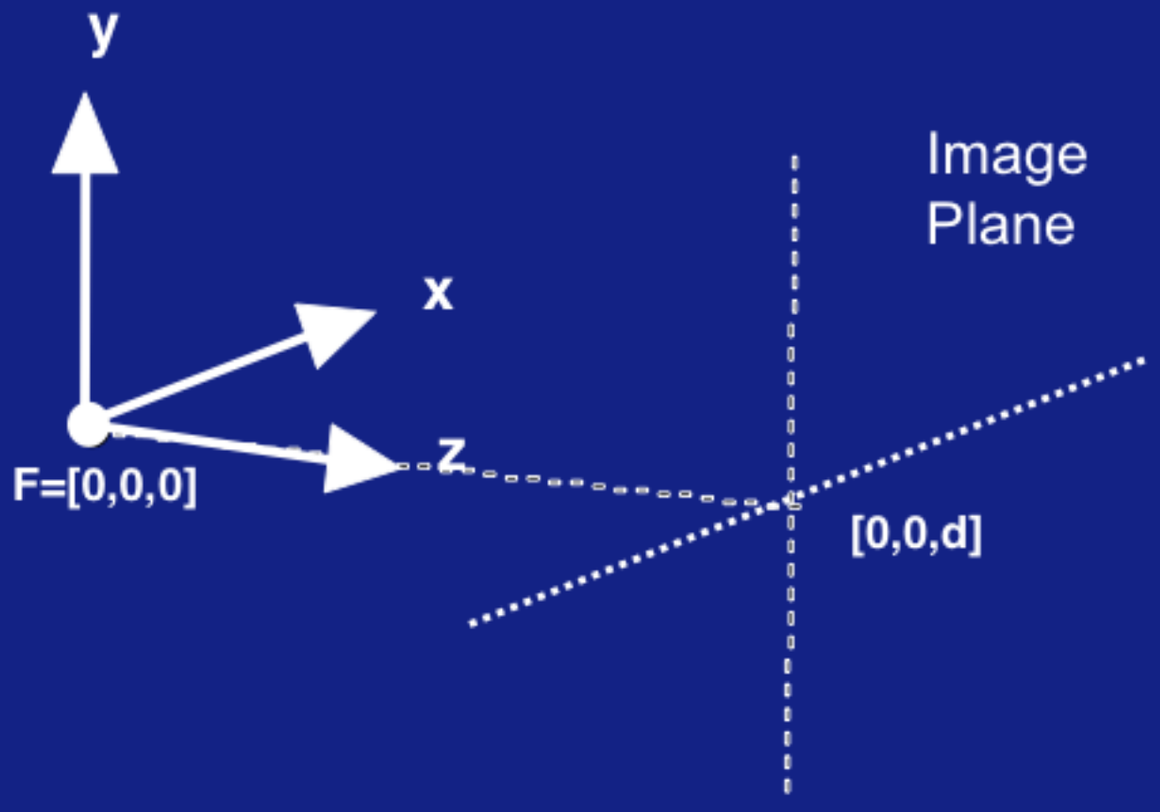
Paint on the window whatever color you see there.

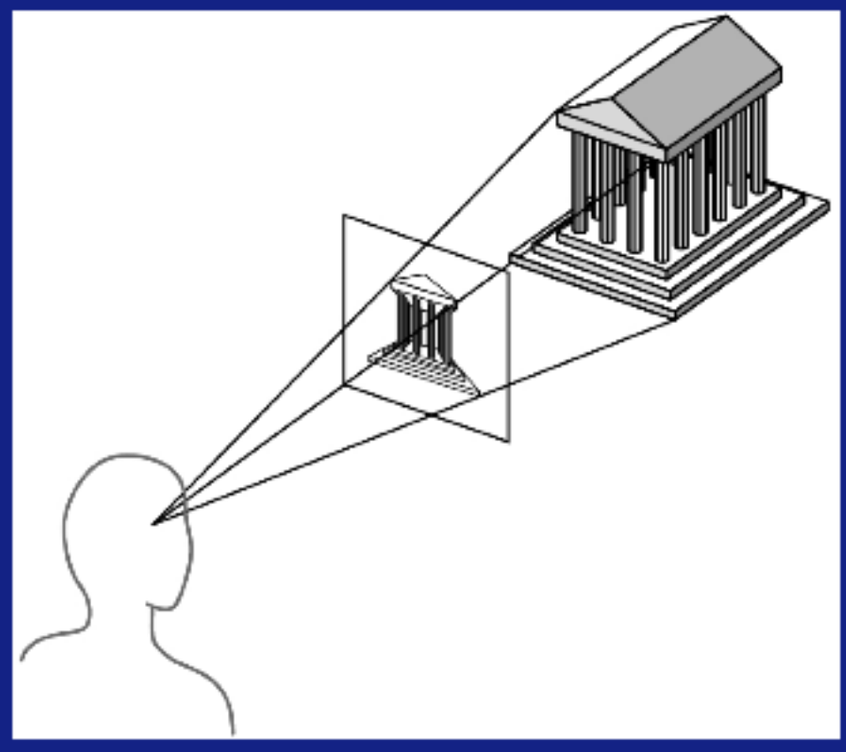
source:
http://blog.mlive.com/flintjournal/newsnow/2007/11/WINDOW_PAINTING_02.jpg

Simple Perspective Camera

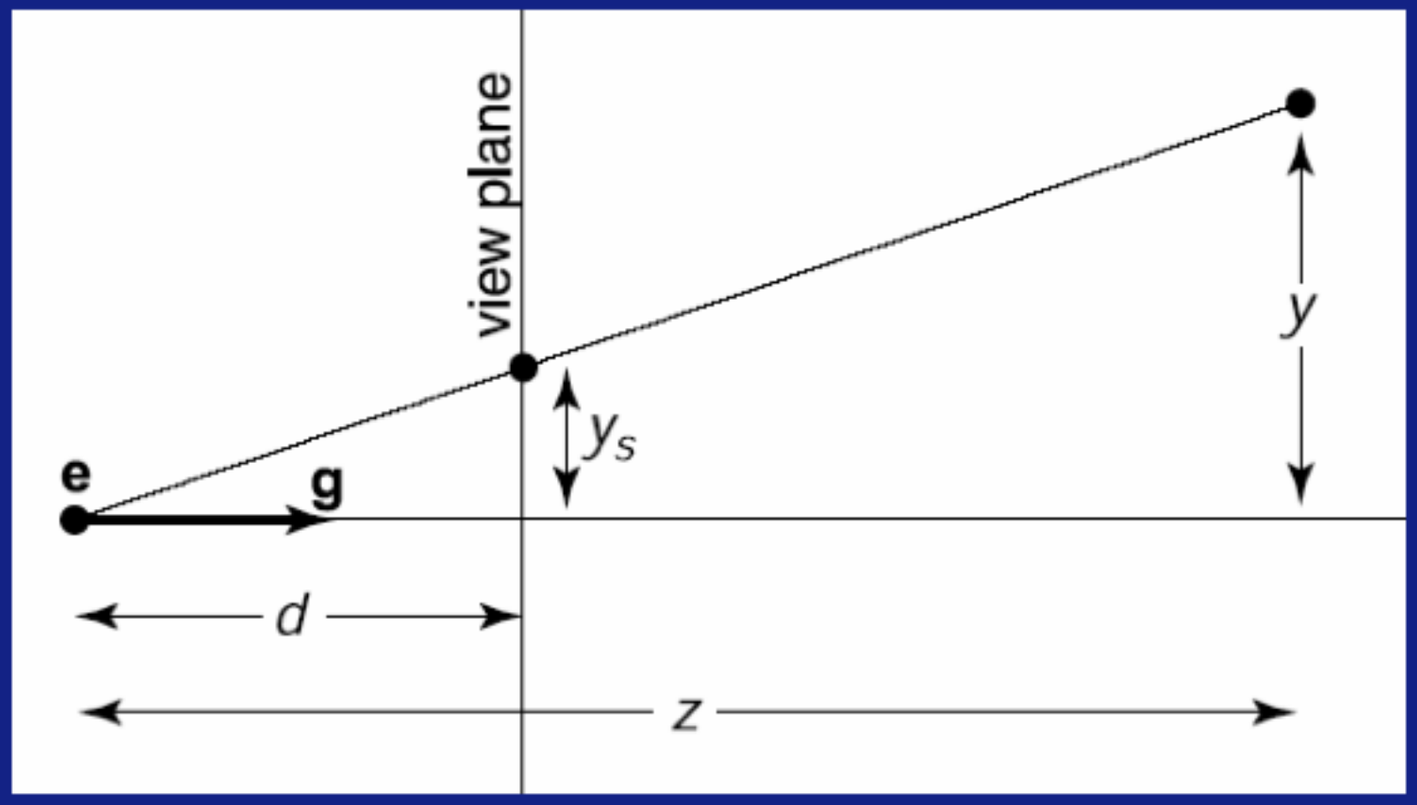
Canonical case:

- camera looks along the *z*-axis
- focal point is the origin
- image plane is parallel to the *xy*-plane at distance *d*
- (We call d the focal length, mainly for historical reasons)





Perspective Projection of a Point

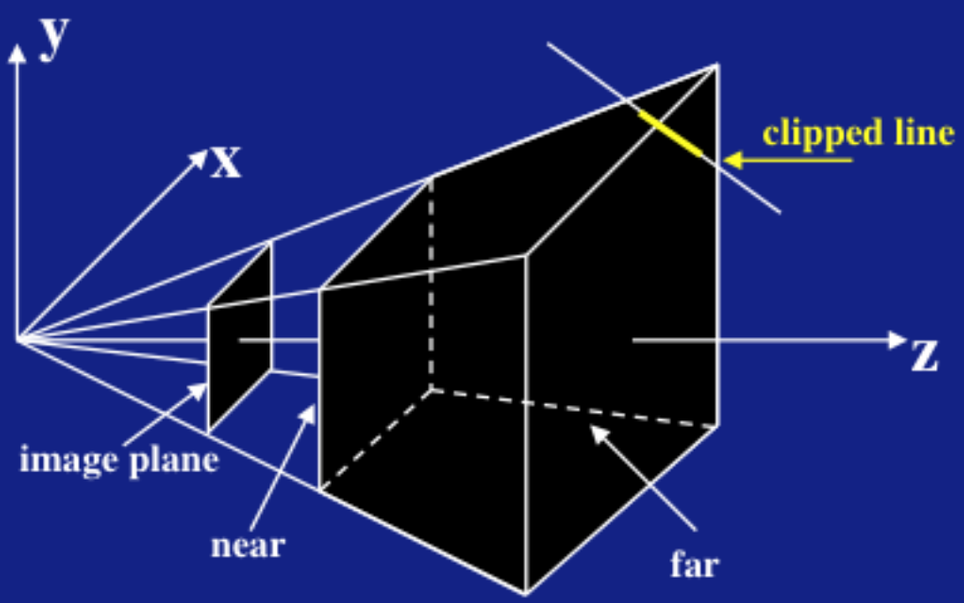


$$y_s = d\frac{y}{z}$$

Clipping

Something is missing between projection and viewing...

Before projecting, we need to eliminate the portion of scene that is outside the viewing frustum

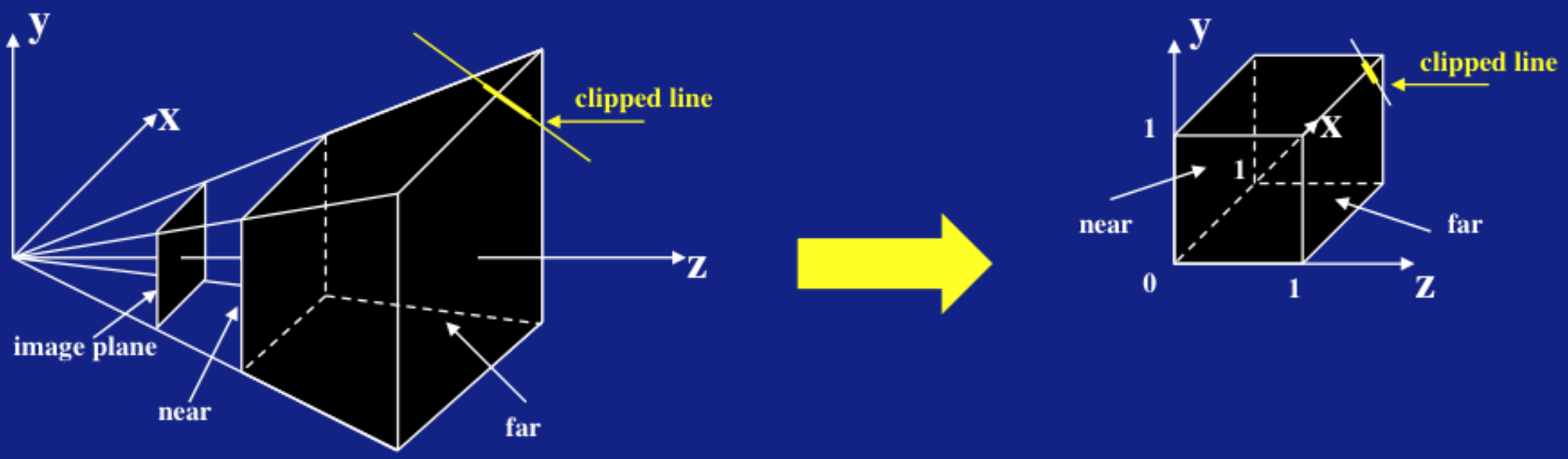


Need to clip objects to the frustum (truncated pyramid)

Now in a canonical position but it still seems kind of tricky...

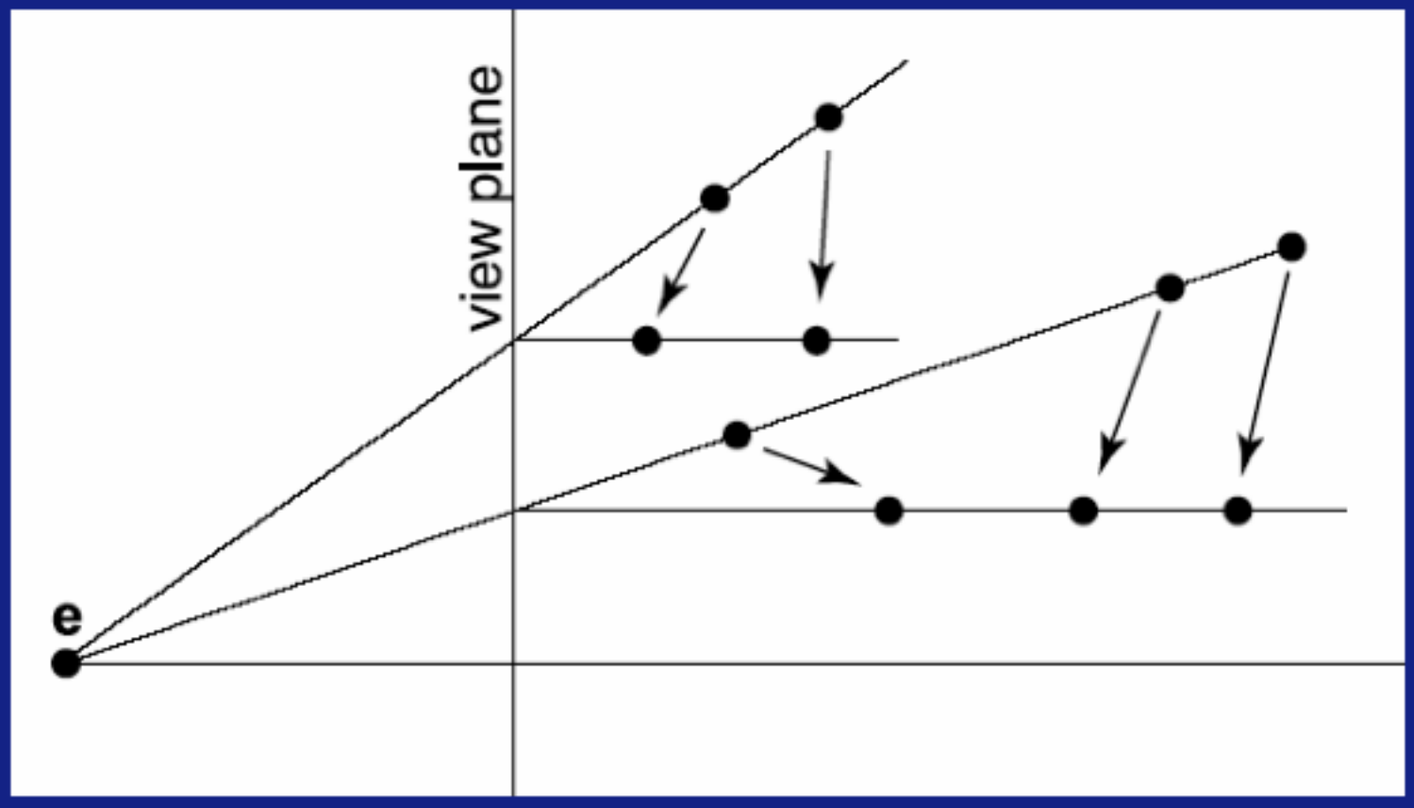
Normalizing the Viewing Frustum

Solution: transform frustum to a cube before clipping



Converts perspective frustum to orthographic frustum
Yet another homogeneous transform!

Perspective Projection



chalkboard

Warping a perspective projection into and orthographic one

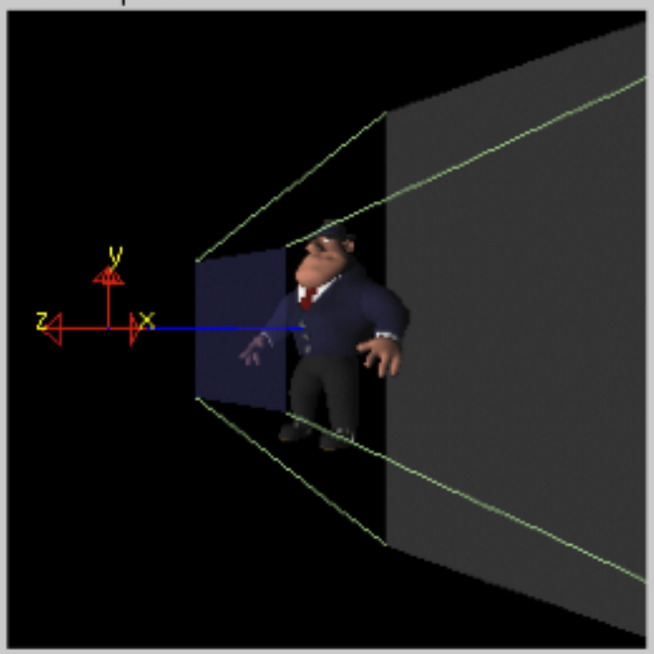
Lines for the two projections intersect at the view plane

How can we put this in matrix form?

Need to divide by z—haven't seen a divide in our matrices so far…

Requires our w from last time (or h in the book)

World-space view



Screen-space view

Command manipulation window

```
                    fovy  aspect  zNear  zFar
gluPerspective(  60.0  , 1.00  , 1.0   , 10.0  );
    gluLookAt(  0.00  , 0.00  , 2.00  ,    <- eye
                0.00  , 0.00  , 0.00  ,    <- center
                0.00  , 1.00  , 0.00  );   <- up
```

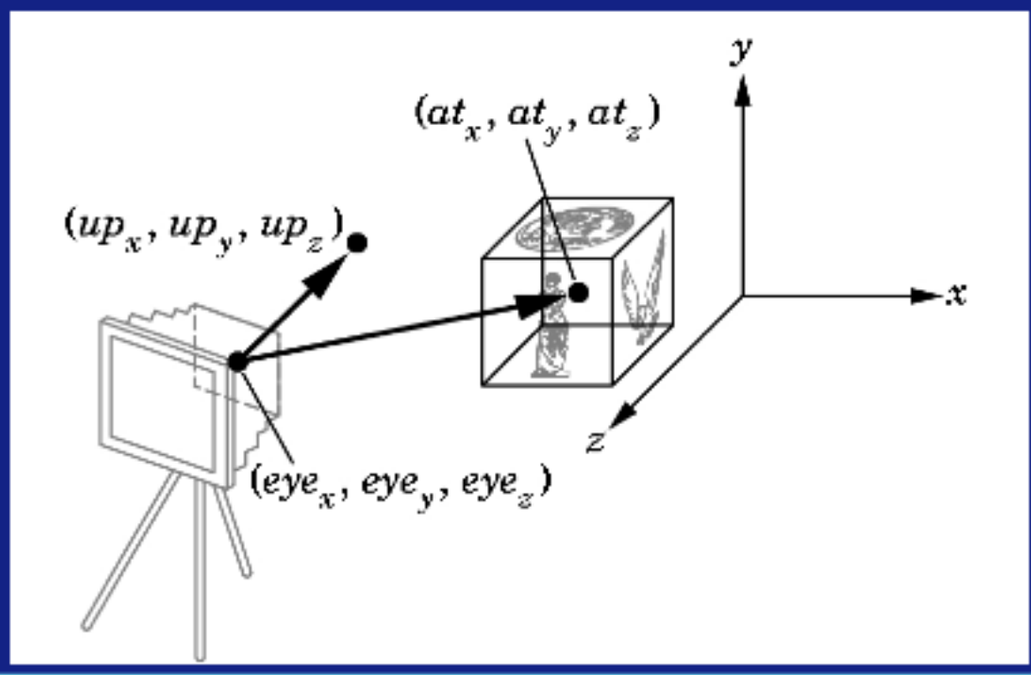
Click on the arguments and move the mouse to modify values.

Camera Control Values

- All we need is a single translation and angle-axis rotation (orientation), but...
- Good animation requires good camera control--we need better control knobs
- Translation knob - move to the *lookfrom* point
- Orientation can be specified in several ways:
  - specify camera rotations
  - specify a *lookat* point (solve for camera rotations)

A Popular View Specification Approach

- Focal length, image size/shape and clipping planes are in the perspective transformation
- In addition:
  - *lookfrom:* where the focal point (camera) is
  - *lookat:* the world point to be centered in the image
- Also specify camera orientation about the *lookat-lookfrom* axis



Implementation

Implementing the *lookat/lookfrom/vup* viewing scheme

(1) Translate by *-lookfrom*, bring focal point to origin

(2) Rotate *lookat-lookfrom* to the *z*-axis with matrix R:

» v = (*lookat-lookfrom*) (normalized) and z = [0,0,1]

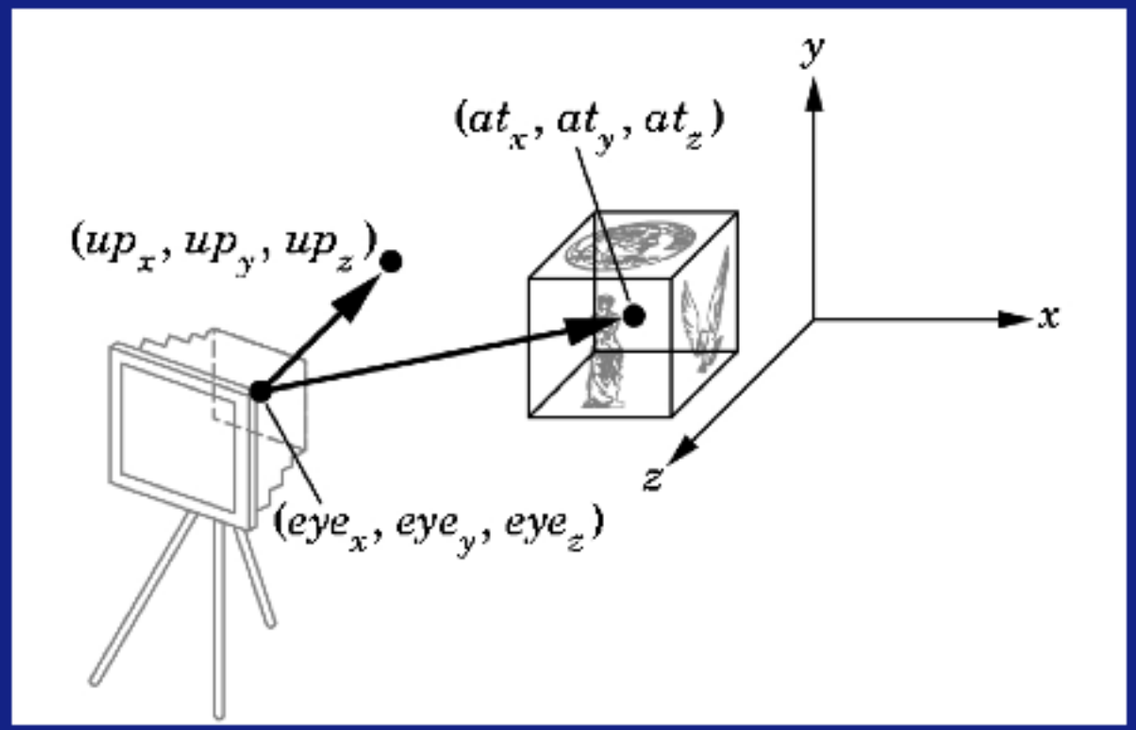
» rotation axis: $a = (v \times z)/|v \times z|$

» rotation angle: $\cos\theta = v \bullet z$ and $\sin\theta = |v \times z|$

glRotate(θ, a_x, a_y, a_z)

(3) Rotate about *z*-axis to get *vup* parallel to the y-axis

The Whole Picture



LOOKFROM: Where the camera is

LOOKAT: A point that should be centered in the image

VUP: A vector that will be pointing straight up in the image

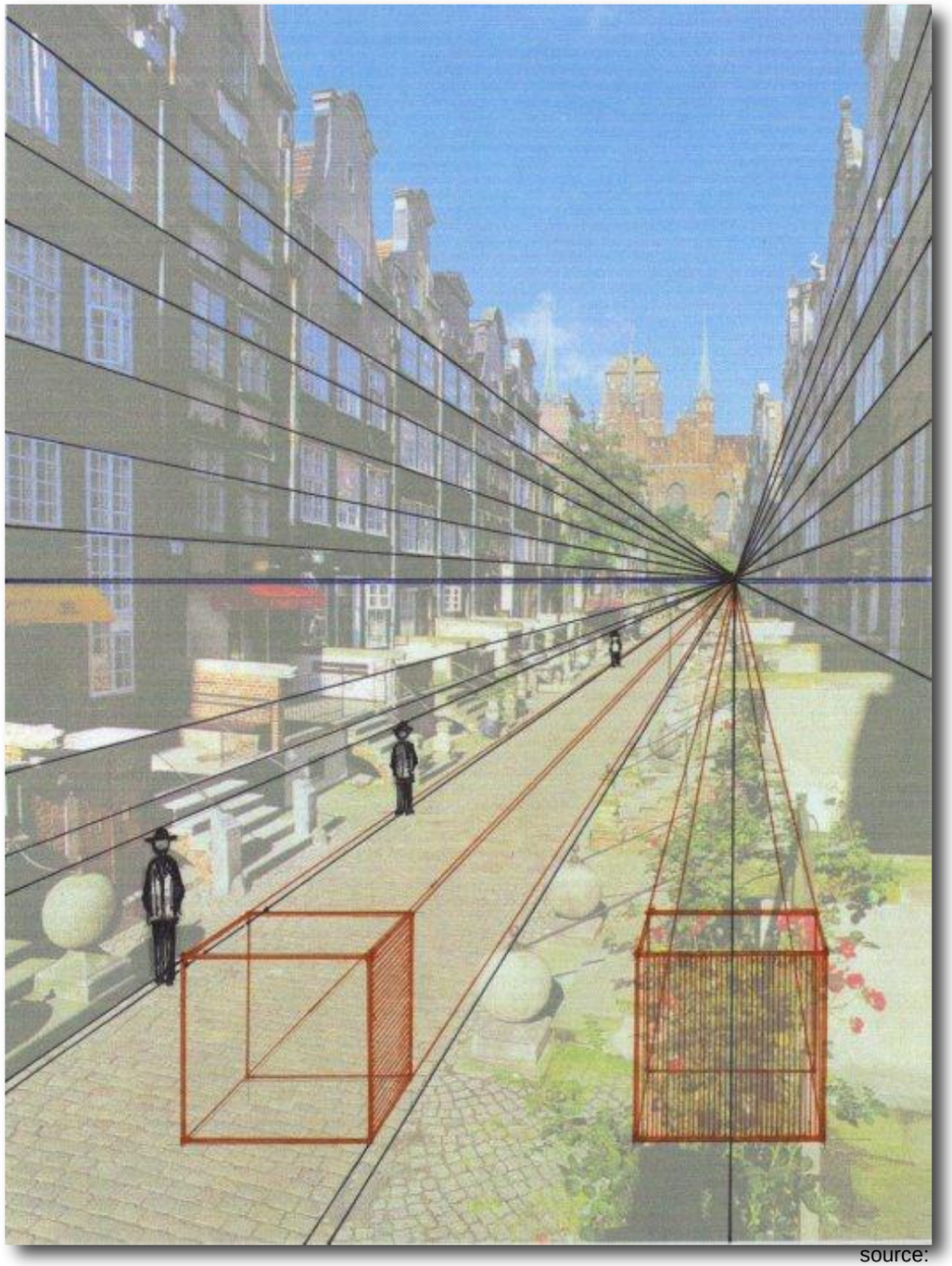
FOV: Field-of-view angle.

d: focal length

WORLD COORDINATES

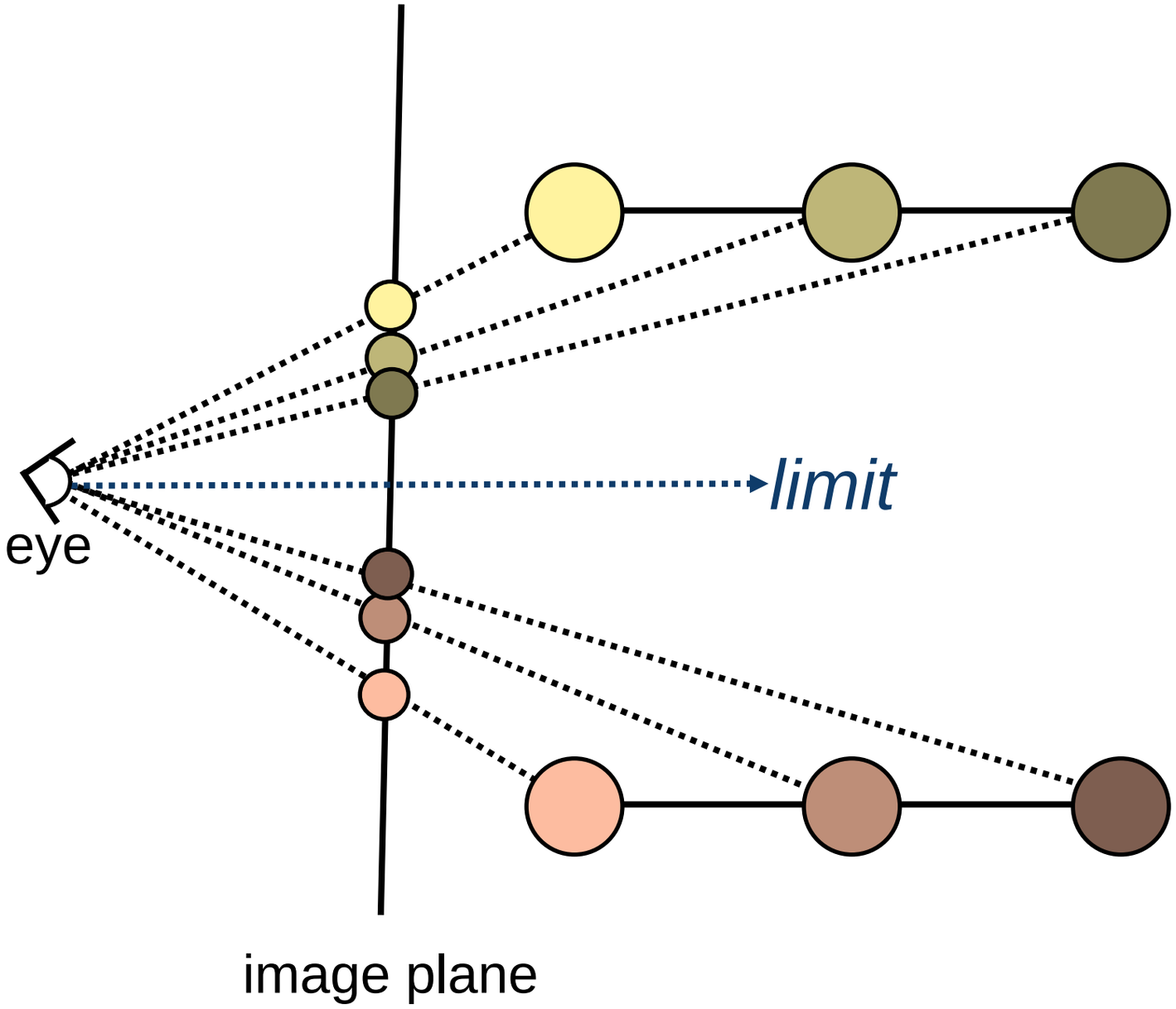
Vanishing Points

source: http://stevewebel.com/photographer/wp-content/uploads/2008/04/vanishing-point.jpg

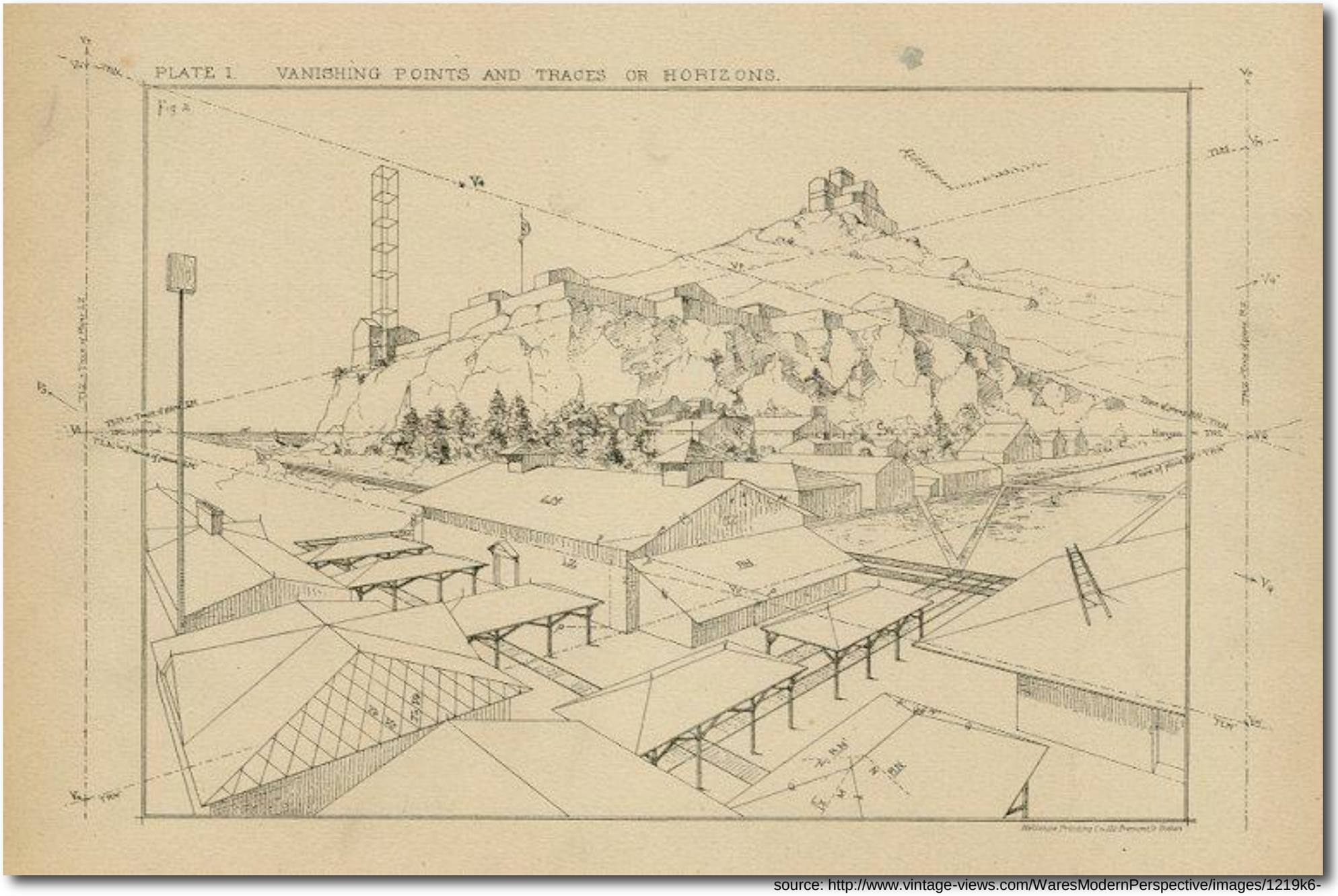


source: http://cavespirit.com/CaveWall/5/vanishing_point_high_horizon.jpg

What Causes Vanishing Points?


limit
eye
image plane

2 Vanishing Points



source: http://www.vintage-views.com/WaresModernPerspective/images/1219k6-Plate1.jpg

How many vanishing points can an image have?

Announcements

Project 1 Due Thursday