Lecture 5

Cameras, Projections, and Clipping

Lecture outline:

1. Pin-hole camera
2. Family of Projections
3. Parallel Projection
4. Perspective Projection

Basic Concepts

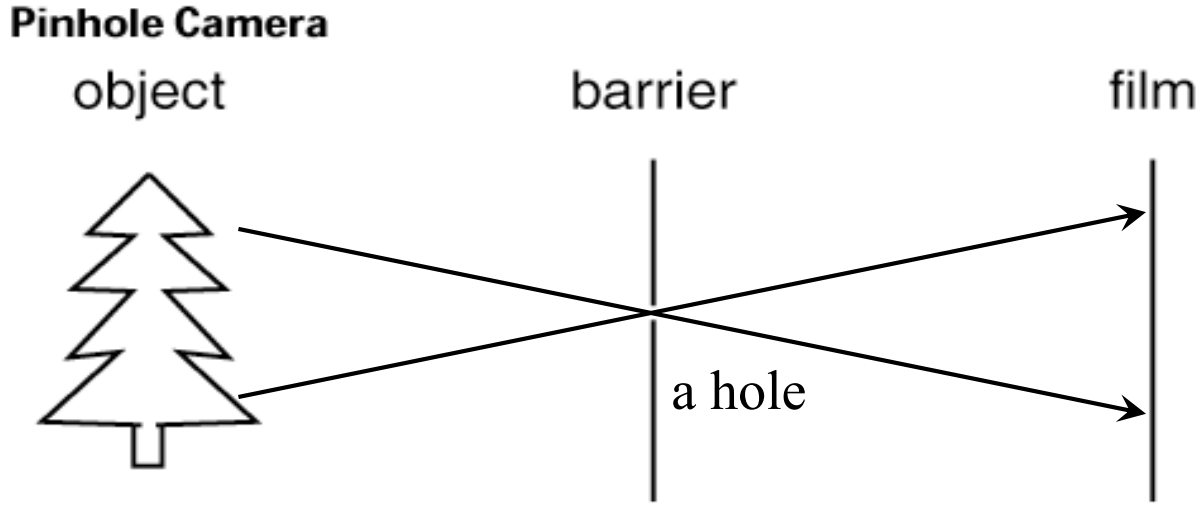
5. OpenGL Projection Model: 4x4 projection matrix
6. Clipping and Perspective Division

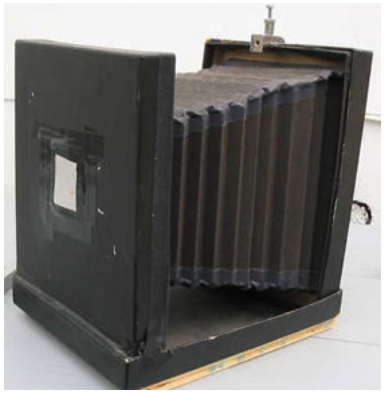
Camera: Forming an image

- Last lecture: we built a 3D geometric world
- Now, we need some sort of sensor to receive and record a picture.
- Is this all we need?

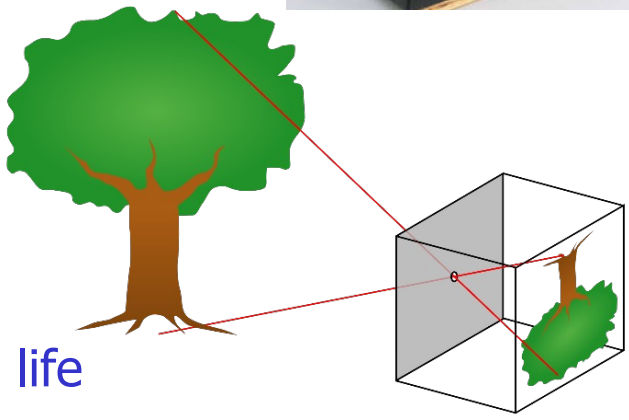
- Do we get a useful image?

Physical Model 1: Pin-hole Camera

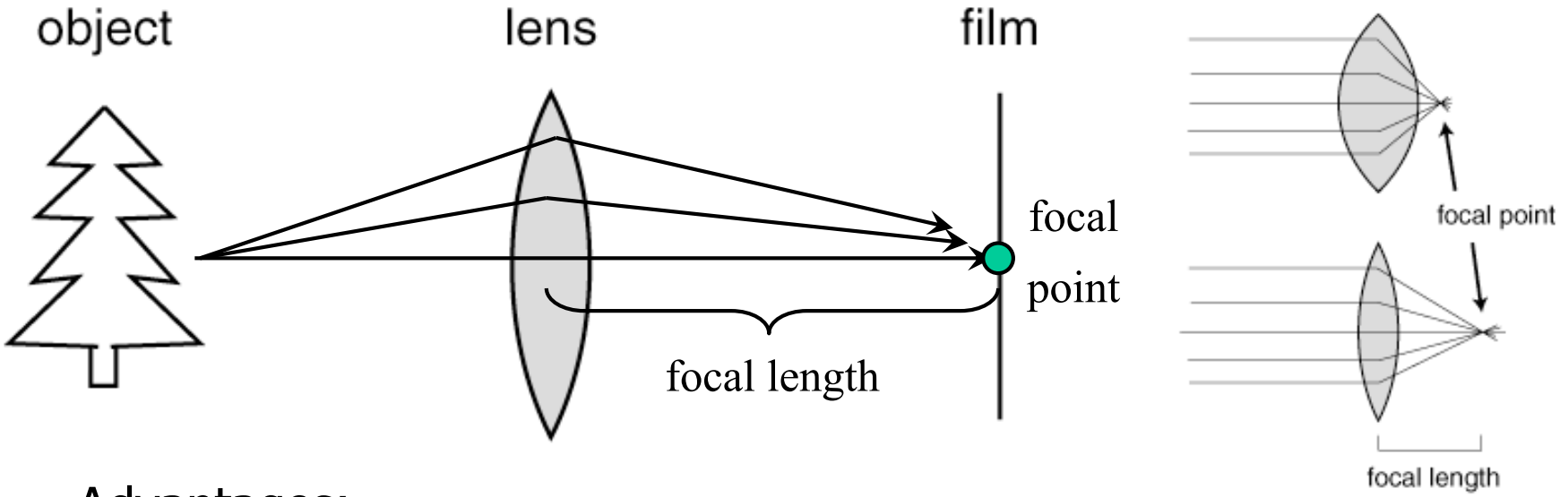




- Advantages:
  - easy to simulate
  - everything is in focus
- Disadvantages:
  - needs a bright scene (or long exposure) – in real life
  - everything is in focus (no depth of field) – not realistic (photography)



Physical Model 2: Camera with Lens



- Advantages:
  - no need to have bright scene - in real life
  - not everything is in focus (more realistic in terms of photography)
- Disadvantages:
  - need more programming to simulate
  - not everything is in focus (can only capture objects within a certain depth range)

Viewing in OpenGL

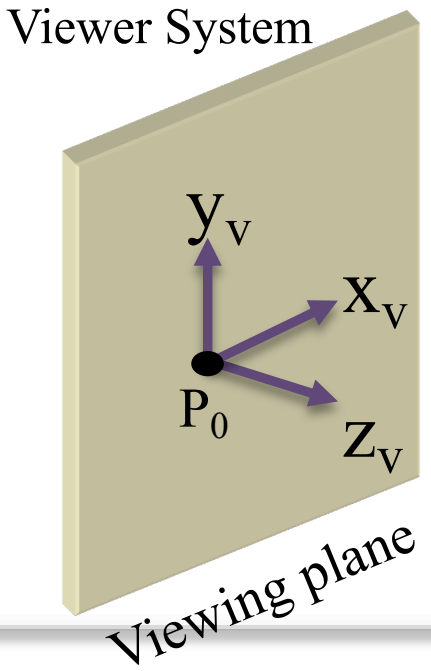
! "#$%&'$( ) *%+,$ - ' . , (/%01/2%02%0%+,3/405%10 6 $30

– What parameters do you need to define a camera?
  - Viewpoint (Center of Projection)
  - View direction
  - Field of view
  - Film size
  - Projection plane

Viewing

! 7,$ - ,(8%3$94,3$2%:%$5$6$(/2;

- Objects to be viewed
- A viewer with a projection surface
- A projection from the objects to the viewing surface



Viewing

! <=0 6 ' 5$;%%>%3$05%10 6 $30

- Objects: Whatever you're taking a picture of: landscape, people, etc.
- Viewer: The camera (with its film as the projection surface)
- Projection: Defined by the lens, maps 3D objects on to the 2D surface

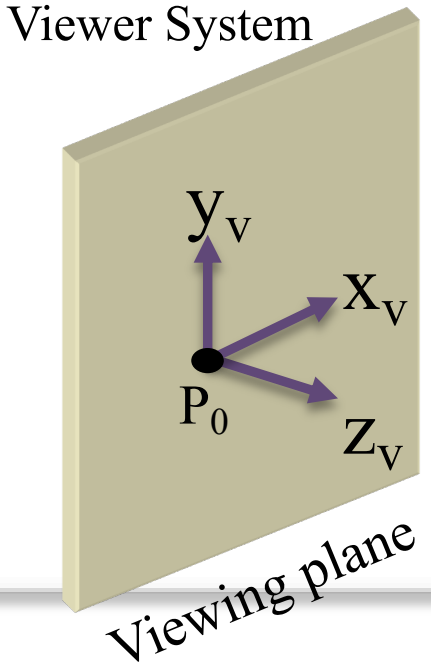
Viewing

! <=06 '5$;%&'$( )*%106$30

- Objects: The input geometry
- Viewer: The OpenGL “camera” (with the view volume as its viewing “surface”)
- Projection: The OpenGL projection matrix (GL_PROJECTION), maps 3D space (world coordinates) into 3D space (eye coordinates)
  - Eventually into normalized device coordinates (NDC)

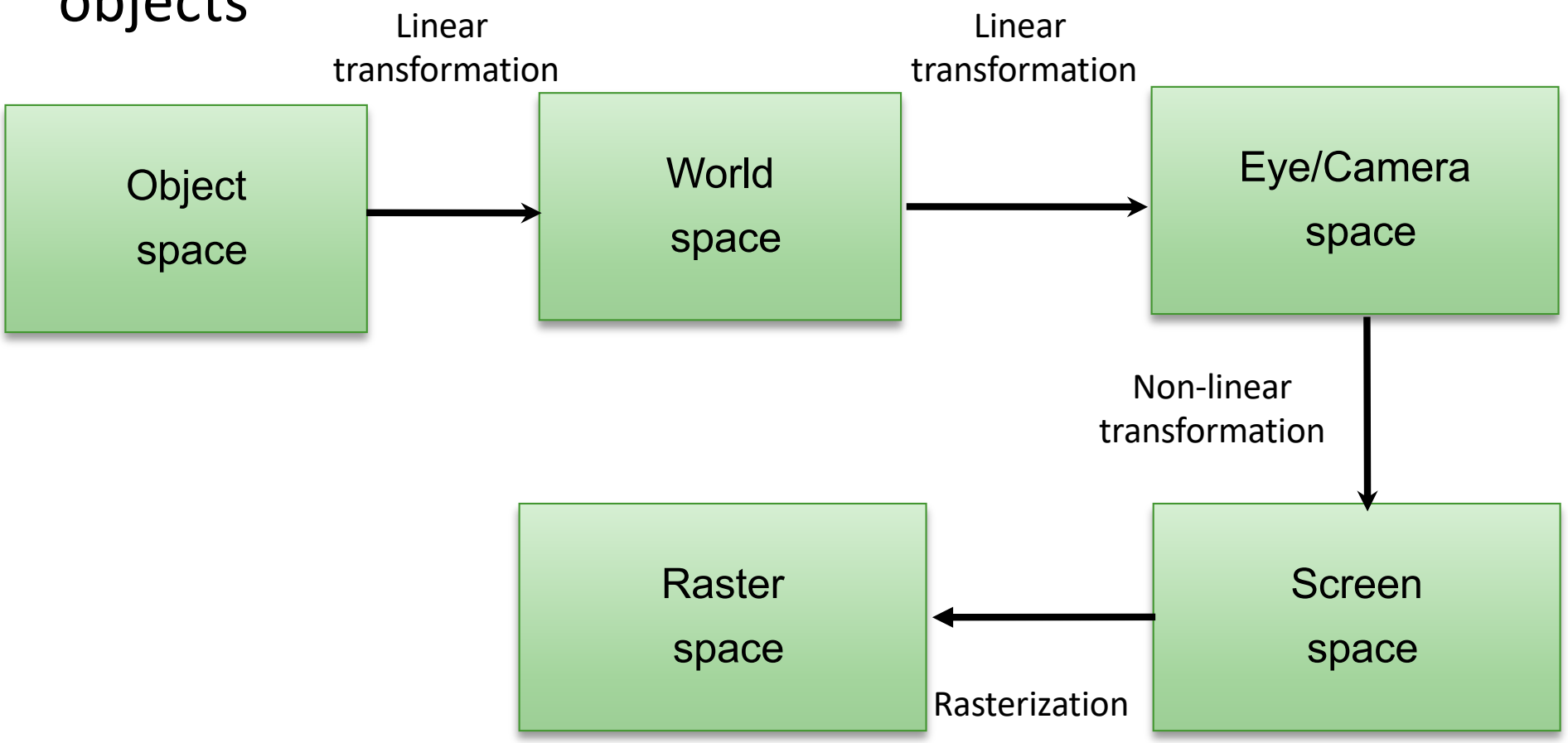
Planar Geometric Projections

- Standard projections are assumed to be onto a single plane
- A projection can be perspective or orthographic
  - In perspective projection, all rays converge at a single point (the center of projection, or COP)
  - In orthographic projection, all rays are parallel

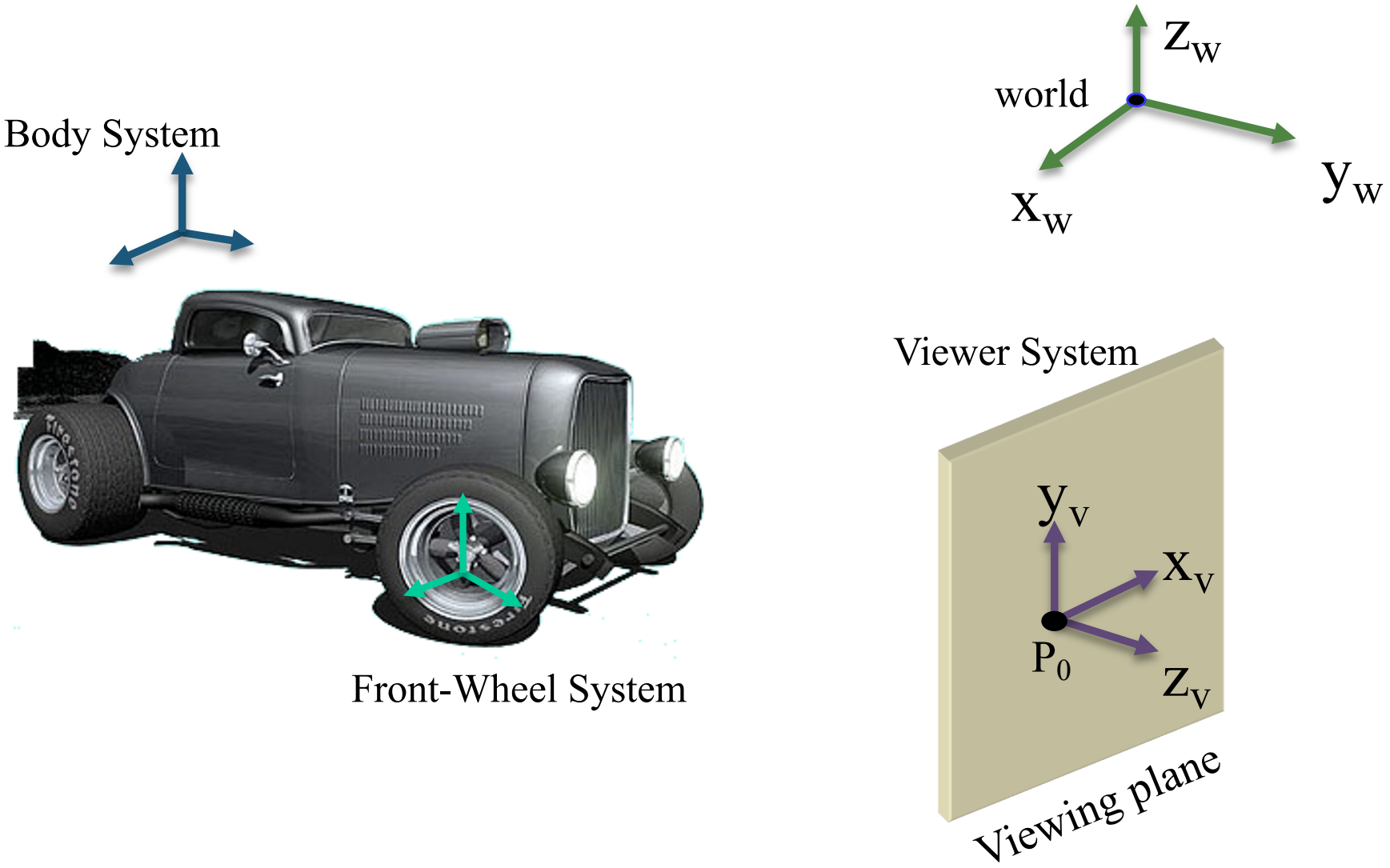


Coordinate Systems

- Coordinate systems used when rendering geometry objects

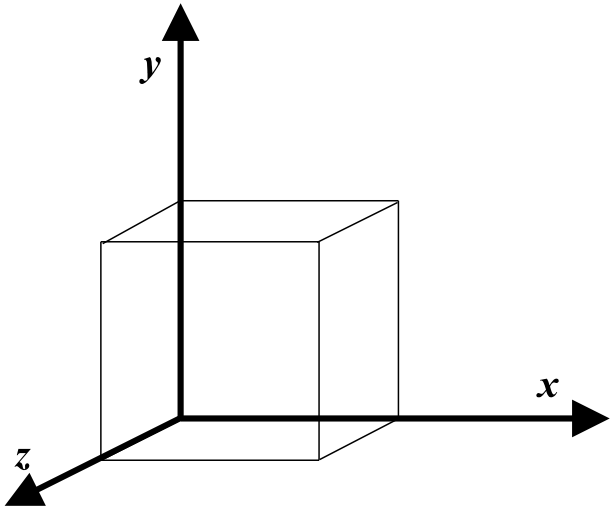


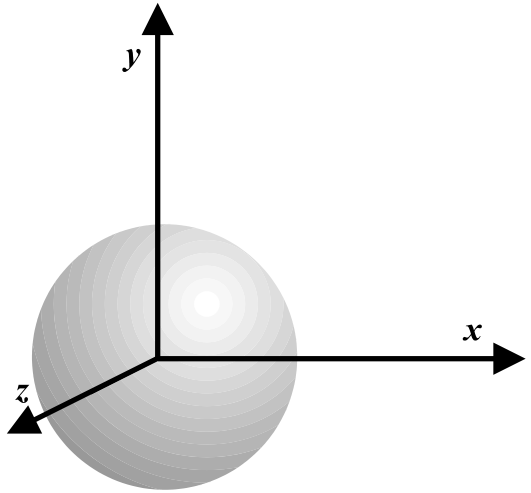
Viewing Coordinate System



Object Space

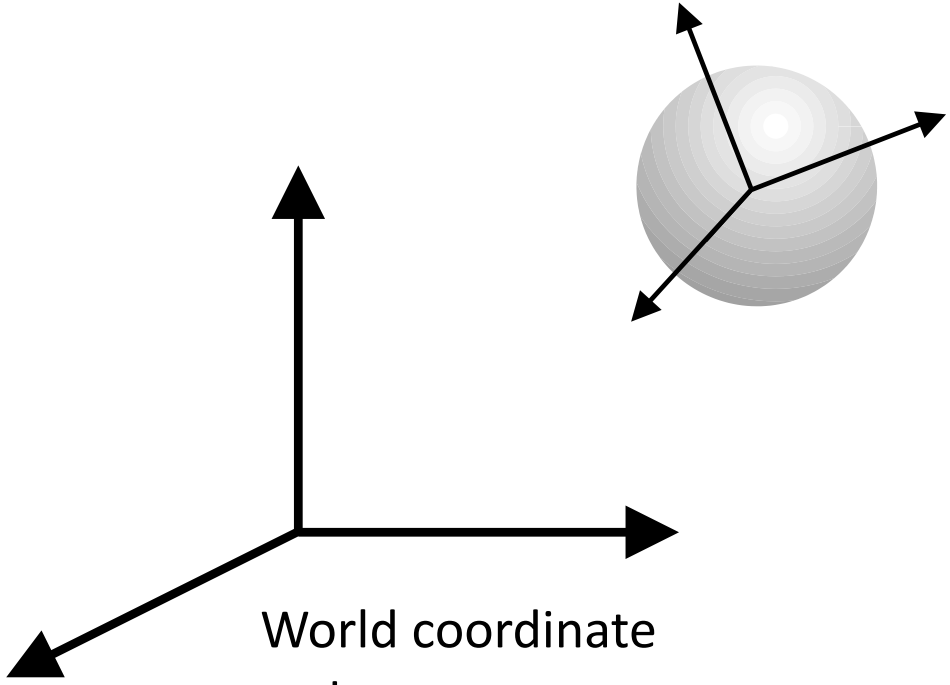
- "#$%%6 .2/%(0/4305%1. .3?,(0/$%2@2/$6%A.3%/#$% 8$. 6$/3@%.BC$1/
- *.105%1. .3?,(0/$%2@2/$6D%
- E*.105F%%/ .%/#$%.BC$1/
- >52.%G(. - (%02%6 .?$5,(8%1. .3?,(0/$%2@2/$6
- $D8D%

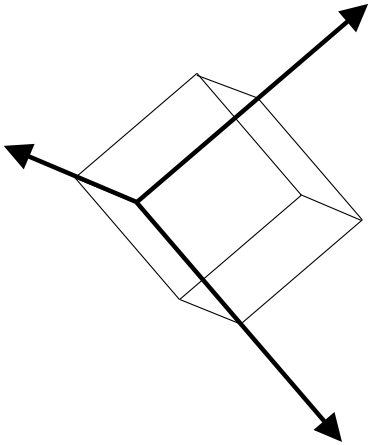




World Space

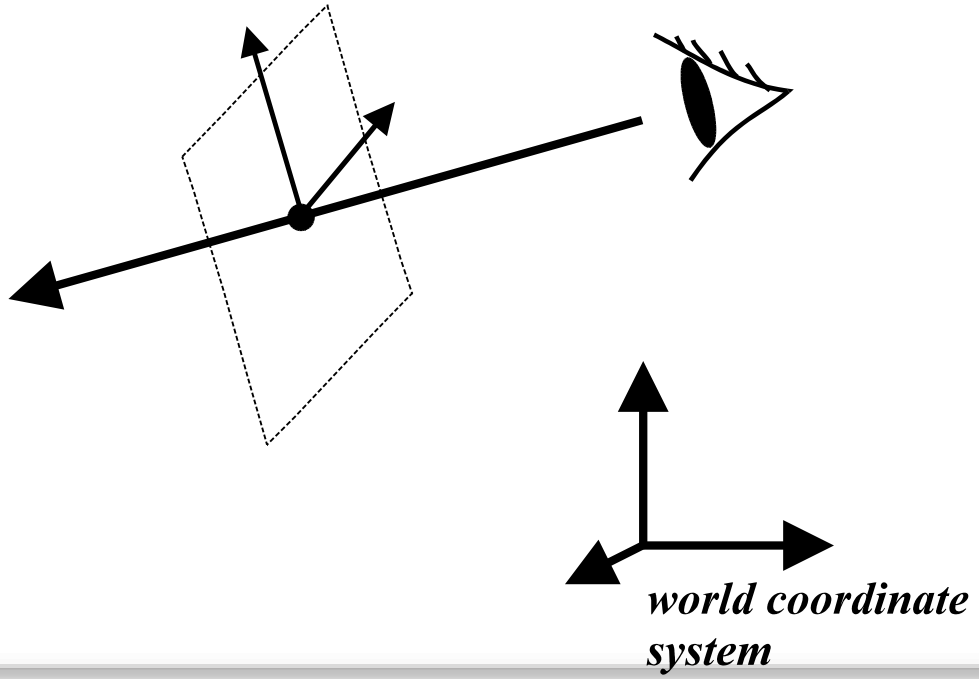
! "#$%85.BO5%H - .35?I%1..3?,(0/$%2@2/$6%/#0/%,2%
2#03$?%B@%O55%.BC$1/2%,(%/#$%6.?$5$?%21$($





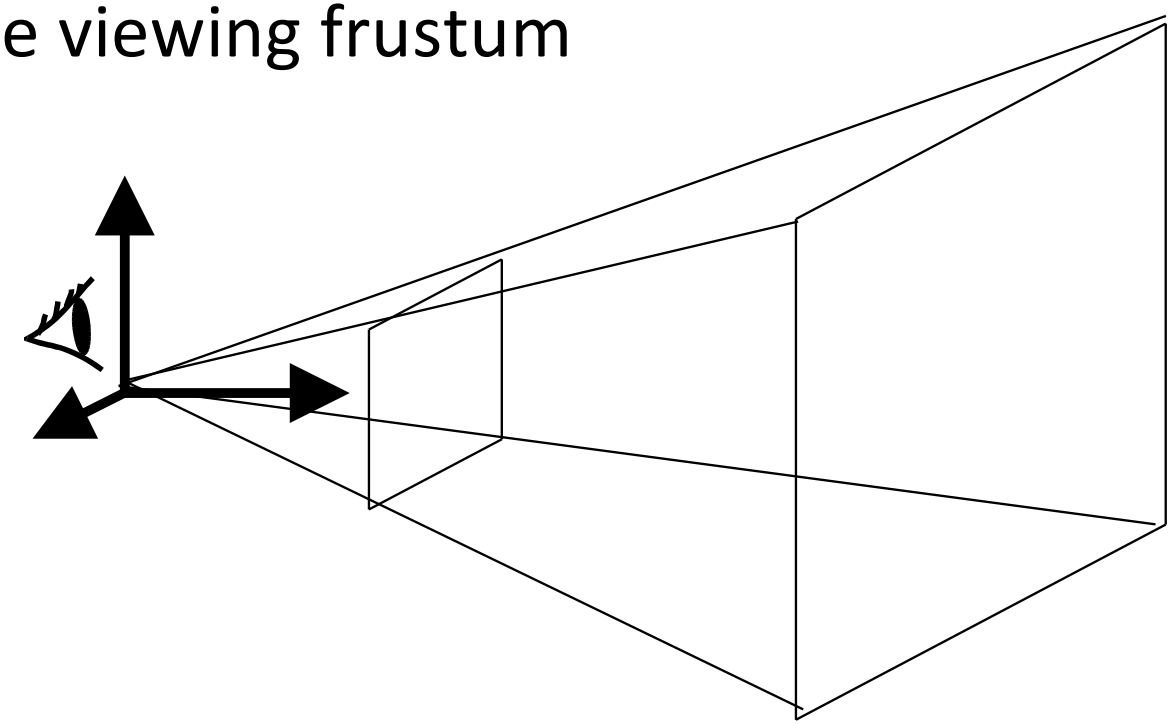
Eye/Camera Space

- The coordinate system with the eye or the center of projection as its origin
- A space in which the viewing volume is established
- Facilitates clipping (removing out-of-view objects)



Screen Space

- Transformation to screen space is a process that describes how light rays reach our eyes
- Screen space is defined to act within a closed volume called the viewing frustum

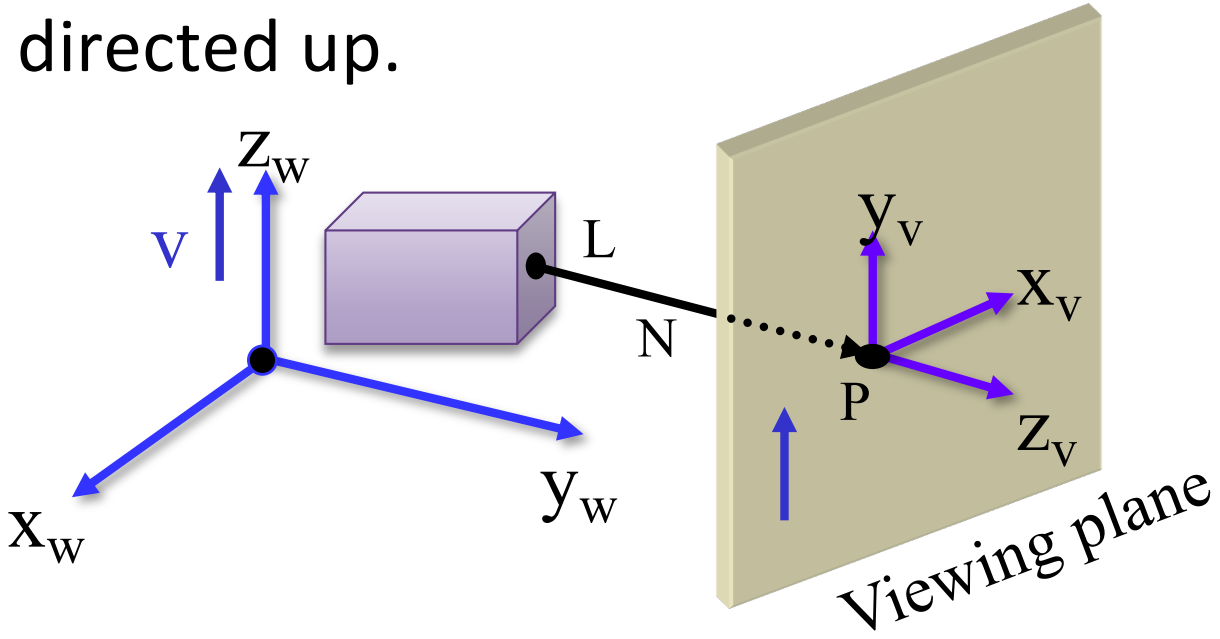


Specifying the Viewing Coordinate System

- *Viewing Coordinates system*J%%K=vJ%@vJ%LvMJ%%?$213,B$2%
  :N%.BC$1/2%-,/#%3$2'$1/%/.%O%+,$-$3D
- >%+,$-,(8%'5O($H*projection plane*I%,2%2$/%4'%
  '$3'$(?,14503%/.%Lv O(?%%%O5,8($?%-,/#%H=vJ@vID

! 0(%.3?$3%/.%2$/%0%+,$-,(8%'50($%-$%#0+$%/.%2'$1,A@;%

- P=(P_x,P_y,P_z) is a point where a camera is located.
- L - is a point to look-at.
- V - is the view up vector, whose projection onto the view-plane is directed up.



Viewing Coordinate System

! P. -%/.%A.36%7,$-,(8%1..3?,(0/$%2@2/$6%;

$$Z_v = \frac{P - L}{|P - L|} \quad ; \quad X_v = \frac{V \times Z_v}{|V \times Z_v|} \quad ; \quad Y_v = Z_v \times X_v$$

! "#$%/30(2A.360/,.(J%QJ%A3.6%-.35?R1..3?,(0/$%
,(/.%+,$-,(8R1..3?,(0/$2%,2;

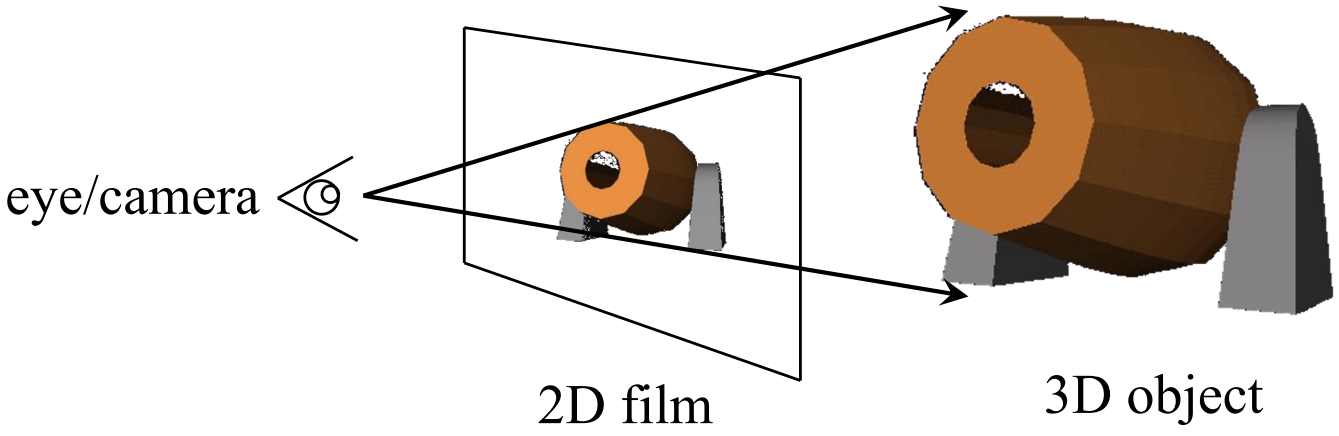
$$M = \begin{bmatrix} x_v^x & x_v^y & x_v^z & 0 \\ y_v^x & y_v^y & y_v^z & 0 \\ z_v^x & z_v^y & z_v^z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -P_x \\ 0 & 1 & 0 & -P_y \\ 0 & 0 & 1 & -P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = R \cdot T$$

3D Projection

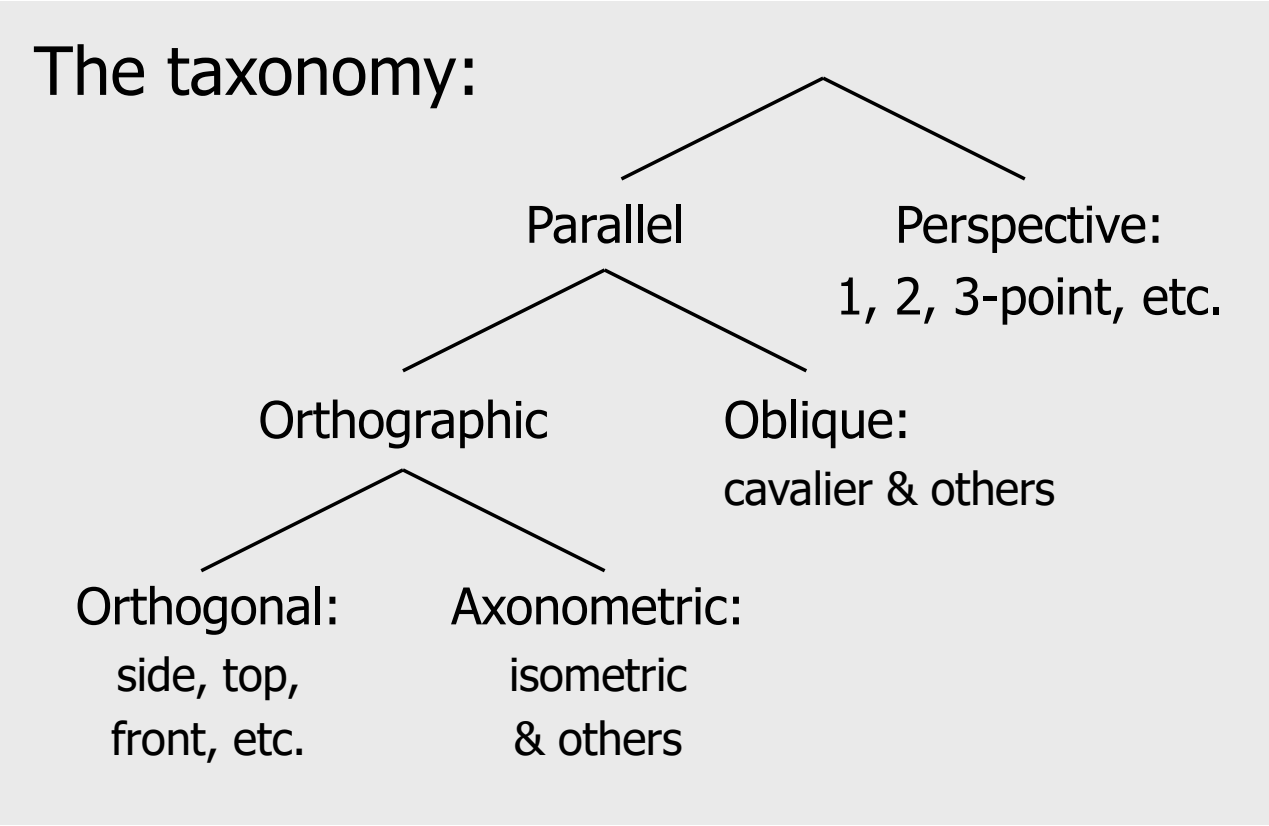
- Even we model the world in 3D, the final display on your monitor is 2D!!!
- Projection refers to the process of projecting the 3D world onto the tiny 2D screen, so that you can visualize on your monitor.
- There are two major types of projections:
  - Perspective projection, simulates your eye
  - Orthogonal projection, preserve length after projection, used in CAD systems.

What is a Projection?

- From a higher dimension to a lower dimension
  - E.g. from *n*-space to *m*-space such that $m < n$
- So, in 3D computer graphics, usually n=3 and m=2



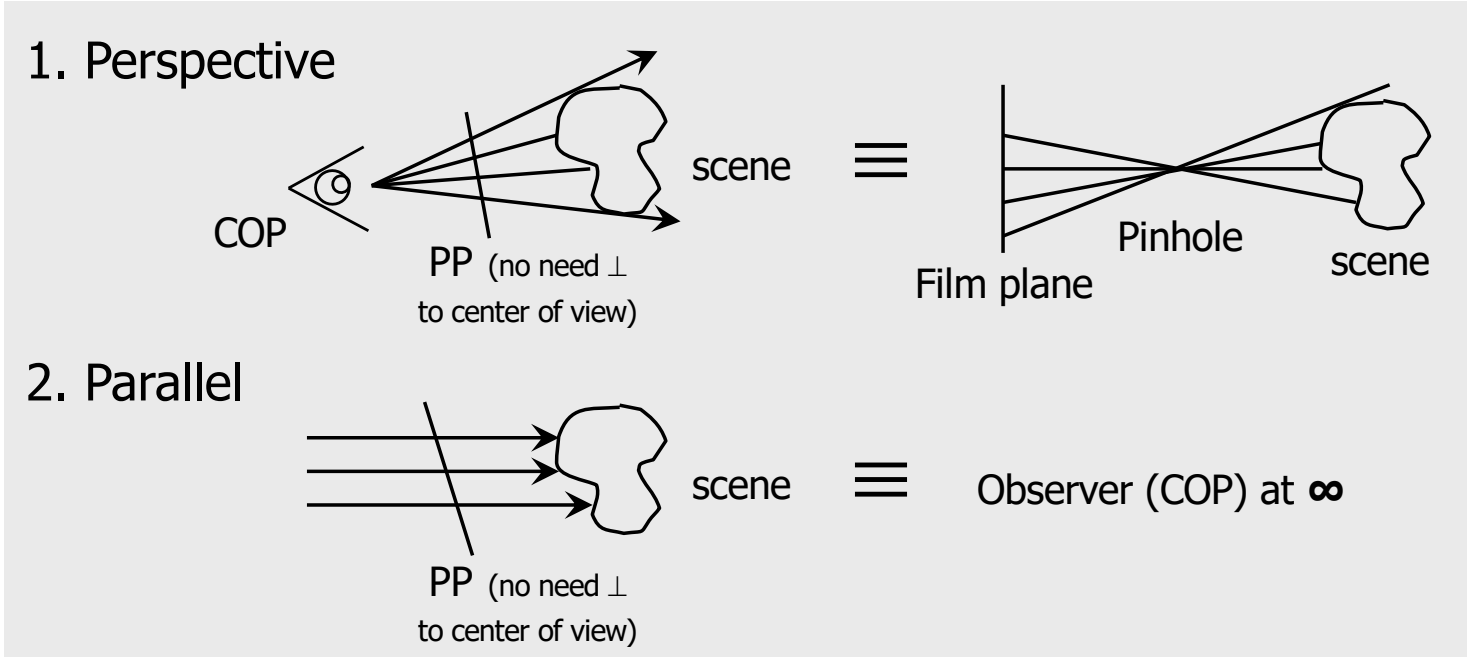
Family of Projections



Three basic terms:
- Center of projector (COP)
- Projection plane (PP)
- Projection line and Direction of Projection (DOP)

Parallel VS Perspective Projections

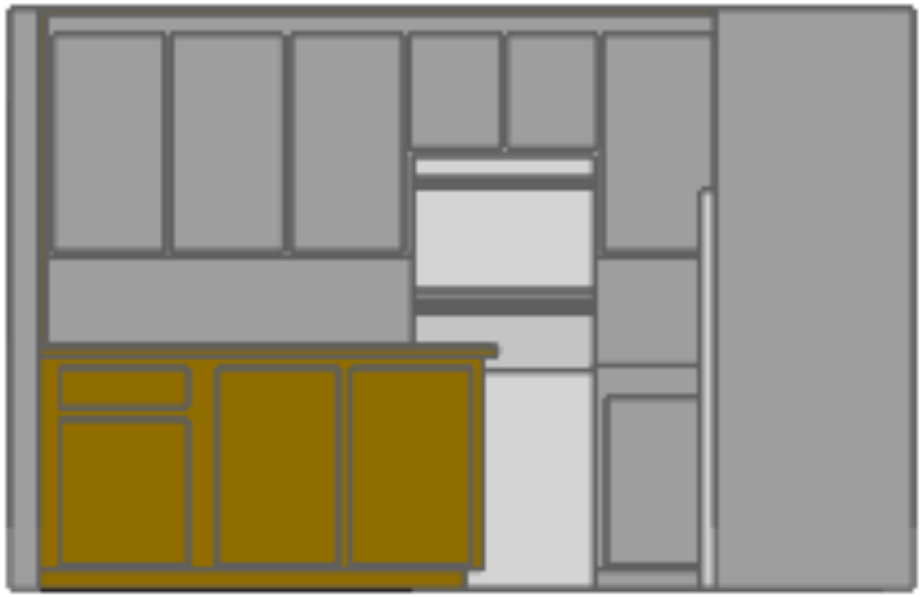
- In 3D, we map points from 3-space to the **projection plane** (PP) along **projection lines** emanating from the **center of projection** (COP):
- The center of projection (COP) is exactly the same as the pinhole in a pinhole camera



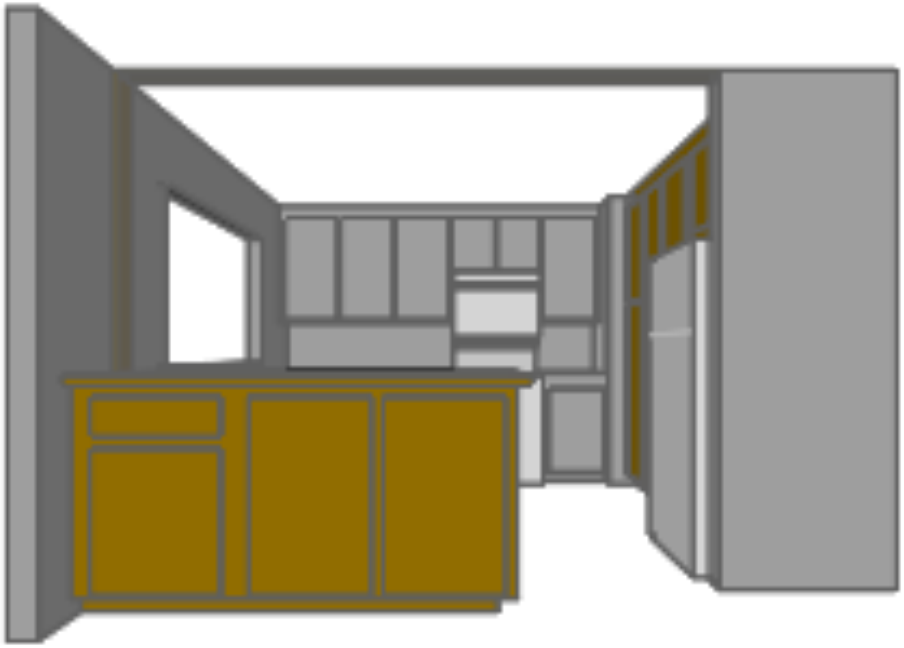
*** Finite distance between PP and COP**

*** Infinite distance between PP and COP**

Examples



Parallel projection

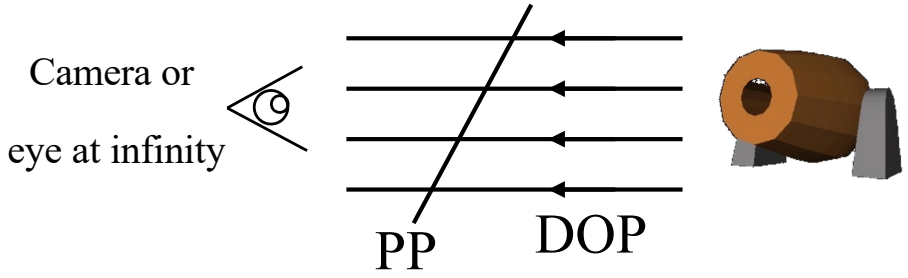


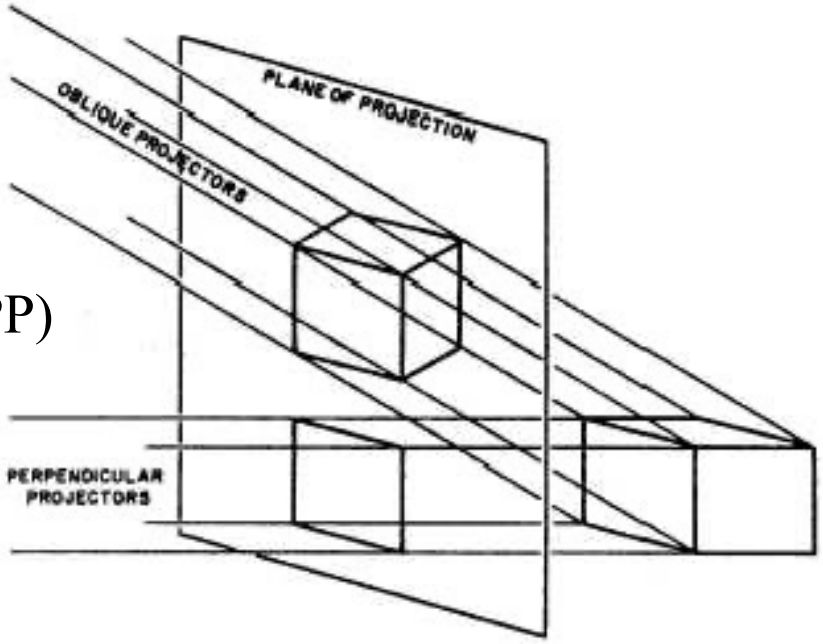
Perspective projection

Parallel Projection

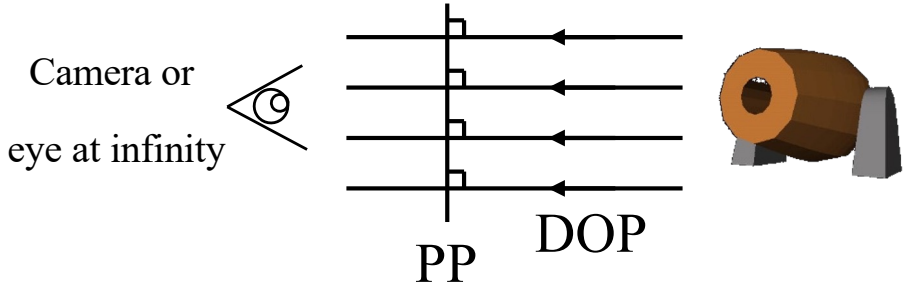
In Parallel Projection, we can specify **Direction of Projection** (DOP) because all projection lines are **parallel**.

1. Oblique Projection (DOP NOT perpendicular to PP)



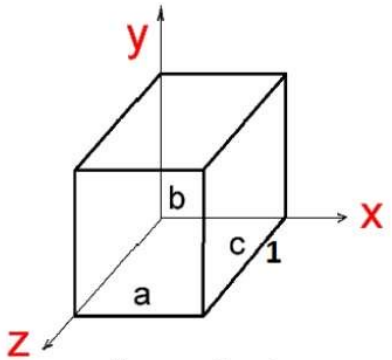


2. Orthographic Projection (DOP perpendicular to PP)

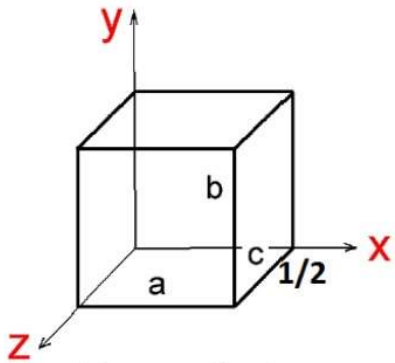


Oblique Projection (parallel)

- It is like shearing the geometry along PP
- Two standards of Oblique Projection:

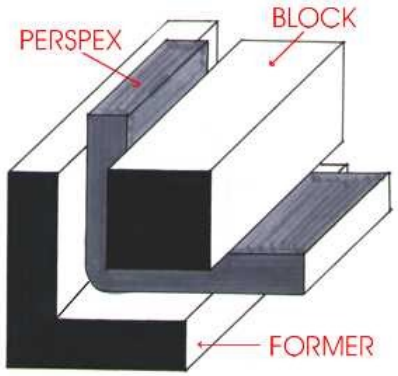


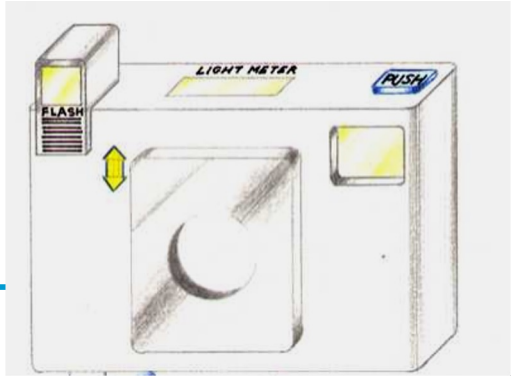
Cavalier Projection



Cabinet Projection

1. Cavalier Projection
   - DOP makes 45° angle with PP
   - Does not foreshorten lines perpendicular to PP
   - So, it preserves lengths
2. Cabinet Projection
   - DOP makes 63.4° angle with PP
   - Foreshorten lines perpendicular to PP by one-half
     * Also known as “Chinese Perspective”





Orthographic Projection (parallel)

1. Orthogonal Projection

   - DOP align with object's X, Y, or Z axes

   - implementation: we can simply throw away the X, Y, or Z component (put it as zero) of the eye coordinate

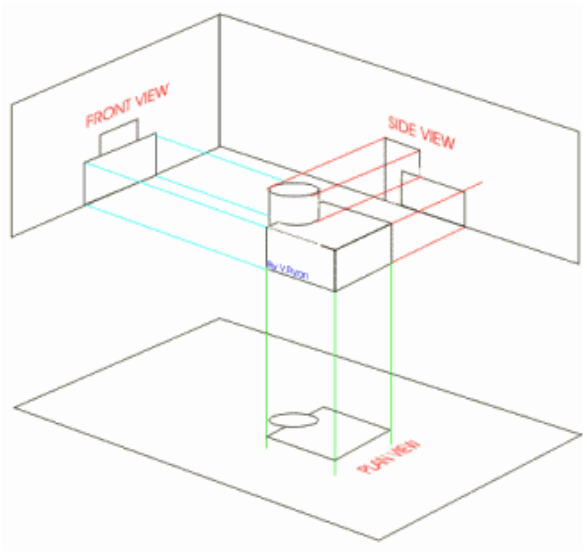
   - "side", "top/plan", or "front" views

2. Axonometric Projection

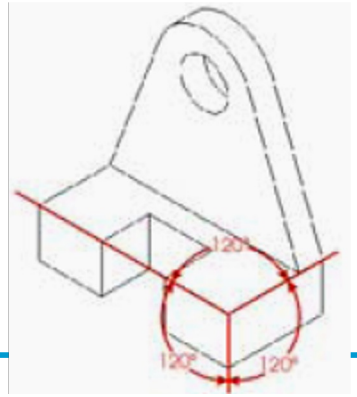
   - arbitrary DOP, not parallel to world X, Y, nor Z axes

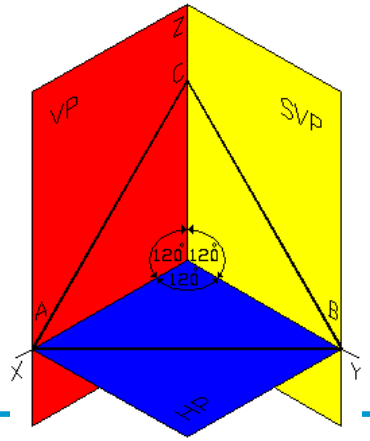
   - special case: isometric (exactly 120°)

Example:



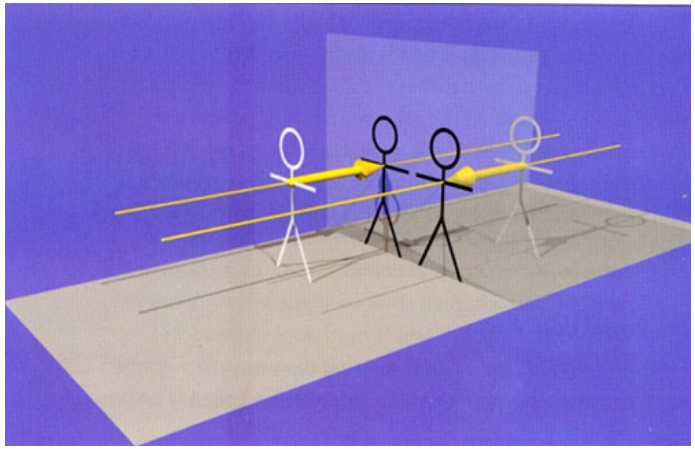
Example (isometric):





simulation games

Parallel Projection



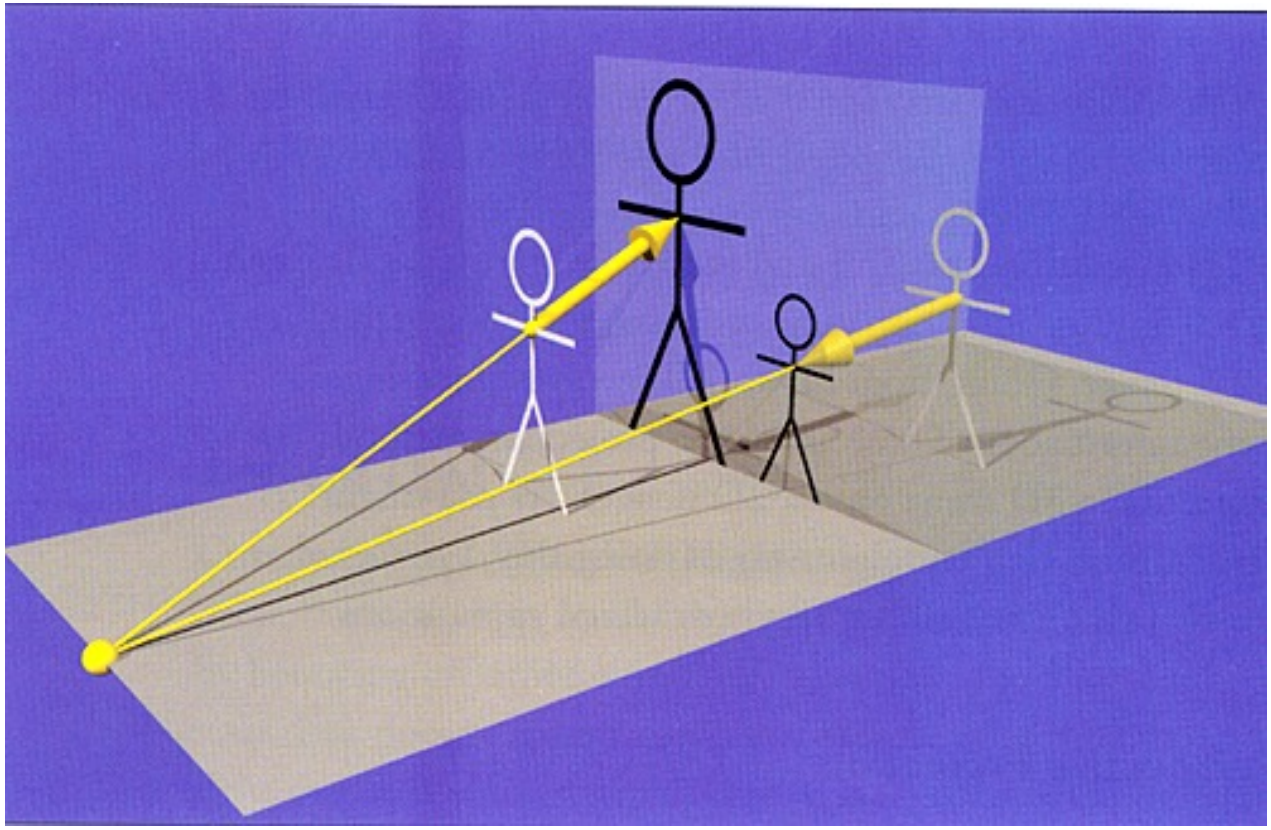
Properties of parallel projection:

- Not realistic looking
- Good for exact measurements
- Are actually a kind of affine transformations
  - Parallel lines remain parallel
  - Angles not (in general) preserved
- Most often used in CAD, architecture drawings,

  etc., where taking exact measurement is important

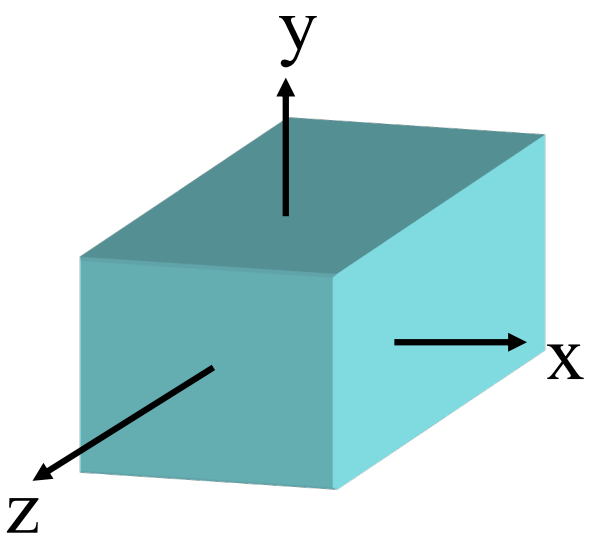
Perspective Projection

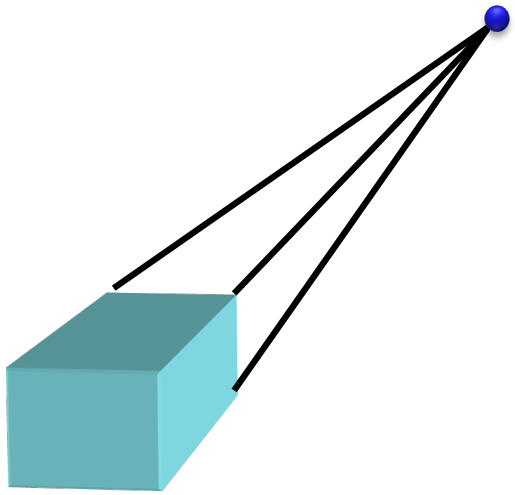
- COP is at a finite distance to the PP, so the further away an object is, the smaller it is
- This is how our eyes see the nature



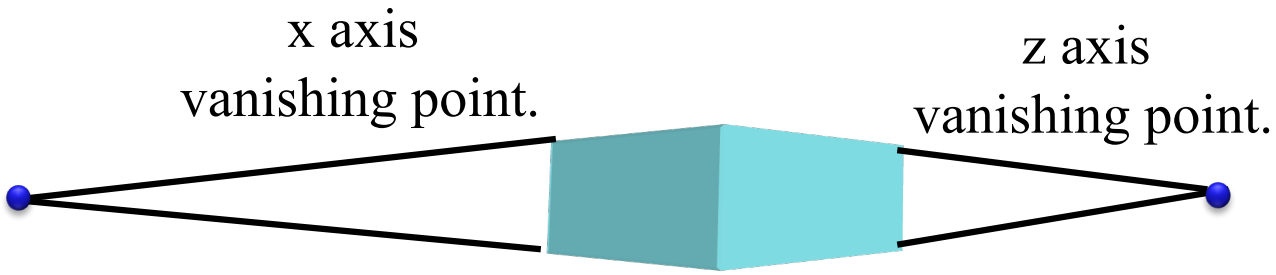
Vanishing Points

- There are infinitely many general vanishing points.
- There can be up to three *principal vanishing points* (axis vanishing points).
- Perspective projections are categorized by the number of principal vanishing points, equal to the number of principal axes intersected by the viewing plane.
- Most commonly used: one-point and two-points perspective.





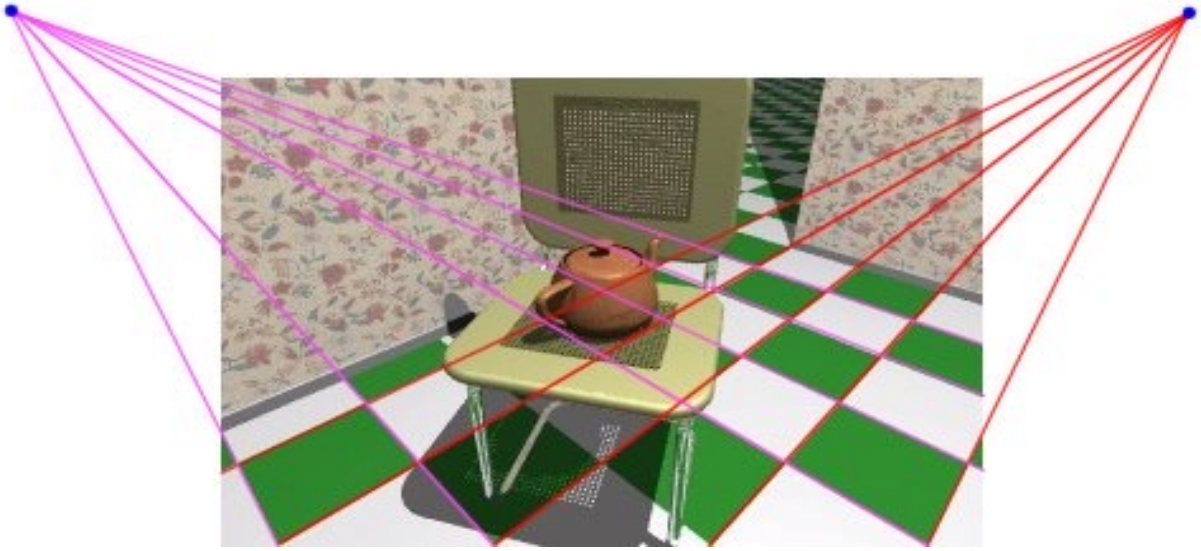
One point (z axis)
perspective projection



Two points
perspective projection

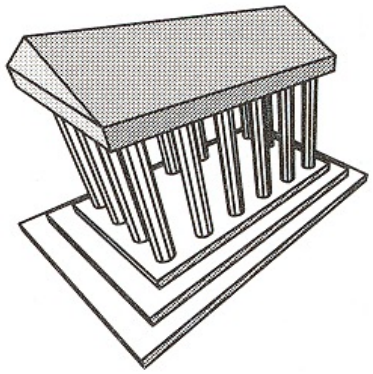
Perspective Transformation

- Parallel lines appear to converge to single point

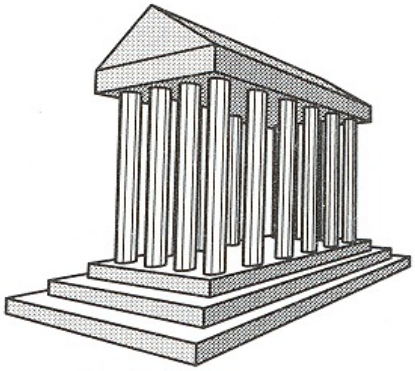


Perspective Projection

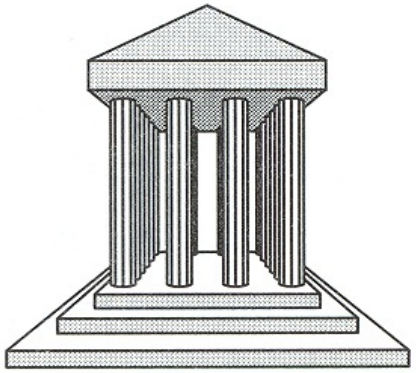
! P. -%60(@%+0(,2#,(8%' .,(/2S



3-Point
Perspective



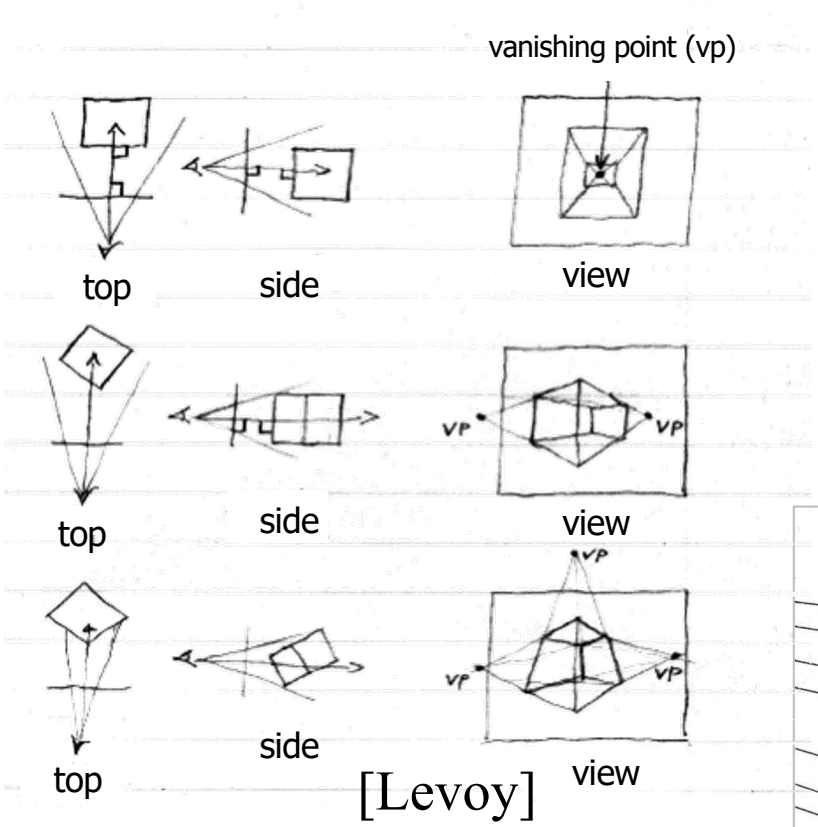
2-Point
Perspective



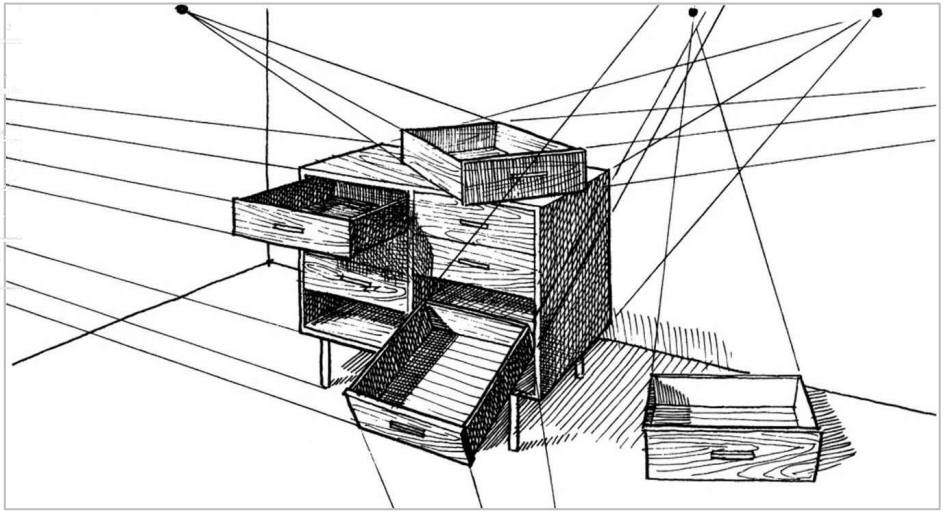
1-Point
Perspective

Perspective Projection

- **Vanishing points** – points at which parallel lines meet on PP



[Levoy]

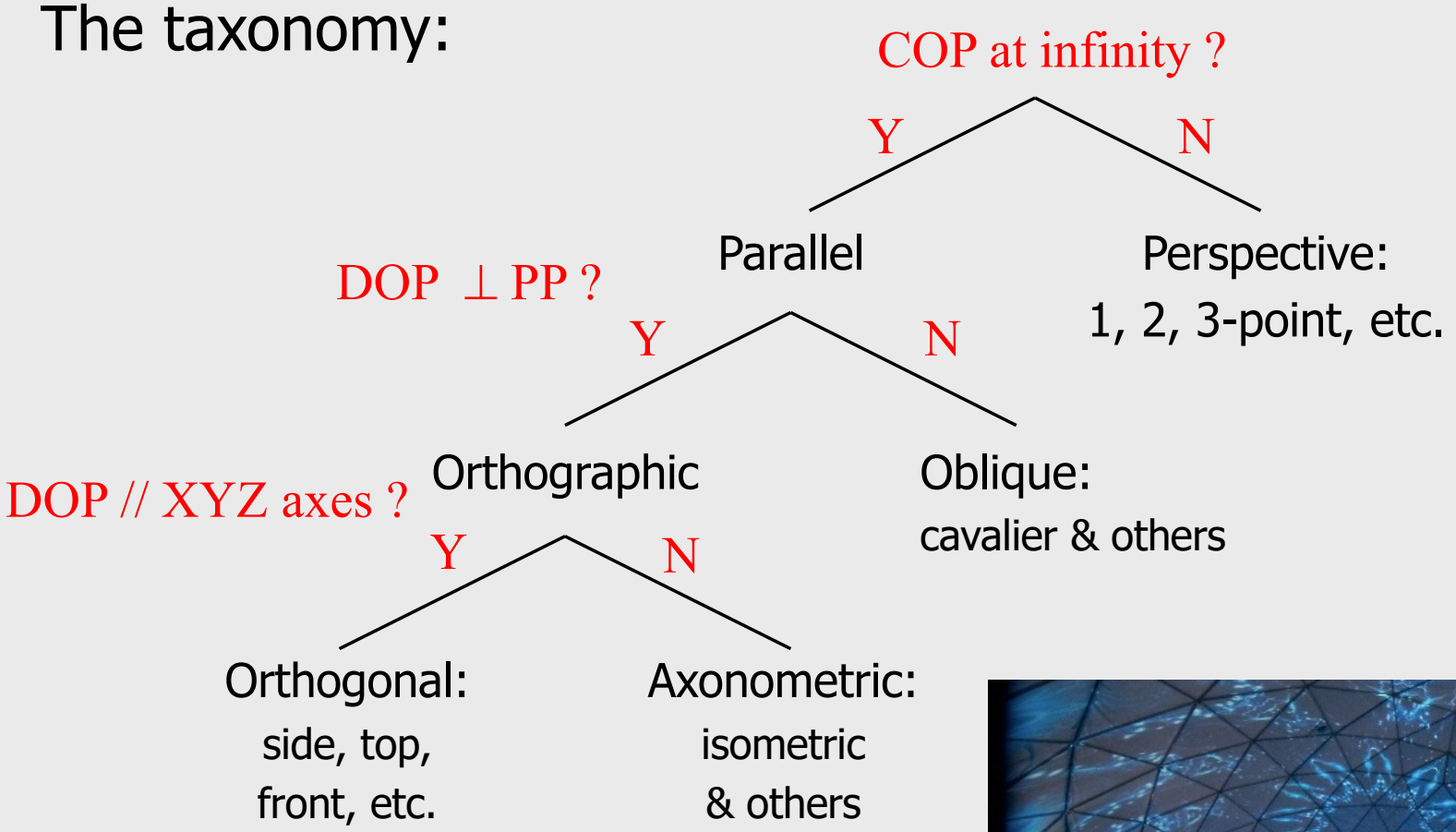


Q. How many vanishing points can a picture contain?

A. One per direction of line in the scene

Summary: Projections

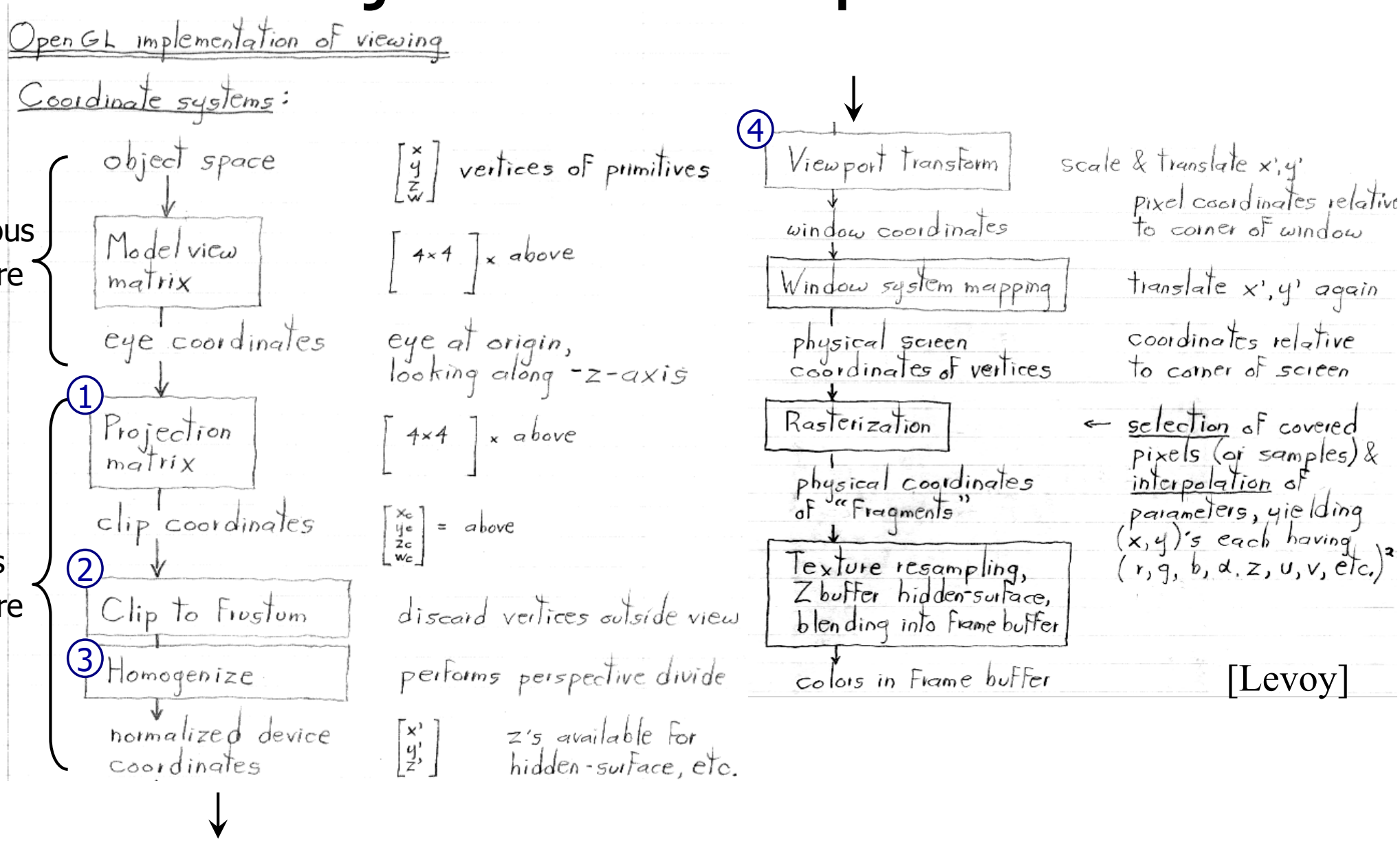
The taxonomy:



Note: all these are planar projections

(other projections: planetarium, etc.)

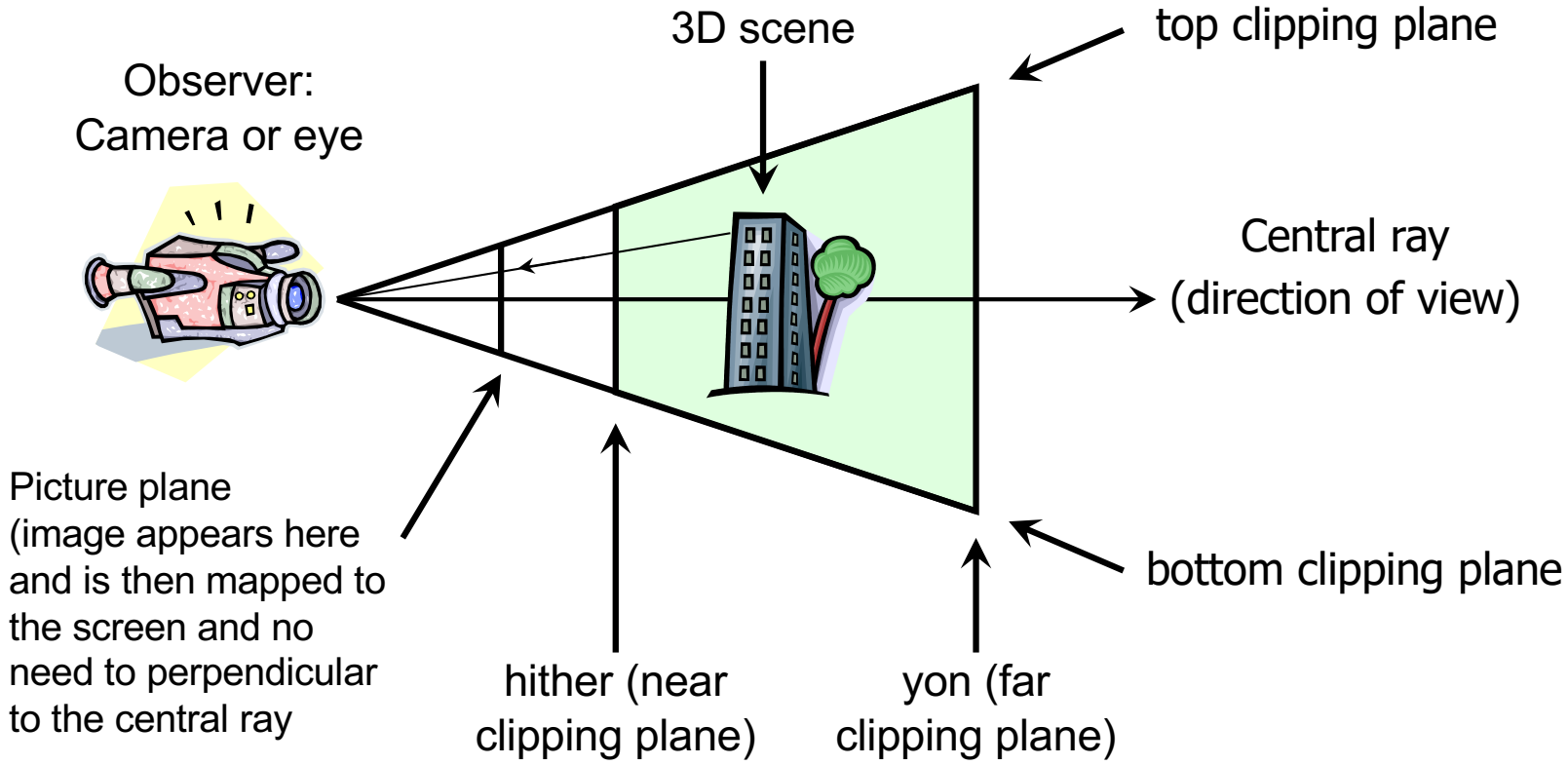
Projections in OpenGL



View Frustum: the "Visible Volume"

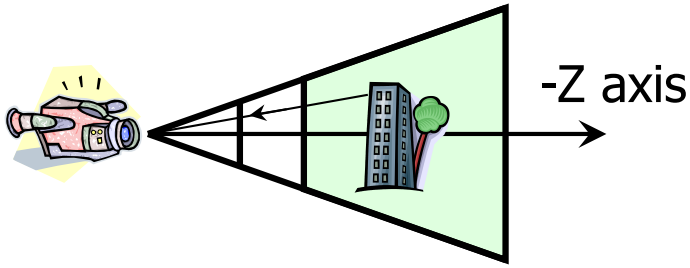
Specifying projection:

The Viewing Frustum



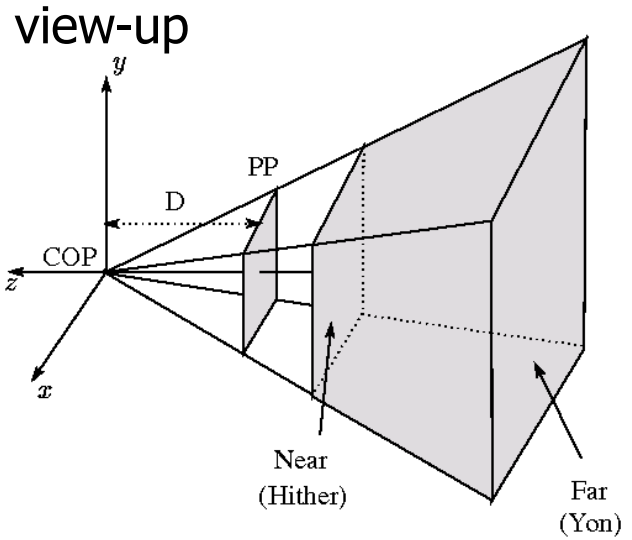
Projection in OpenGL

(all in eye space)



OpenGL Projection Settings:

1. Eye/Camera position: Origin
2. Viewing direction: Negative Z-axis
3. View-up direction: Positive Y-axis
4. Viewing region (viewing frustum):
   - between near and far clipping planes
   - between bottom and top clipping planes
   - between left and right clipping planes
5. Visible depth range: Hither-yon distance (far – near)



#1: Orthographic Projection in OpenGL

glOrtho(l,r,b,t,n,f) or gluOrtho2D(l,r,b,t)

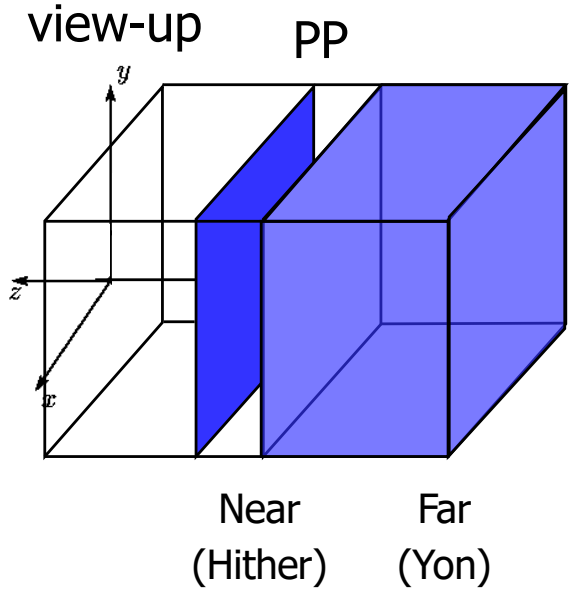
- Left/right/bottom/top/near/far
- gluOrtho2D: near = 0 and far = 1

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & -\frac{2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Clip coordinate

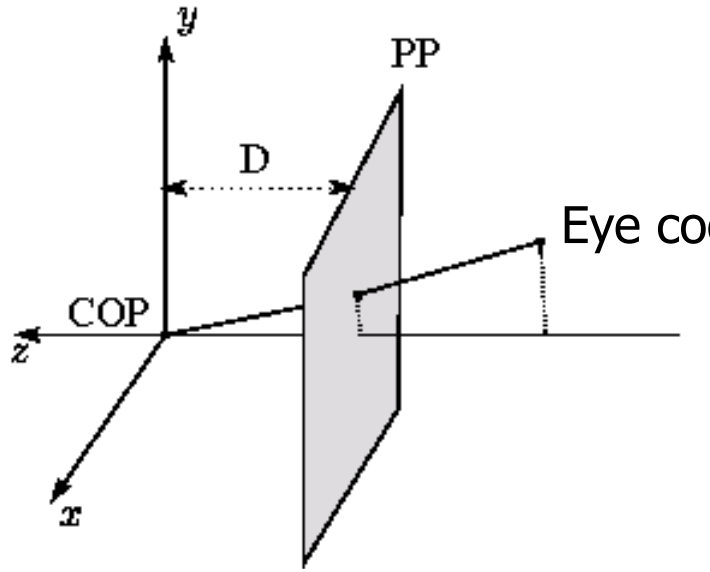
Projection Matrix (Orthographic)

Eye coordinate (after Modelview transformation)

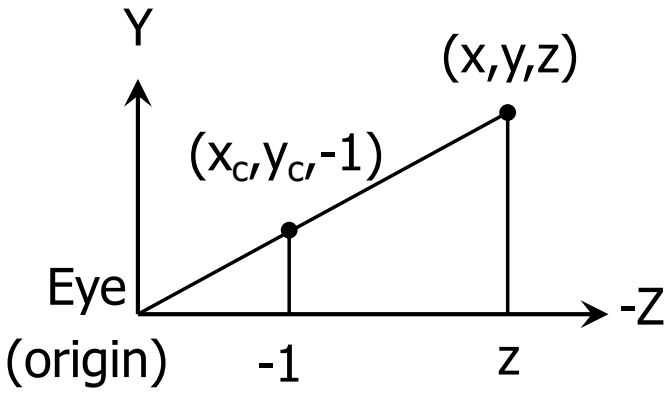


#2: Perspective Concept in OpenGL

Consider the projection of a point onto the projection plane:



Eye coordinate



By similar triangles, we can compute how much the x and y coordinates are scaled:

By similar triangles: $x_c = -\frac{x}{z}$ and $y_c = -\frac{y}{z}$

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ * & * & * & * \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

#2: Perspective Concept in OpenGL

How about z_c?

- We need it for hidden-surface removal
- The smaller z_c is, the closer the clip coordinate is from the eye

  (note: Z is flipped after this matrix multiplication)

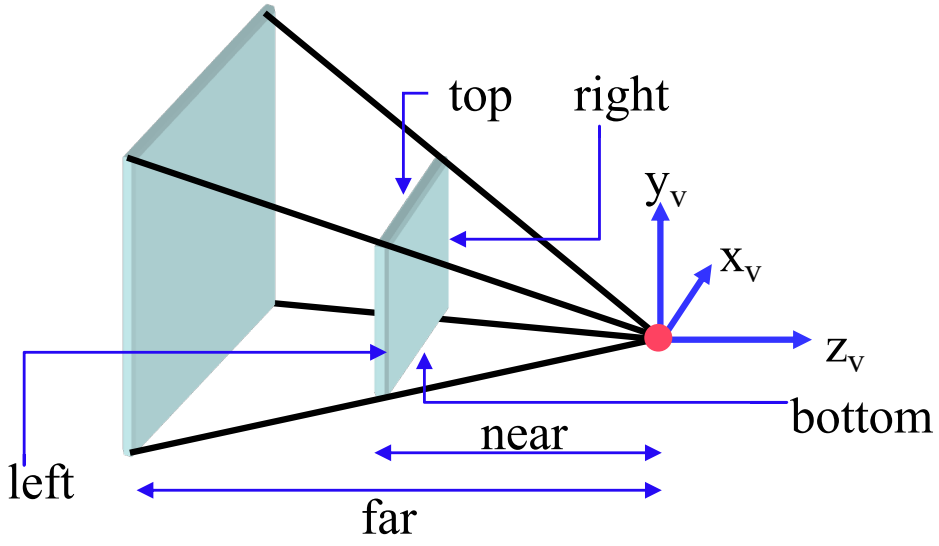
We use *glFrustum(l,r,b,t,n,f)*.

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

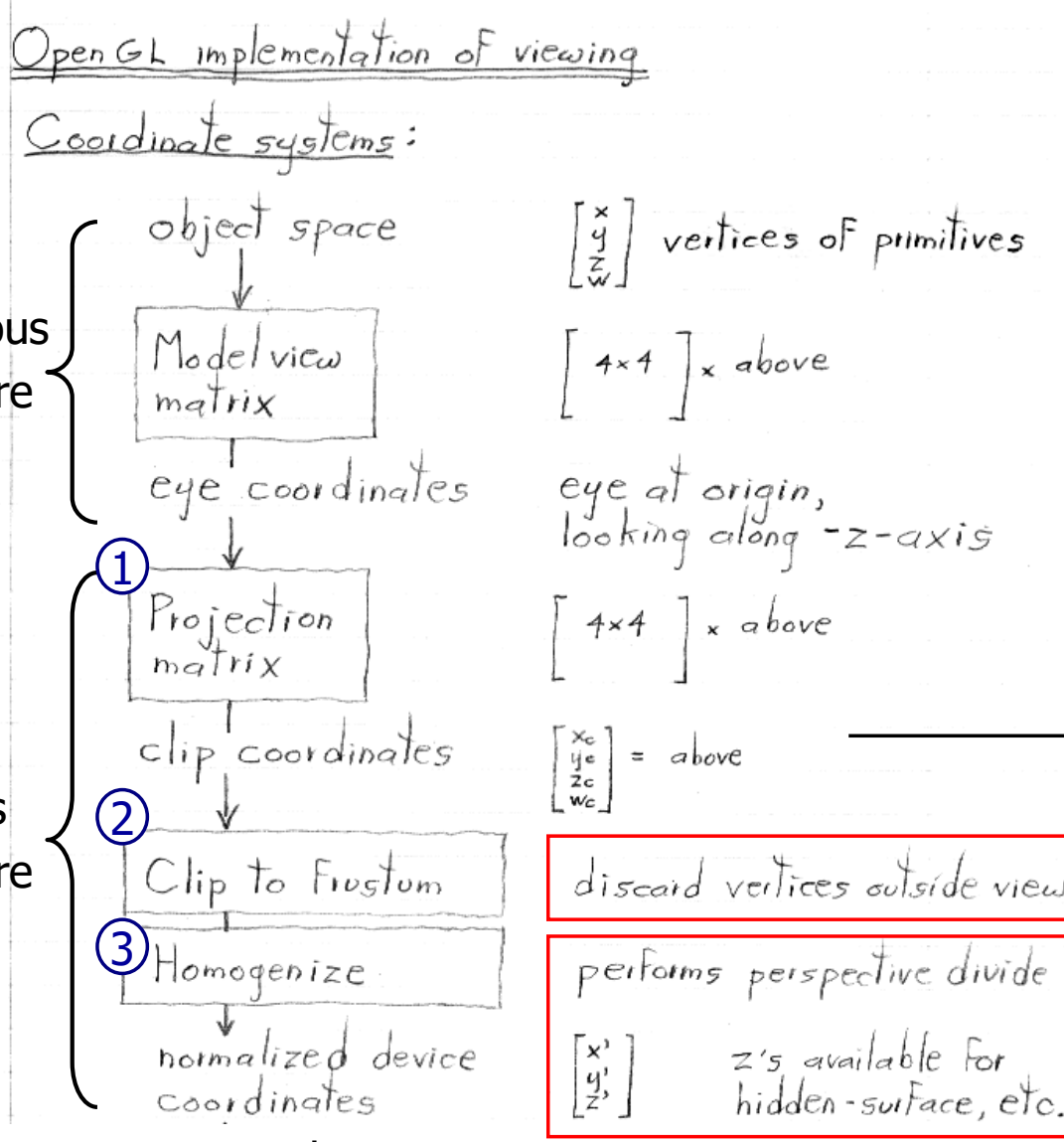
Clip coordinate

Projection Matrix (Perspective)

Eye coord. (after Modelview)



Clipping and Perspective Division



So, after multiplying the eye coord. with the Projection matrix, we obtain the clip coord. (for clipping and Perspective division)

[Levoy]

Clipping and Perspective Division

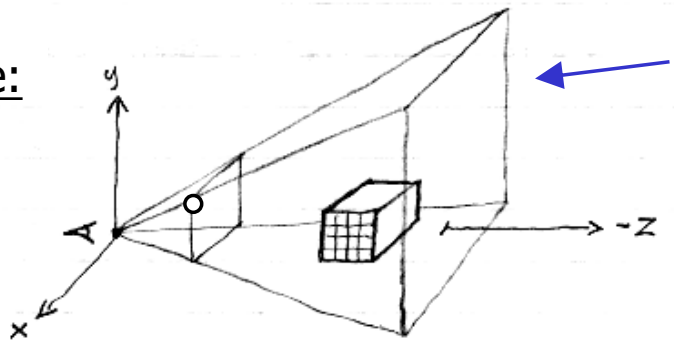
Perspective Division (or homogenization):

$$x' = \frac{x_c}{w_c} \quad y' = \frac{y_c}{w_c} \quad z' = \frac{z_c}{w_c}$$

Normalized device coordinates (NDC)

"Perspective" interpreted as a distortion of 3-space, i.e., view frustum

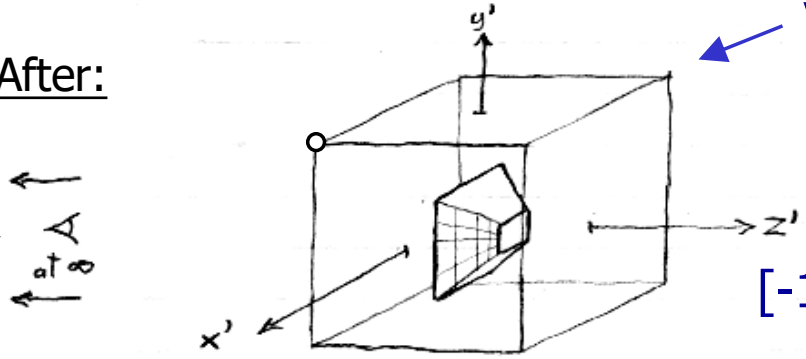
- Before:



View frustum

Eye space

- After:



View frustum after transformation

NDC Space
[-1,+1] along X/Y/Z

View frustum becomes a cube

[Levoy] Note: compression of grid lines in x', y' and z'

Clipping and Perspective Division

Clipping before VS after homogenization:

- After:

$$\text{If } \begin{cases} -1 \le x' \le 1 \text{ and} \\ -1 \le y' \le 1 \text{ and} \\ -1 \le z' \le 1 \end{cases} \text{ then in else out}$$

- Before:

$$\text{If } w_c > 0 \text{ then}$$

$$\text{if } \begin{cases} -w_c \le x_c \le w_c \\ -w_c \le y_c \le w_c \\ -w_c \le z_c \le w_c \end{cases} \text{ then in else out}$$

multiplying through by w_c

$$\text{else}$$

$$\text{if } \begin{cases} -w_c \ge x_c \ge w_c \\ -w_c \ge y_c \ge w_c \\ -w_c \ge z_c \ge w_c \end{cases} \text{ then in else out}$$

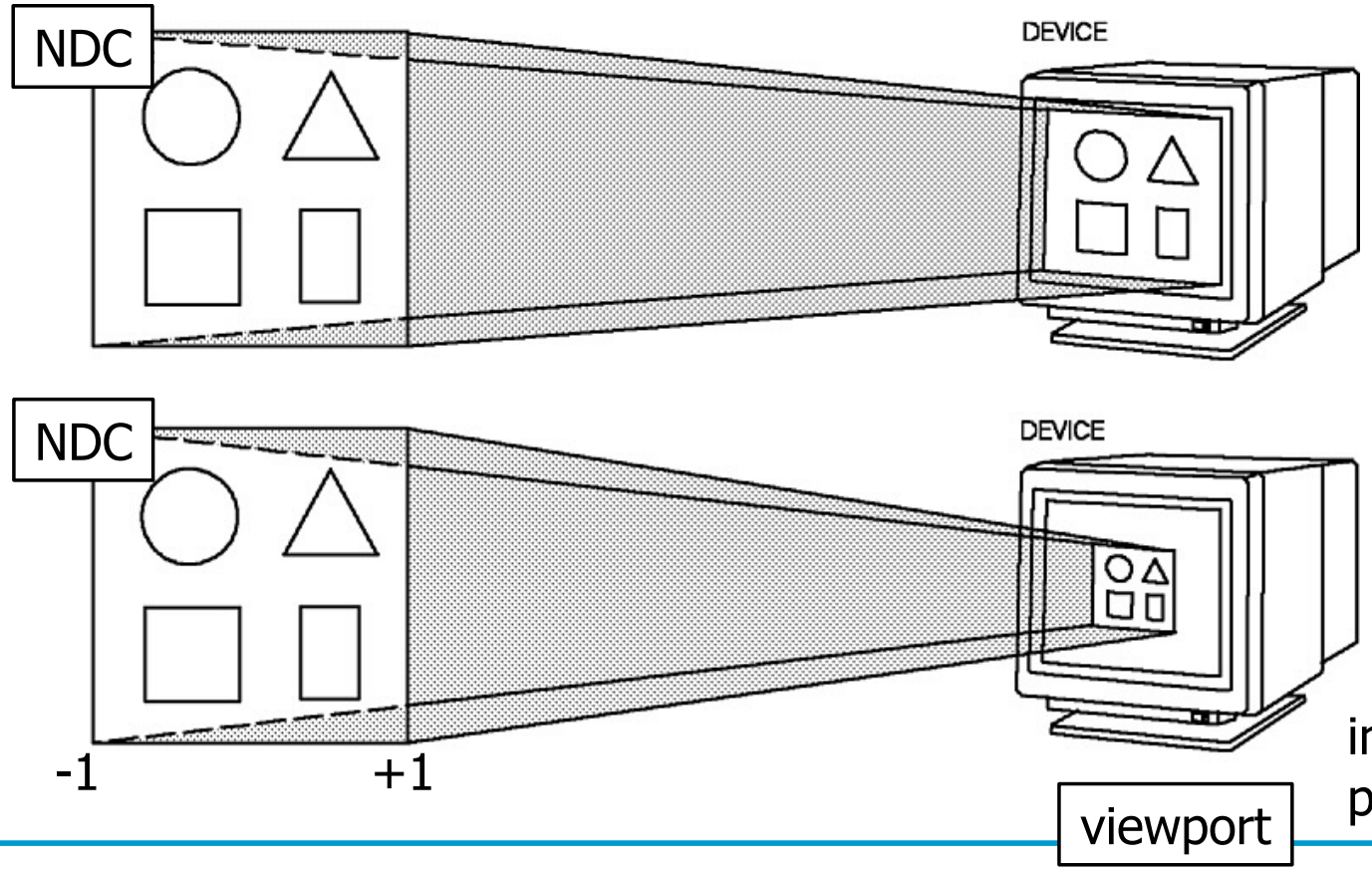
[Levoy]

Q. Why clip beforehand?

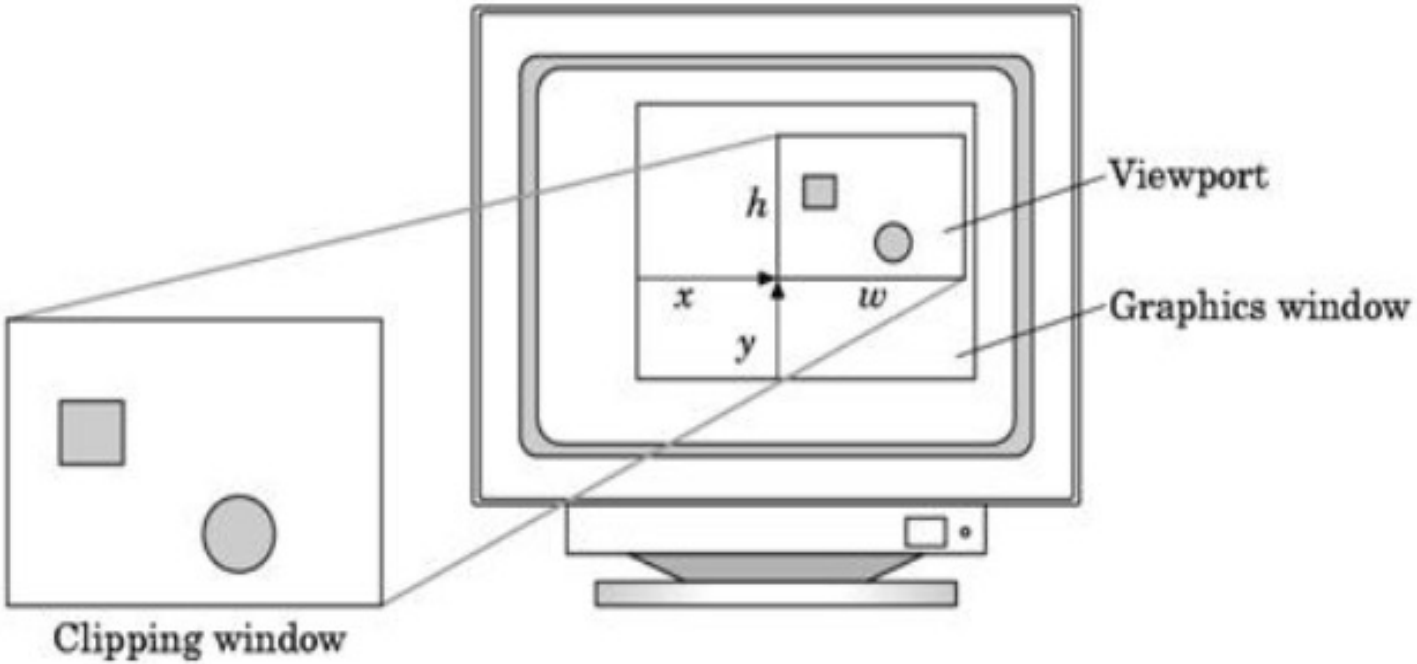
A. Avoids unnecessary divisions

Viewport Transformation

glViewport() command is used to define the rectangle of the rendering area where the final image is mapped



Viewport Transformation

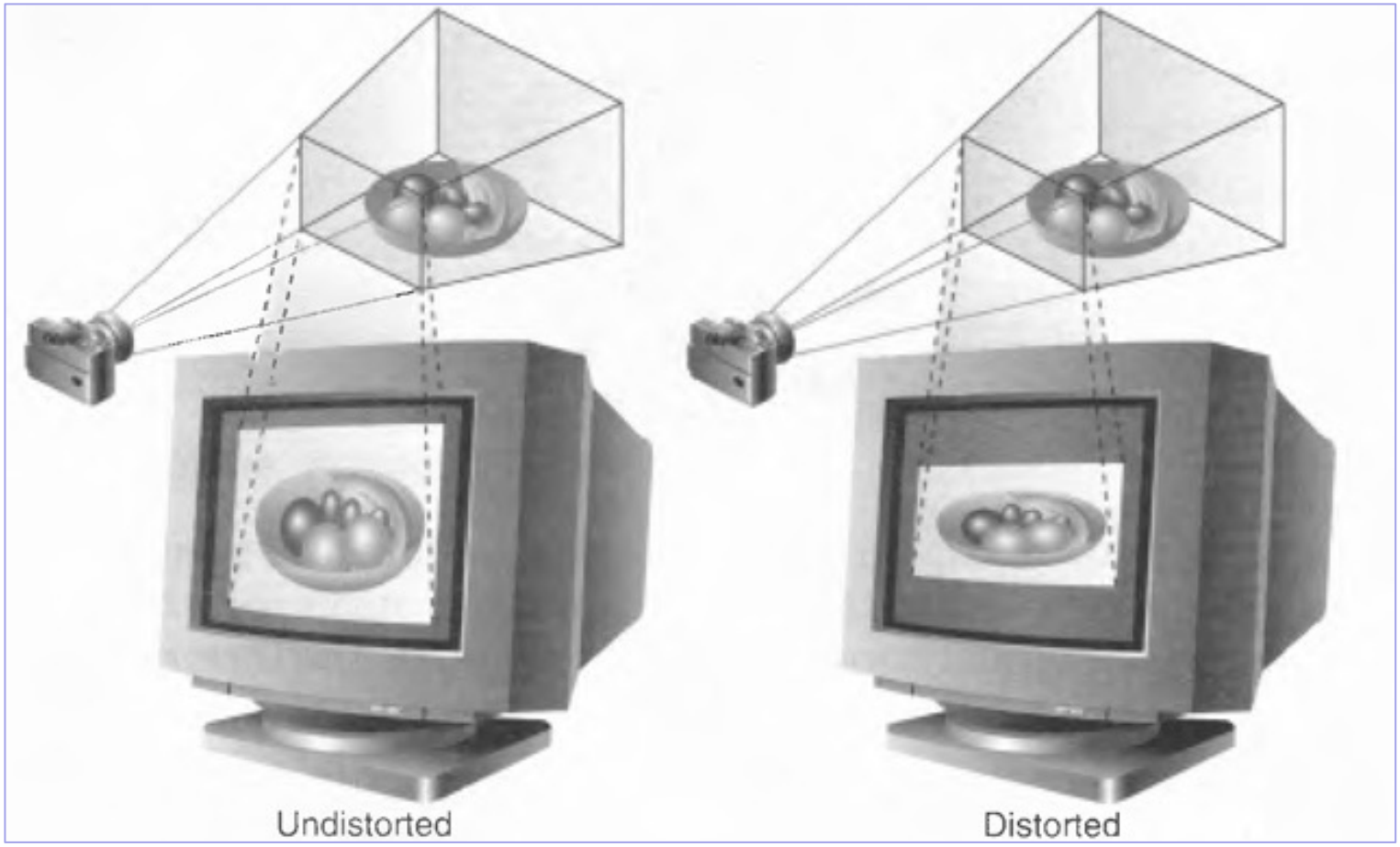


glViewport (x, y, width, height)

- x, y – specify the lower left corner of the viewport rectangle in pixels. The default is (0,0)
- width, height – specify the width and height of the viewport. When a scene is first attached to a window, width and height are set to the dimensions of that window.

Viewport Transformation

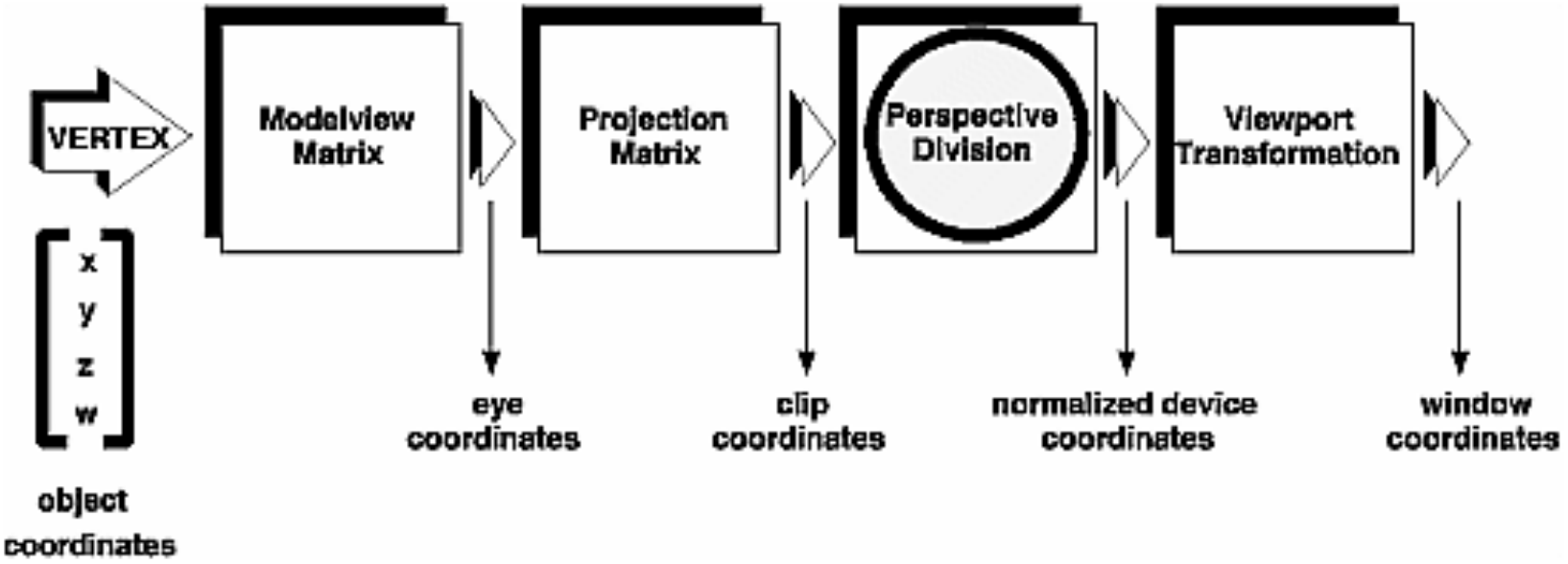
Pay attention to the aspect ratio!!! Why?



Conclusion

- Family of Projections (parallel and perspective, oblique and orthographic, etc.)
- Projection Models in OGL (Orthographic and Perspective)
- Projection Mechanism in OGL:
  1. Eye coordinate to Clip coordinate (Projection Matrix)
  2. Clipping against w_c (inside/outside view frustum)
  3. Perspective Division (homogenization):
     Clip coordinate to NDC
  4. Viewport transformation: to window coordinates

Conclusion

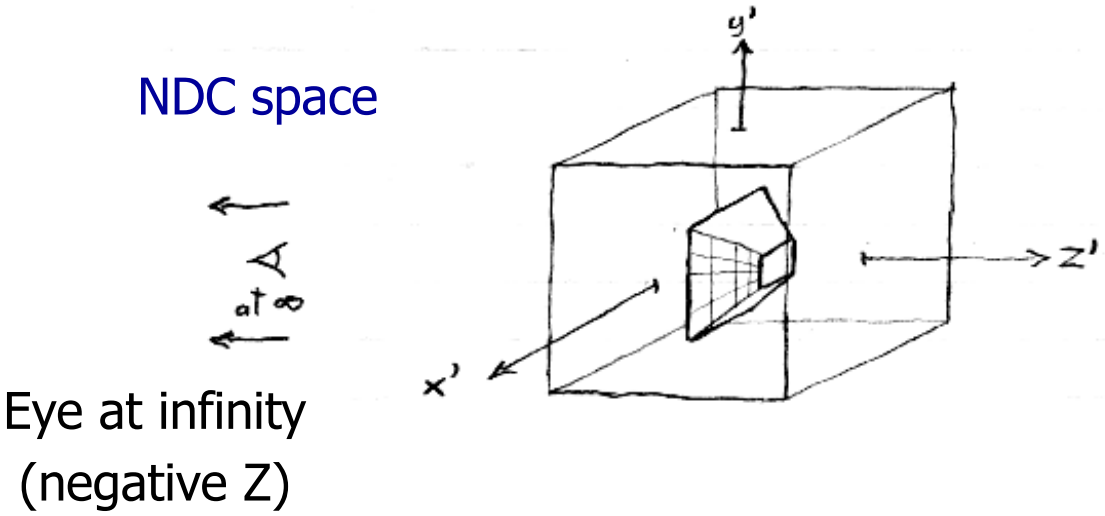

VERTEX
x
y
z
w
object coordinates
Modelview Matrix
eye coordinates
Projection Matrix
clip coordinates
Perspective Division
normalized device coordinates
Viewport Transformation
window coordinates

More about NDC

Note:

- xyz of NDC is [-1,+1] (inside view frustum)
- Z of NDC is important for hidden surface removal while X&Y of NDC (next: viewport transform) is for scan conversion

to be continue…… let's look at this in the next lecture……



More: Useful references (optional)

- About the internal matrix layout in OGL
  http://sjbaker.org/steve/omniv/matrices_can_be_your_friends.html
- More about viewing in OGL
  http://www.opengl.org/resources/faq/technical/viewing.htm#view0030
- A Good article: Projection abuse
  http://sjbaker.org/steve/omniv/projection_abuse.html