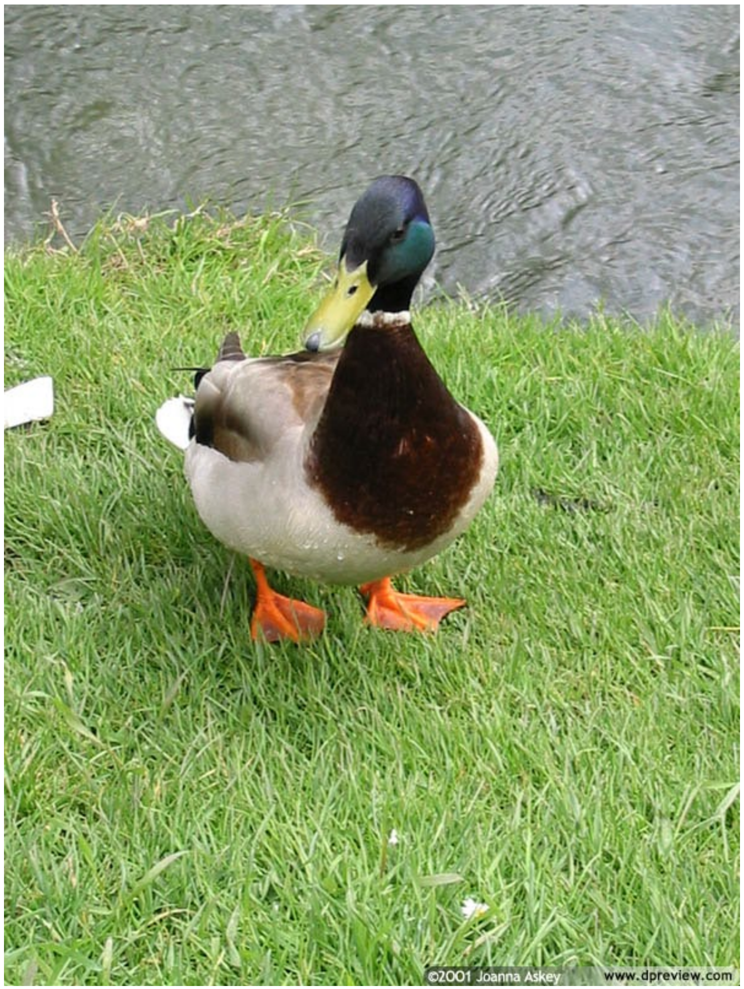
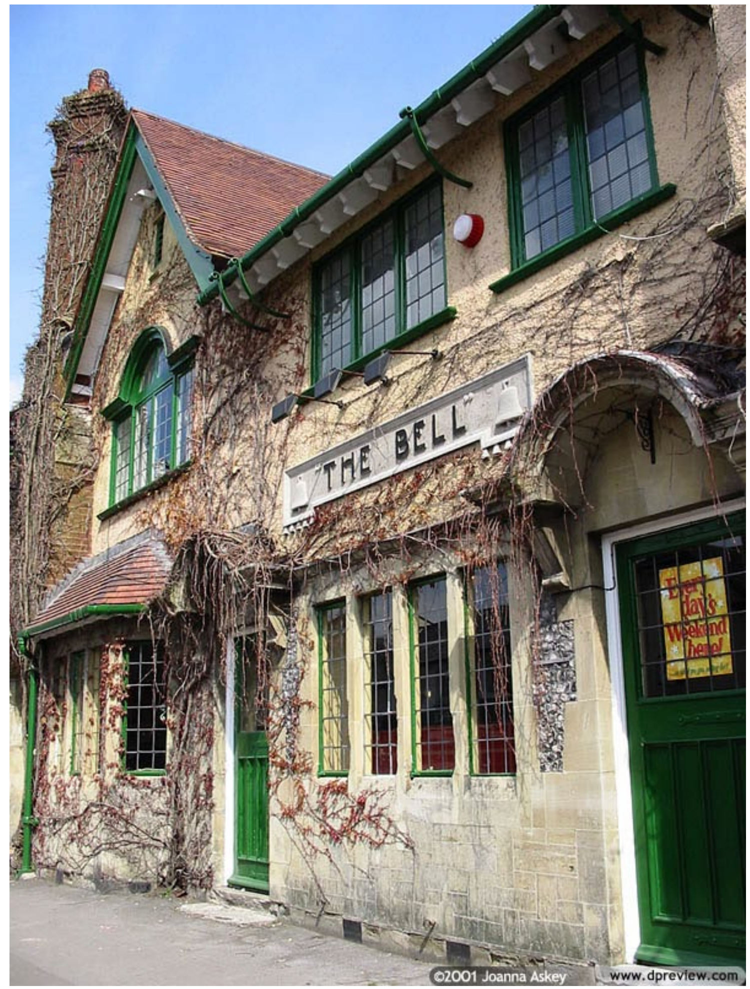
Camera Models

CS535

Daniel G. Aliaga
Department of Computer Science
Purdue University

Our (3D) World…

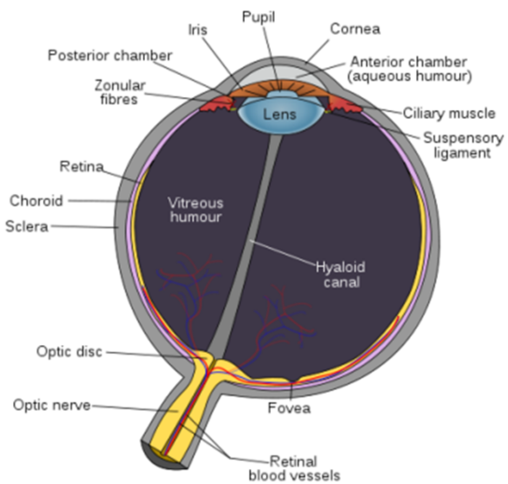

©2001 Joanna Askey
www.dpreview.com


THE BELL
©2001 Joanna Askey
www.dpreview.com

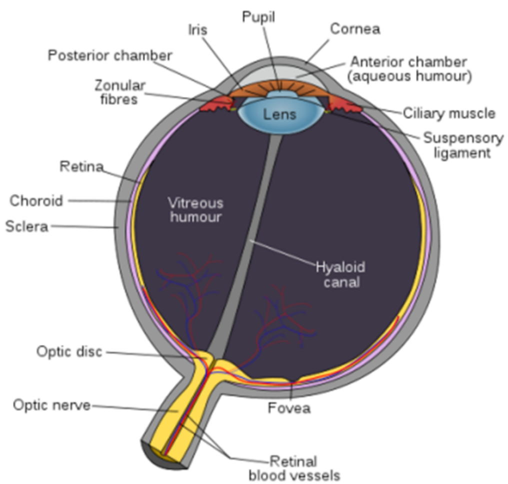
What makes it 3D?

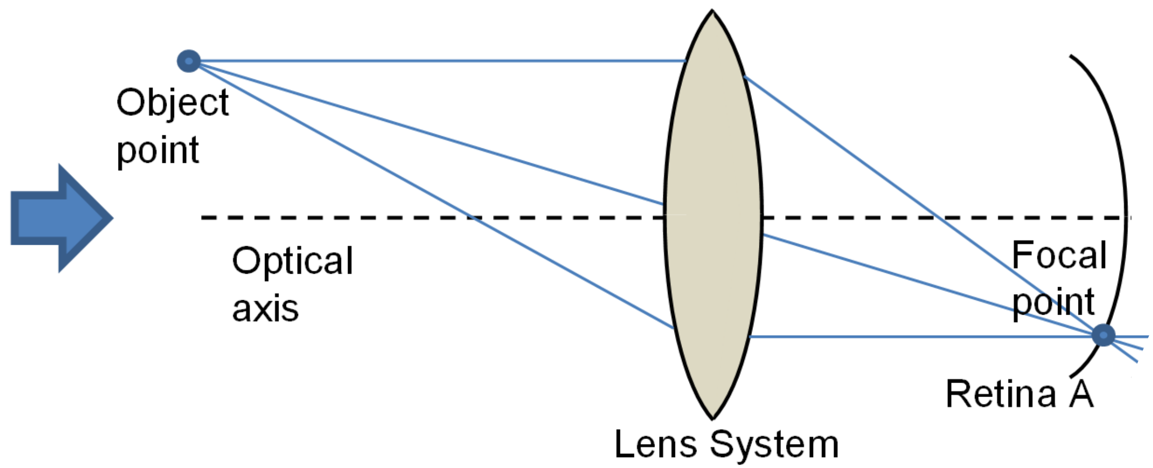
- Depth (…illusion)

Human Eye

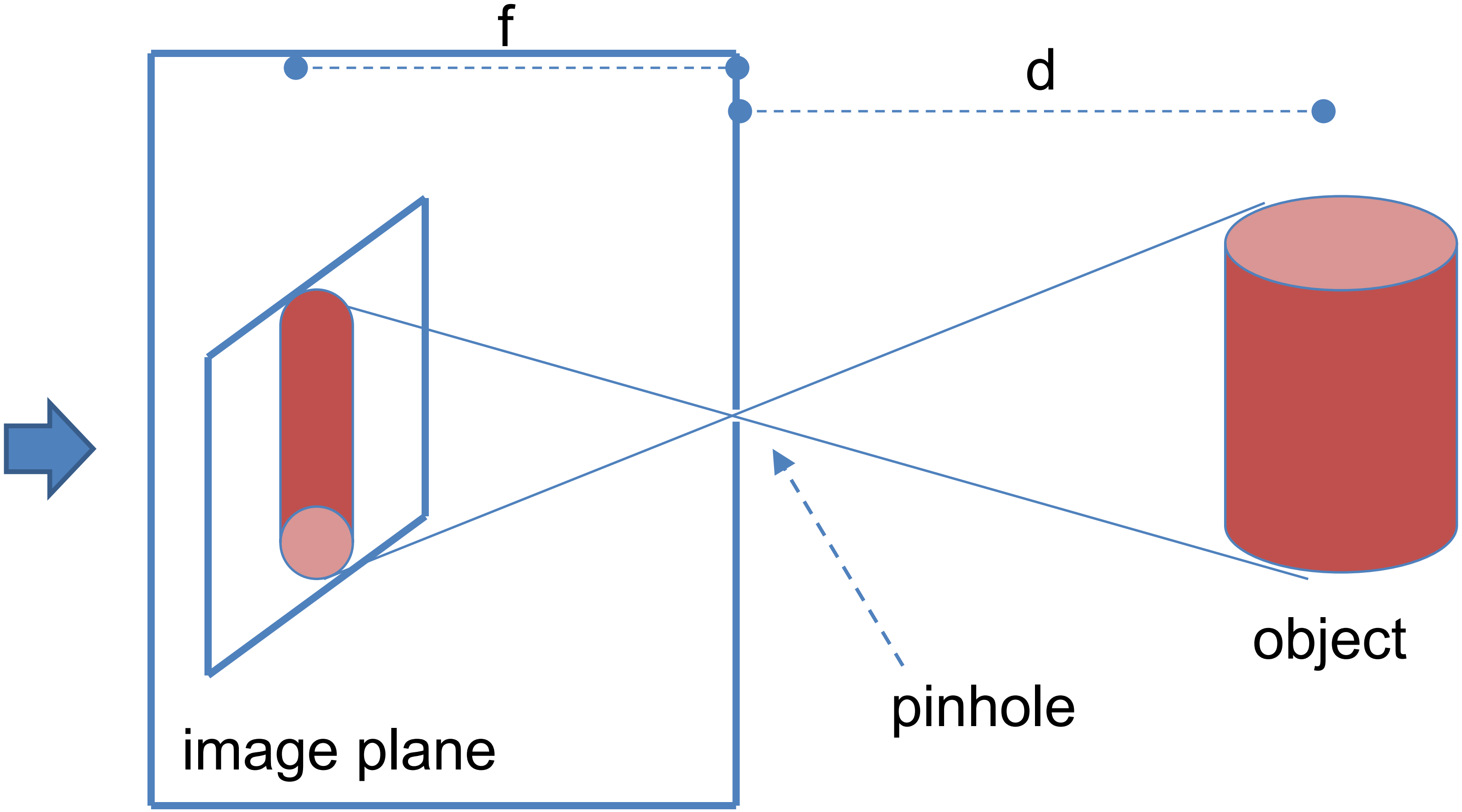

Pupil
Iris
Cornea
Posterior chamber
Anterior chamber
(aqueous humour)
Zonular
fibres
Lens
Ciliary muscle
Suspensory
ligament
Retina
Choroid
Sclera
Vitreous
humour
Hyaloid
canal
Optic disc
Optic nerve
Fovea
Retinal
blood vessels

Thin Lens System



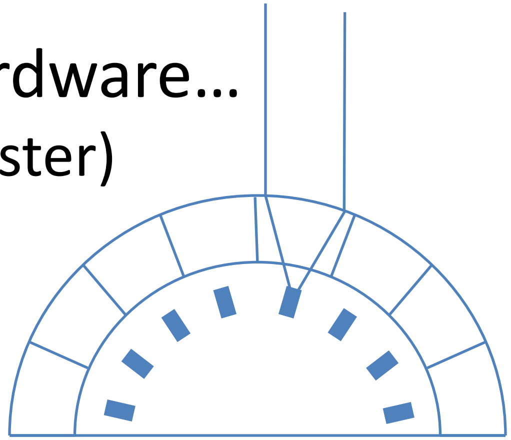


Pinhole Camera Model

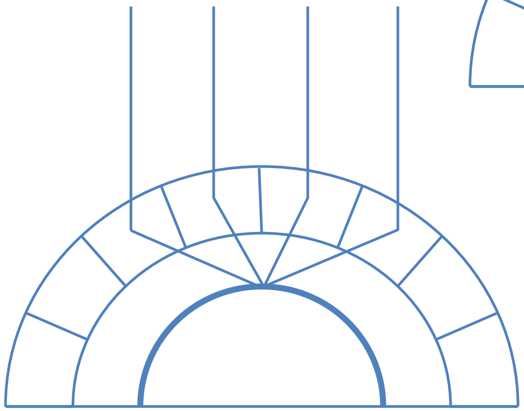


Biology 101

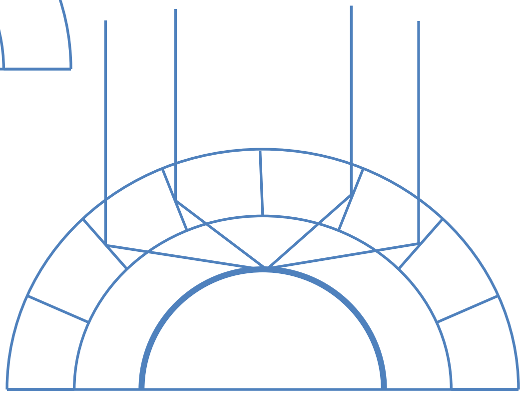
- Not all animals have the same vision hardware…
  - e.g., certain insects, crustaceans (e.g., lobster)
- Diurnal Insect Vision



- Nocturnal Insect Vision



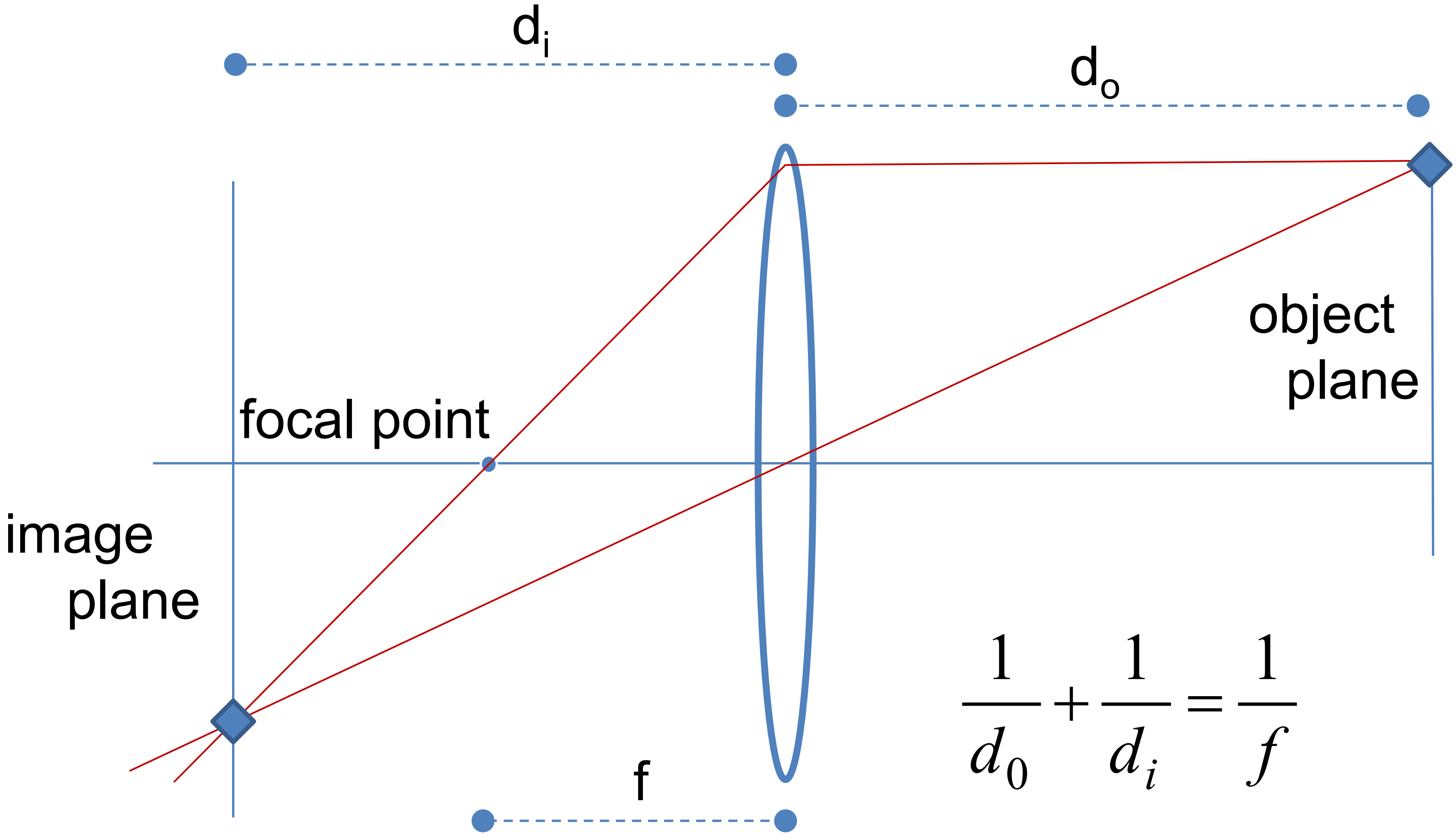
- Crustacean Vision



Optics: Terminology

- Dioptric
  - All elements are refractive (lenses)
- Catoptric
  - All elements are reflective (mirrors)
- Catadioptric
  - Elements are refractive and reflective (mirrors + lenses)

Thin Lens Equation



$$\frac{1}{d_0} + \frac{1}{d_i} = \frac{1}{f}$$

Thin Lens Equation

- What happens if image plane is not at the focal point (distance)?

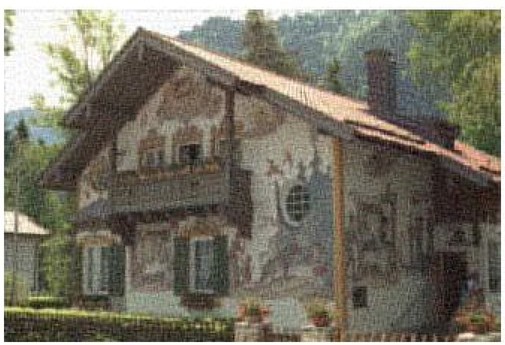
Digression: Deblurring

Digression: Deblurring

original image

observed image

proposed method
25.69dB, 1.65sec



deconvwnr
22.47dB, 0.12sec

deconvlucy
23.91dB, 6.38sec

deconvreg
24.11dB, 0.5sec

[Chan et al., IEEE IP 2011]

Digression: Synthetic Focusing

- Ray tracing:

- Lightfield:
  - http://lightfield.stanford.edu/aperture.swf?lightfield=data/chess_lf/preview.zip&zoom=1

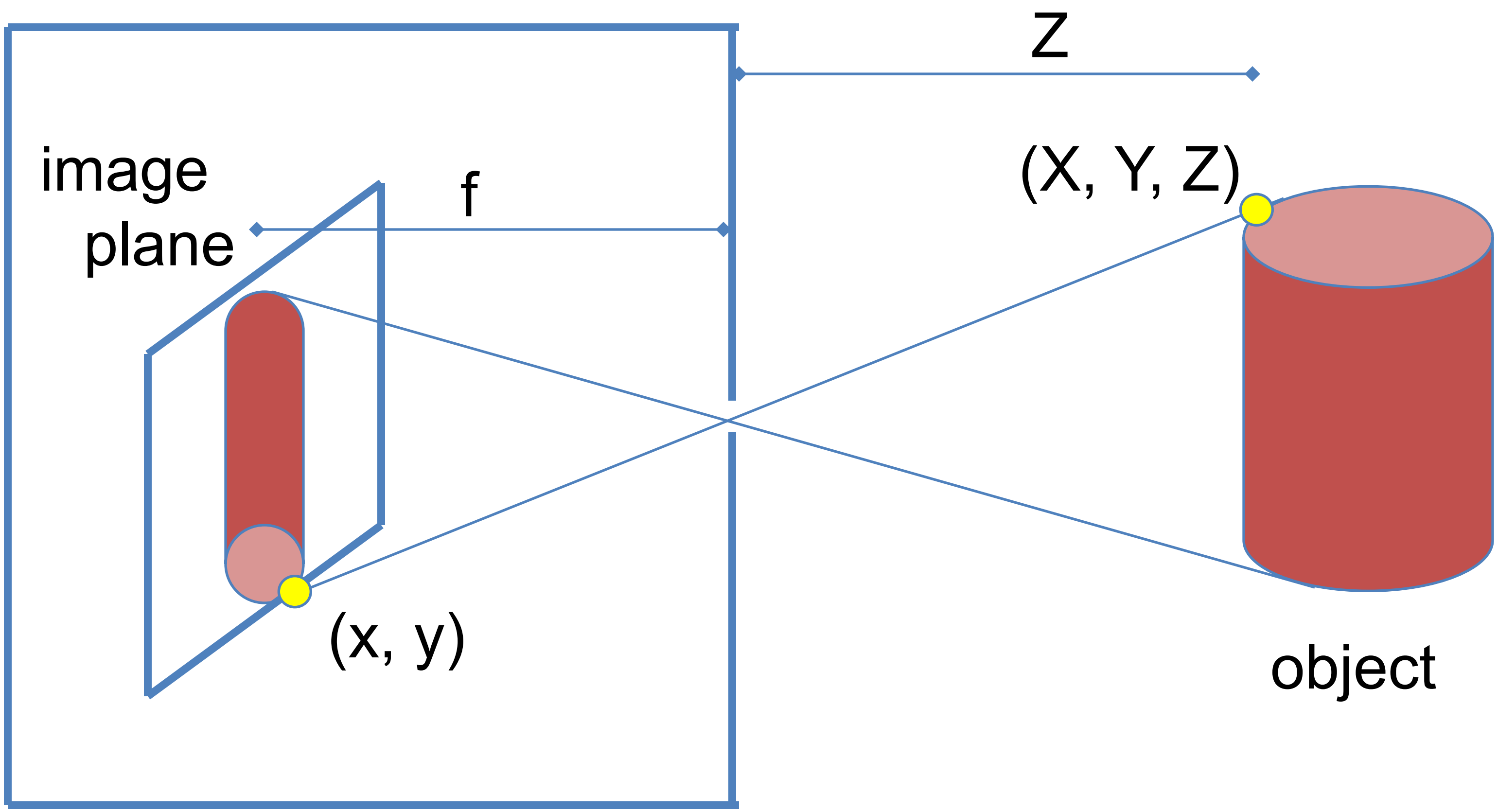
- Lytro Camera:
  - http://lightfield.stanford.edu/aperture.swf?lightfield=data/chess_lf/preview.zip&zoom=1

Digression: Non-Pinhole Camera Models…

- Why restrict the camera to a pinhole camera model?
  – Aperture is large: lightfields/lumigraphs
  – Multi-perspective imaging
  – Sample-based camera
  – Tailored camera designs
    - Occlusion-resistant cameras (kinda…)
    - Graph cameras
    - and more!

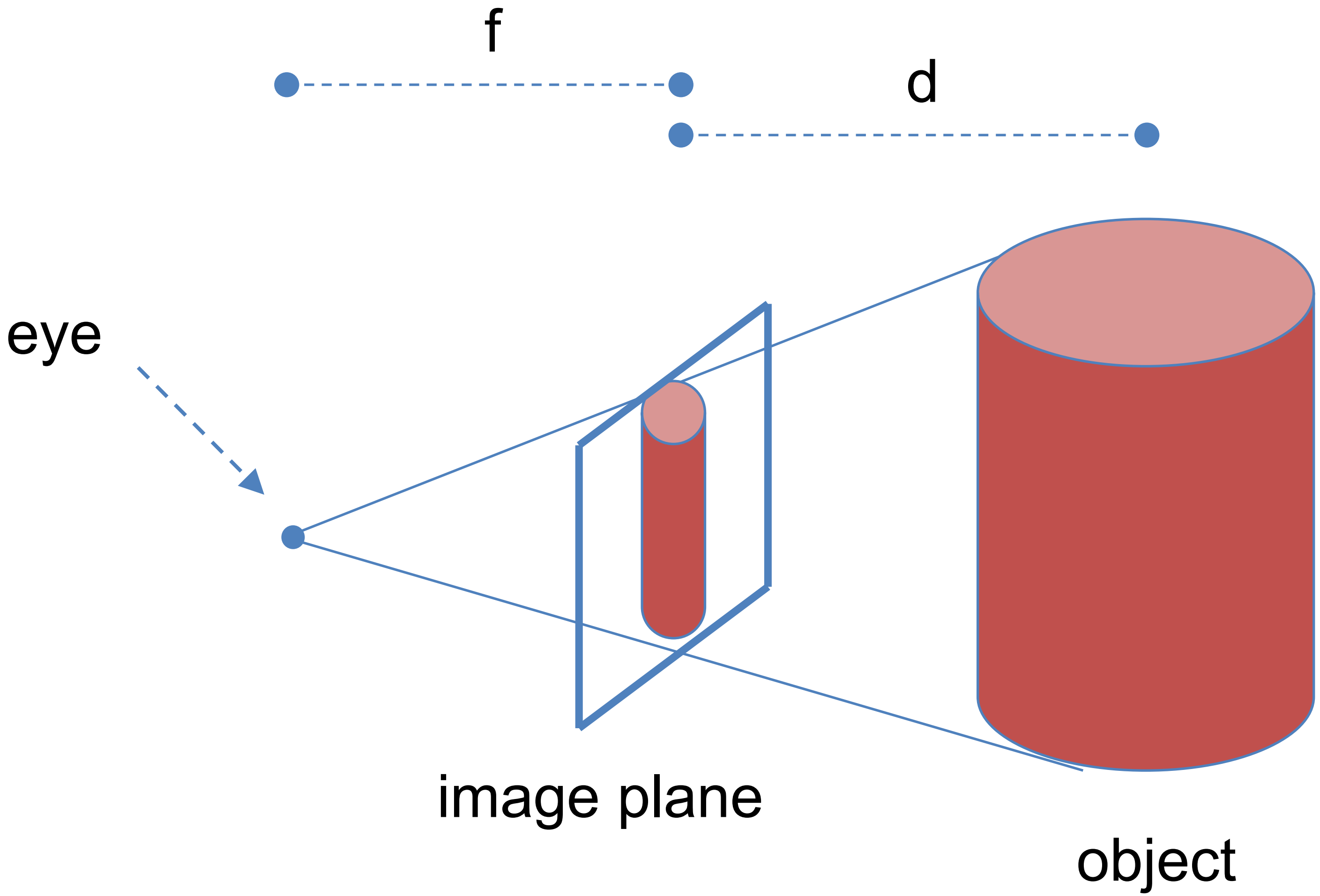
[**switch to non-standard**…]

Pinhole Camera Model

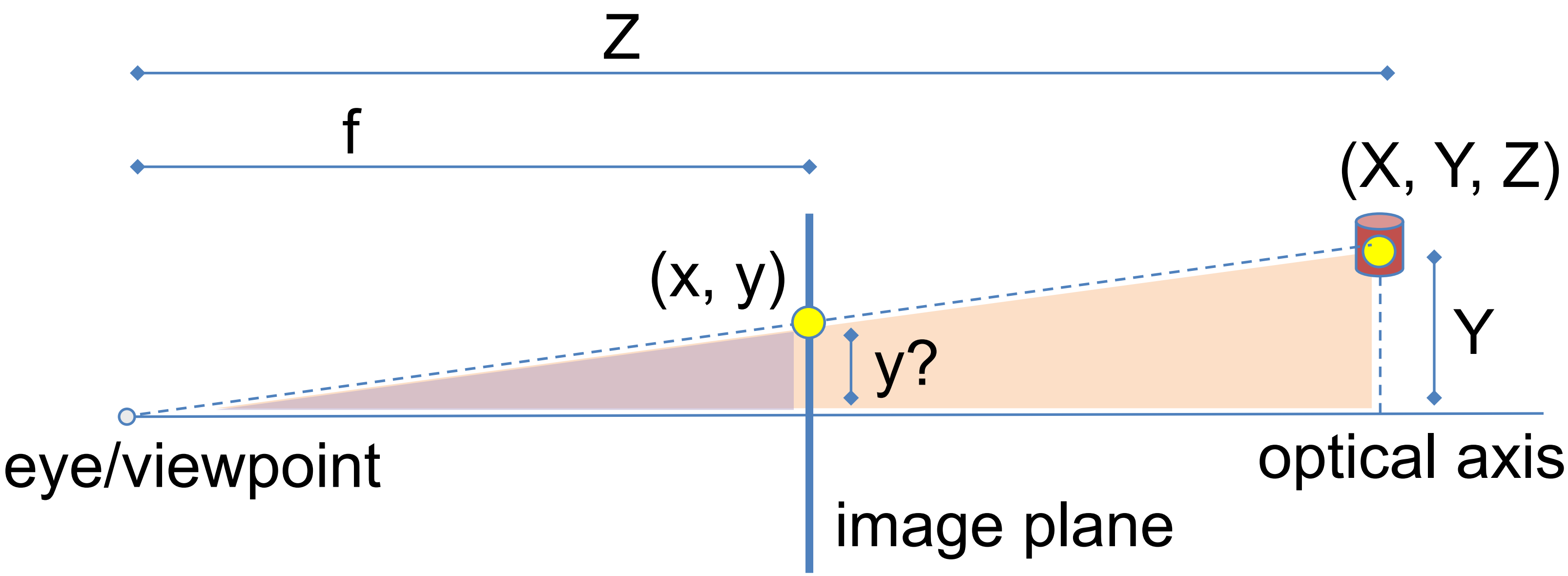


$$x = f\frac{X}{Z} \qquad y = f\frac{Y}{Z}$$

"Computer Graphics" Pinhole Camera Model

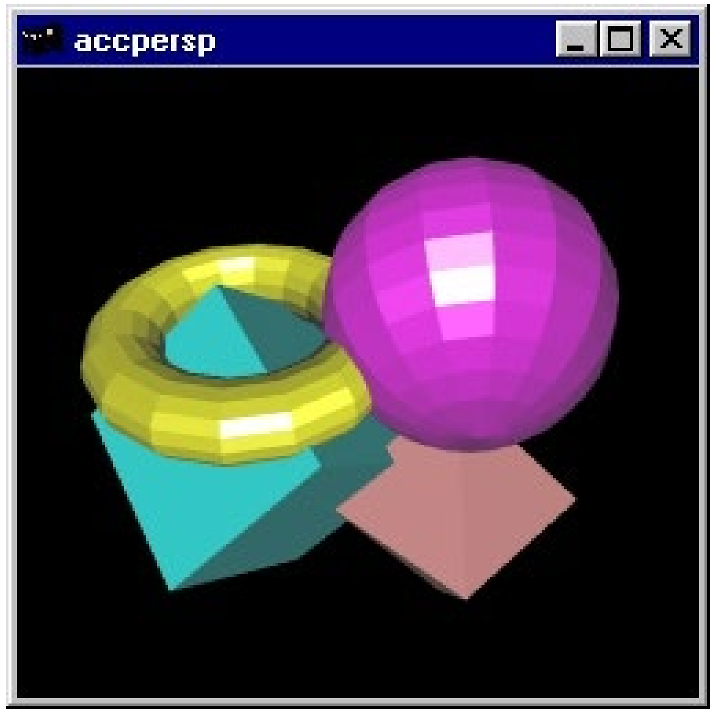


3D Perspective Projection



$$\frac{y}{f} = \frac{Y}{Z} \Rightarrow y = f\frac{Y}{Z} \quad \& \quad x = f\frac{X}{Z}$$

Example OpenGL Rendering



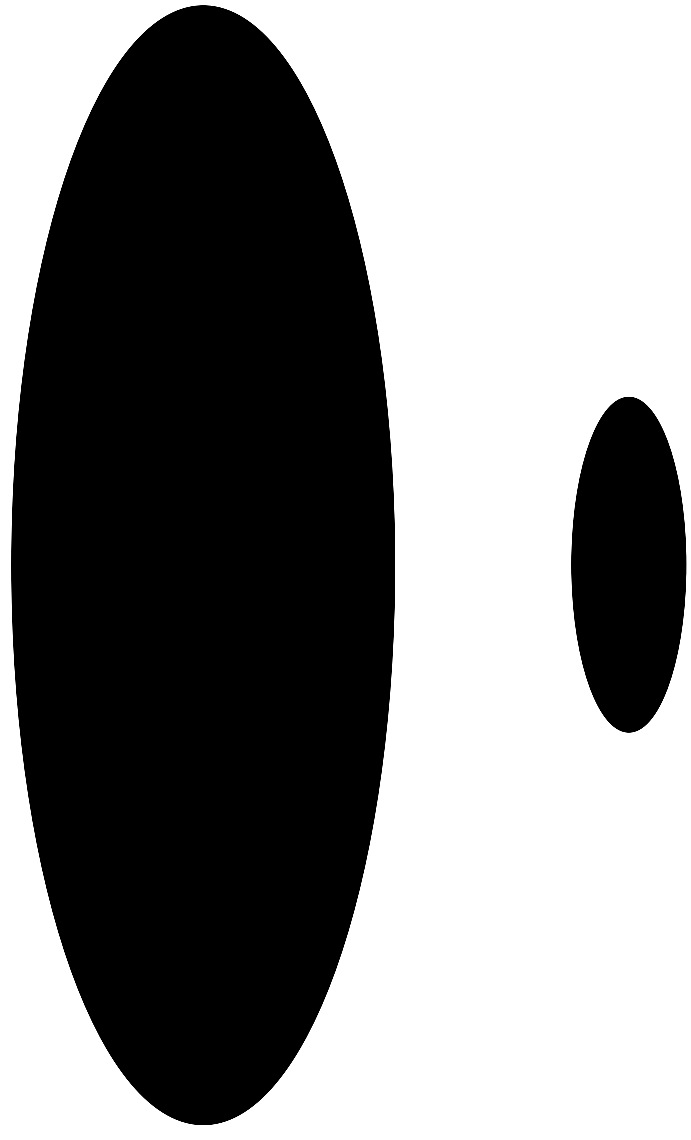
- Perspective is “mathematically” accurate…

(3D) Perception

- There is lot more to it…

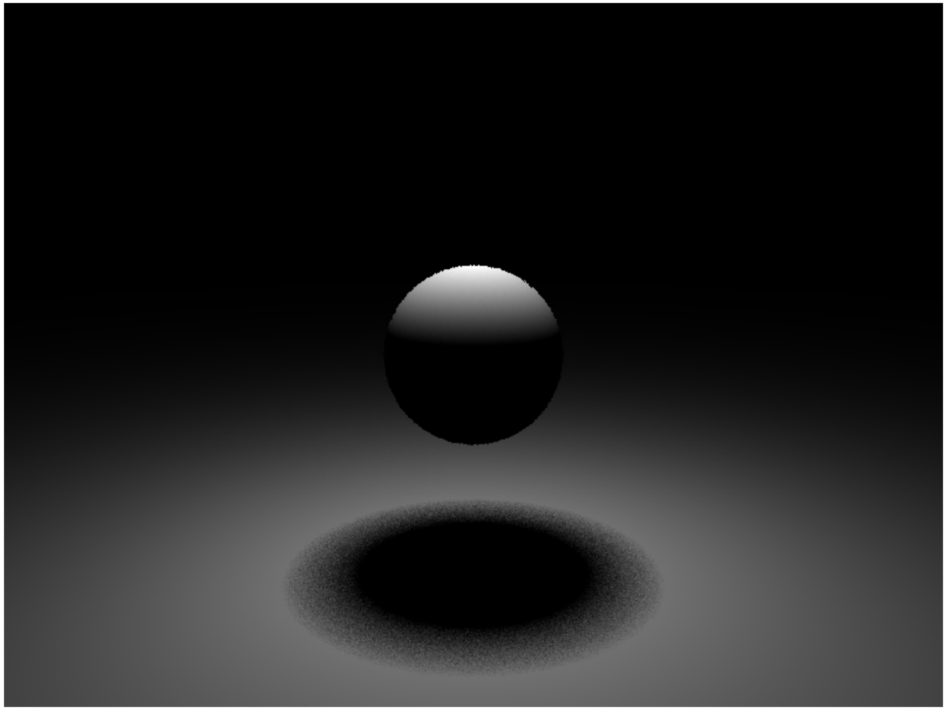
Perception

- Size-distance relationship
  - Same size object at two distances…



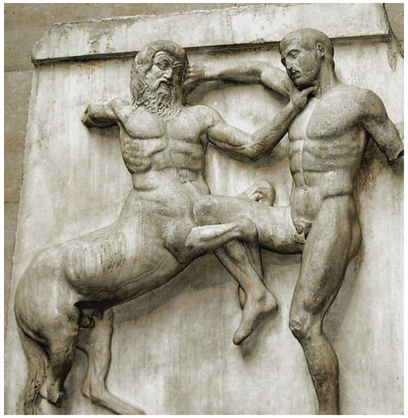
Perception

- Importance of ground plane, shadows…



Perception

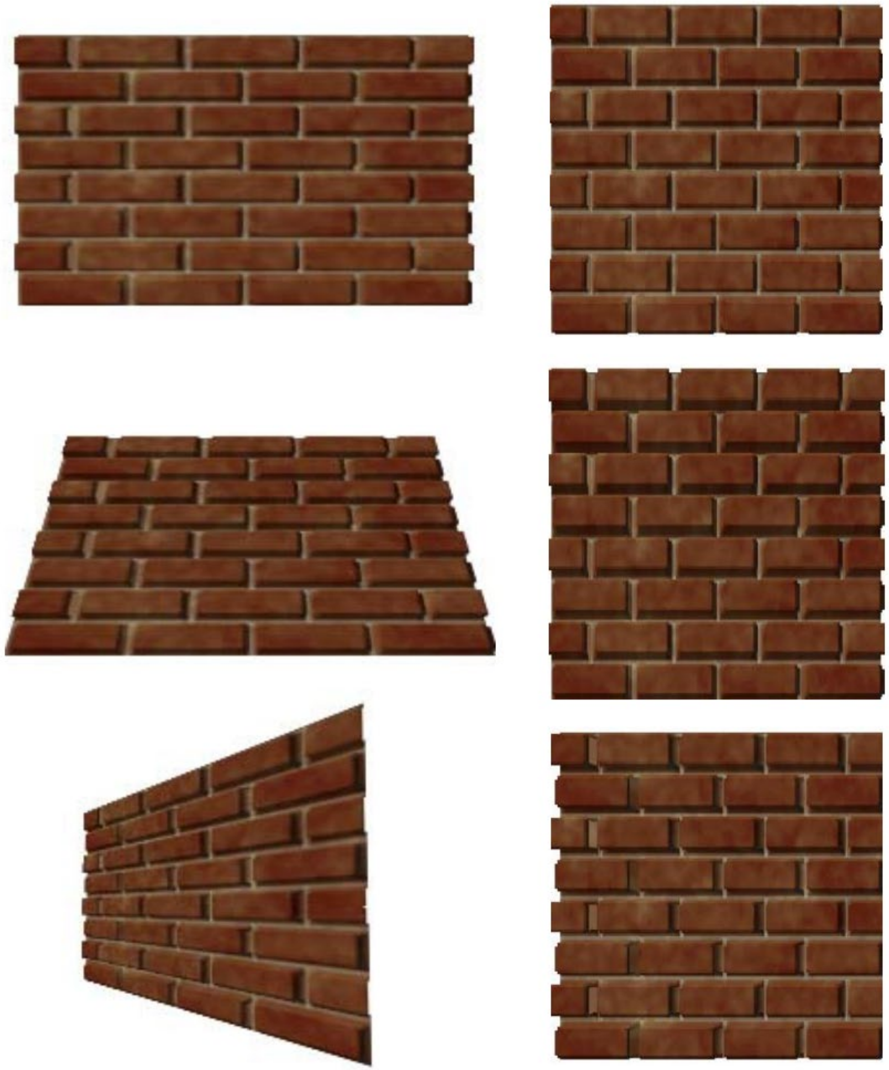
- Alto Relief



- Sunken Relief

(Relief Images)

- We will see this type of things later…



Perception: Bas-Relief

Perception: Bas-Relief

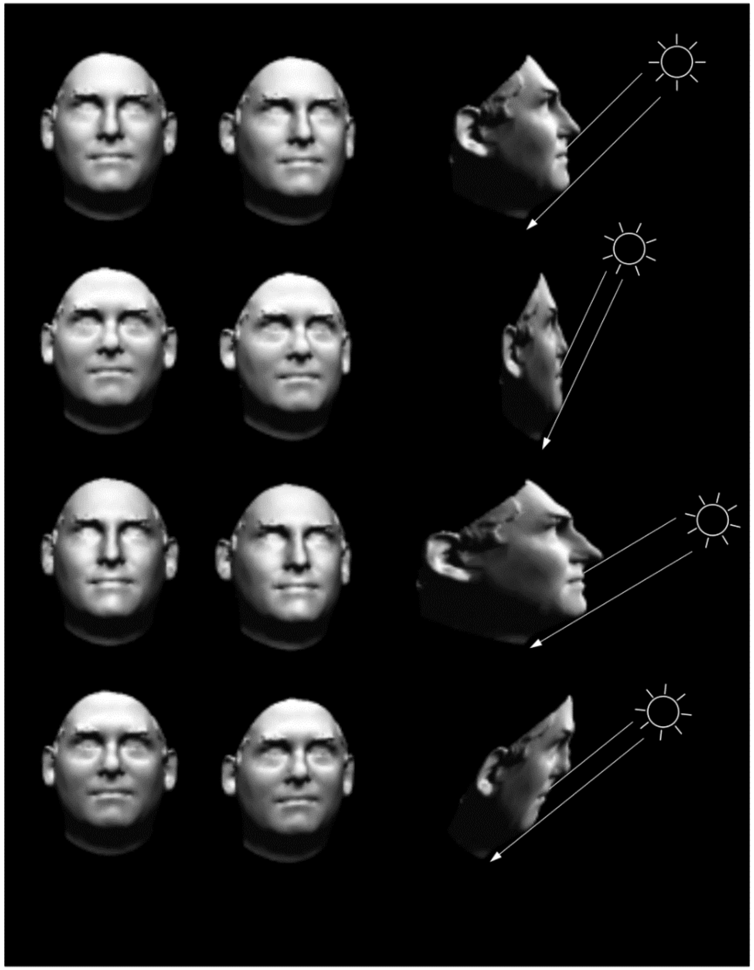
Perception: Bas Relief Ambiguity

$$NL^T = C$$

or

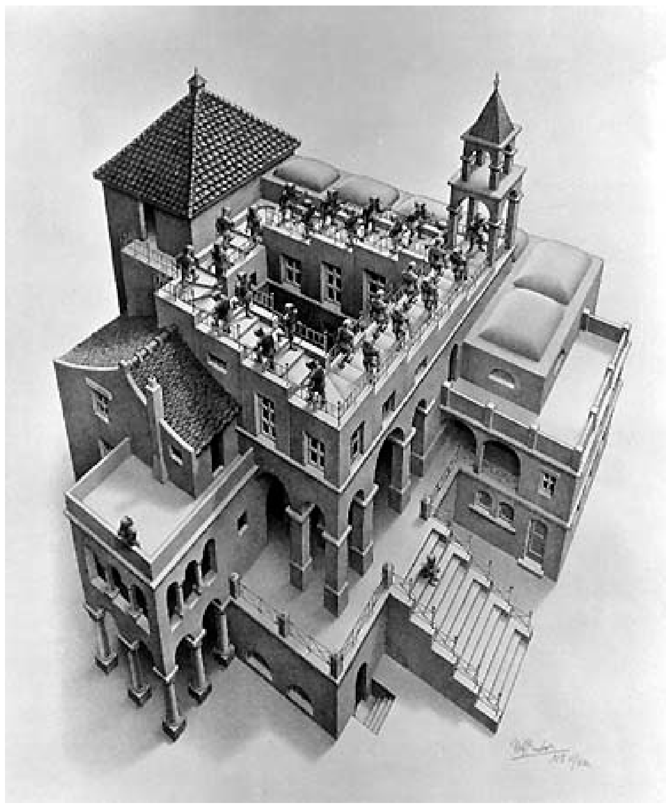
$$NRR^TL^T = C$$
$$NGG^{-1}L^T = C$$

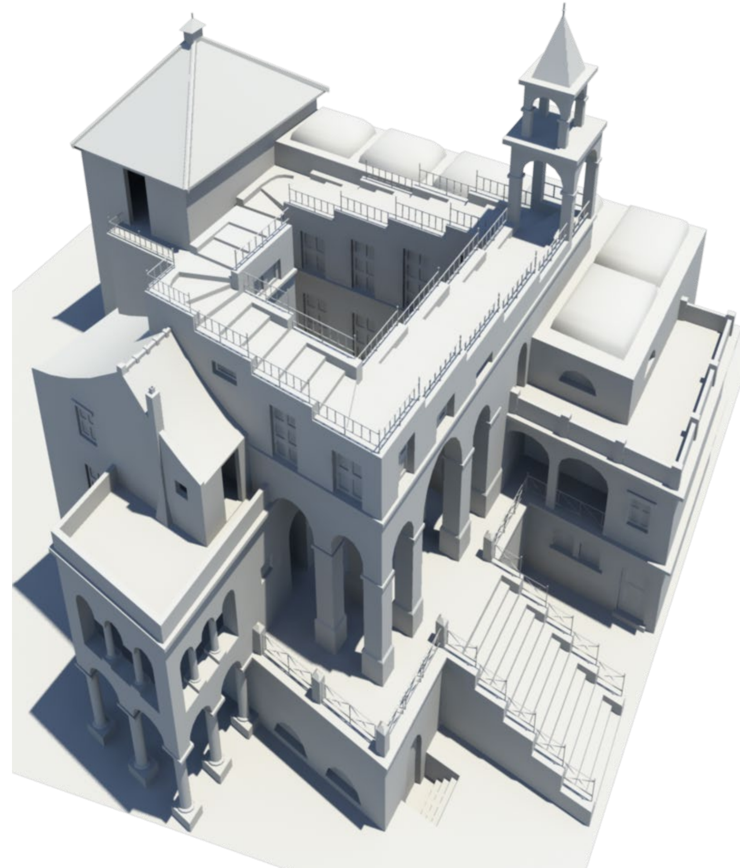
??



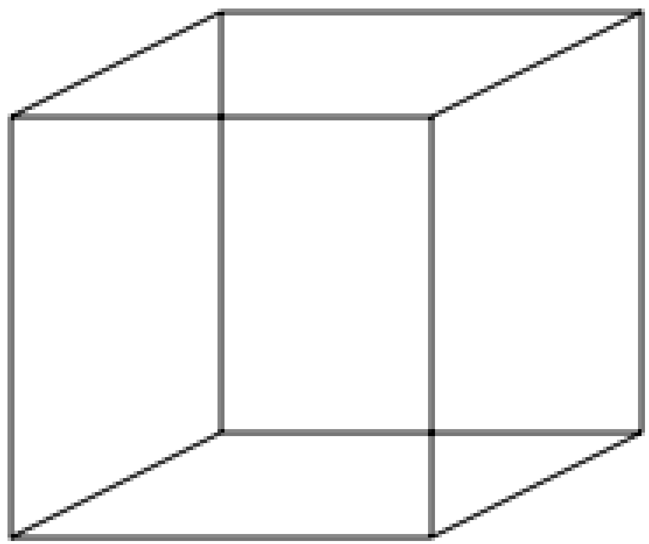
Perception: Inconsistencies

- Escher

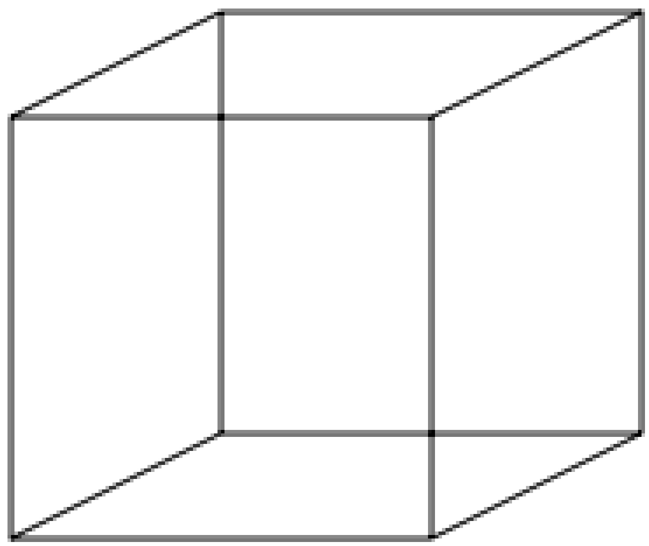


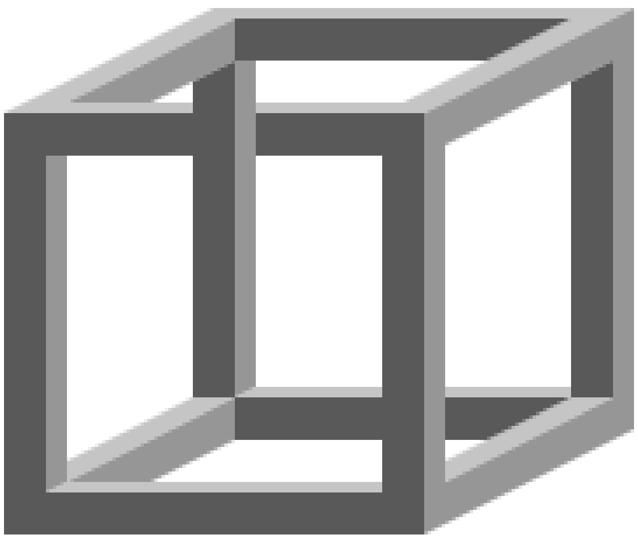


Perception: Nekker Cube

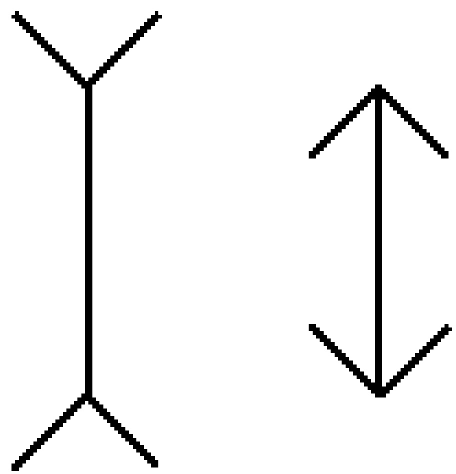


Perception: Nekker Cube

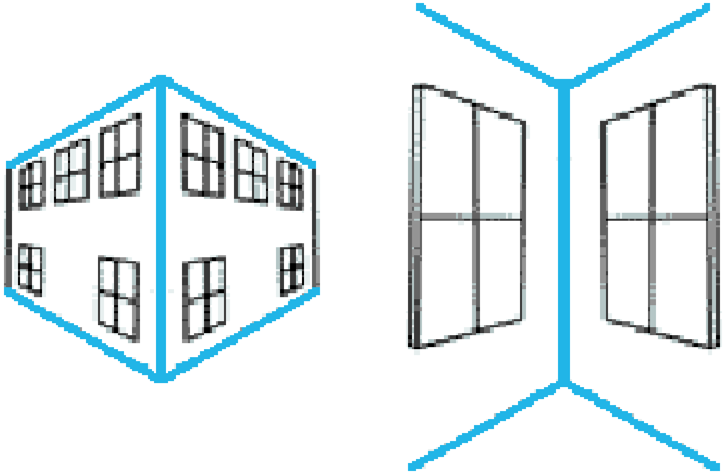




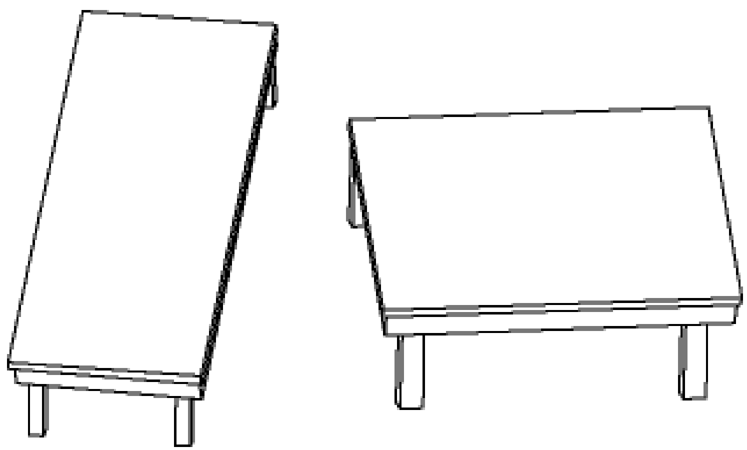
Perception: Muller-Lyer Illusion



Perception: Muller-Lyer Illusion



Perception: Muller-Lyer Illusion



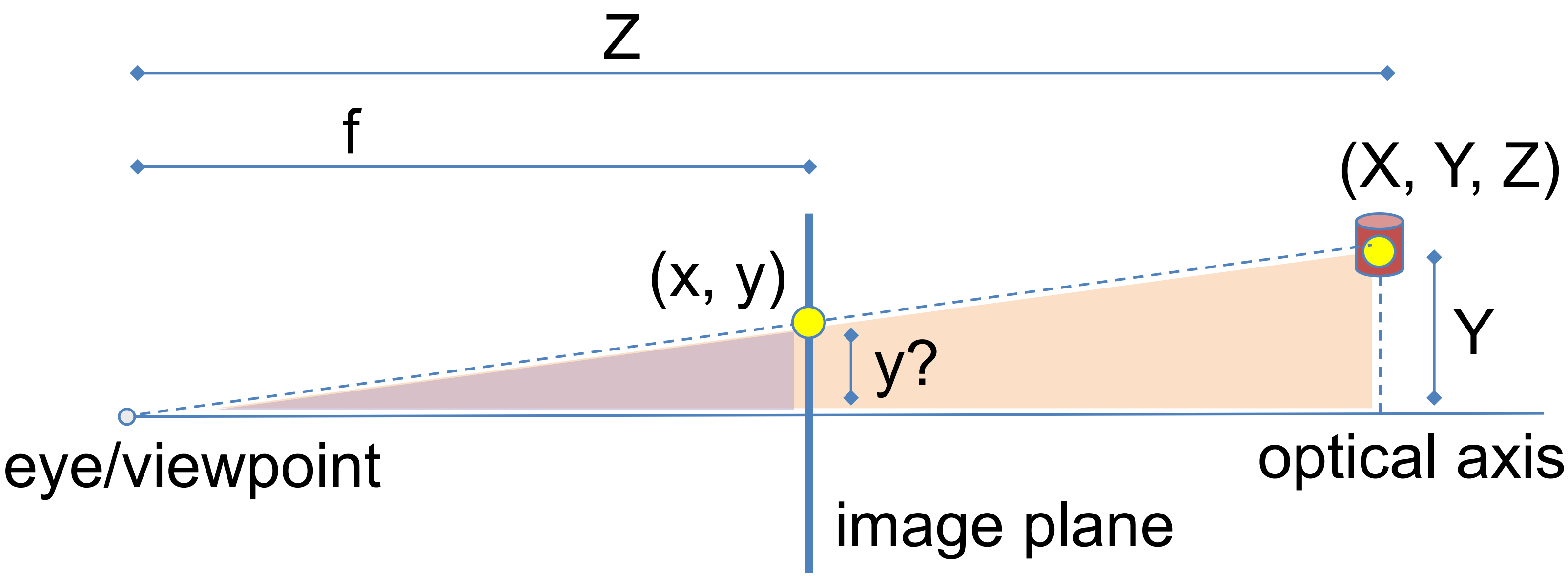
Perspective Camera Parameters

- Intrinsic/Internal
  - Focal length f
  - Principal point (center) p_x, p_y
  - Pixel size s_x, s_y
  - (Distortion coefficients) $k_1, \ldots$
- Extrinsic/External
  - Rotation ϕ, φ, ψ
  - Translation t_x, t_y, t_z

Perspective Camera Parameters

- Intrinsic/Internal
  - Focal length f
  - Principal point (center) (=middle of image)
  - Pixel size (=1, irrelevant)
  - (Distortion coefficients) (=0, assuming no bugs ☺)
- Extrinsic/External
  - Rotation ϕ, φ, ψ
  - Translation t_x, t_y, t_z

Recall: 3D Perspective Projection

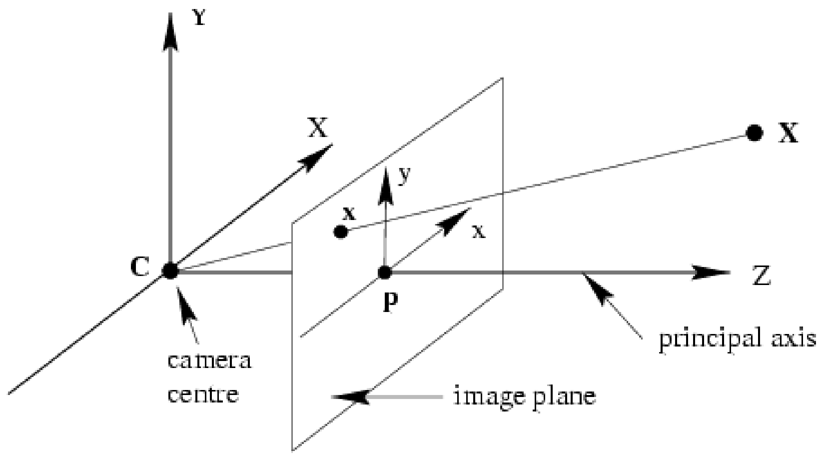


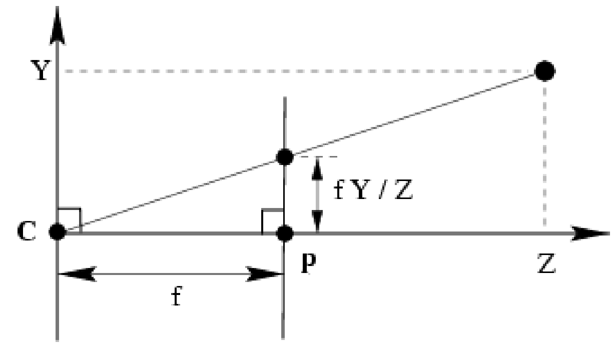
$$\frac{y}{f} = \frac{Y}{Z} \Rightarrow y = f\frac{Y}{Z} \quad \& \quad x = f\frac{X}{Z}$$

Matrix Encoding

- How do we put the perspective divide process into the matrix pipeline?

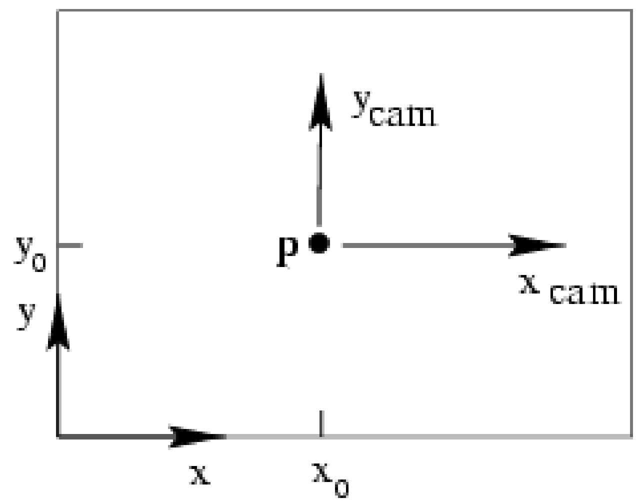
Focal Length





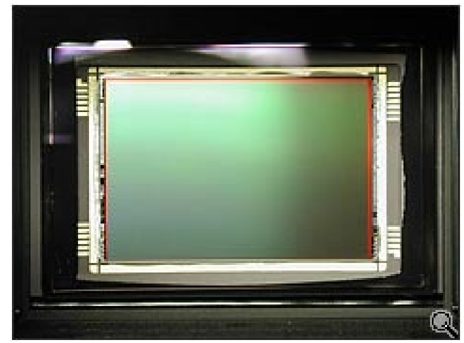
$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} fX/Z \\ fY/Z \end{pmatrix} \Leftarrow \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} \quad \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Principal Point



$$\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \Leftarrow \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} & & & \\ & & & \\ & & & \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

CCD Camera: Pixel Size



$$P = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix} \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

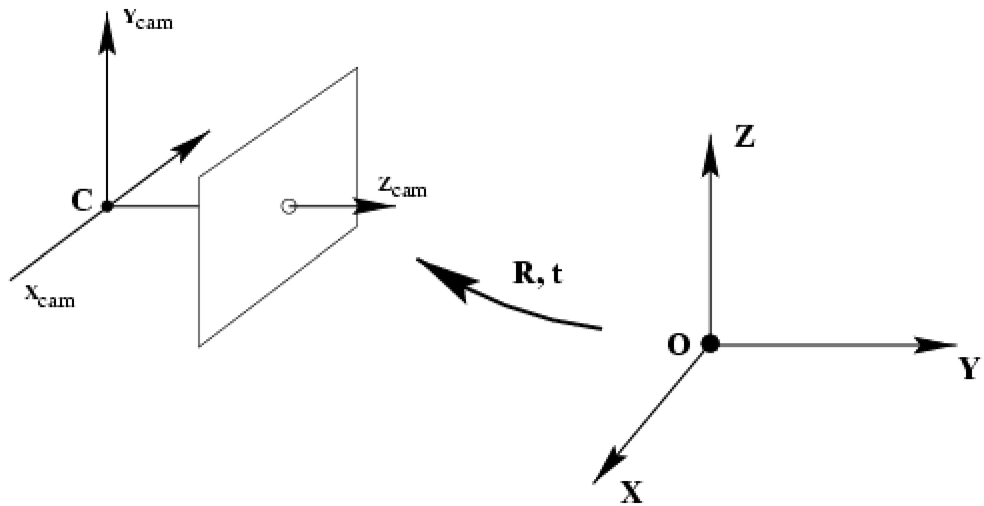
In graphics, often

$$s_x = s_y = 1$$
$$p_x = p_y = 0$$

$$P = \begin{bmatrix} \alpha_x & 0 & p_x & 0 \\ 0 & \alpha_y & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Projection matrix

Translation & Rotation



$$\tilde{x}_c = R(\tilde{X} - C)$$

$$\tilde{x}_c = R\tilde{X} - RC$$

$$-RC \Rightarrow -t$$

$$\tilde{x}_c = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

World-to-camera matrix M

$$R = R_x R_y R_z$$

3x3 rotation matrices

$$t = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T$$

translation vector

Perspective Projection Process

- Given $\tilde{X} = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$

...the perspective projection is

$$\tilde{x}_p = PM \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \Rightarrow \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \tilde{x}_{p_x} / \tilde{x}_{p_z} \\ \tilde{x}_{p_y} / \tilde{x}_{p_z} \end{bmatrix}$$

Projection Matrices

- Basic Perspective Projection:

$$\begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Basic Orthographic:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4x4 Matrices

- How do we encode into 4x4 matrix/vector multiplication pipeline?
- See black board…

Projection Matrices

- Passthrough Z Perspective Projection:

$$\begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -nf \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

If $(Z = n), then\ z = \left(n + f - \frac{nf}{n}\right) = n$

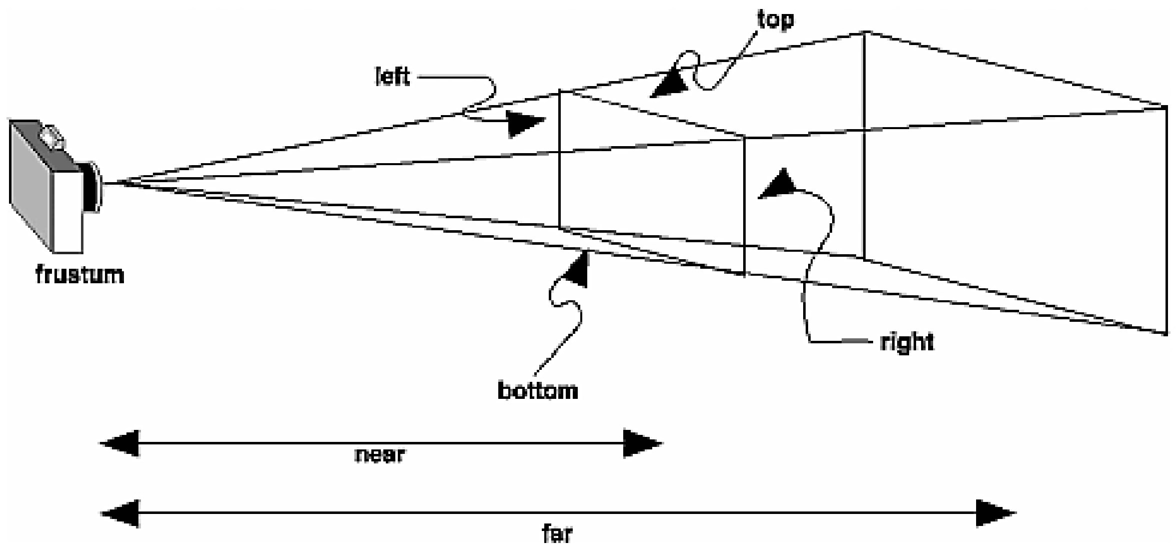
If $(Z = f), then\ z = \left(n + f - \frac{nf}{f}\right) = f$

Projection Matrices

- To map [-1,+1] to viewing frustum (l,r,b,t,n,f):

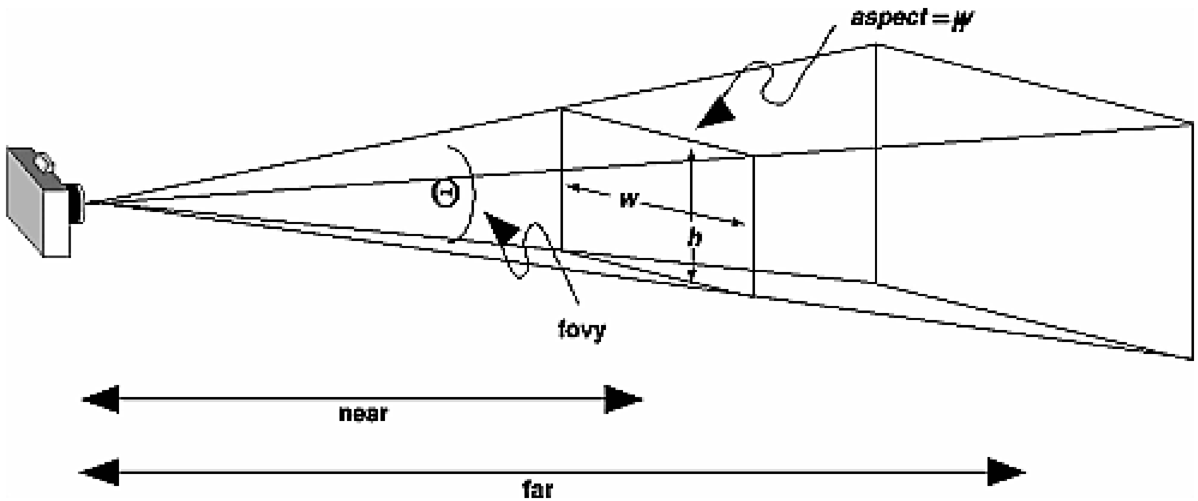
$$\begin{bmatrix} 2n/(r-l) & 0 & 0 & -(r+l)/(r-l) \\ 0 & 2n/(t-b) & 0 & -(t+b)/(t-b) \\ 0 & 0 & 2n/(n-f) & -(n+f)/(n+f) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Projection Transformations



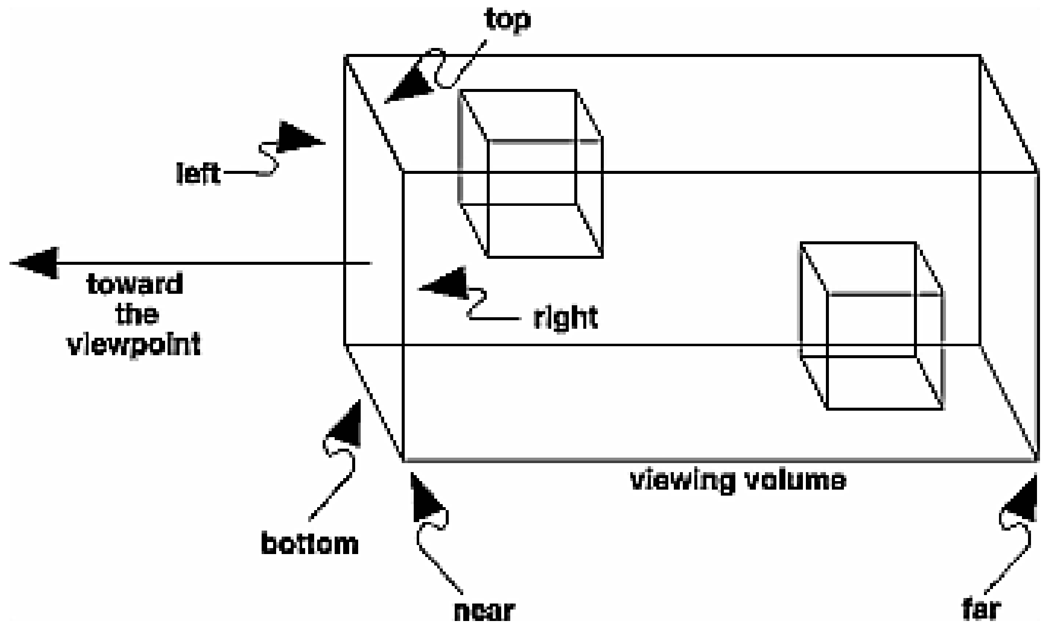

```
void glFrustum(GLdouble left, GLdouble right, GLdouble
  bottom, GLdouble top, GLdouble near, GLdouble far);
```

Projection Transformations




```
void gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble
                    near, GLdouble far);
```

Projection Transformations




```
void glOrtho(GLdouble left, GLdouble right, GLdouble
                      bottom,
    GLdouble top, GLdouble near, GLdouble far);

  void gluOrtho2D(GLdouble left, GLdouble right,
        GLdouble bottom, GLdouble top);
```

OpenGL Equivalent

```
…
glMatrixMode(GL_PROJECTION);
…
gluPerspective(60, 1.0, 0.1, 1000.0);
…
glMatrixMode(GL_MODELVIEW);
…
glTranslatef(tx,ty,tz);
glRotatef(rx,1,0,0);
glRotatef(ry,0,1,0);
glRotatef(rz,0,0,1);

/* or glLoadMatrixf(mat); */
…
```

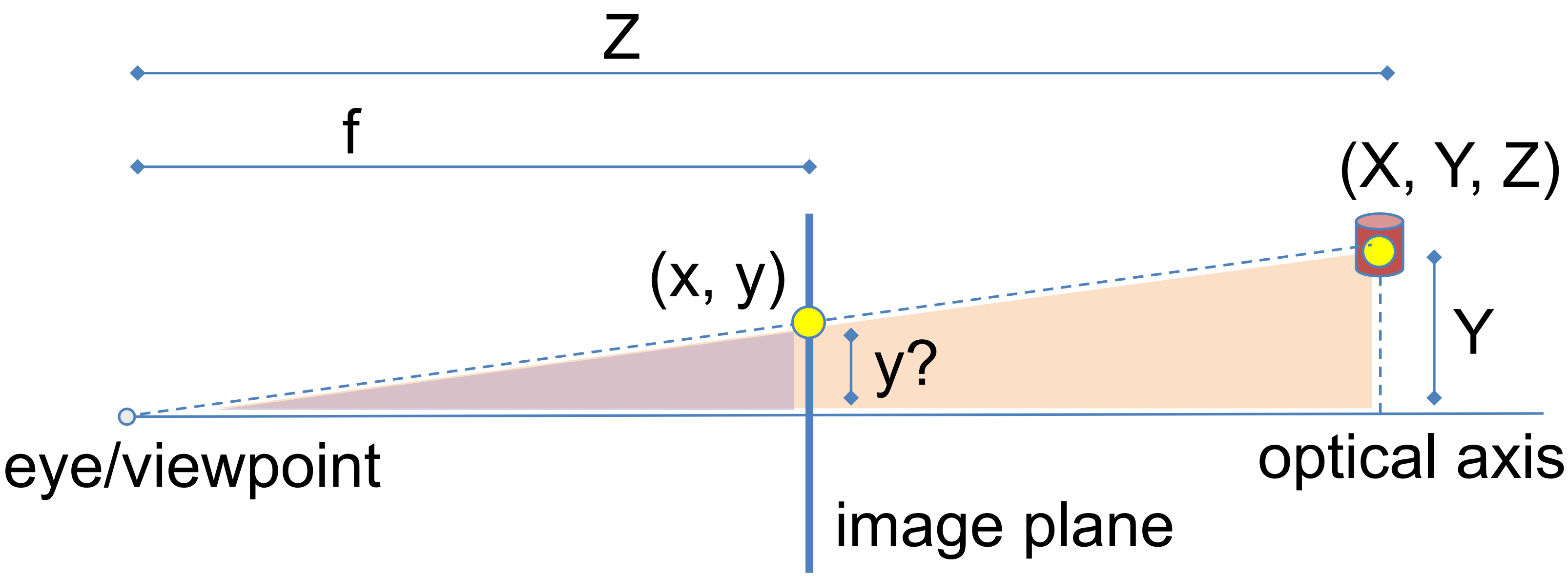
OpenGL Note

- OpenGL is row major and left-multiplies
- This means from the previous (more intuitive explanation) the actual matrix is the transpose of what I wrote; e.g. $M_{OpenGL} = M^T$
- Also, the order of multiplication is from most recent matrix command to least recent matrix command; e.g.
  - "glRotate(rx,1,0,0); glRotate(ry,0,1,0)" rotates a vertex about the y-axis and then the x-axis

Producing a 3D World

- Camera
  - **Zero lens model (i.e., no lens)**
  - Thin lens model
  - Thick lens model

(Zero Lens) Perspective Projection

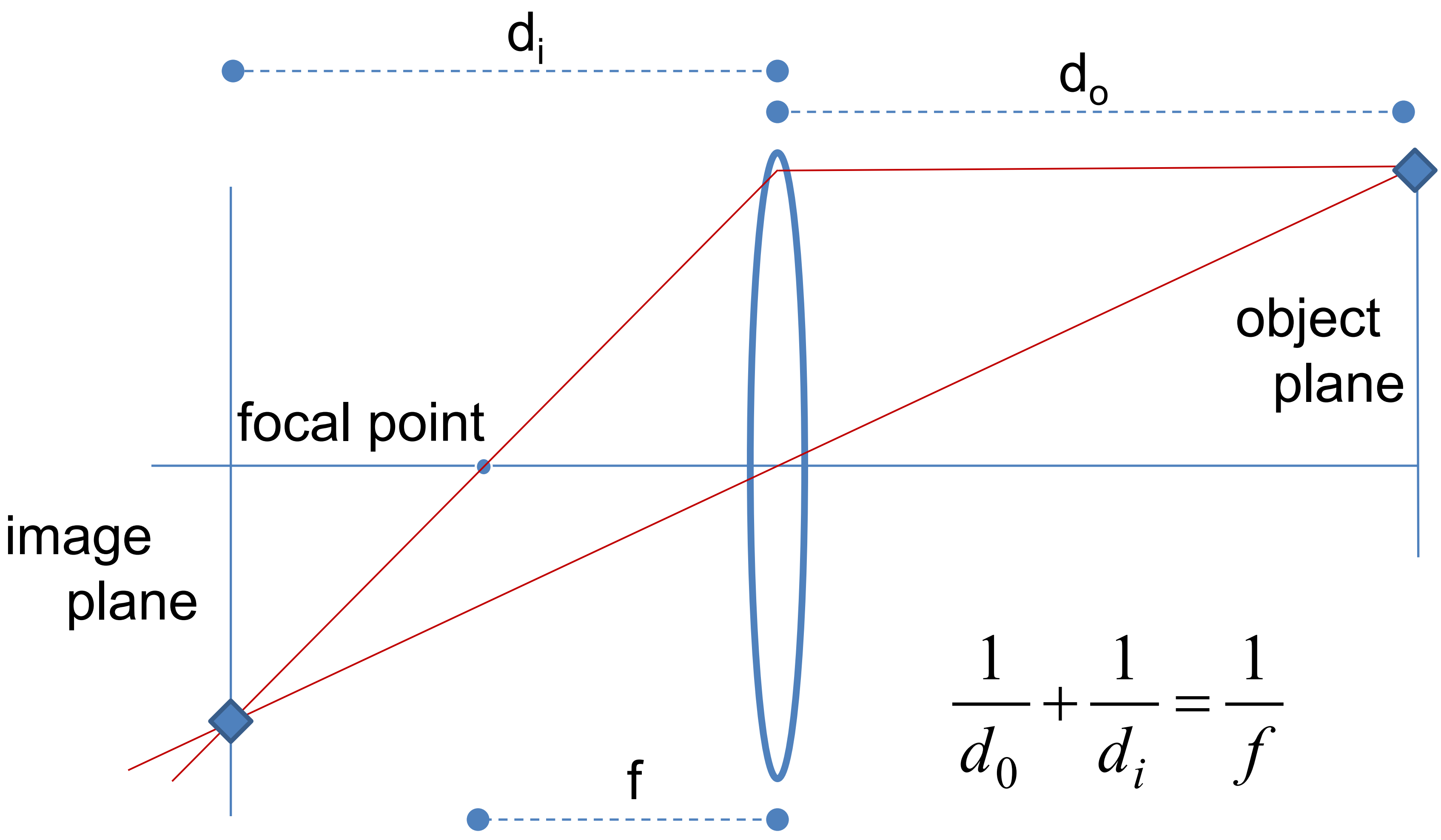


$$\frac{y}{f} = \frac{Y}{Z} \Rightarrow y = f\frac{Y}{Z} \quad \& \quad x = f\frac{X}{Z}$$

Producing a 3D World

- Camera
  - Zero lens model (i.e., no lens)
  - **Thin lens model**
  - Thick lens model

Thin Lens Camera Model



$$\frac{1}{d_0} + \frac{1}{d_i} = \frac{1}{f}$$

Lens Aberrations

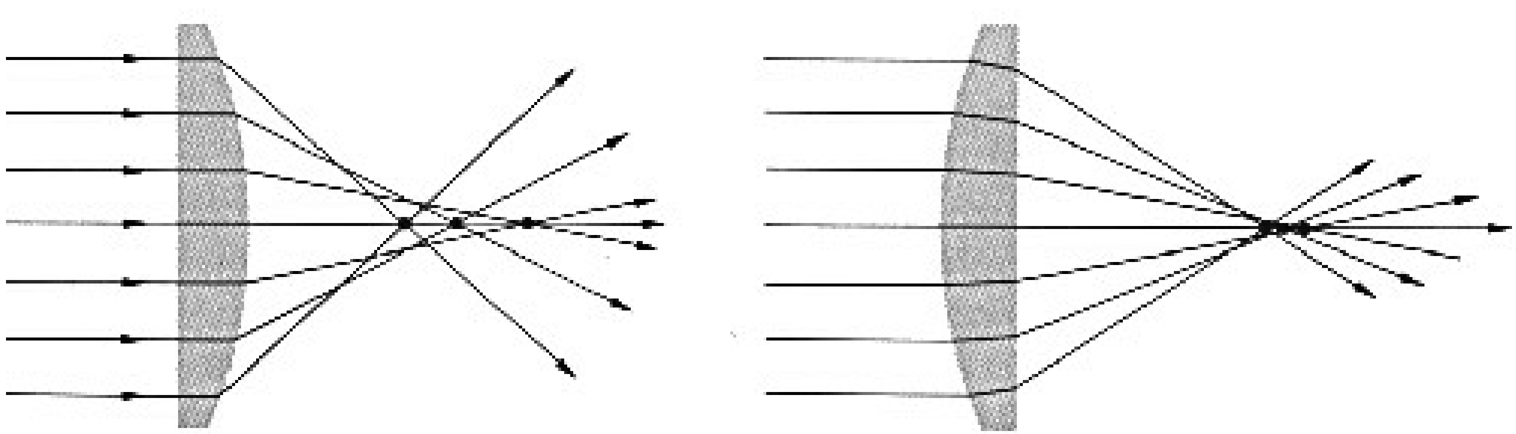
- A "real" lens system does not produce a perfect image
- Aberrations are caused by imperfect manufacturing and by our approximate models
  - Lenses typically have a spherical surface
    - Aspherical lenses would better compensate for refraction but are more difficult to manufacture
  - Typically 1st order approximations are used
    - Remember $\sin \Omega = \Omega - \Omega^3/3! + \Omega^5/5! - \ldots$
    - Thus, thin-lens equations only valid iff $\sin \Omega \approx \Omega$

Aberrations

- Most common aberrations:
  - Spherical aberration
  - Coma
  - Astigmatism
  - Curvature of field
  - Chromatic aberration
  - **Distortion**

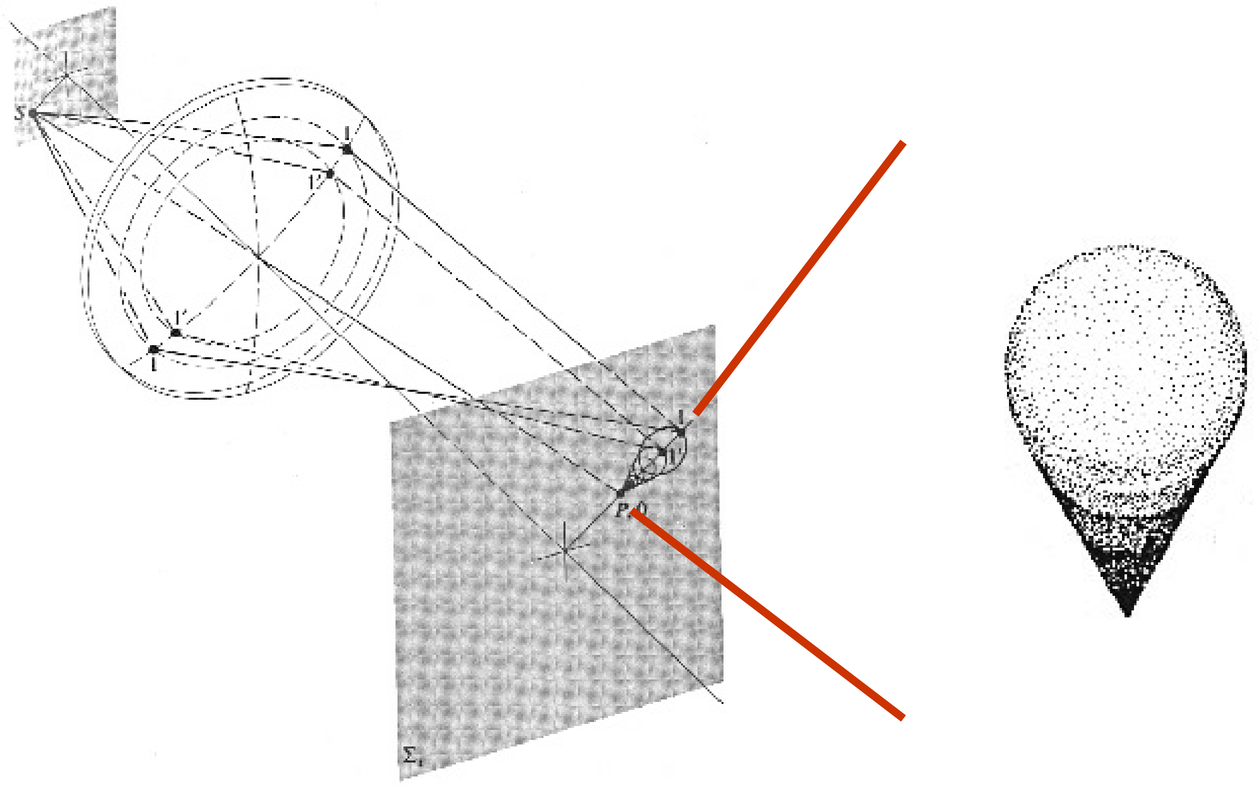
Spherical Aberration

- Deteriorates axial image



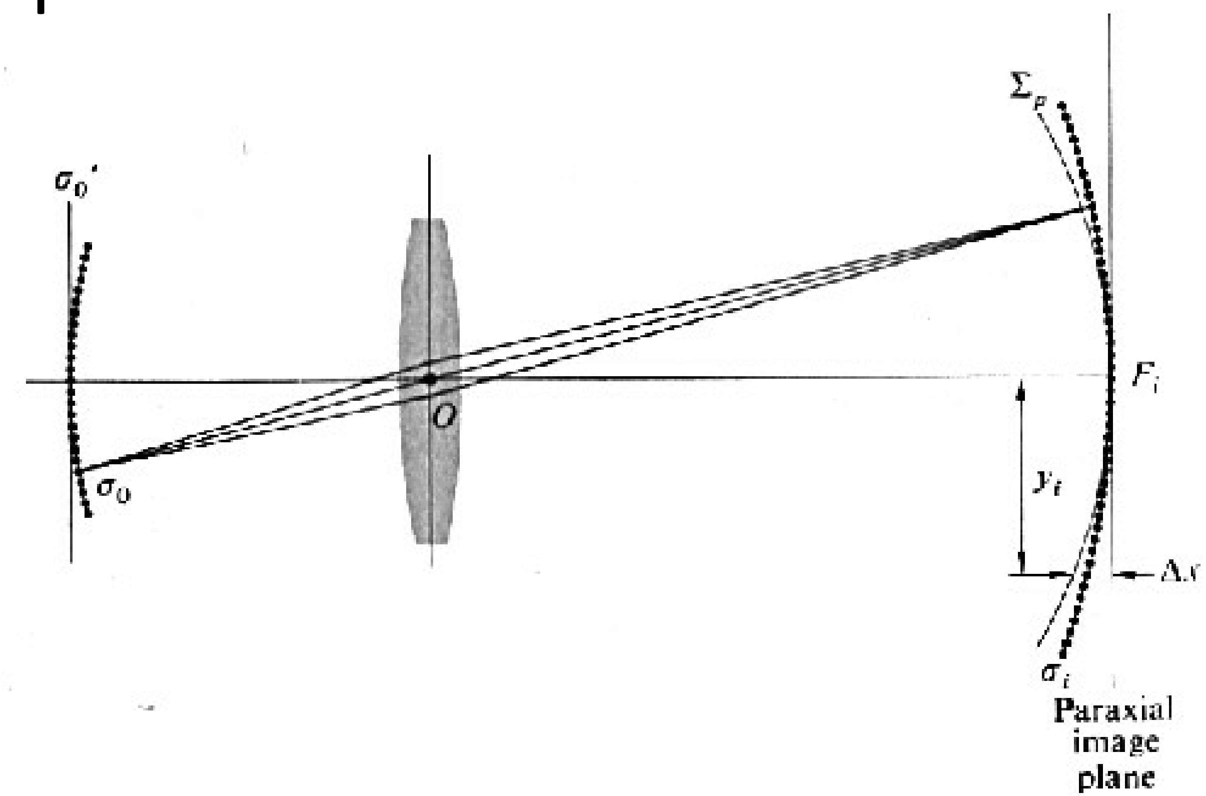
Coma

- Deteriorates off-axial bundles of rays



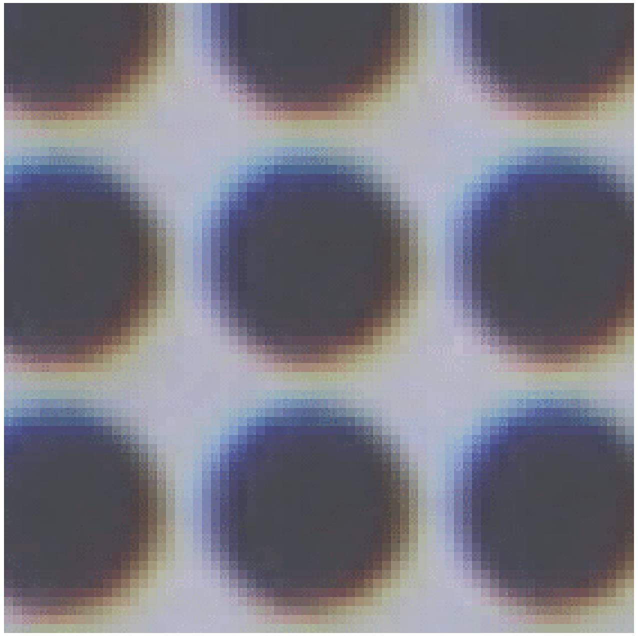
Astigmatism and Curvature of Field

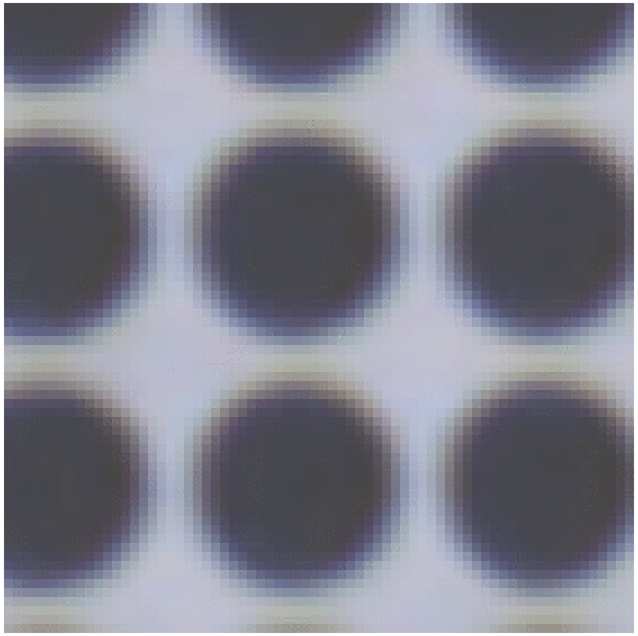
- Produces multiple (two) images of a single object point



Chromatic Aberration

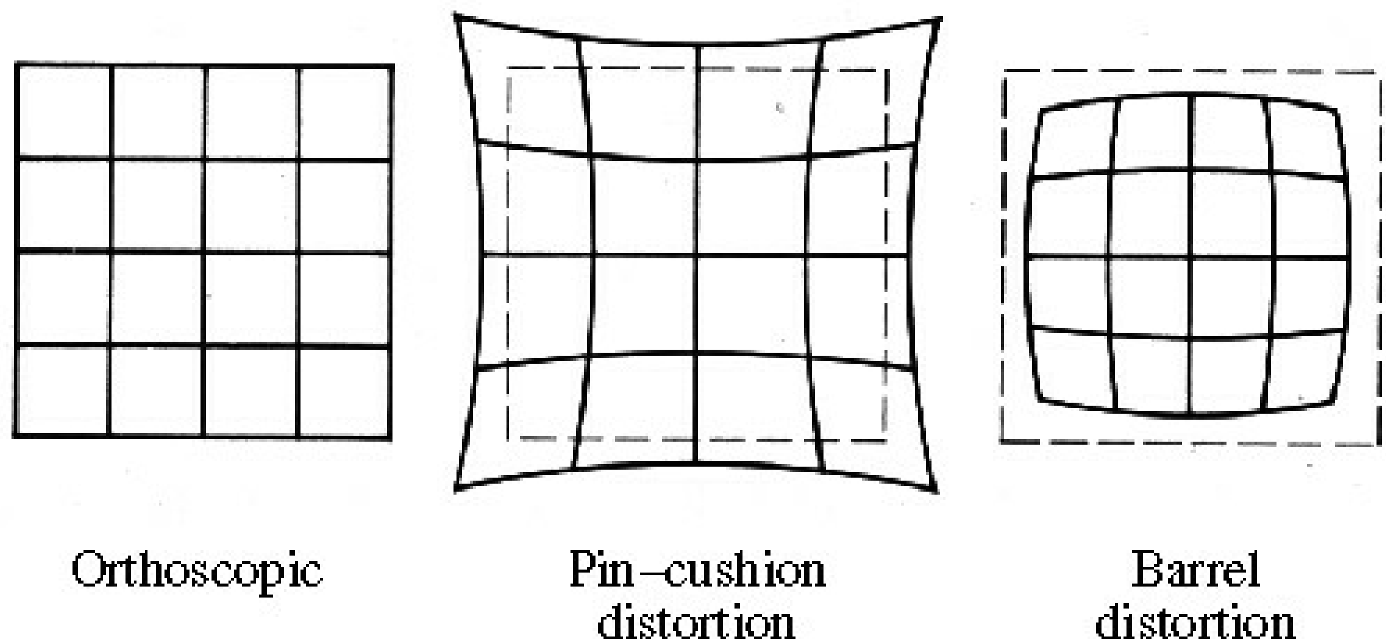
- Caused by wavelength dependent refraction
  - Apochromatic lenses (e.g., RGB) can help





Distortion

- Radial (and tangential) image distortions



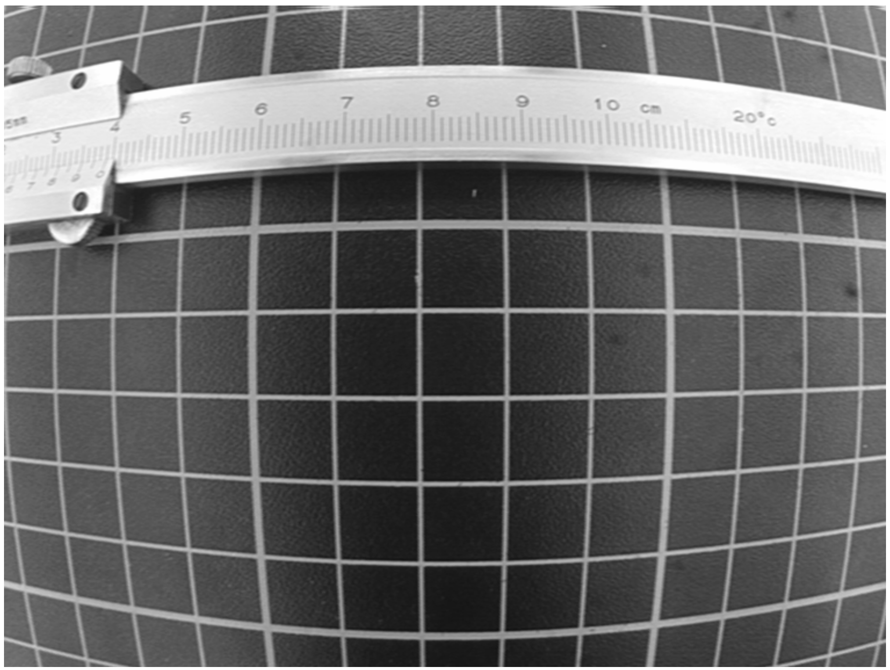
Radial Distortion

- (x, y) pixel before distortion correction
- (x', y') pixel after distortion correction
- Let $r = (x^2 + y^2)^{-1}$

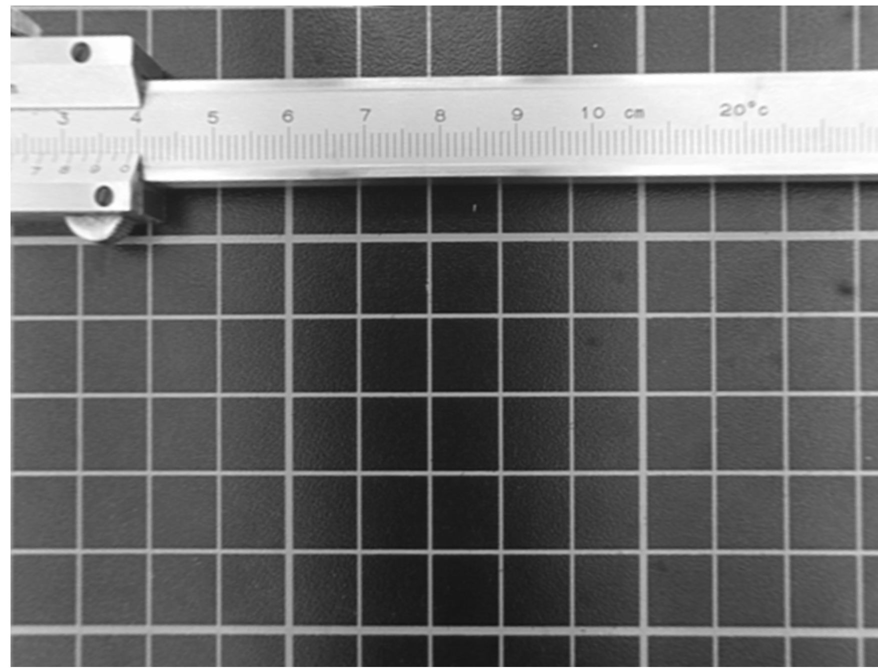
- Then
  - $x' = x(1 - \Delta r/r)$
  - $y' = y(1 - \Delta r/r)$
  - where $\Delta r = k_0 r + k_1 r^3 + k_2 r^5 + \ldots$

- Finally,
  - $x' = x(1 - k_0 - k_1 r^2 - k_2 r^4 - \ldots)$
  - $y' = y(1 - k_0 - k_1 r^2 - k_2 r^4 - \ldots)$

Radial Distortion



before



after

Tsai Camera Model and Calibration

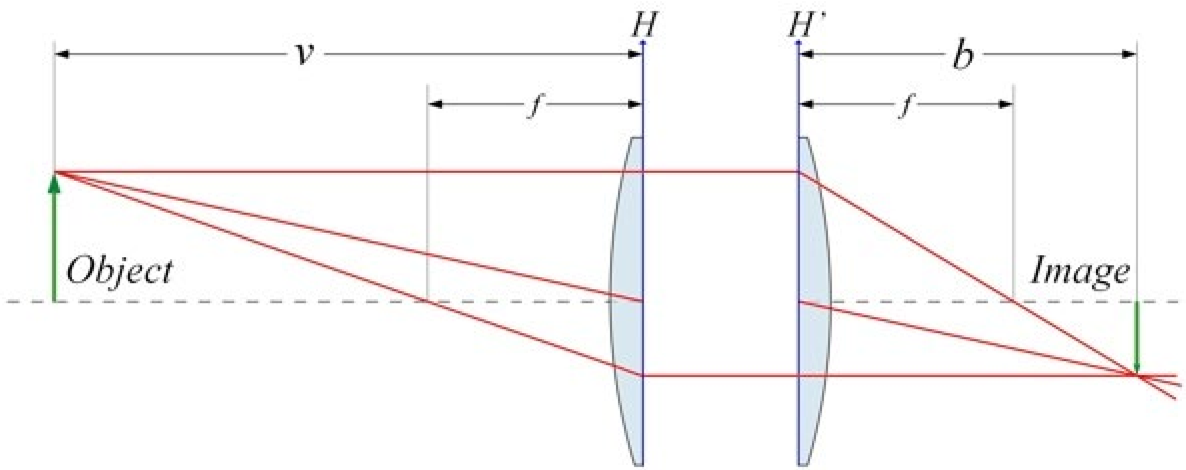
- A widely used camera model to calibrate conventional cameras based on a pinhole camera

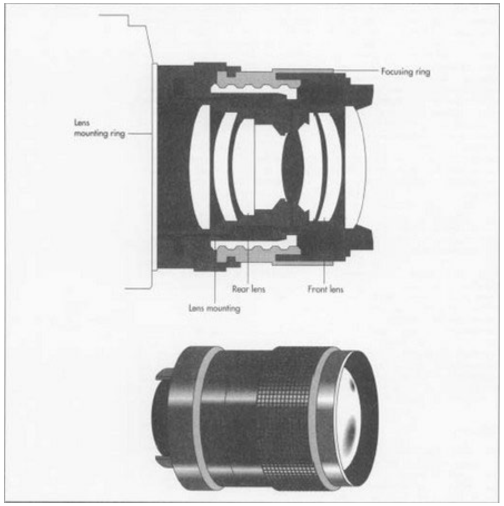
- Reference
  - "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses", Roger Y. Tsai, IEEE Journal of Robotics and Automation, Vol. 3, No. 4, August 1987

Producing a 3D World

- Camera
  - Zero lens model (i.e., no lens)
  - Thin lens model
  - **Thick lens model**

Thick Lens Camera Model



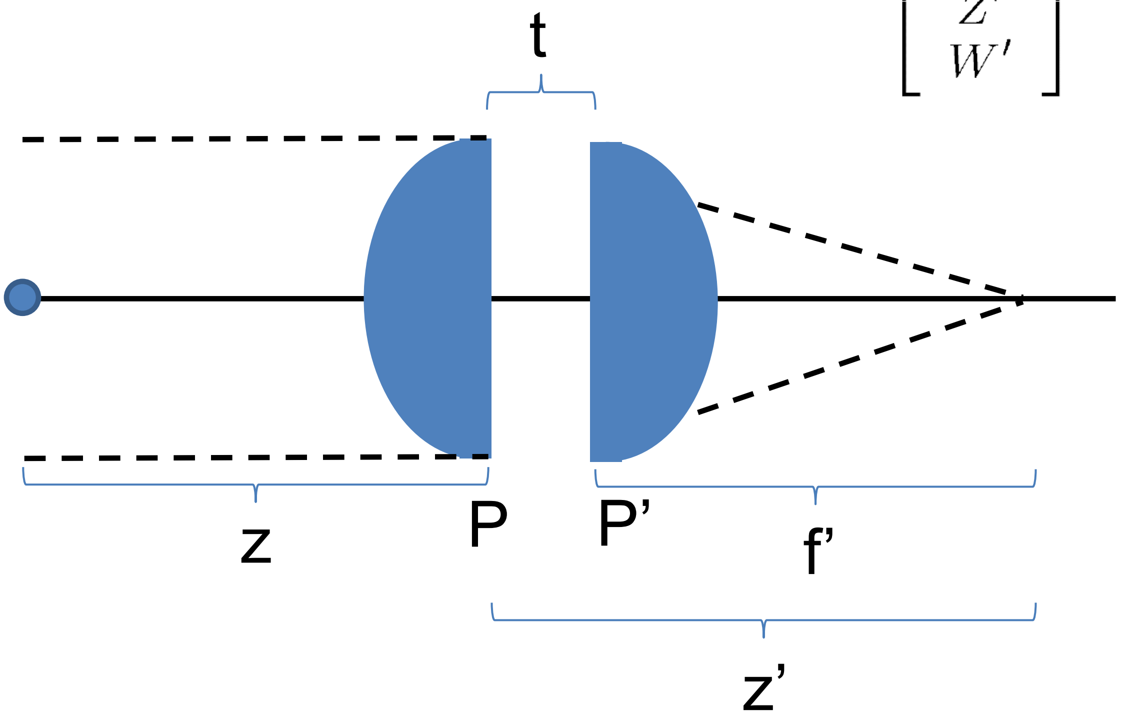


- http://www.cabrillo.edu/~jmccullough/Applets/optics.html
- Even more parameters…

Thick Lens Camera Model

- A simplification for 4x4 matrix usage

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ W' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1+\frac{t}{f'} & t \\ 0 & 0 & \frac{1}{f'} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



[Kolb et al. 1995]