Course Updates

- **New Instructor:** **Leonid Sigal** (call me Leon)

- Contact Info:

E-mails:	lsigal@utsc.utoronto.ca
	ls@cs.toronto.edu
Web Page:	http://www.cs.toronto.edu/~ls/

- Office hours:

Mondays:	**12-1 pm**
	(or by appointment)
Office:	**S625A**

- Assignment and Exams
  - Assignment 1 was collected on
  - Assignment 2 will be out by Wednesday (individual)
  - Mid-term is as scheduled on October 24, 5:15-6:30 in HW-214

3D Transformations

Computer Graphics, CSCD18

Fall 2007

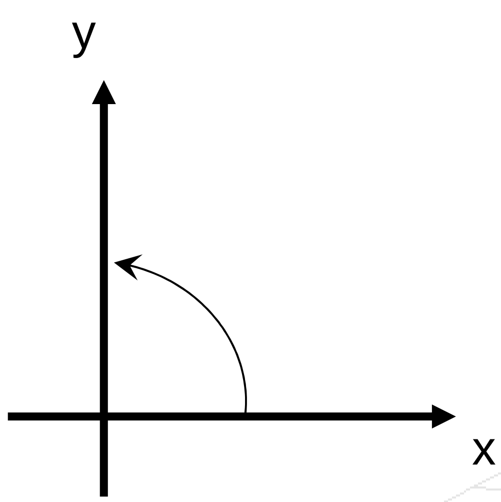
Instructor: Leonid Sigal

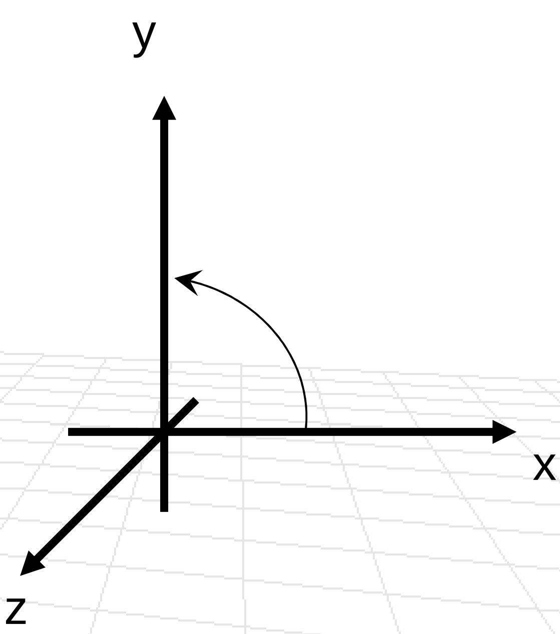
3D Transformations

- Why do we need them?
  - Coordinate transforms
  - Shape modeling (e.g. surfaces of revolution)
    - Alex will do this in the tutorial this week (also on Assignment 2)
  - Hierarchical object models
  - Camera modeling

3D Coordinate Frame

- In 3D there are two conventions for coordinate frames





Right-handed Coordinate System
(OpenGL uses this convention)

Affine Transformations

- Affine transformations in 3D look the same as in 2D

$$F(\vec{p}) = A\vec{p} + \vec{t}$$

$\vec{p}$ - point mapped, $\in \boldsymbol{R}^3$

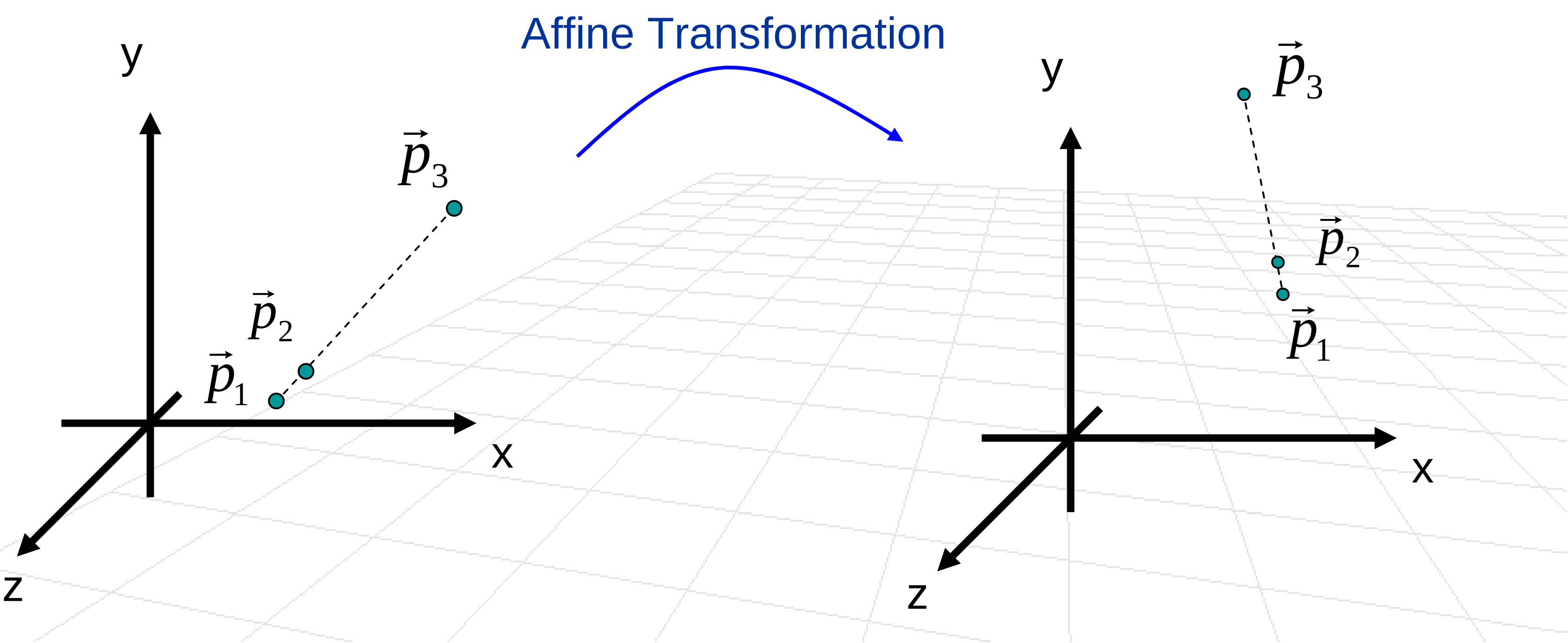
$\vec{t}$ - translation, $\in \boldsymbol{R}^3$

A - transformation matrix, $\in \boldsymbol{R}^{3x3}$

- Many of the transformations we will talk about today are of this type

Properties of Affine Transformations

- Collinearity of points is preserved
- Ratio of distances along the line is preserved
- Concatenation of affine transformations is also an affine transformation



Homogeneous Affine Transformations

- We can rewrite the affine transformation

$$F(\vec{p}) = A\vec{p} + \vec{t}$$

as follows:

$$F(\hat{p}) = M\hat{p}$$

$$M = \begin{bmatrix} A & \vec{t} \\ [0,0,0] & 1 \end{bmatrix} \qquad \hat{p} = \begin{bmatrix} \vec{p} \\ 1 \end{bmatrix}$$

- This has nice properties, we will explore them later

3D Translation

- Simple extension of the 2D translations

$$\boldsymbol{T}_{2\boldsymbol{D}} = \begin{bmatrix} 1 & 0 & \boldsymbol{t}_x \\ 0 & 1 & \boldsymbol{t}_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{T}_{3\boldsymbol{D}} = \begin{bmatrix} 1 & 0 & 0 & \boldsymbol{t}_x \\ 0 & 1 & 0 & \boldsymbol{t}_y \\ 0 & 0 & 1 & \boldsymbol{t}_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Scaling

- Simple extension of the 2D translations

$$\boldsymbol{S}_{2\boldsymbol{D}} = \begin{bmatrix} \boldsymbol{s}_x & 0 & 0 \\ 0 & \boldsymbol{s}_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{S}_{3\boldsymbol{D}} = \begin{bmatrix} \boldsymbol{s}_x & 0 & 0 & 0 \\ 0 & \boldsymbol{s}_y & 0 & 0 \\ 0 & 0 & \boldsymbol{s}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Rotation

- In general, rotations in 3D are much more complicated then 2D rotations
  - There is typically no unique rotation that does what you want
  - You can specify rotations in variety of ways that are convenient for different tasks (e.g. Euler angles, Axis/Angle, Quaternion, Exponential Map)
- We will only consider elementary rotations

3D Rotation

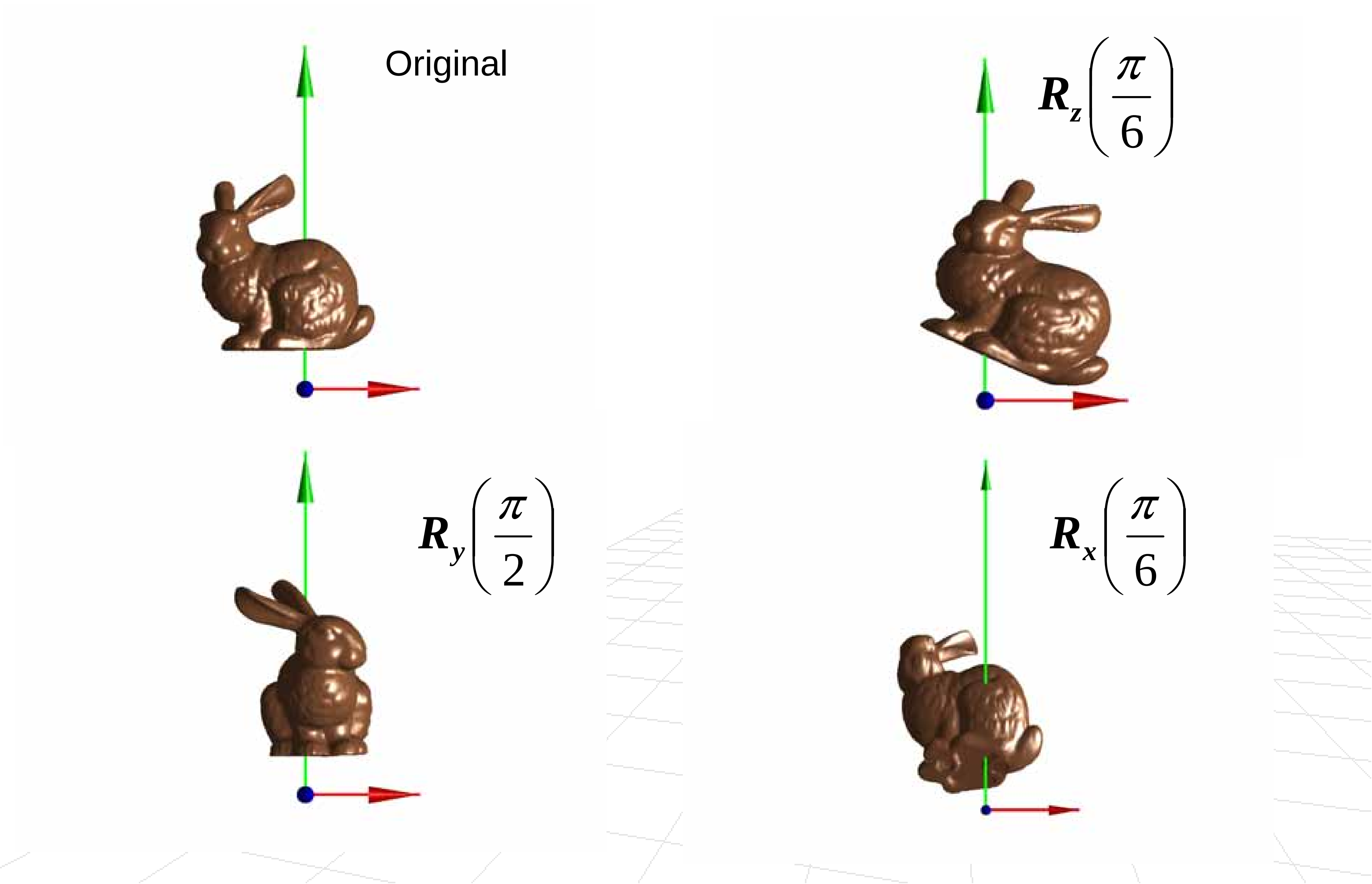
- 2D rotation introduced previously is simply a 3D rotation about the Z-axis

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- But we also have rotations about the X- and Y-axis

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Rotation - Examples



Composing Rotations

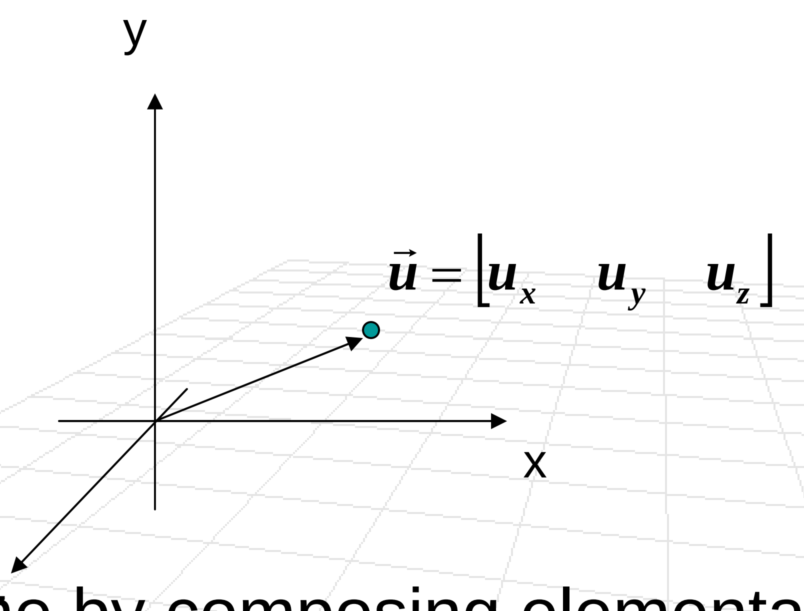
- Rotation order matters !!!
- For example,

$$\boldsymbol{R}_z(\theta_z)\boldsymbol{R}_x(\theta_x) \neq \boldsymbol{R}_x(\theta_x)\boldsymbol{R}_z(\theta_z)$$

- So one needs to be careful

Rotation about Arbitrary Axis

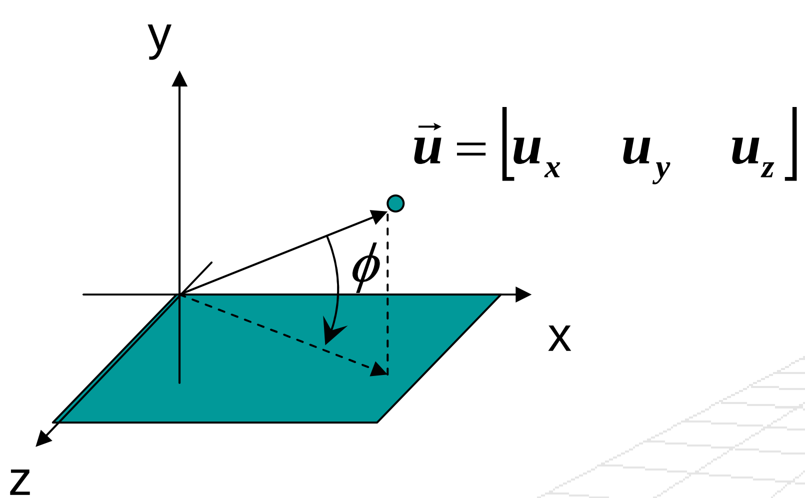
- In general we want to rotate a point or an object about arbitrary axis $\vec{u}$ by some θ
- How do we do this using what we already know?

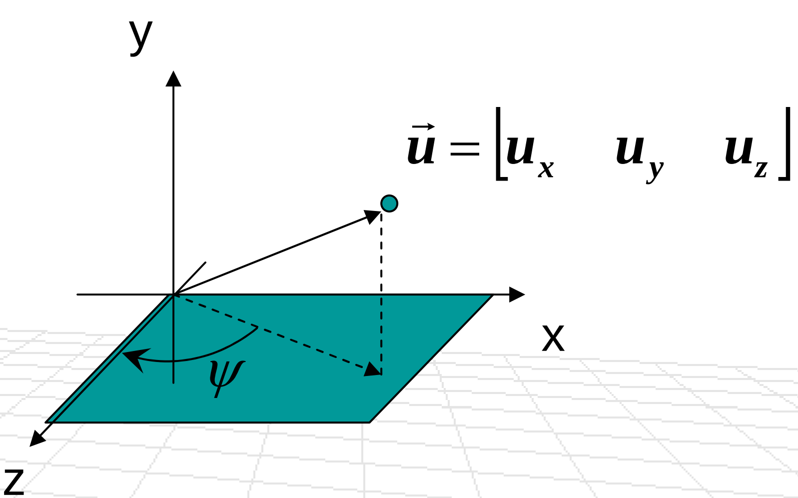


- **Hint:** Can be done by composing elementary rotations

Rotation about Arbitrary Axis

- **Idea:** Align $\vec{u}$ with z-axis, then rotate about z-axis by desired angle θ





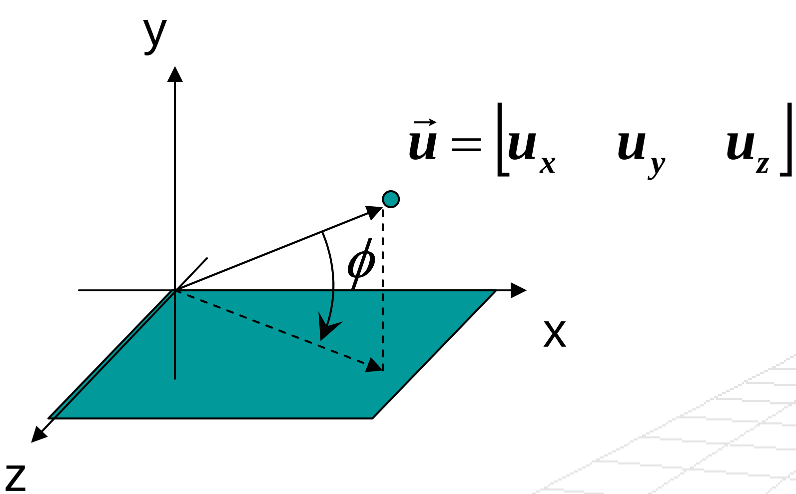
1) Rotate $\vec{u}$ into x-z plane $R_z(\phi)$

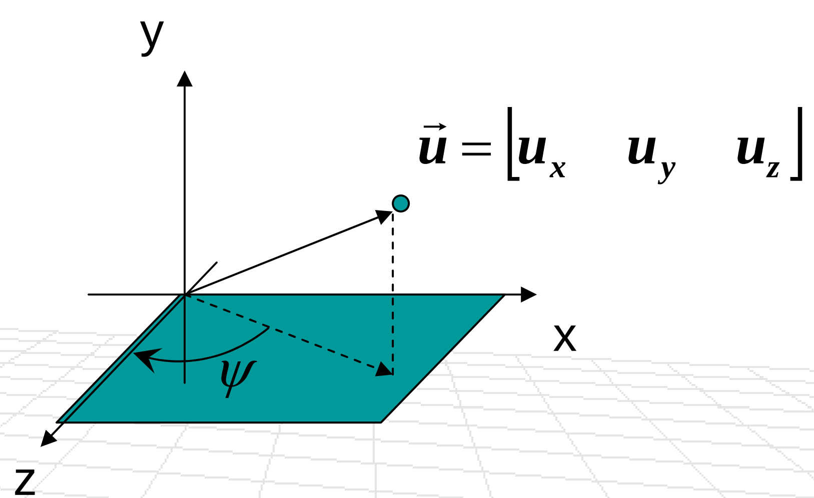
2) Rotate $\vec{u}$ in x-z plane $R_y(\psi)$

3) Rotate by θ about z-axis

Rotation about Arbitrary Axis

- **Idea:** Align $\vec{u}$ with z-axis, then rotate about z-axis by desired angle θ





1) Rotate $\vec{u}$ into x-z plane $\boldsymbol{R}_z(\phi)$

2) Rotate $\vec{u}$ in x-z plane $R_y(\psi)$

3) Rotate by θ about z-axis

4) Undo (1) and (2), i.e. $\left(R_z(\phi)R_y(\psi)\right)^{-1} = \left(R_y(\psi)\right)^{-1}\left(R_z(\phi)\right)^{-1} = R_y(-\psi)R_z(-\phi)$

Rotation about Arbitrary Axis

- Hence rotation about an arbitrary axis can always be expressed as a series of elementary rotations

$$R(\vec{u},\theta) = R_z(\phi)R_x(\psi)R_z(\theta)R_x(-\psi)R_z(-\phi)$$

- How do we obtain values for angles ϕ, ψ?
  - Alex will cover this in the tutorial this week

Non-Linear Transformations

- Affine transformations

$$F(\vec{p}) = A\vec{p} + \vec{t}$$

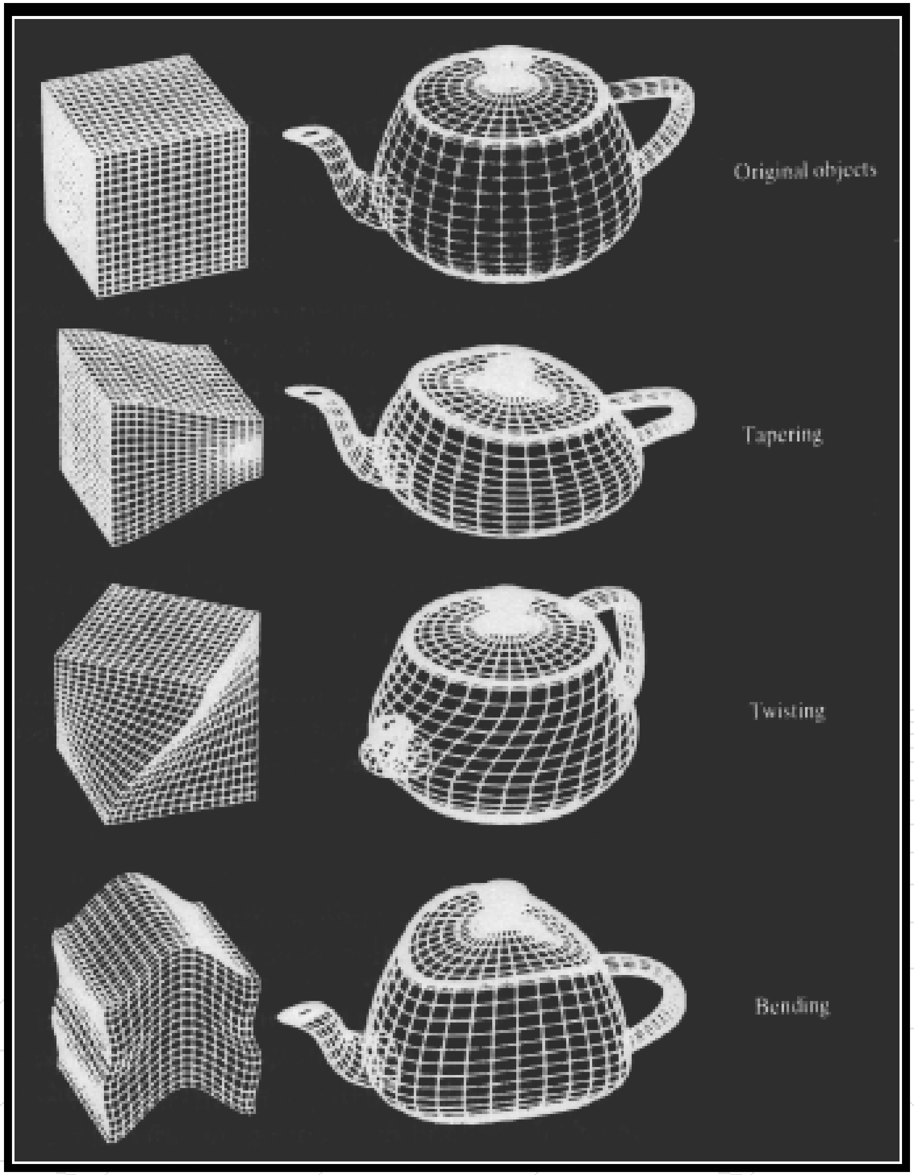
are 1st order shape deformations

- Higher order deformations are also possible, let's consider general differentiable deformation $F(\vec{p})$ then we can express deformation as a Taylor series

$$F(\vec{p}) = \vec{t} + A\vec{p} + B\vec{p}^{\,2} + \ldots$$

- Common non-linear transformations: tapering, twisting, bending

Non-Linear Transformations



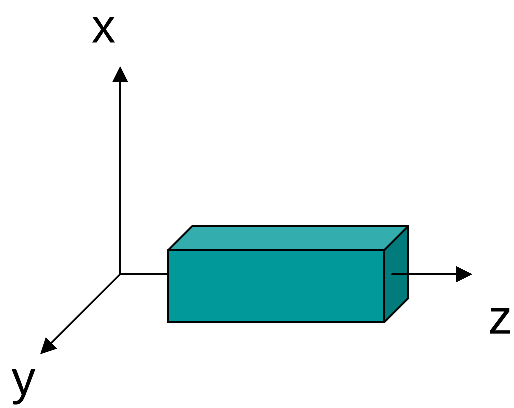
Original

Tapering

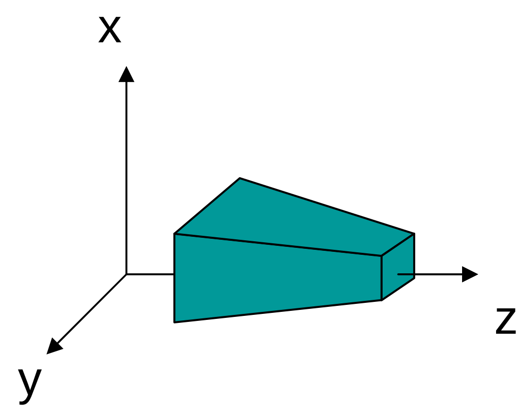
Twisting

Bending

Tapering



Scaling



Linear Taper

$$\boldsymbol{S}_{3D} = \begin{bmatrix} \boldsymbol{s}_x & 0 & 0 & 0 \\ 0 & \boldsymbol{s}_y & 0 & 0 \\ 0 & 0 & \boldsymbol{s}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{Taper}_{3D} = \begin{bmatrix} \boldsymbol{s}_x(\boldsymbol{p}_z) & 0 & 0 & 0 \\ 0 & \boldsymbol{s}_y(\boldsymbol{p}_z) & 0 & 0 \\ 0 & 0 & \boldsymbol{s}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{s}_x(\boldsymbol{p}_z) = \alpha_0 + \alpha_1 \boldsymbol{p}_z$$

$$\boldsymbol{s}_y(\boldsymbol{p}_z) = \alpha_0 + \alpha_1 \boldsymbol{p}_z$$

Big Picture

- What can we do so far?
  - Model a 3D object (hierarchical objects)
  - Transform a 3D object
- What else do we need?
  - Camera
- Why?
  - We need to project model of the 3D world to 2D film plane (or screen)

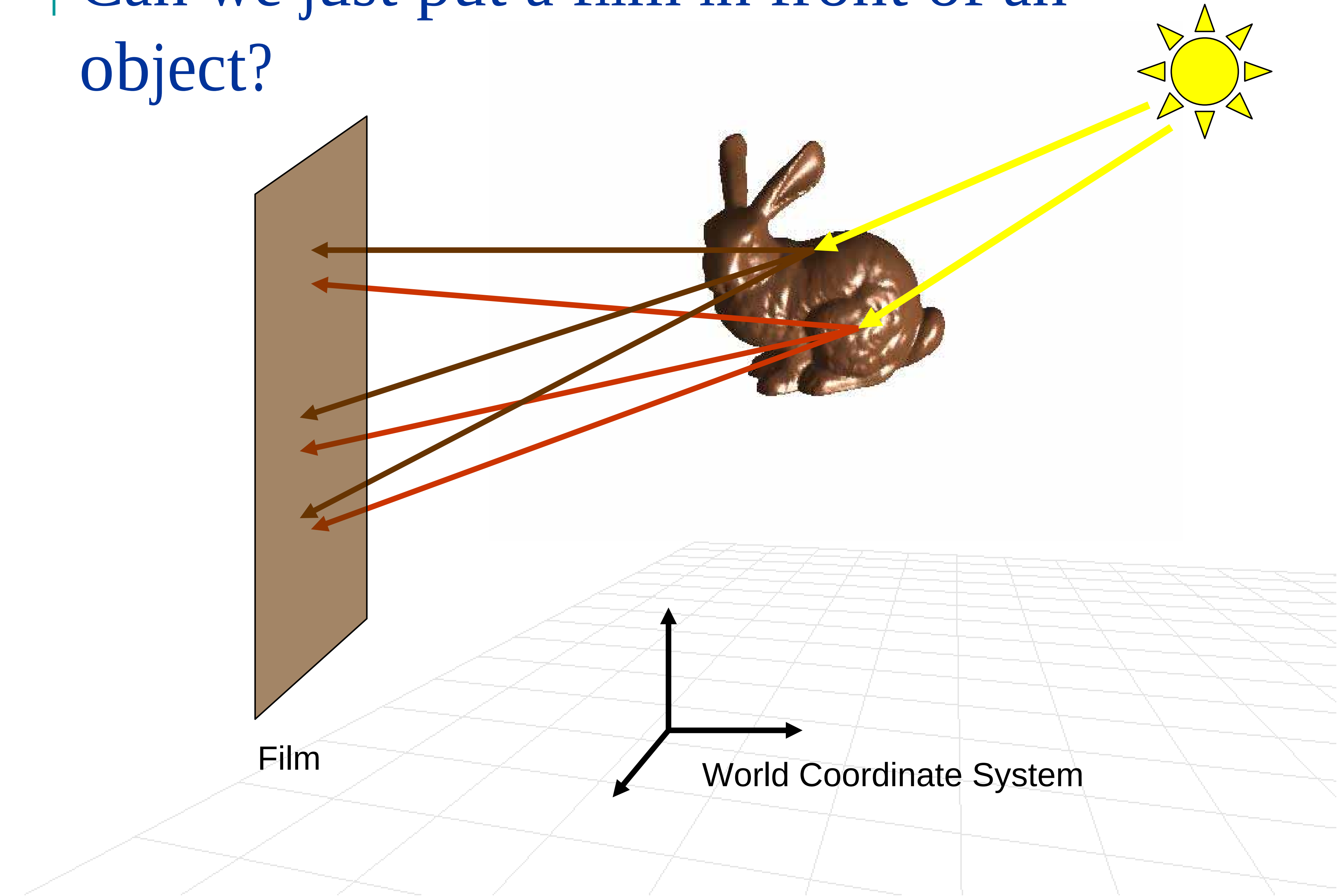
Camera Models
Part 1

Computer Graphics, CSCD18

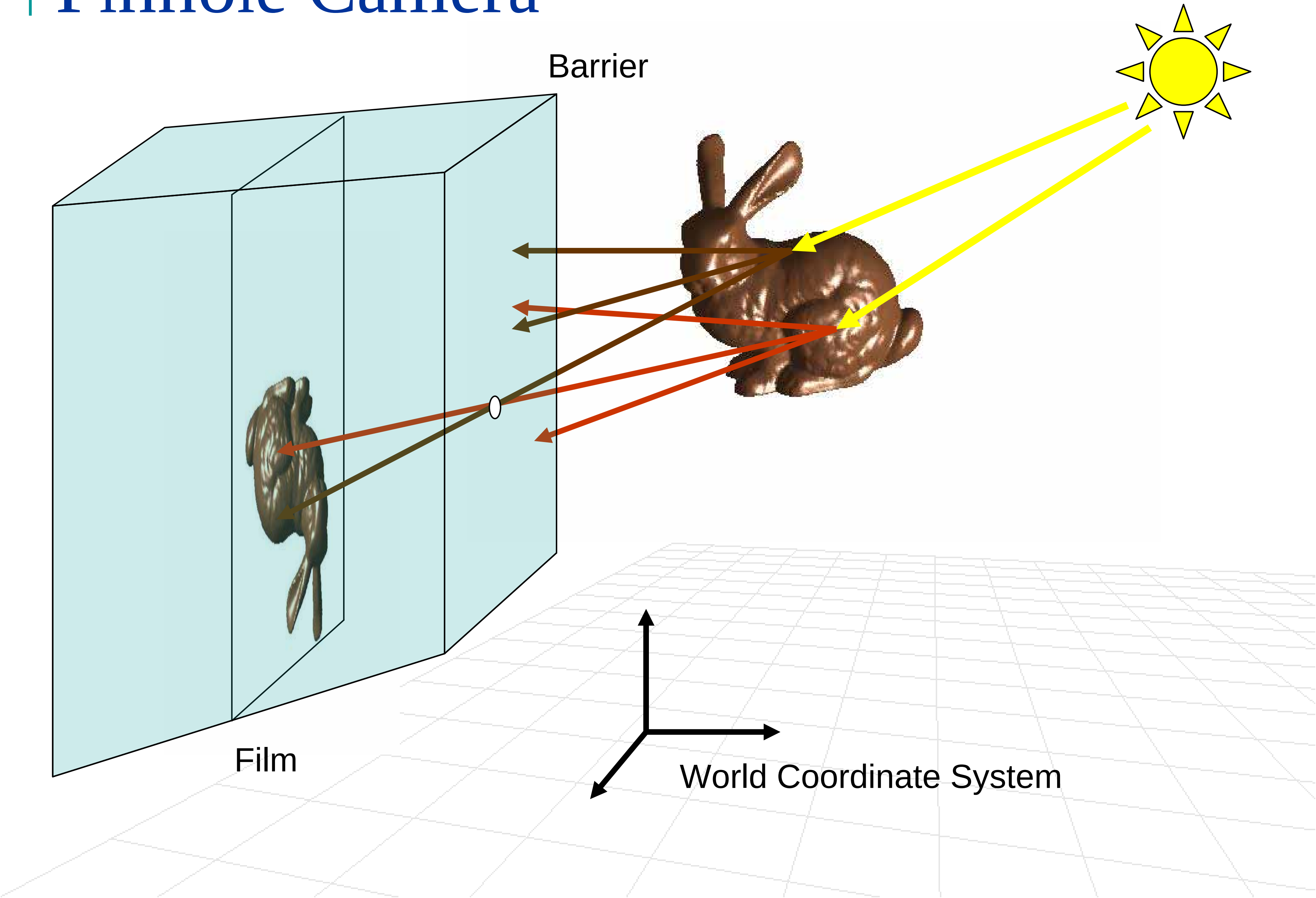
Fall 2007

Instructor: Leonid Sigal

Can we just put a film in front of an object?

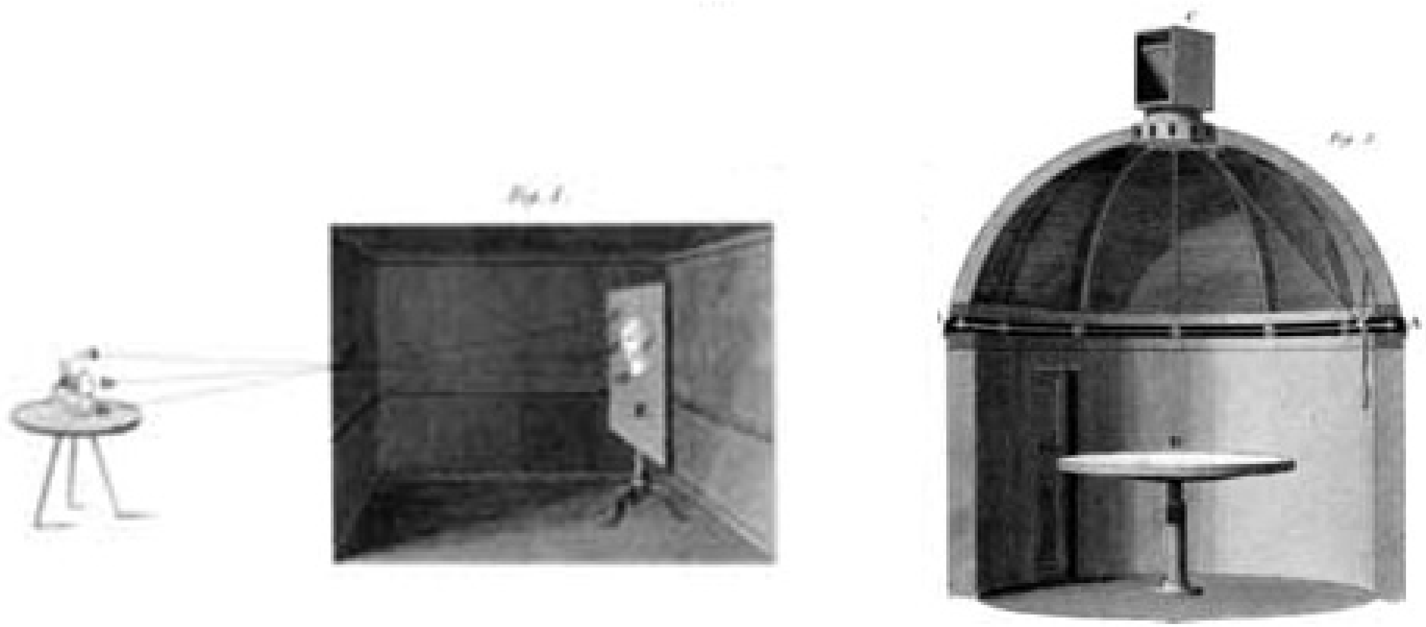


Pinhole Camera



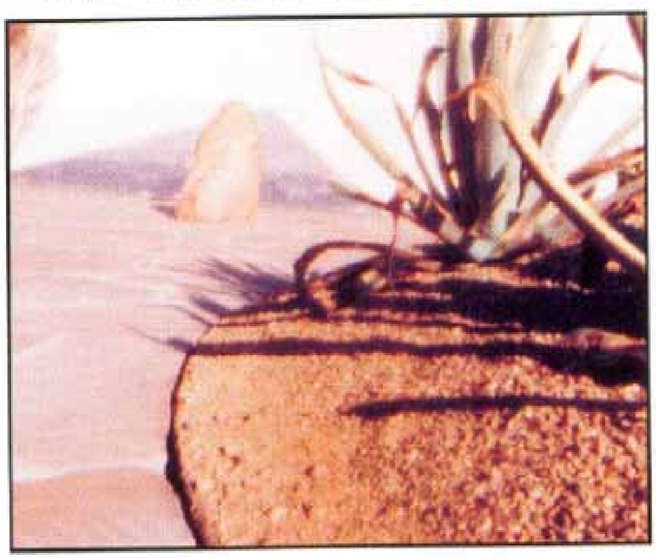
Pinhole Camera

- Room size pinhole cameras date back to 18th century



Pinhole camera

- Problems
  - Small pinhole -> sharp image, but little light, slow image acquisition
  - Large pinhole -> reduces sharpness, but faster acquisition


Photograph made with small pinhole


Photograph made with larger pinhole

Images from lecture notes of Matthias Zwicker

Lenses

- Focus the light, so that enough light can be captured in sufficiently short amount of time (i.e. allows the pinhole to be made larger)

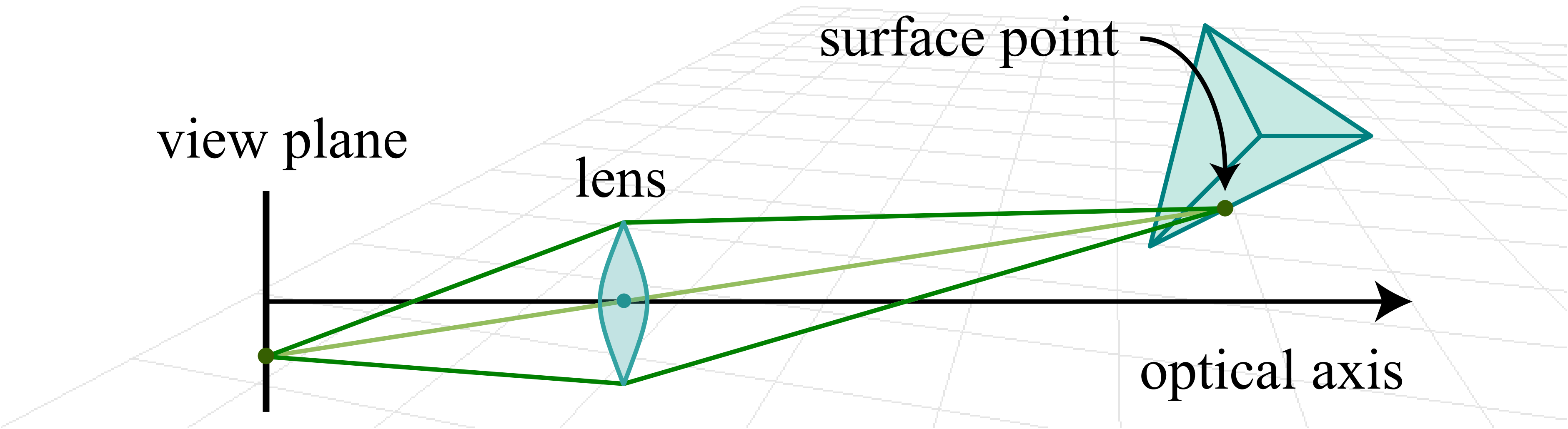
6 sec. exposure

0.01 sec exposure

Images from lecture notes of Matthias Zwicker

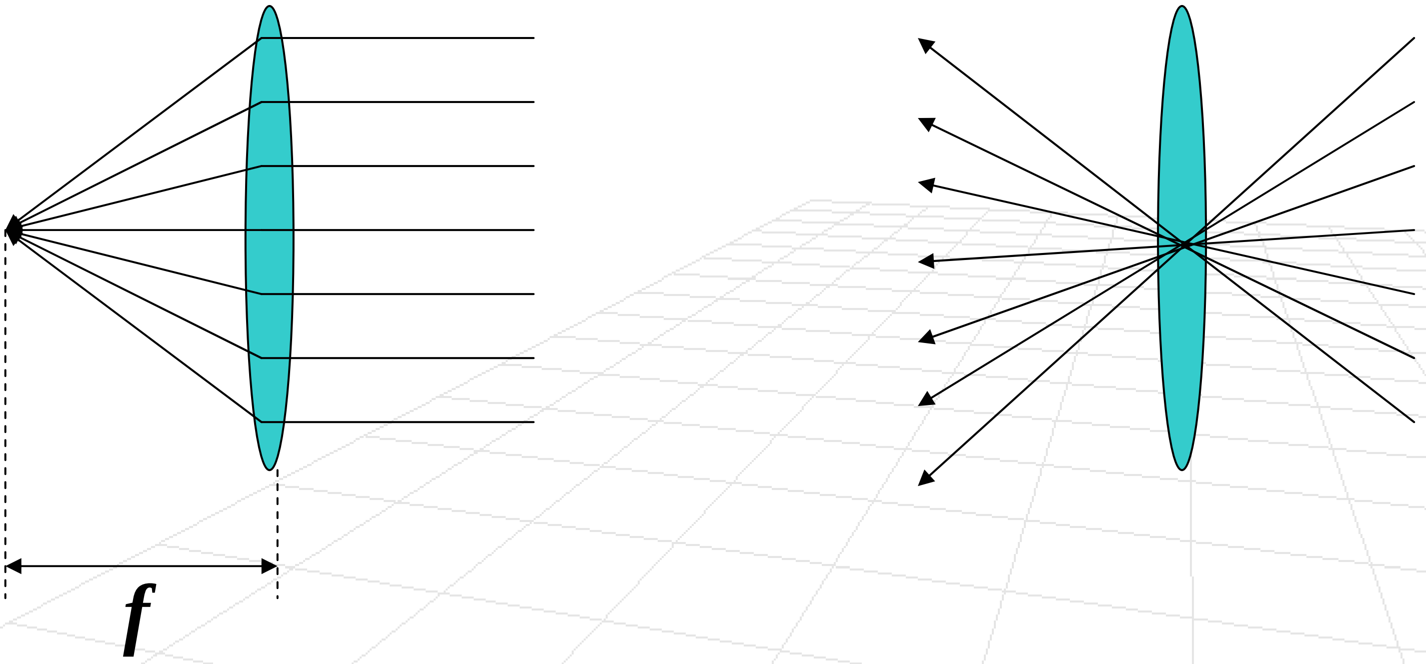
Lenses

- Lens models in real cameras can be very complex
- We will only consider a simple "Thin Lens" model



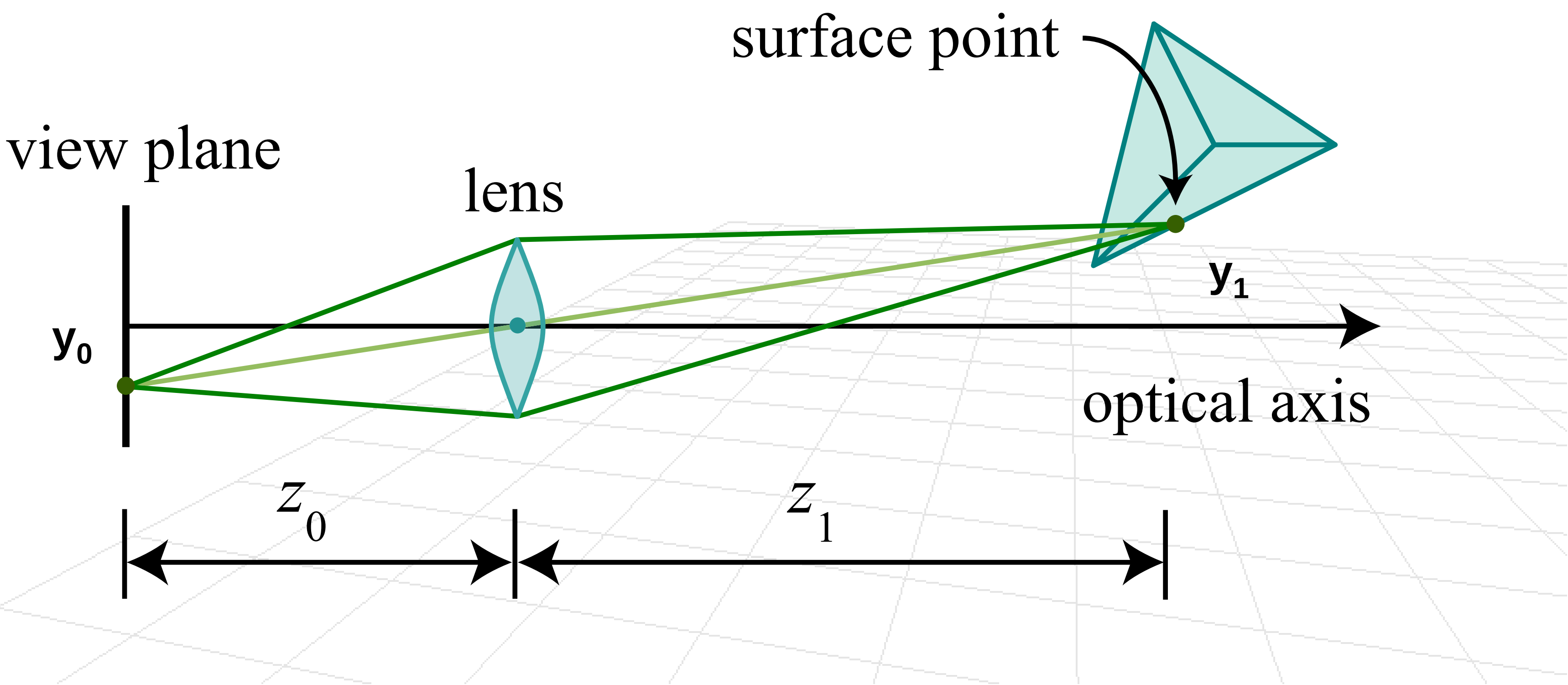
Thin Lens Model

- All parallel rays converge at focal length *f*
- Rays through the center are not deflected



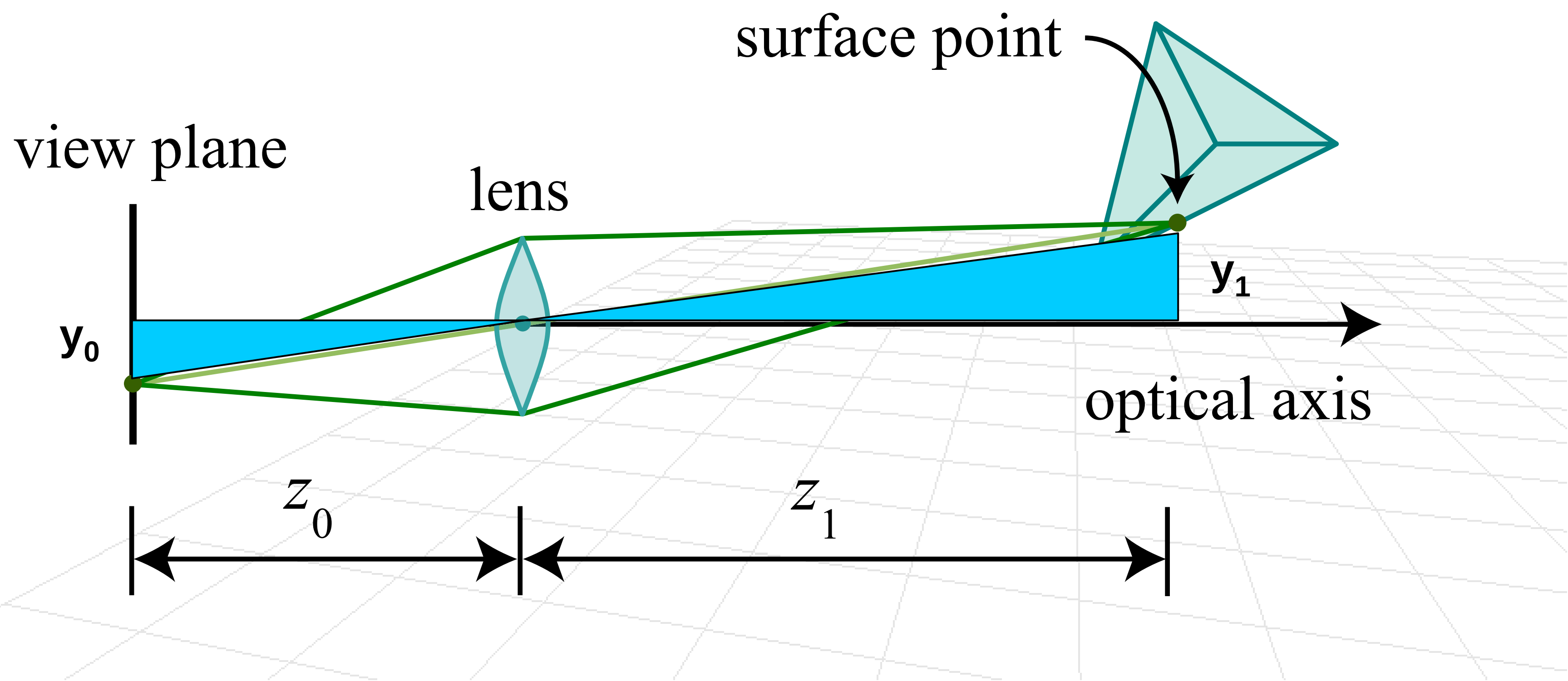
Thin Lens Model

- For rays that are not parallel, we can derive the thin lens equation



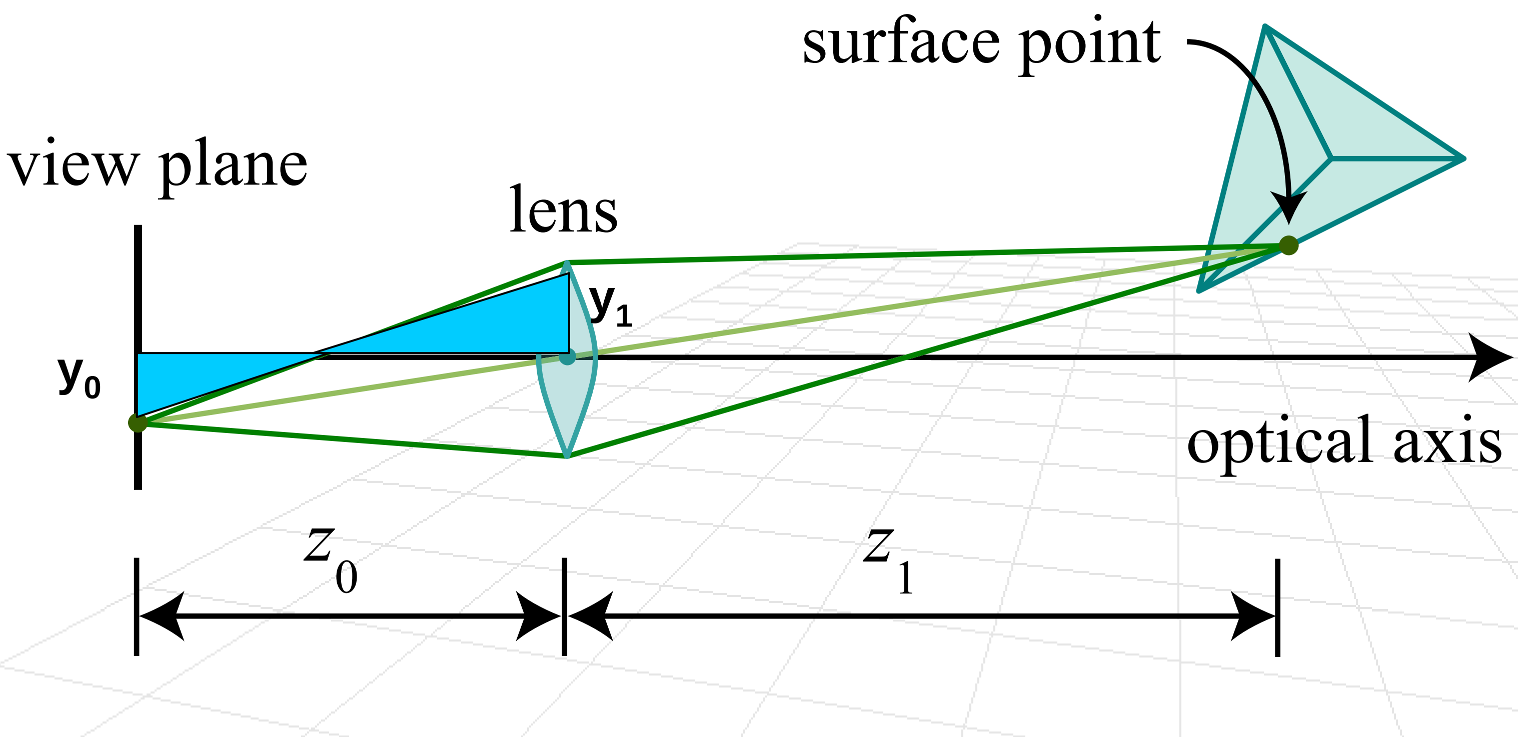
Thin Lens Model

- For rays that are not parallel, we can derive the thin lens equation
- Similar triangles: $\mathbf{y_0 / y_1 = z_0 / z_1}$



Thin Lens Model

- For rays that are not parallel, we can derive the thin lens equation
- Similar triangles: $y_0/ y_1= z_0/ z_1$
- Similar triangles: $y_0/ y_1= (z_0-f)/f$



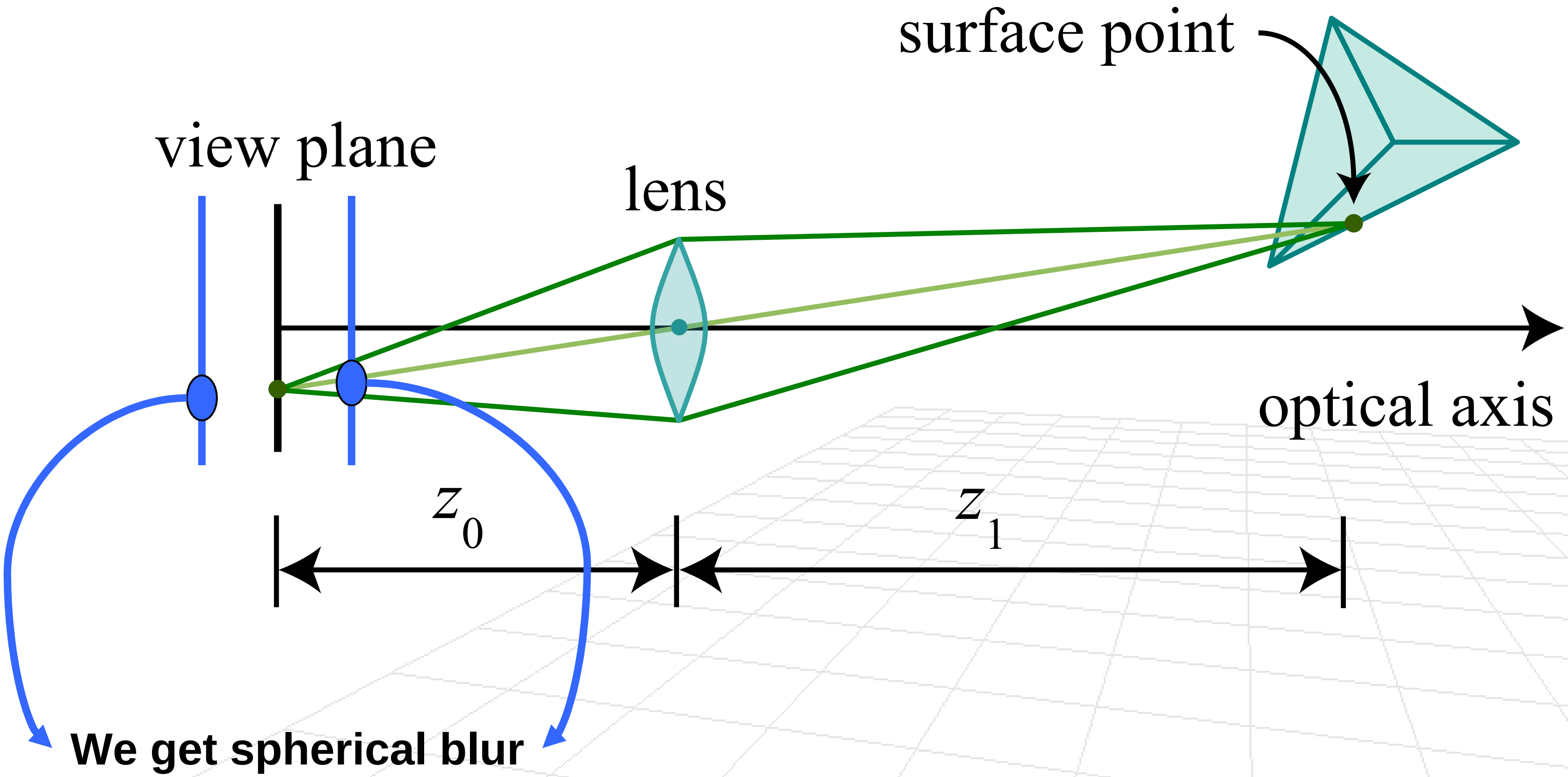
Thin Lens Model

- For rays that are not parallel, we can derive the thin lens equation
- Similar triangles: $y_0 / y_1 = z_0 / z_1$
- Similar triangles: $y_0 / y_1 = (z_0 - f) / f$

$$\frac{z_0}{z_1} = \frac{z_0 - f}{f}$$

$$\boxed{\frac{1}{z_0} + \frac{1}{z_1} = \frac{1}{f}}$$

What if we put view plane elsewhere?



Relationship of Thin Lens Camera and Pinhole Camera

- Pinhole camera is the idealization of the thin lens camera model, where the aperture shrinks to a tiny hole.
- Let's go back to the pin hole camera, it is simpler to deal with

Conceptual Pinhole Camera