

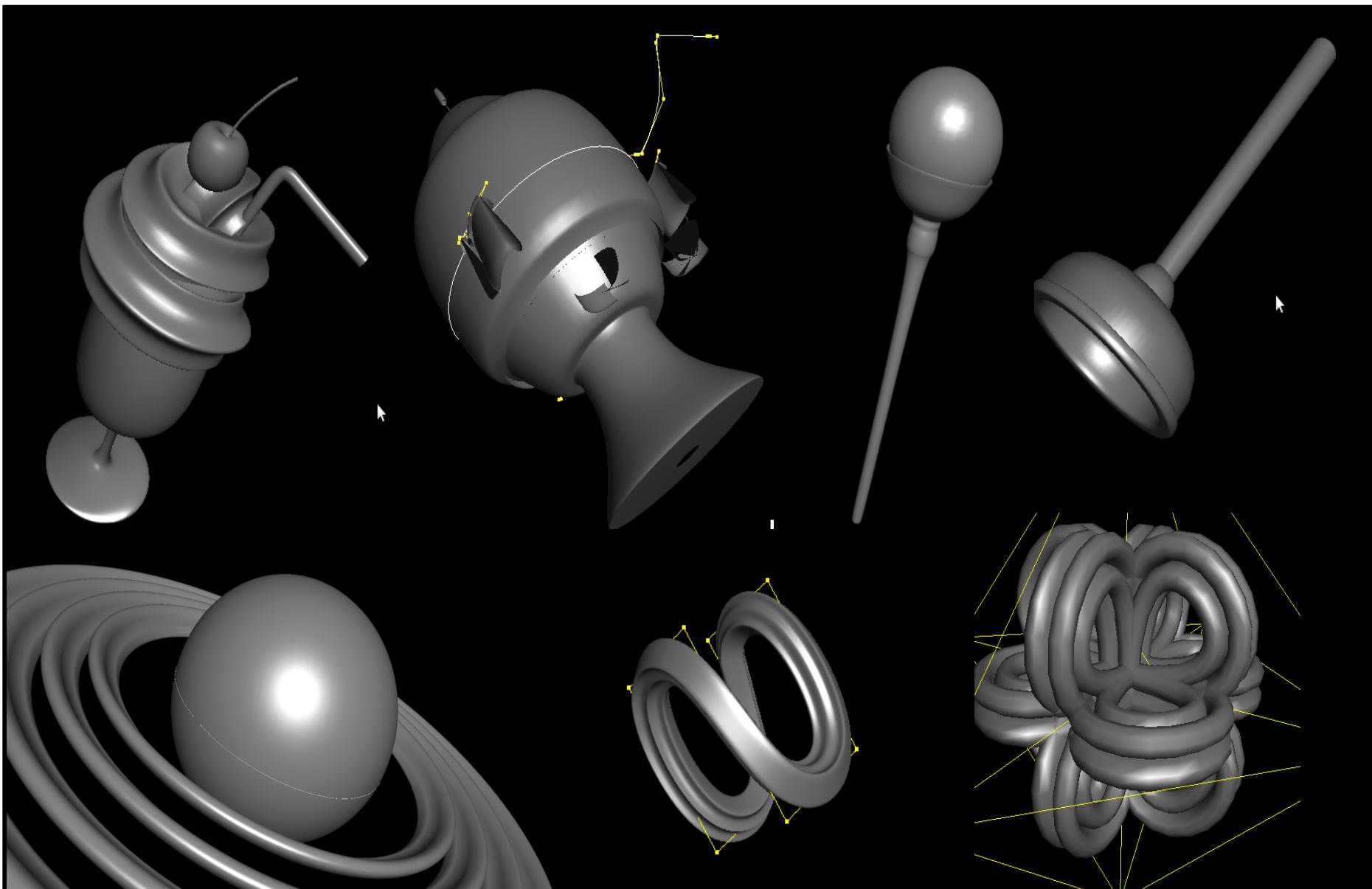
# MIT EECS 6.837 Computer Graphics

## Part 2 – Rendering

### Today: Intro to Rendering, Ray Casting



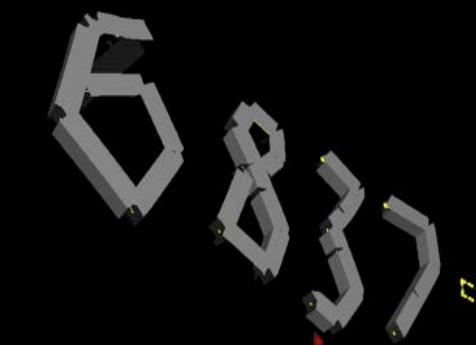
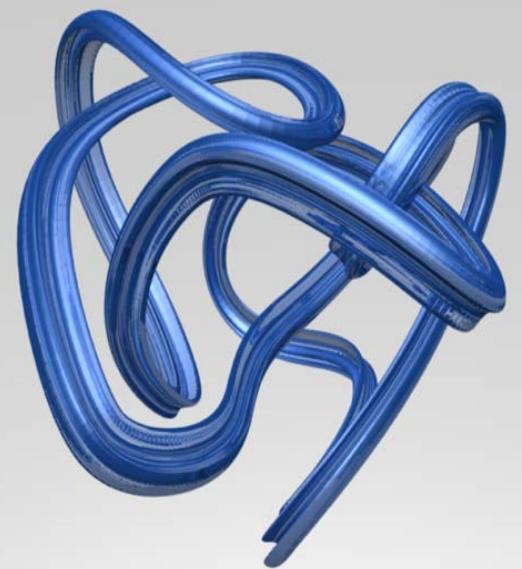
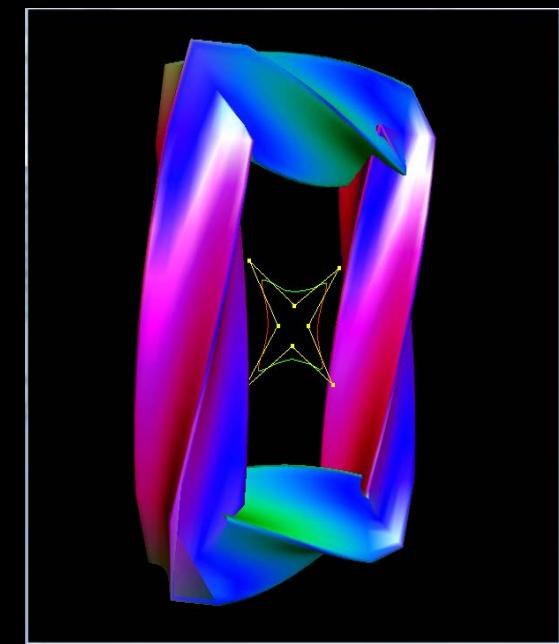
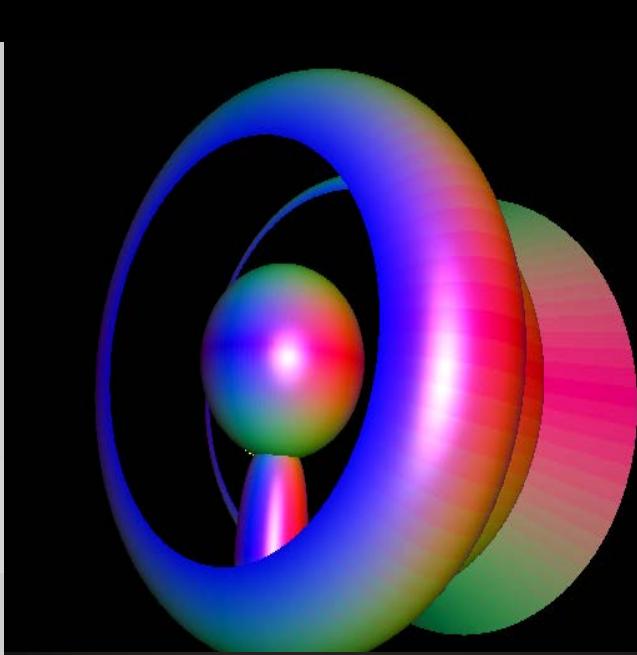
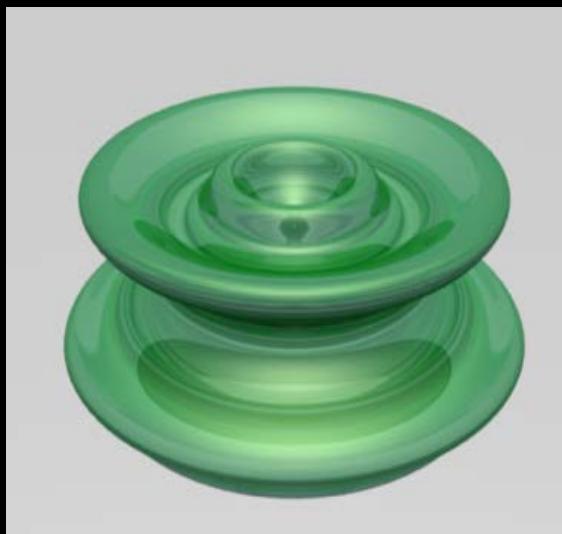
# Cool Artifacts from Assignment 1



# Cool Artifacts from Assignment 1



# Cool Artifacts from Assignment 1



# The Story So Far

---

- Modeling
  - splines, hierarchies, transformations, meshes, etc.
- Animation
  - skinning, ODEs, masses and springs
- **Now we'll see how to generate an image given a scene description!**

# The Remainder of the Term

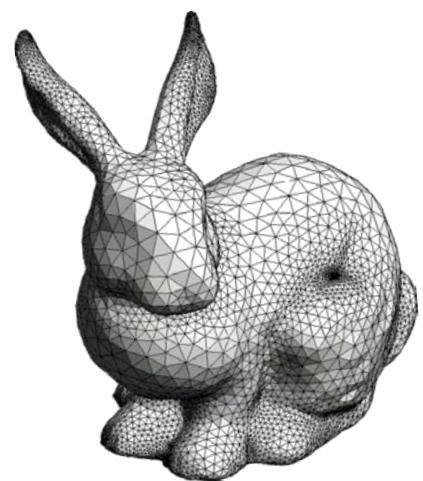
---

- Ray Casting and Ray Tracing
- Intro to Global Illumination
  - Monte Carlo techniques, photon mapping, etc.
- Shading, texture mapping
  - What makes materials look like they do?
- Image-based Rendering
- Sampling and antialiasing
- Rasterization, z-buffering
- Shadow techniques
- Graphics Hardware

# Today

---

- What does *rendering* mean?
- Basics of ray casting



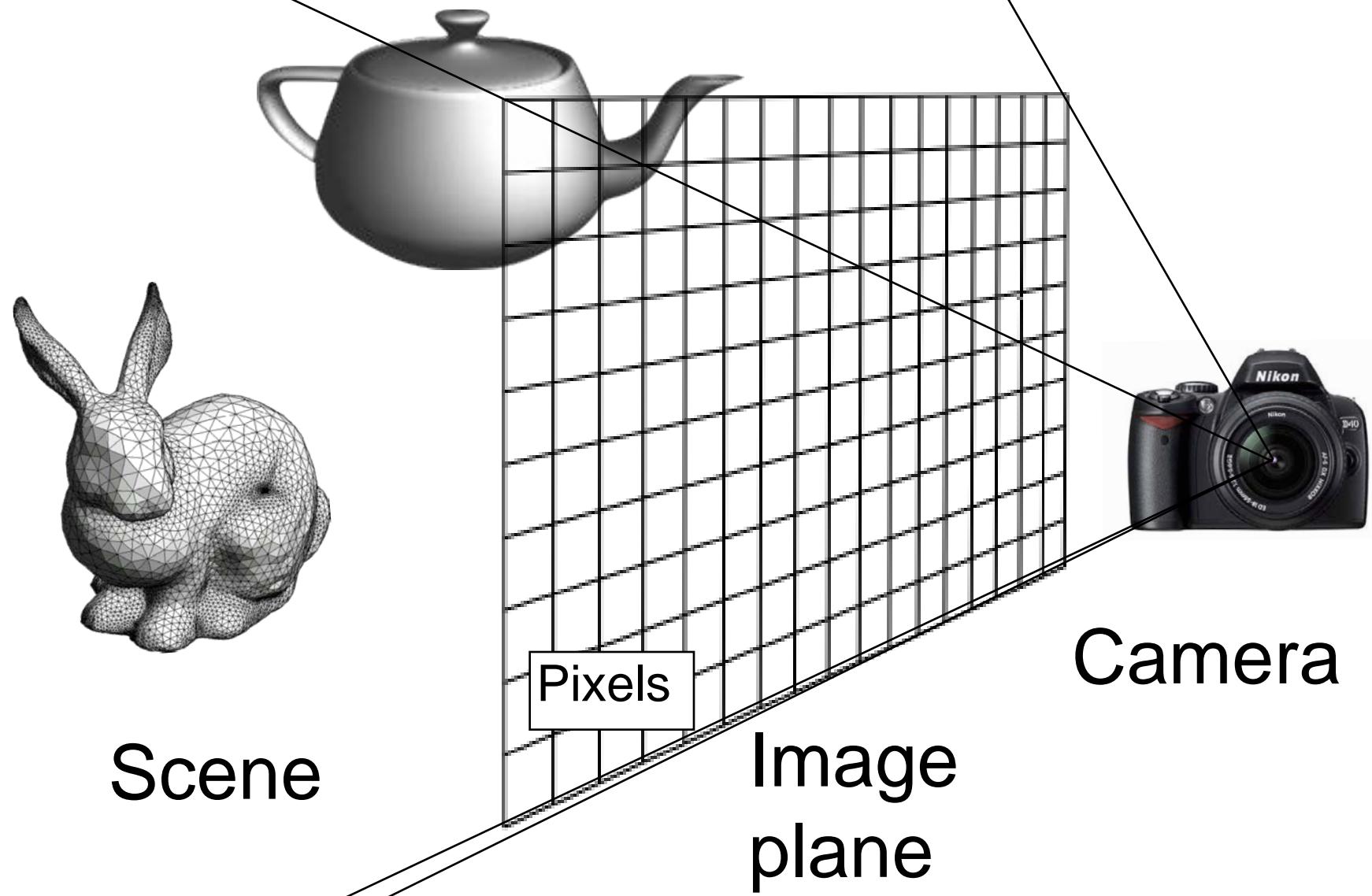
Scene



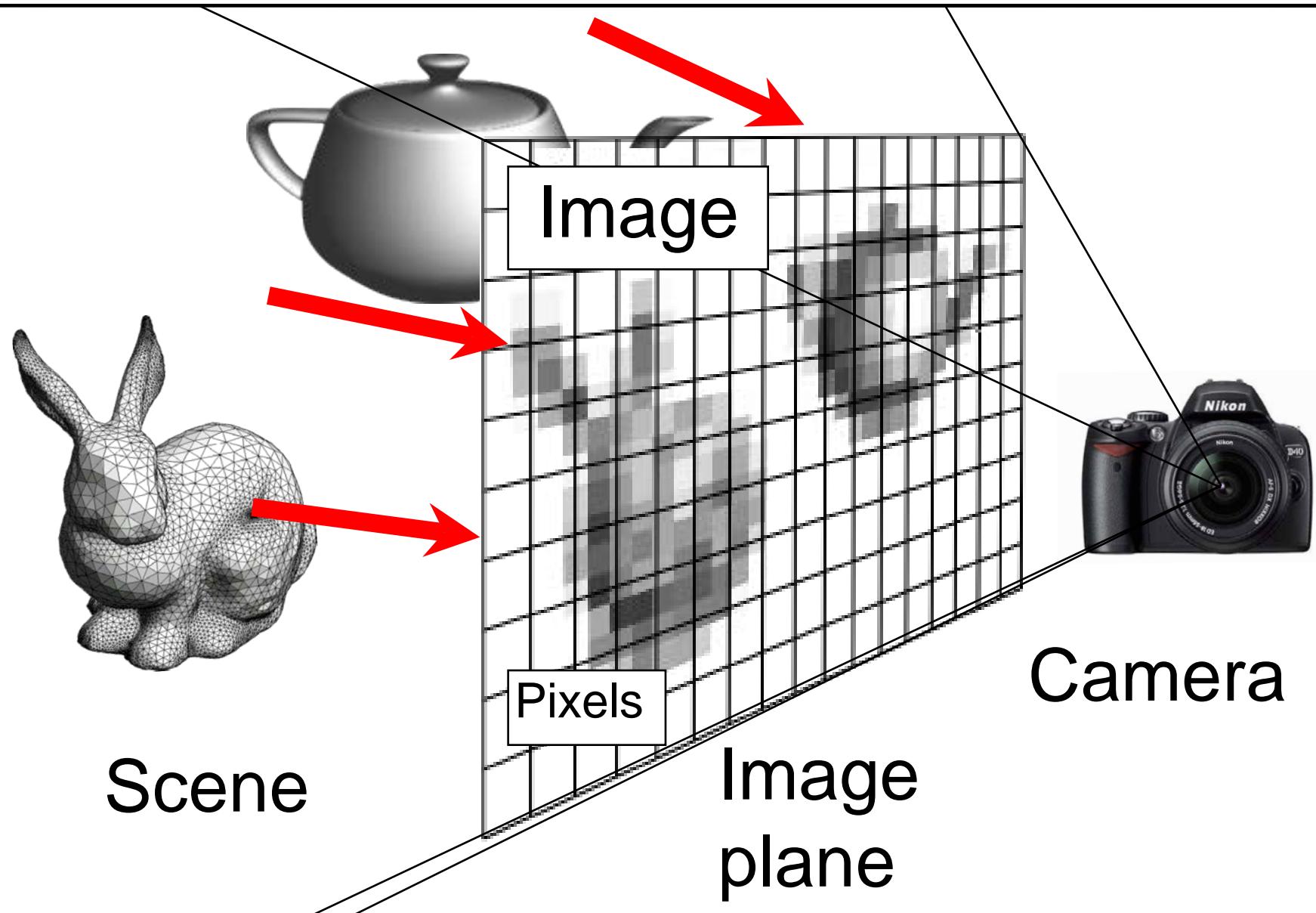


Scene

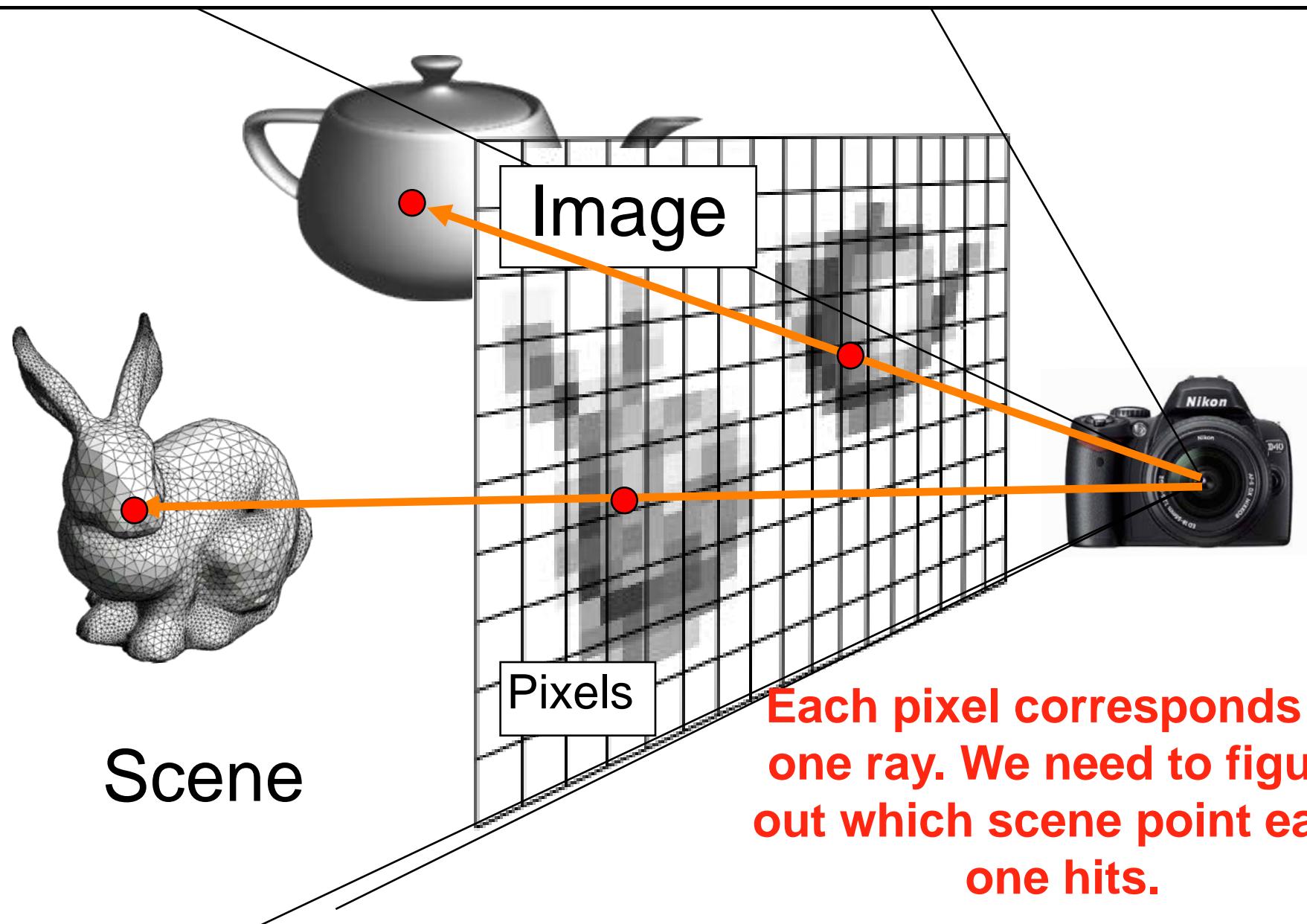
Camera



# Rendering = Scene to Image

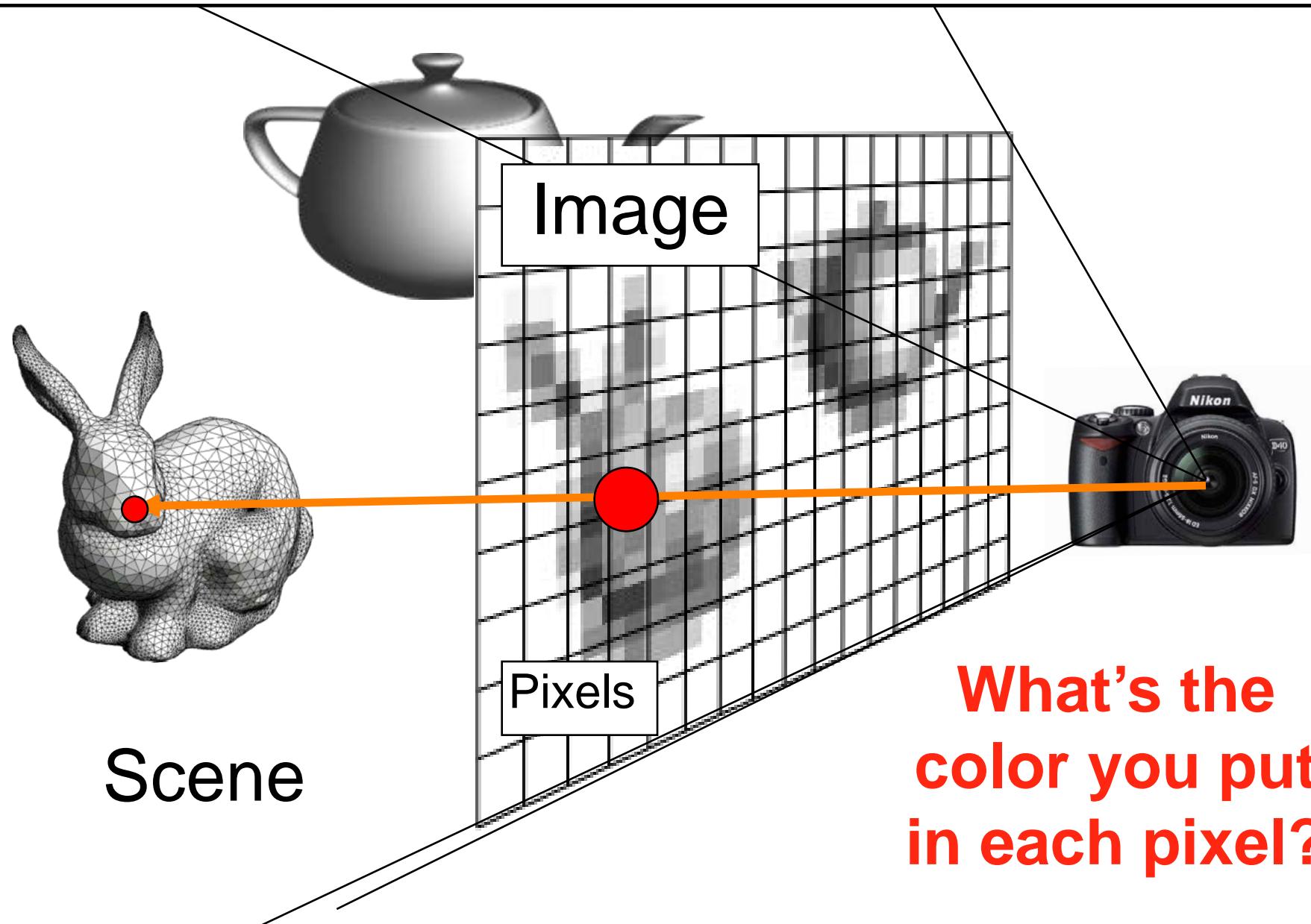


# Rendering – Pinhole Camera



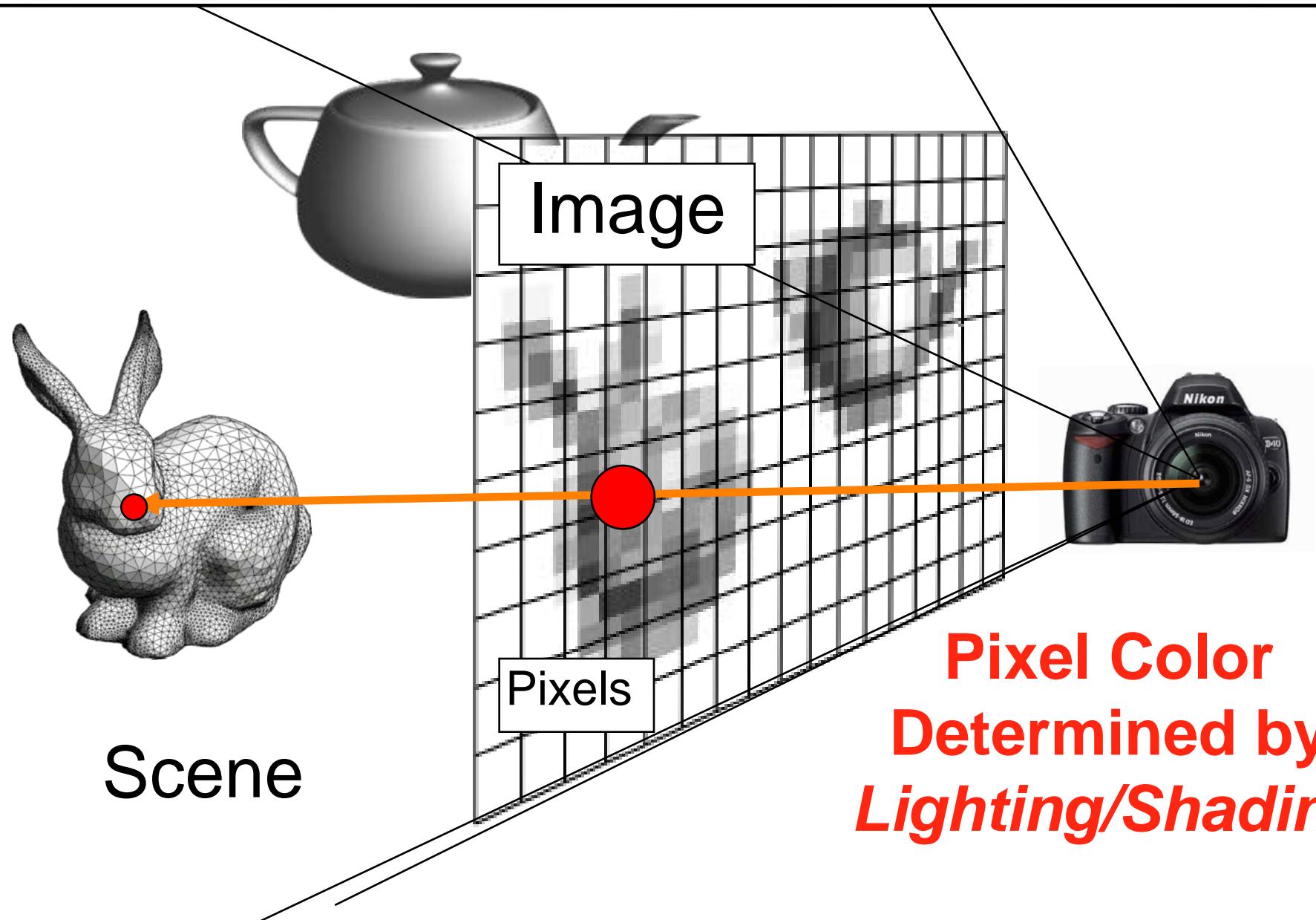
Each pixel corresponds to one ray. We need to figure out which scene point each one hits.

# Rendering



What's the  
color you put  
in each pixel?

# Rendering



# Rendering

---

- “Rendering” refers to the entire process that produces color values for pixels, given a 3D representation of the scene
- Pixels correspond to rays; need to figure out the **visible** scene point along each ray
  - Called “hidden surface problem” in older texts
  - “Visibility” is a more modern term
  - Also, we assume (for now) a single ray per pixel

# Rendering

---

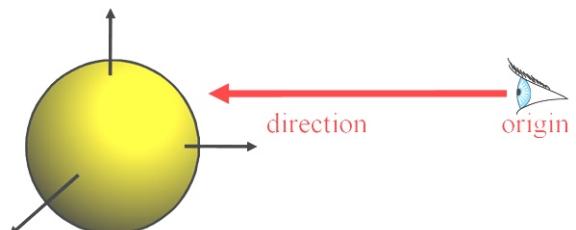
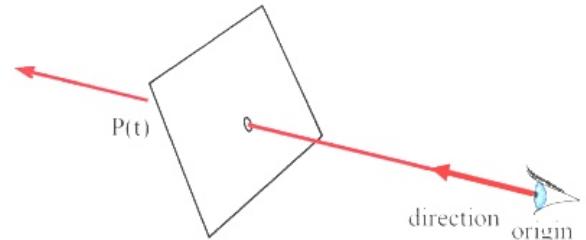
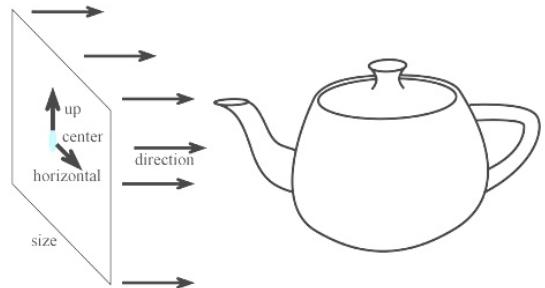
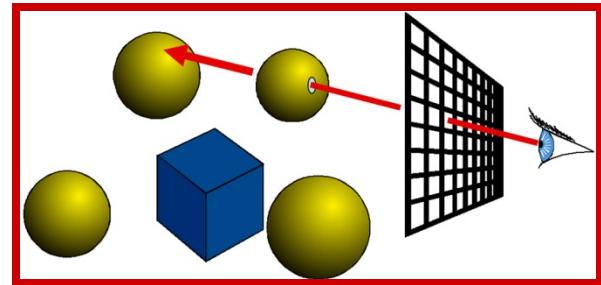
- “Rendering” refers to the entire process that produces color values for pixels
- Pixels correspond to rays; need to figure out the **visible** scene point along each ray
  - Called “hidden surface problem” in older texts
  - “Visibility” is a more modern term
  - Also, we assume (for now) a single ray per pixel
- Major algorithms: **Ray casting and rasterization**
- Note: We are assuming a pinhole camera (for now)

# Questions?

---

# Ray Casting

- Ray Casting Basics
- Camera and Ray Generation
- Ray-Plane Intersection
- Ray-Sphere Intersection



# Ray Casting

---

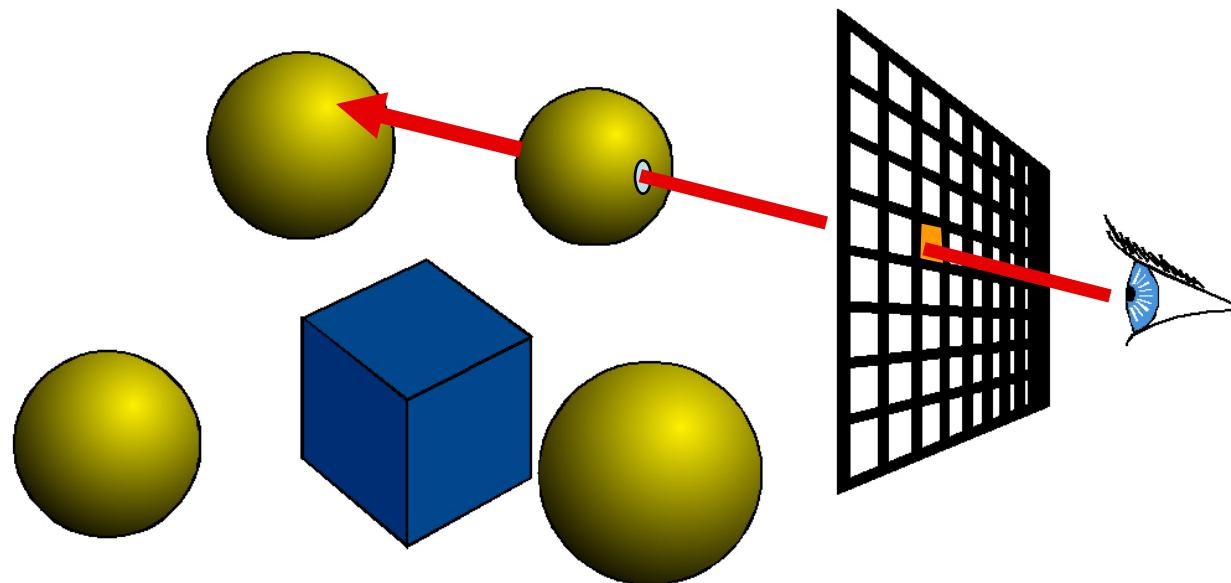
For every pixel

Construct a ray from the eye

For every object in the scene

Find intersection with the ray

Keep if closest



# Shading

---

For every pixel

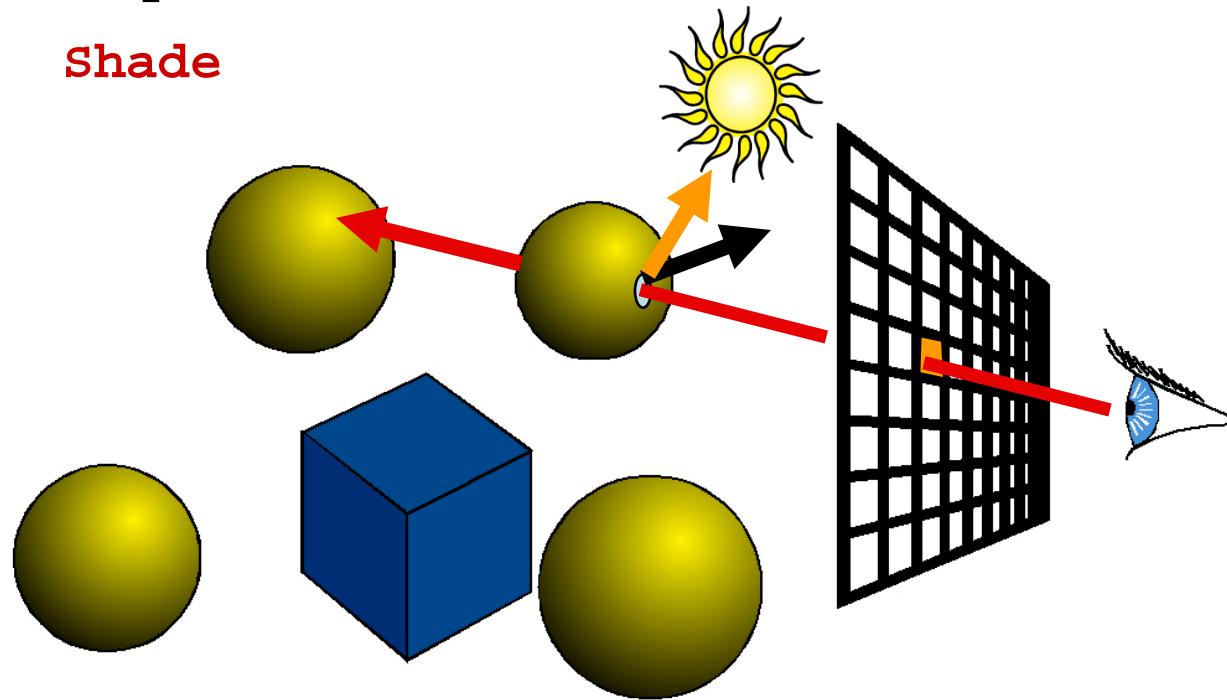
Construct a ray from the eye

For every object in the scene

Find intersection with the ray

Keep if closest

Shade



# Shading = What Surfaces Look Like

- Surface/Scene Properties

- surface normal
  - direction to light
  - viewpoint

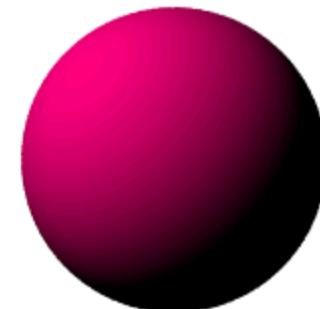
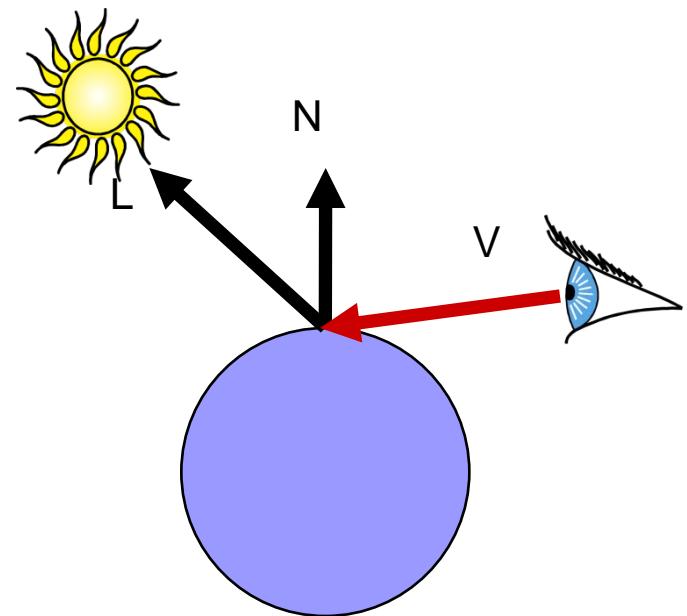
- Material Properties

- Diffuse (matte)
  - Specular (shiny)
  - ...

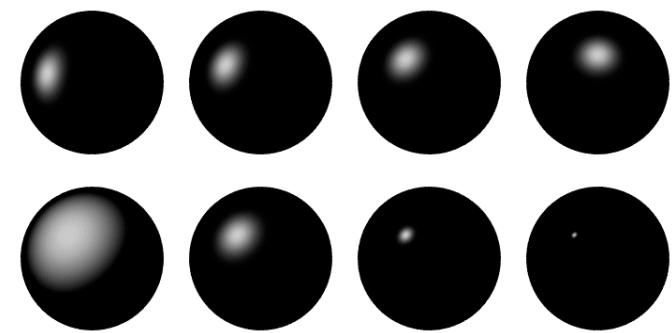
- Light properties

- Position
  - Intensity, ...

- Much more!



*Diffuse sphere*

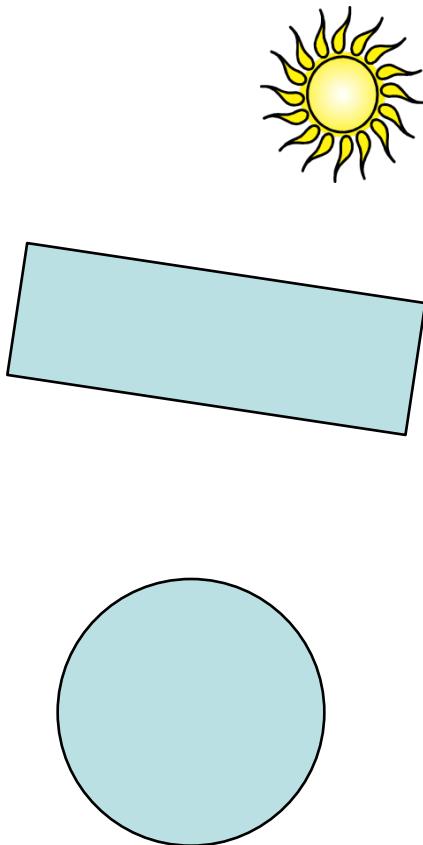


*Specular spheres*

# Ray Casting vs. Ray Tracing

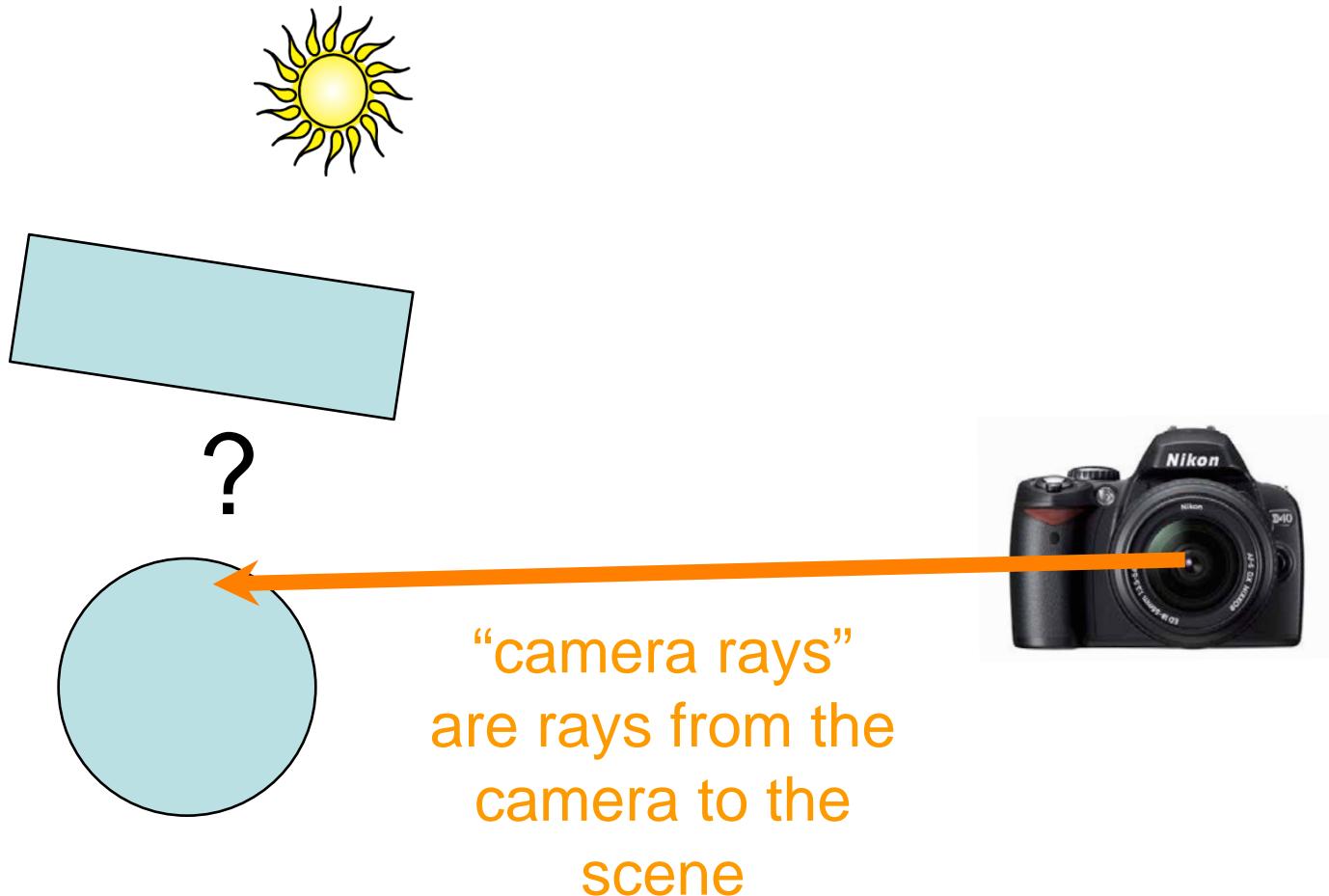
---

- Let's think about shadows...



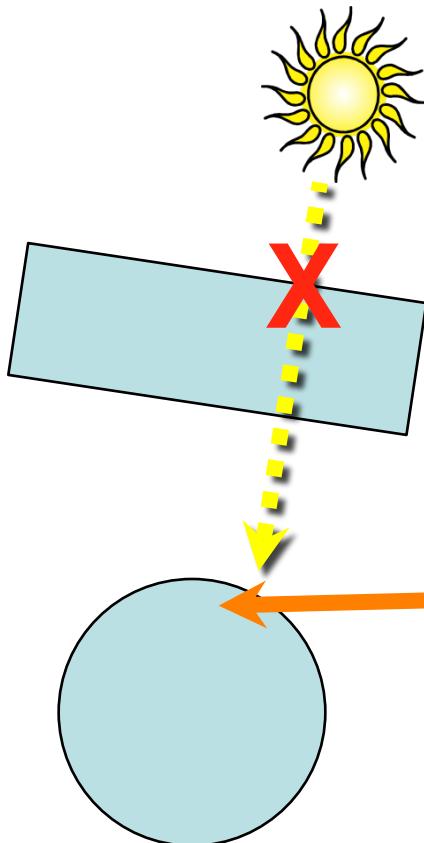
# Ray Casting vs. Ray Tracing

---



# Ray Casting vs. Ray Tracing

---



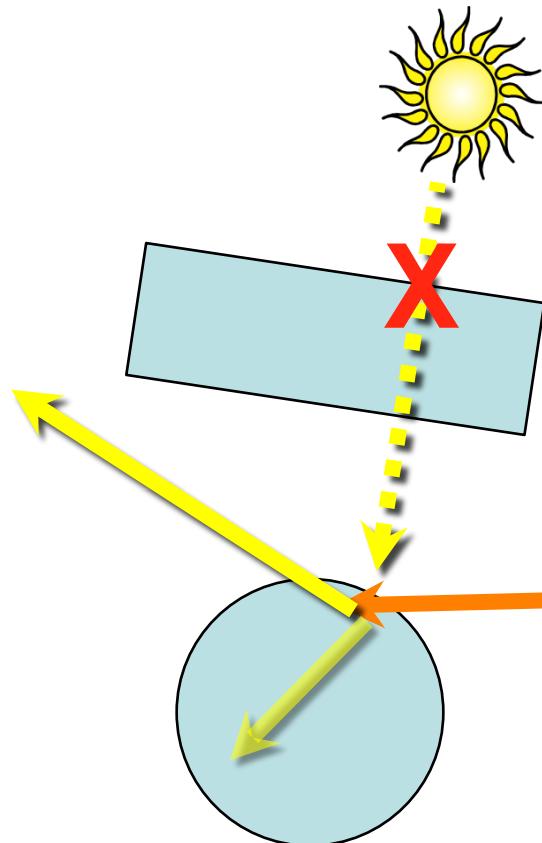
ray from light to hit  
point is blocked, i.e.,  
**point is in shadow**



# Ray Casting vs. Ray Tracing

---

- Ray casting = eye rays only, tracing = also secondary



**Secondary rays are used for testing shadows, doing reflections, refractions, etc.**



**We'll do all this a little later!**

# Secondary Rays

Indirect illumination

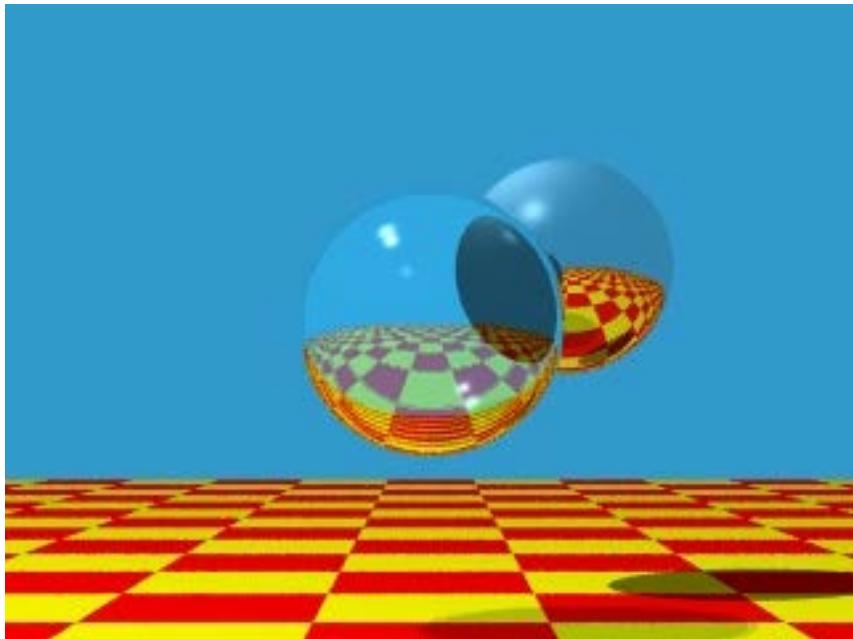
Reflections

Refractions

Shadows

Caustics

# Ray Tracing



Reflections, refractions



Caustics

HENRIK WANN JENSEN 2000

# Questions?

---

# Ray Casting

---

For every pixel

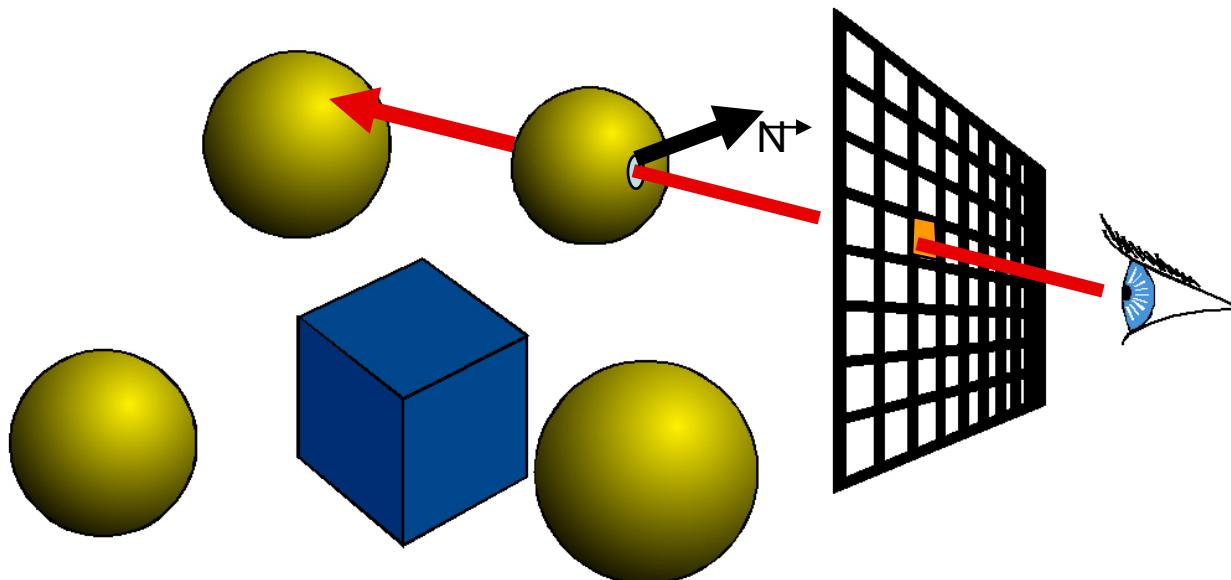
Construct a ray from the eye

For every object in the scene

**Find intersection with the ray**

Keep if closest

Shade depending on light and **normal** vector



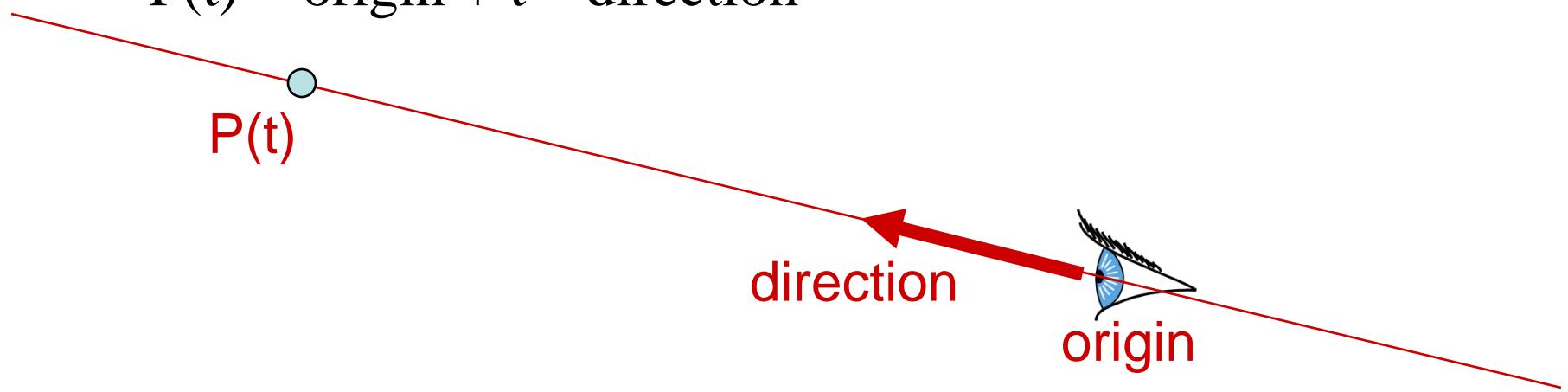
**Finding the  
intersection point  
and **normal** is the  
central part of ray  
casting**

# Ray Representation

---

- Origin – Point
- Direction – Vector
  - normalized is better
- Parametric line
  - $P(t) = \text{origin} + t * \text{direction}$

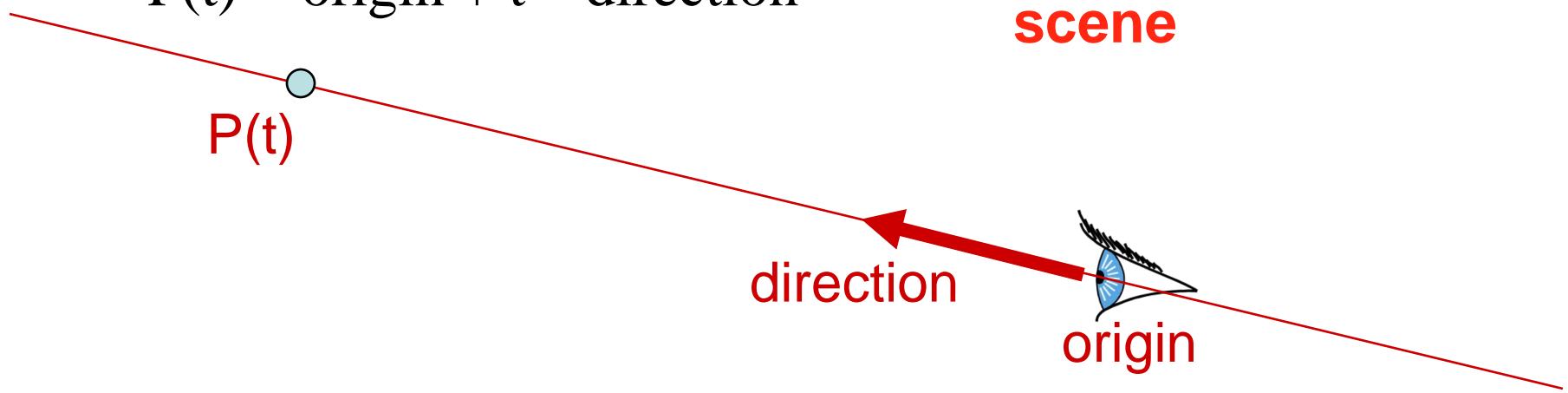
**How would you represent a ray?**



# Ray Representation

---

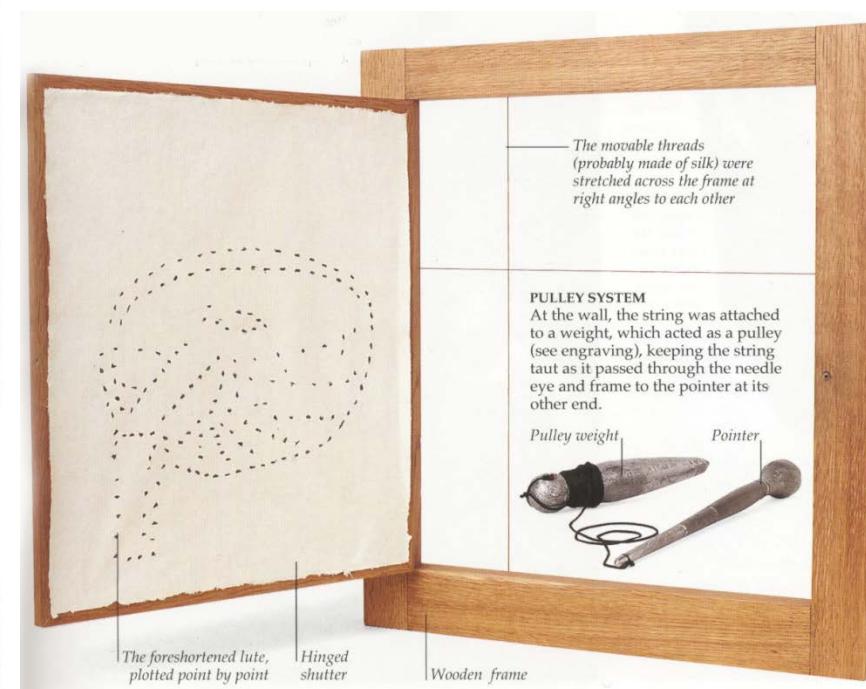
- Origin – Point
- Direction – Vector
  - normalized is better
- Parametric line
  - $P(t) = \text{origin} + t * \text{direction}$



**Another way to put  
the ray casting  
problem statement:  
Find smallest  $t > 0$   
such that  $P(t)$  lies  
on a surface in the  
scene**

# Dürer's Ray Casting Machine

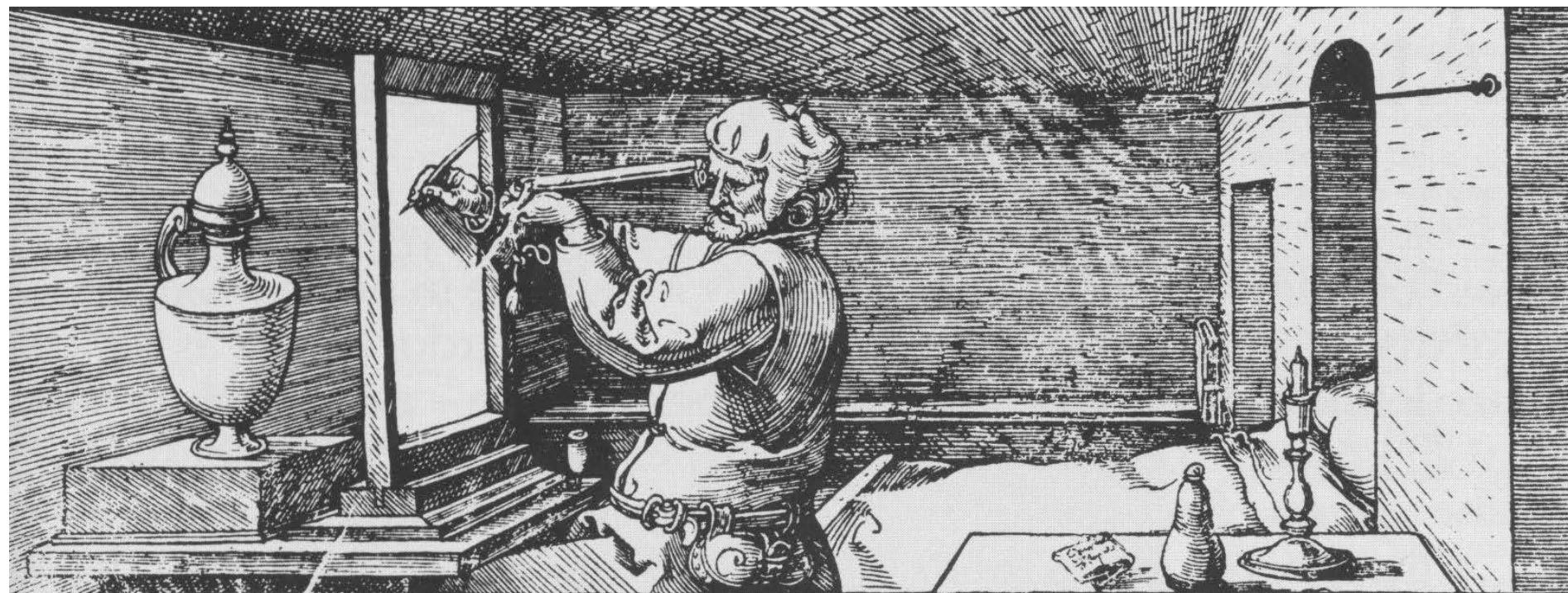
- Albrecht Dürer, 16<sup>th</sup> century



# Dürer's Ray Casting Machine

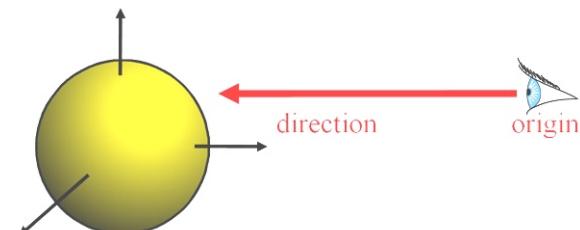
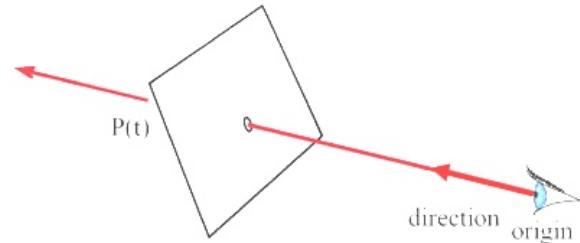
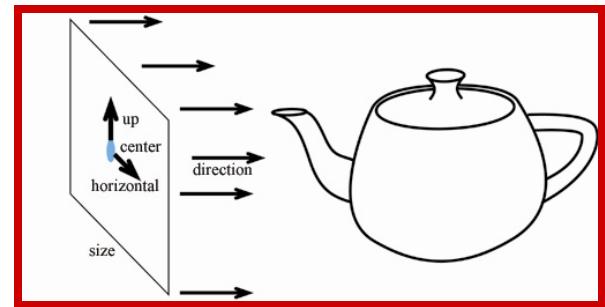
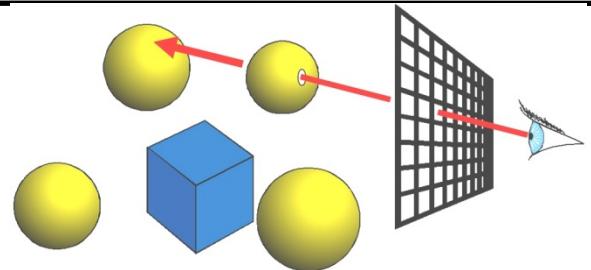
---

- Albrecht Dürer, 16<sup>th</sup> century



# Ray Casting

- Ray Casting Basics
- Camera and Ray Generation
- Ray-Plane Intersection
- Ray-Sphere Intersection



# Cameras

---

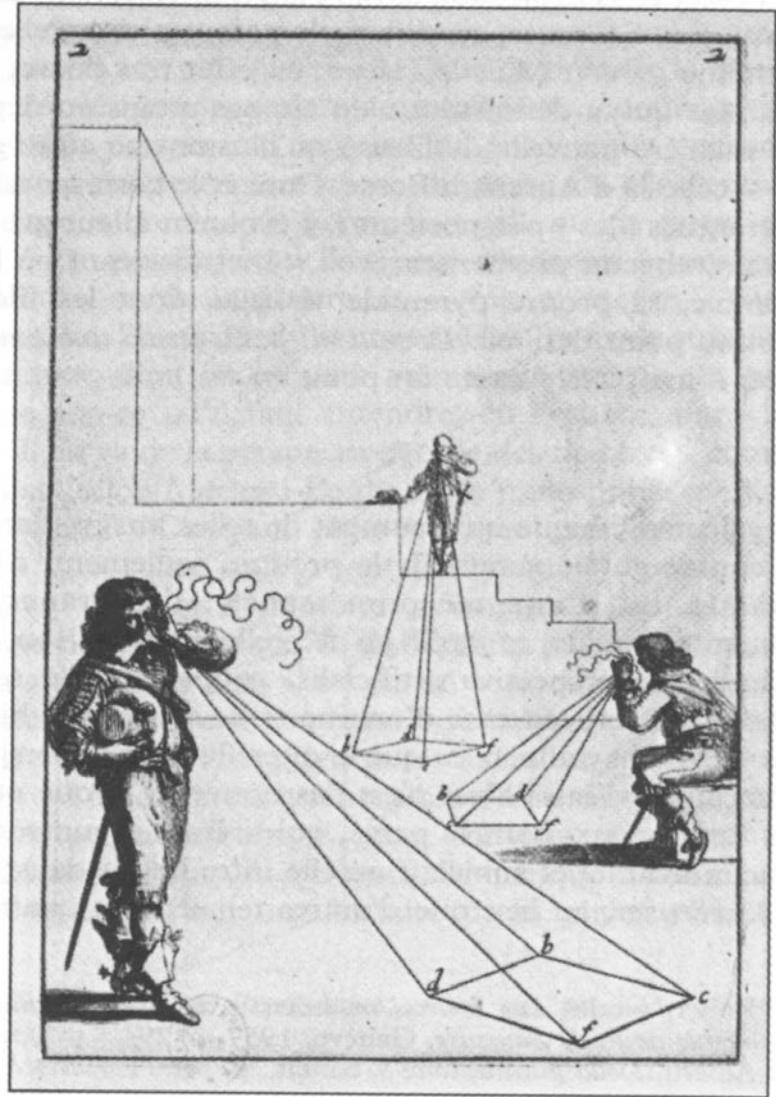
For every pixel

Construct a ray from the eye

For every object in the scene

Find intersection with ray

Keep if closest

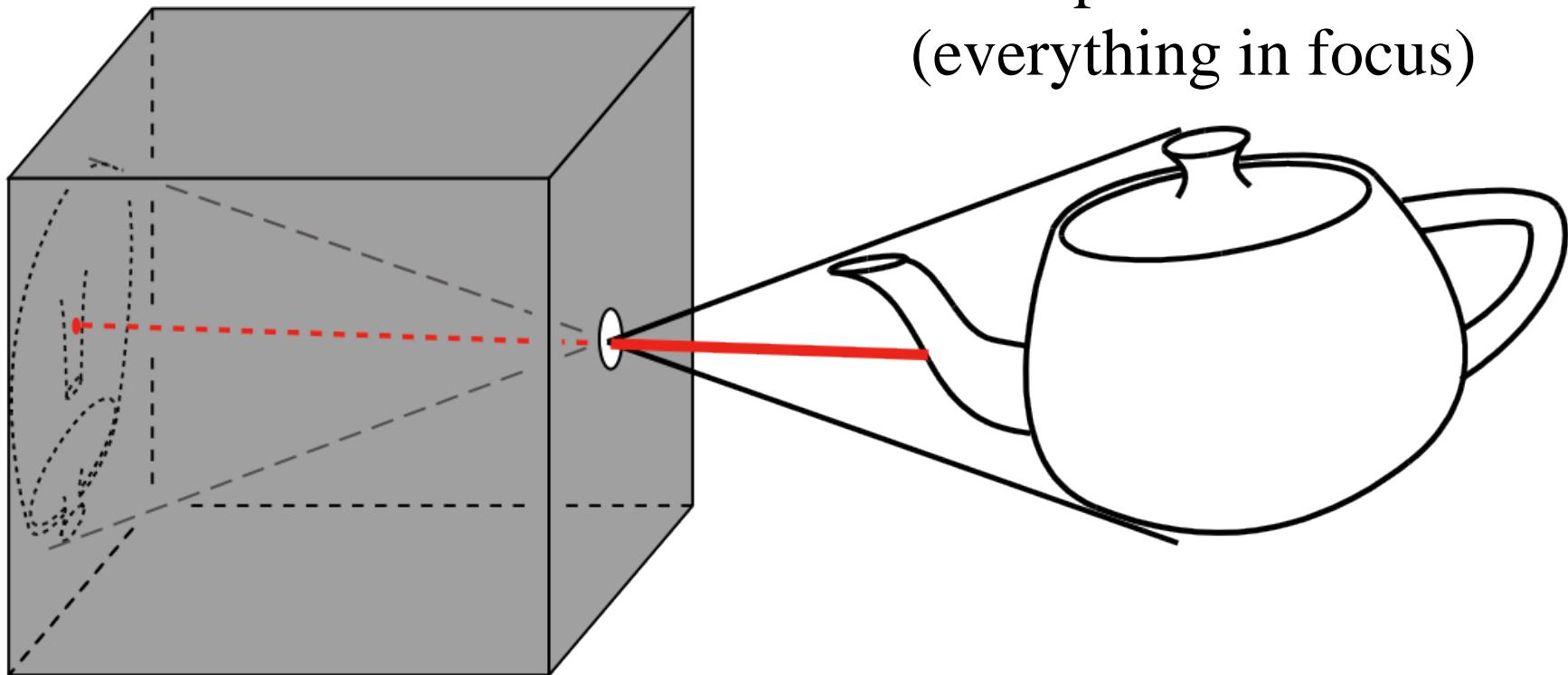


Abraham Bosse, *Les Perspecteurs*. Gravure extraite de la *Manière*

# Pinhole Camera

---

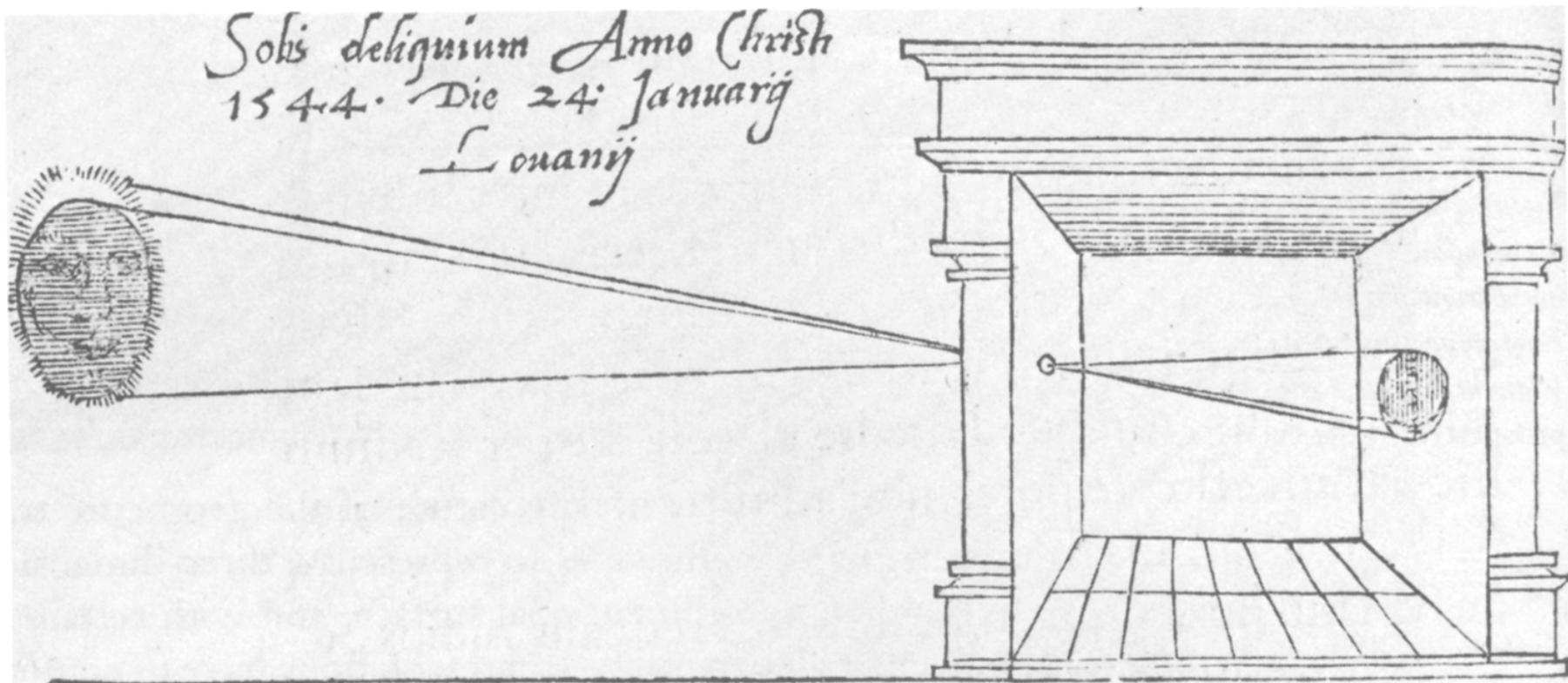
- Box with a tiny hole
- Inverted image
- Similar triangles
- Perfect image if hole infinitely small
- Pure geometric optics
- No depth of field issue (everything in focus)



# Oldest Illustration

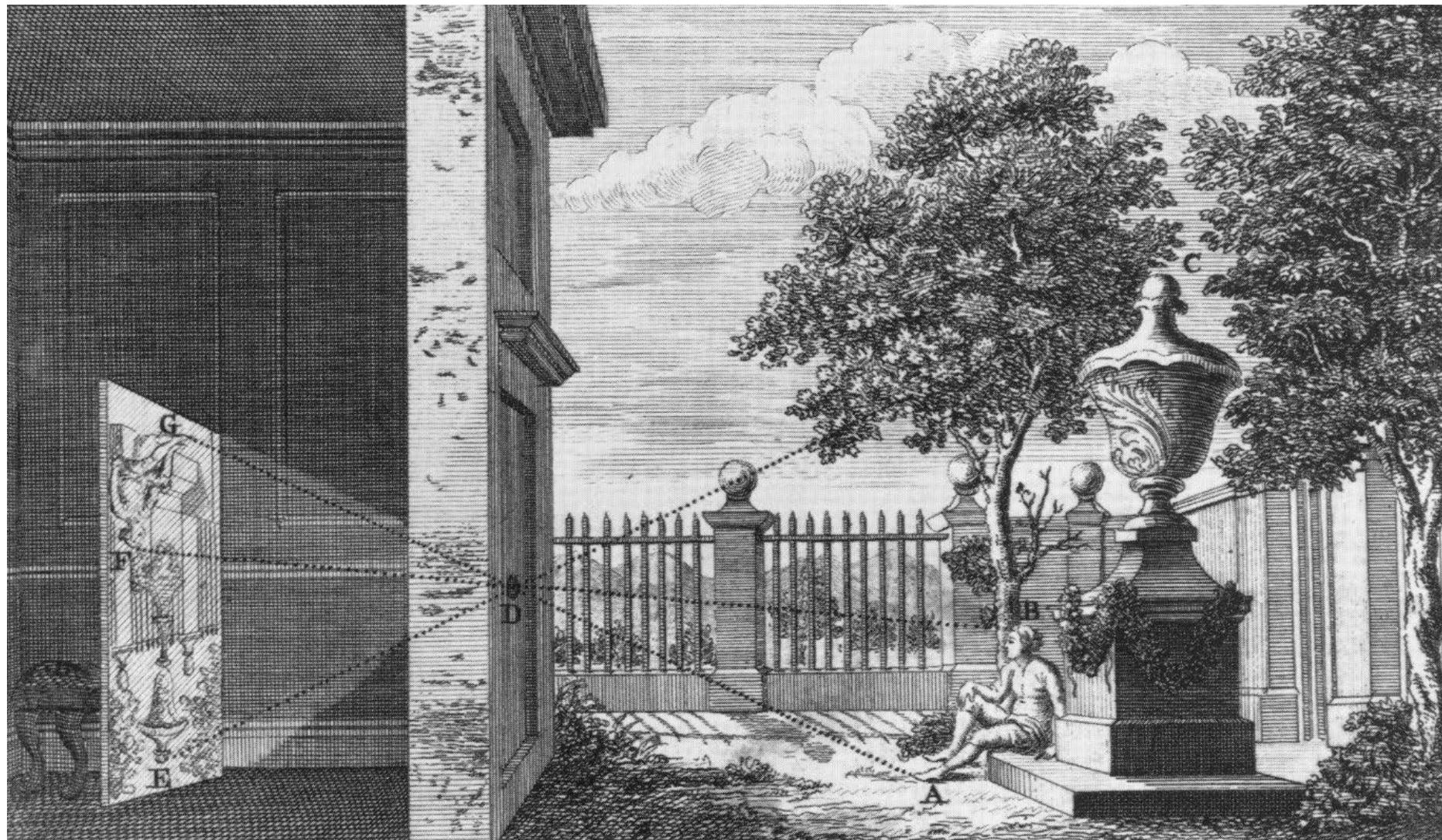
---

- From Gemma Frisius, 1545



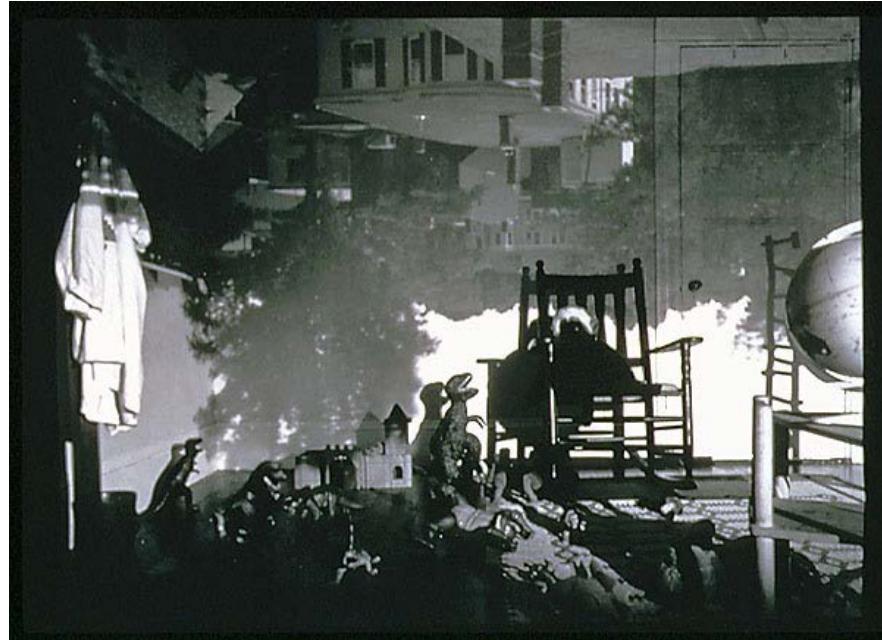
# Also Called “Camera Obscura”

---



# Camera Obscura Today

---



Abelardo Morell

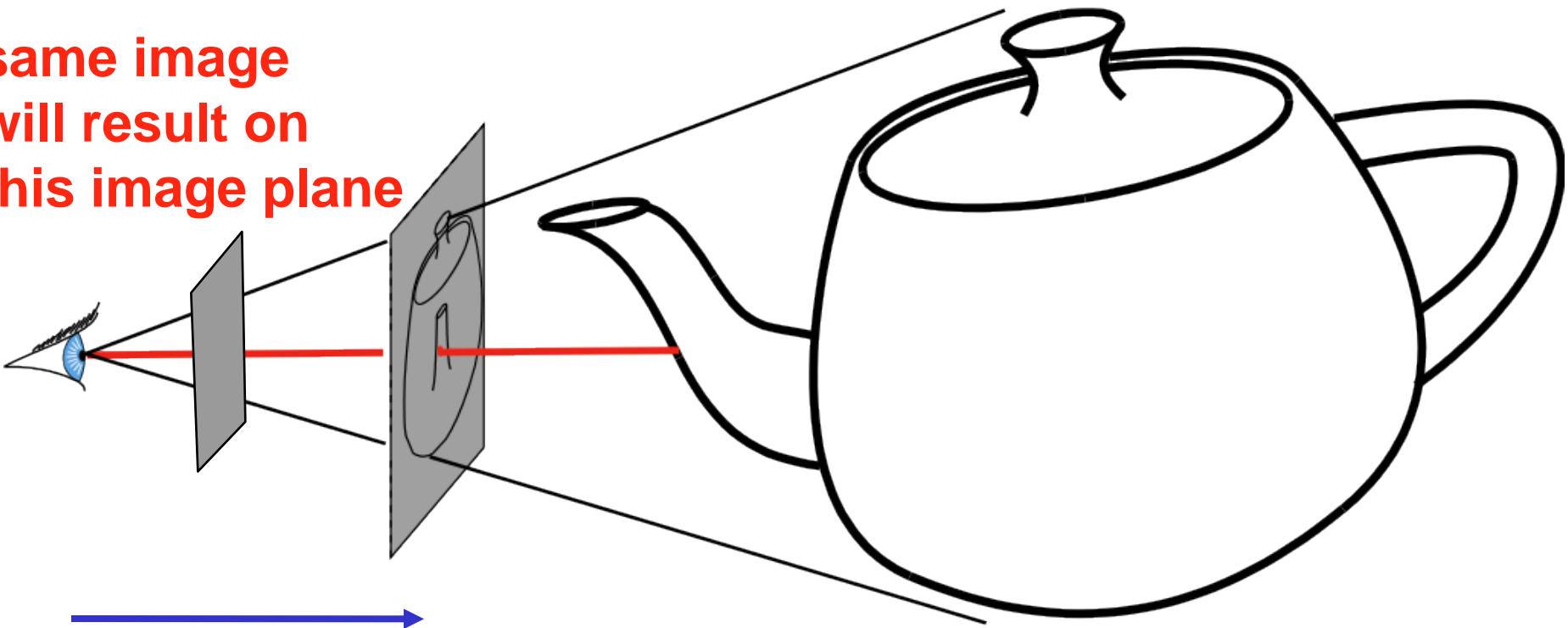
[www.abelardomorell.net](http://www.abelardomorell.net)

# Simplified Pinhole Camera

---

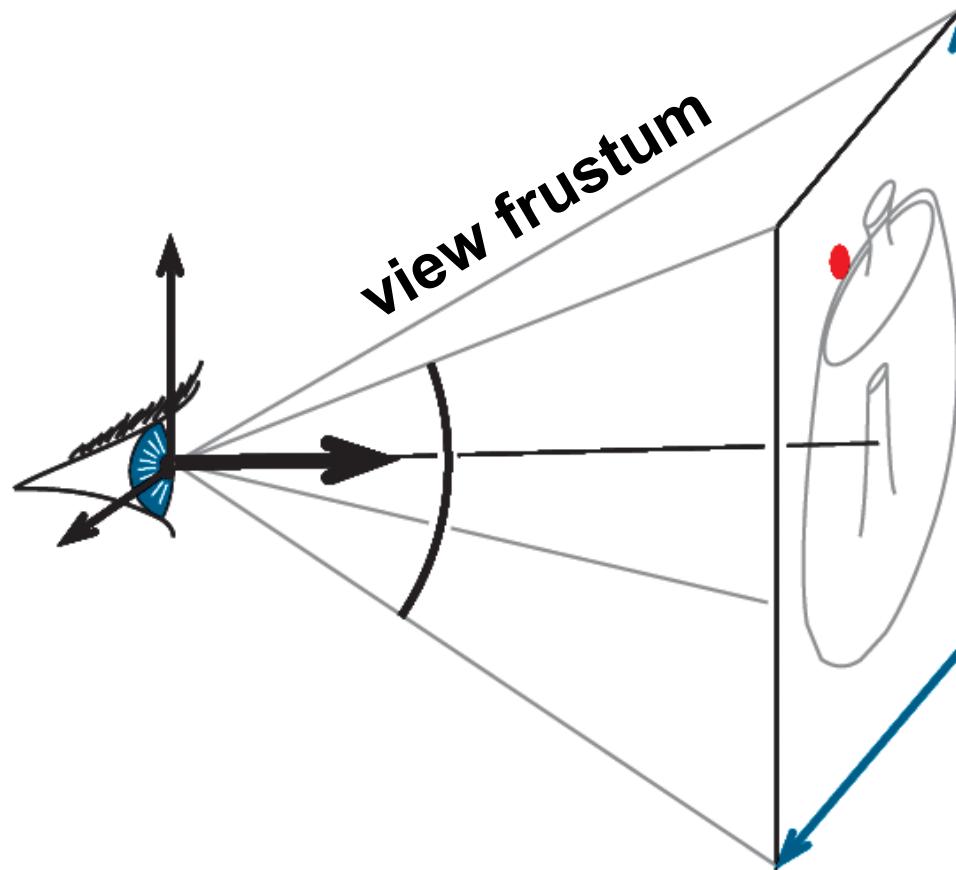
- Eye-image pyramid (view frustum)
- Note that the distance/size of image are arbitrary

same image  
will result on  
this image plane



# Camera Description?

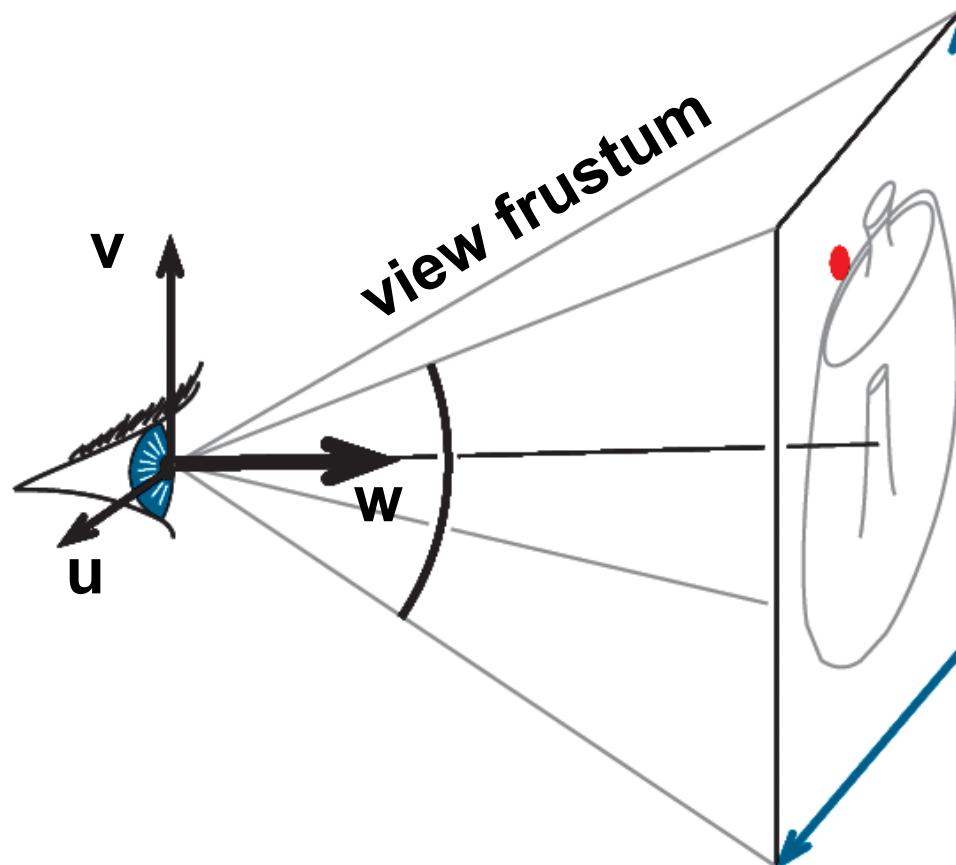
---



# Camera Description?

- Eye point  $e$  (*center*)
- Orthobasis  $u, v, w$  (*horizontal, up, direction*)

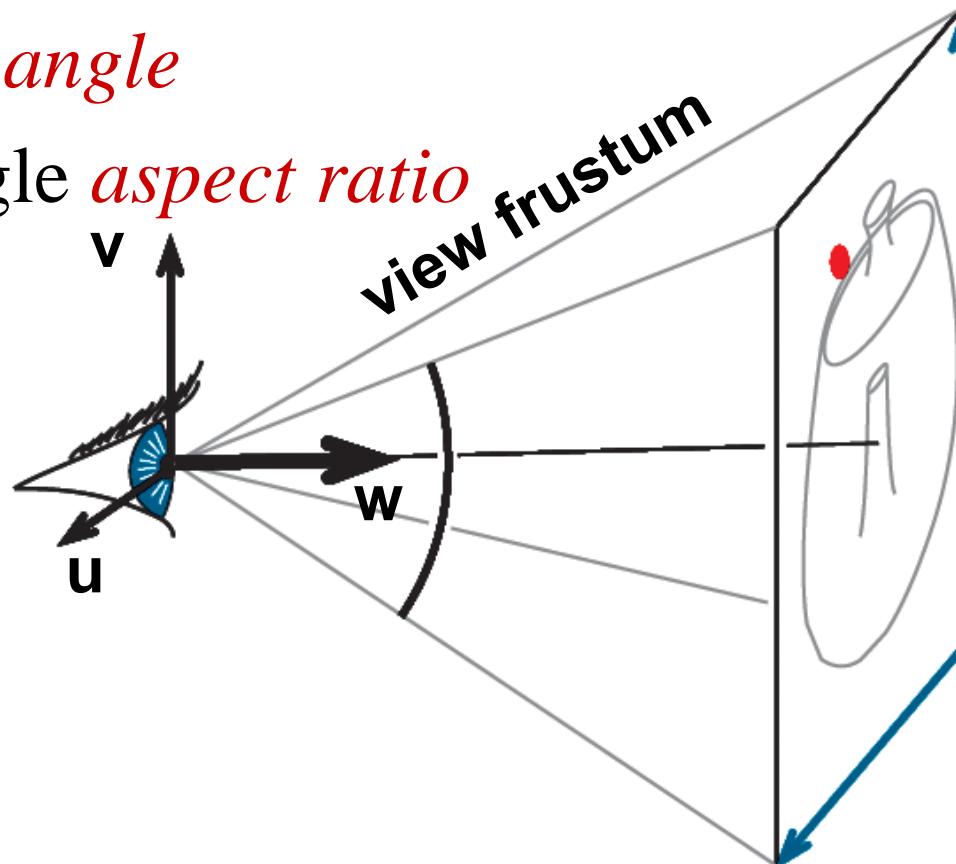
Object  
coordinates  
World  
coordinates  
**View**  
**coordinates**  
Image  
coordinates



# Camera Description?

- Eye point  $e$  (*center*)
- Orthobasis  $u, v, w$  (*horizontal, up, direction*)
- Field of view *angle*
- Image rectangle *aspect ratio*

Object  
coordinates  
World  
coordinates  
**View**  
**coordinates**  
Image  
coordinates



# Image Coordinates

$$-1 \leq y \leq 1$$

Convenient to define “normalized image coordinates” such that the **x, y coordinates run from -1 to 1 regardless of the dimensions and aspect ratio of the image rectangle.**

$$-1 \leq x \leq 1$$

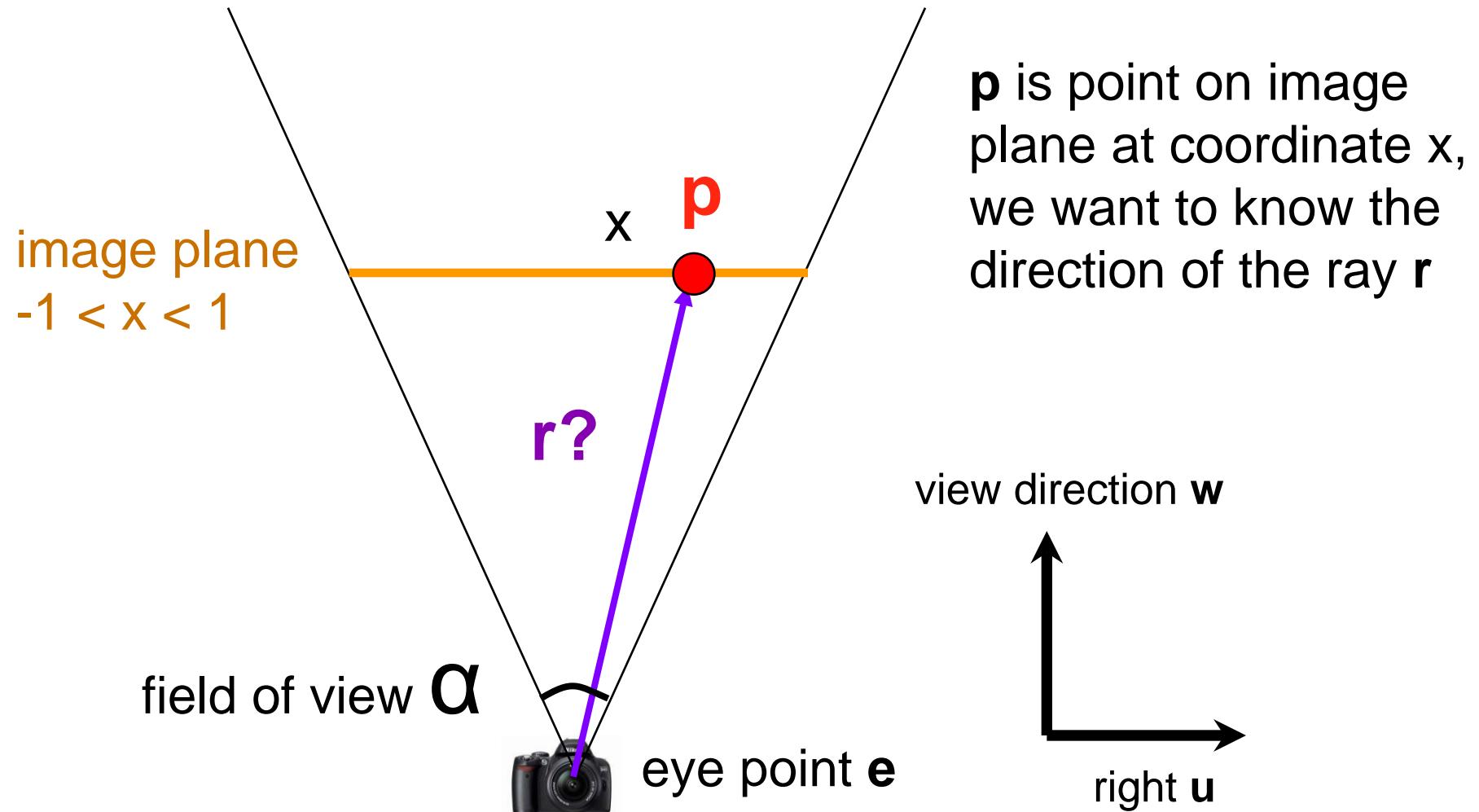
Image plane



Camera

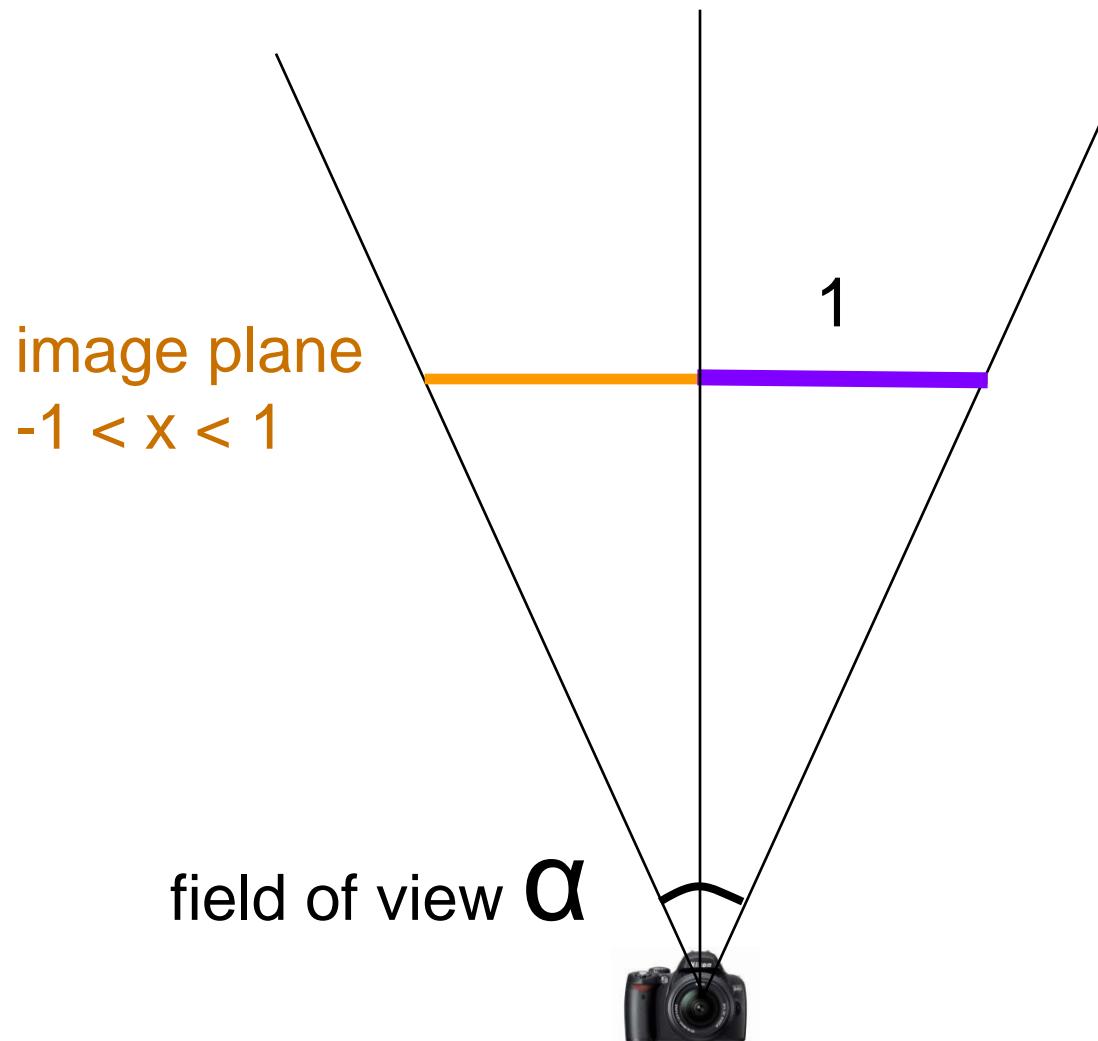
# Ray Generation in 2D

---



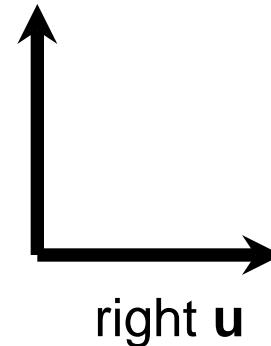
# Ray Generation in 2D

---

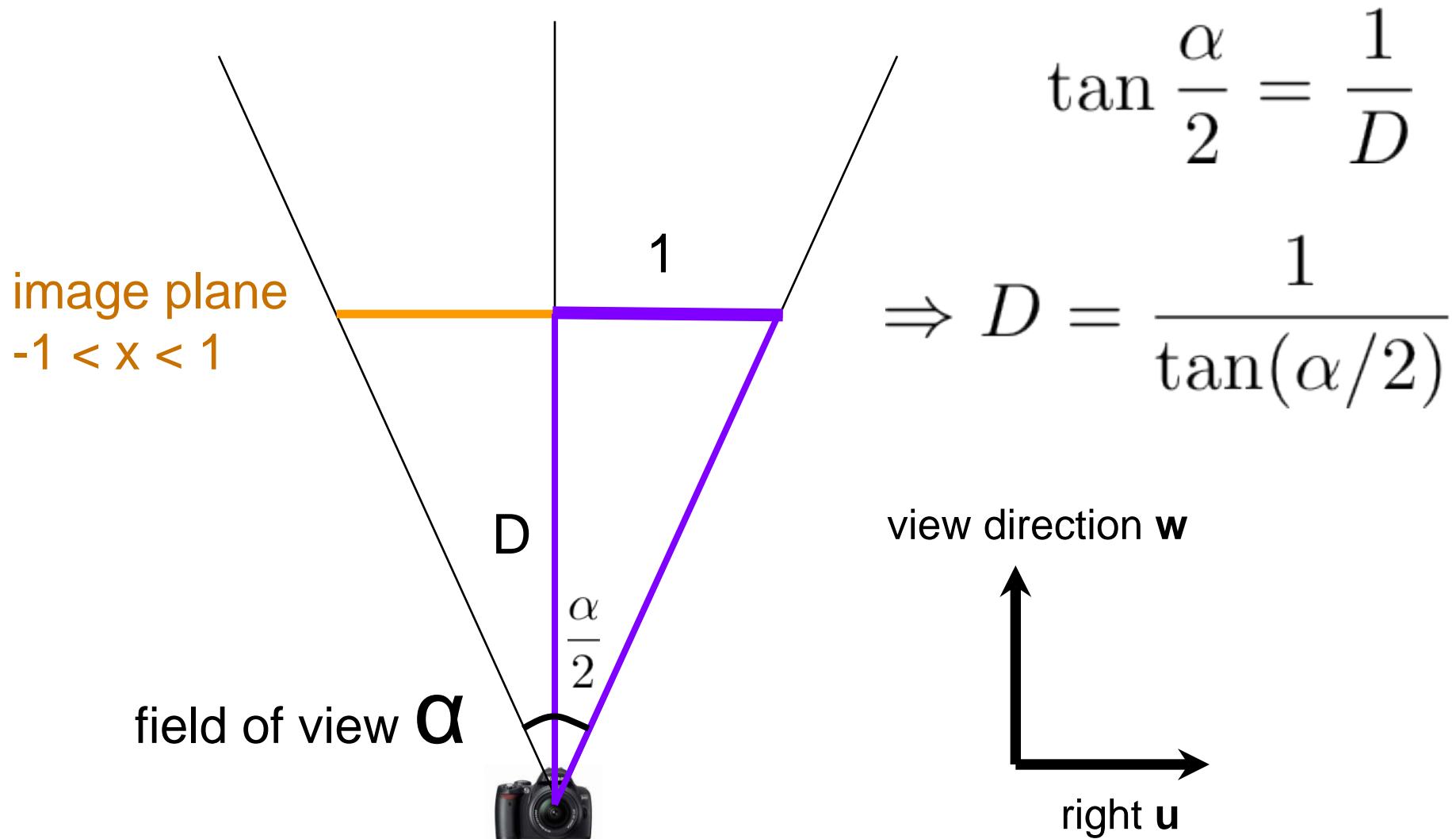


What is the distance  $D$  to the screen so that the normalized coordinates go to 1?

view direction  $w$

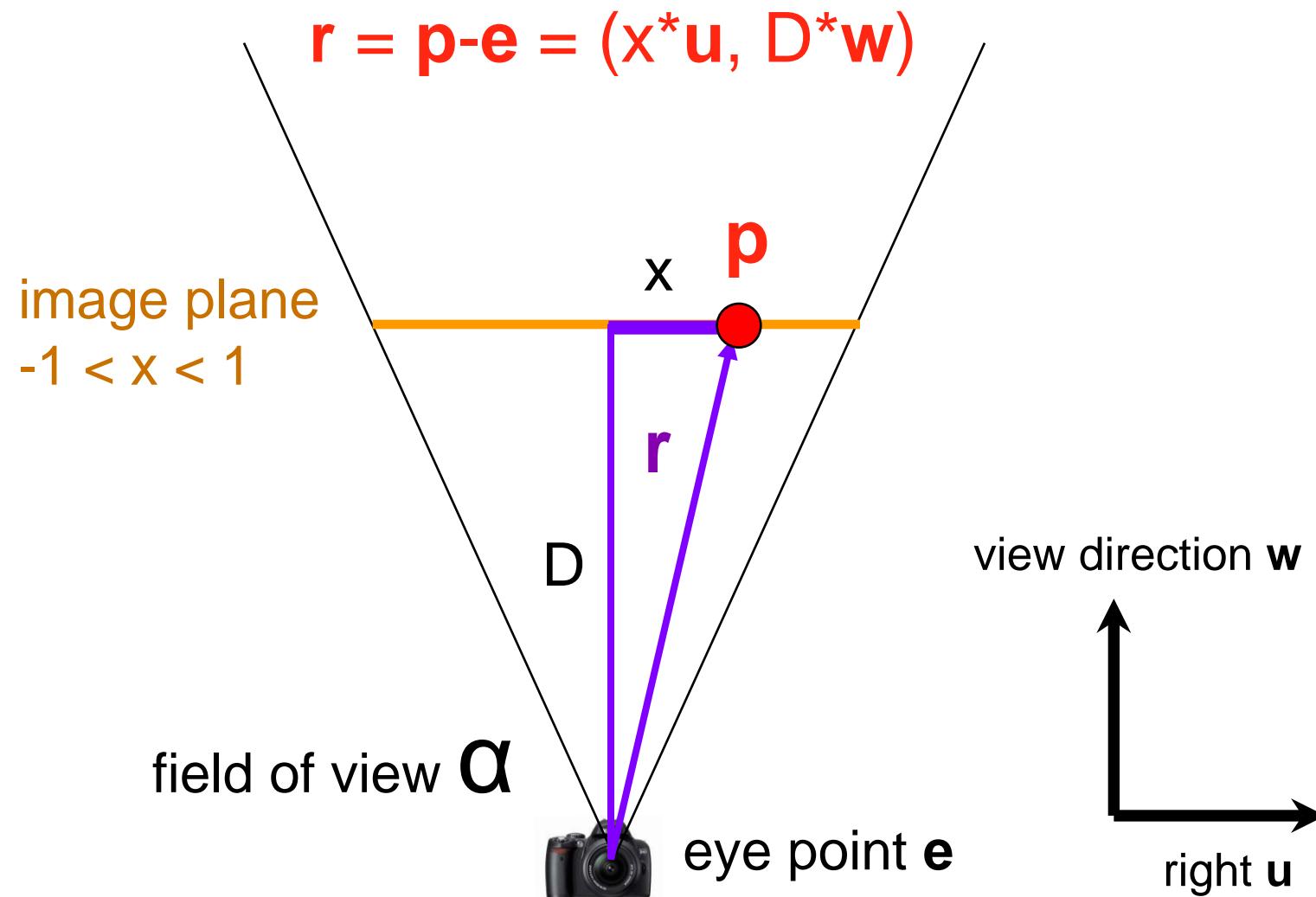


# Ray Generation in 2D

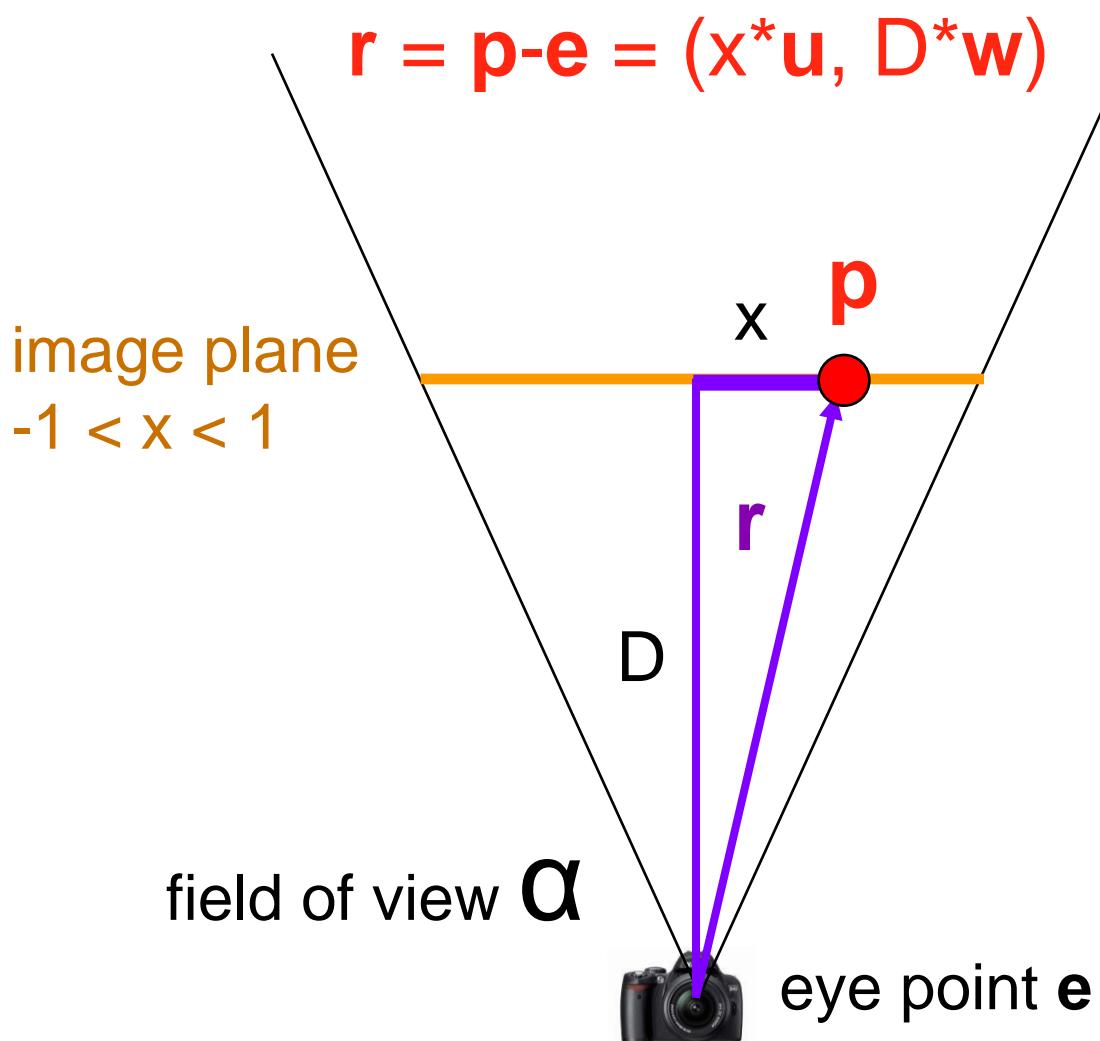


# Ray Generation in 2D

---



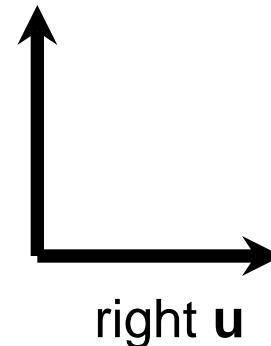
# Ray Generation in 2D



then we just normalize  $\mathbf{r}$  to get the ray

$$\mathbf{P}(t) = \mathbf{e} + t\mathbf{d}$$
$$(\mathbf{d} = \mathbf{r}/\|\mathbf{r}\|)$$

view direction  $\mathbf{w}$



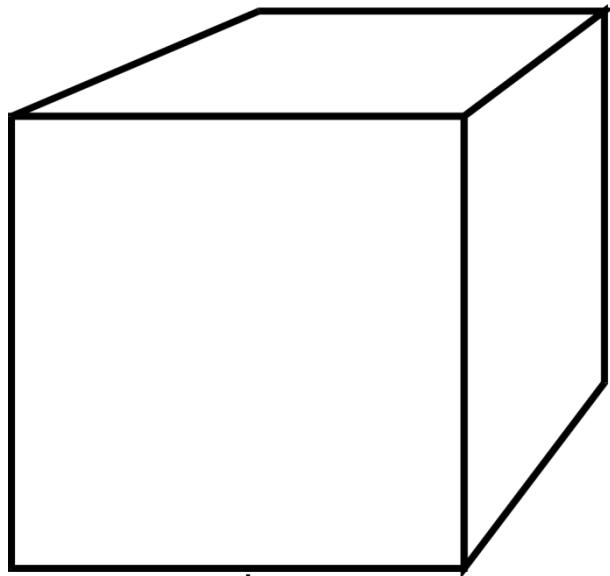
# That was 2D, 3D is just as simple

---

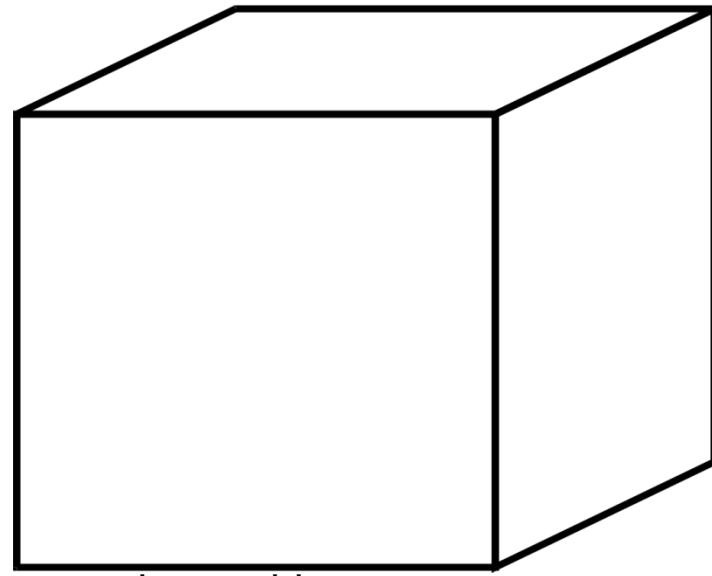
- $y$  coordinate is treated just like  $x$ , except accounting for aspect ratio
  - $\mathbf{r} = (x^*\mathbf{u}, \text{aspect}*y^*\mathbf{v}, D^*\mathbf{w})$
  - Again,  $\mathbf{u}$ ,  $\mathbf{v}$ ,  $\mathbf{w}$  are the basis vectors of the view coordinate system
  - Aspect ratio handles non-square viewports
    - Think of your 16:9 widescreen TV
- The point of the exercise with computing  $D$  was to allow us to use the  $[-1,1]$  image coordinate system regardless of field of view.

# Perspective vs. Orthographic

---



perspective

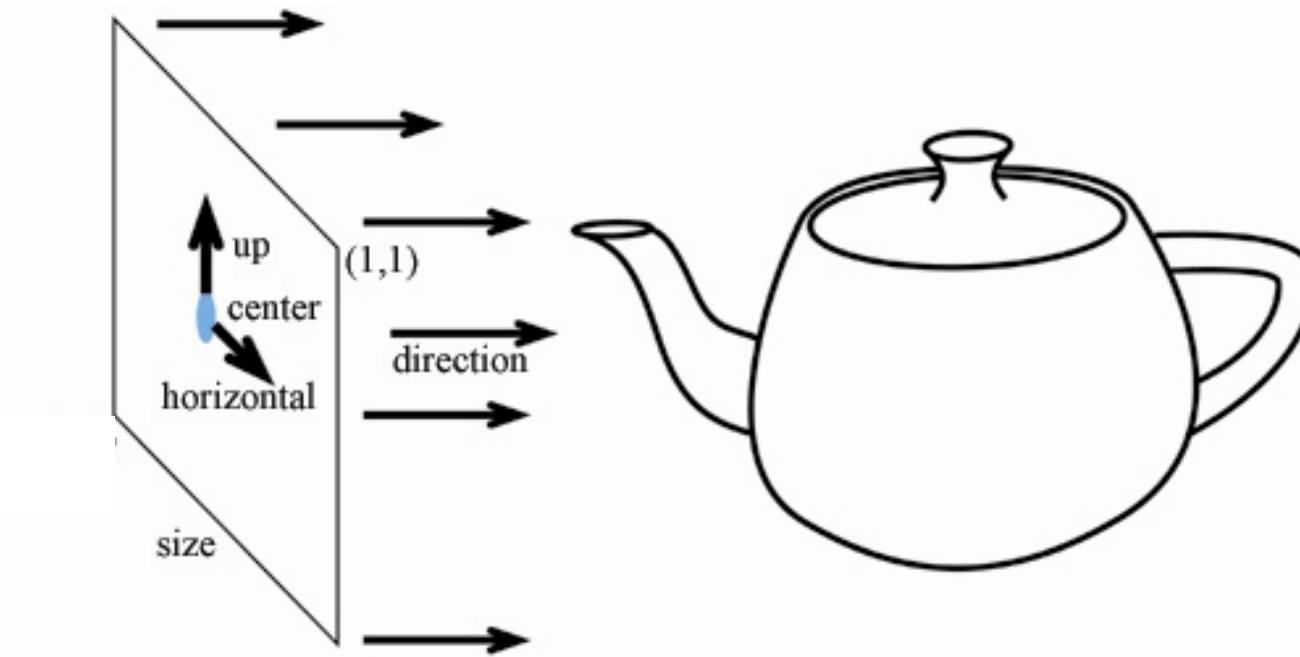


orthographic

- Parallel projection
- No foreshortening
- No vanishing point

# Orthographic Camera

---



- Ray Generation?
  - Origin =  $\mathbf{e} + x * \text{size} * \mathbf{u} + y * \text{size} * \mathbf{v}$
  - Direction is constant:  $\mathbf{w}$

# Other Weird Cameras

---

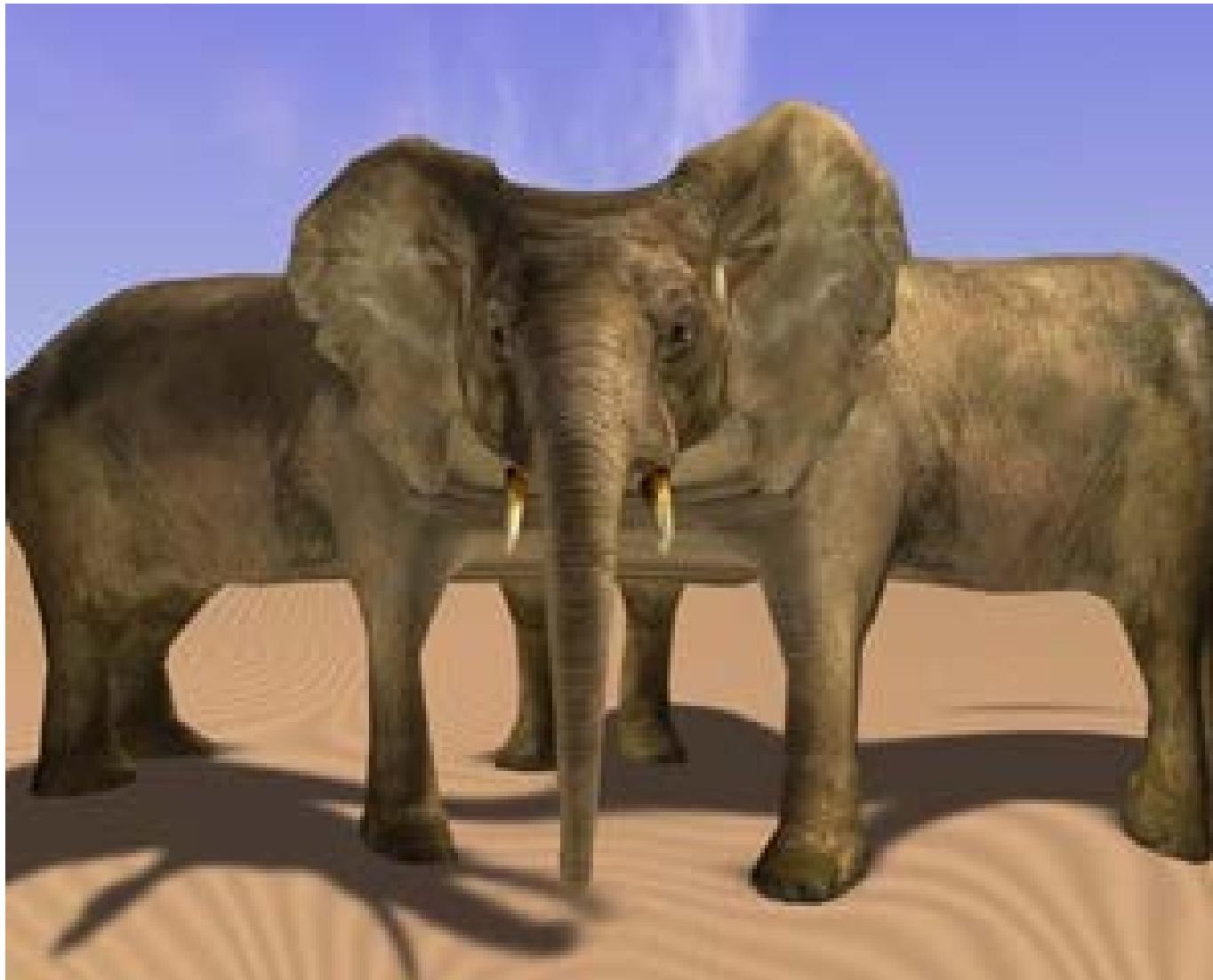
- E.g. fish eye, omnimax, parabolic



CAVE Columbia University

# Questions?

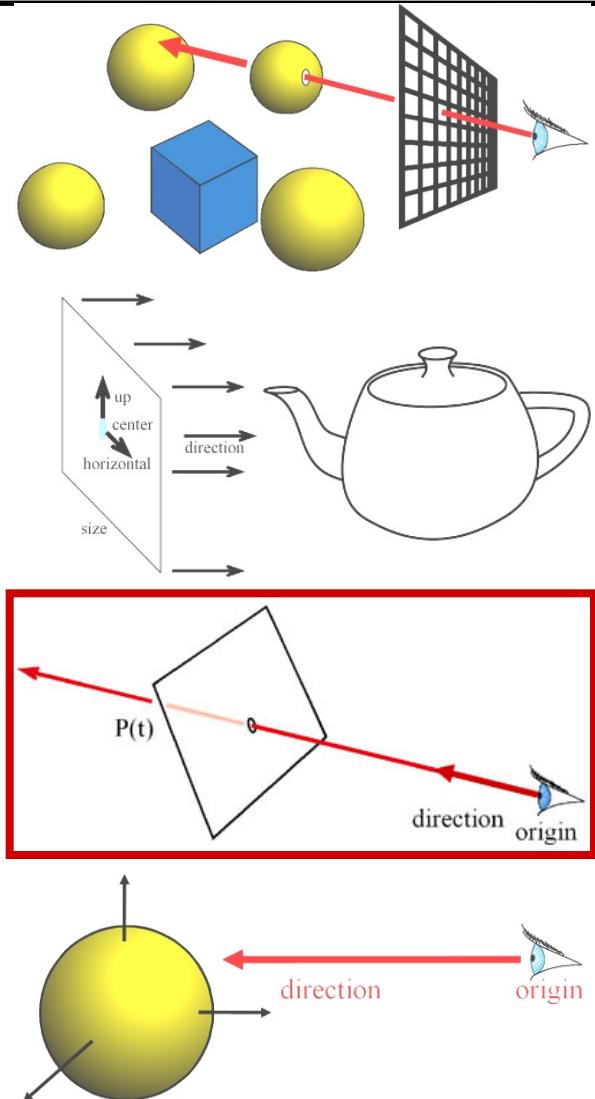
---



Even Funkier  
Multiperspective  
Imaging

# Ray Casting

- Ray Casting Basics
- Camera and Ray Generation
- Ray-Plane Intersection
- Ray-Sphere Intersection



# Ray Casting

---

For every pixel

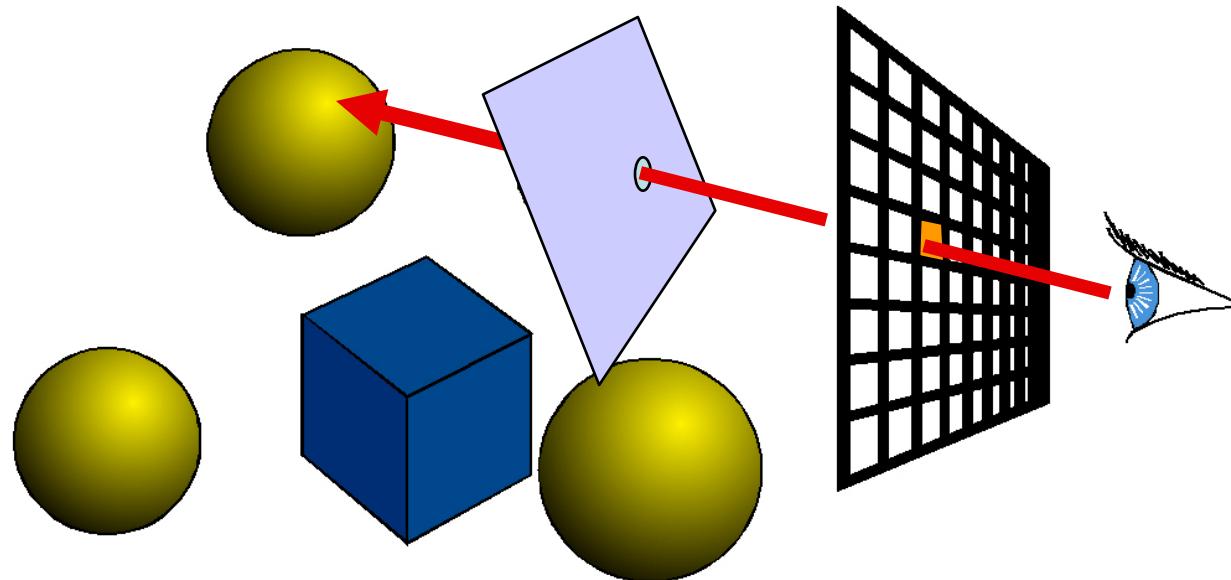
Construct a ray from the eye

For every object in the scene

**Find intersection with the ray**

Keep if closest

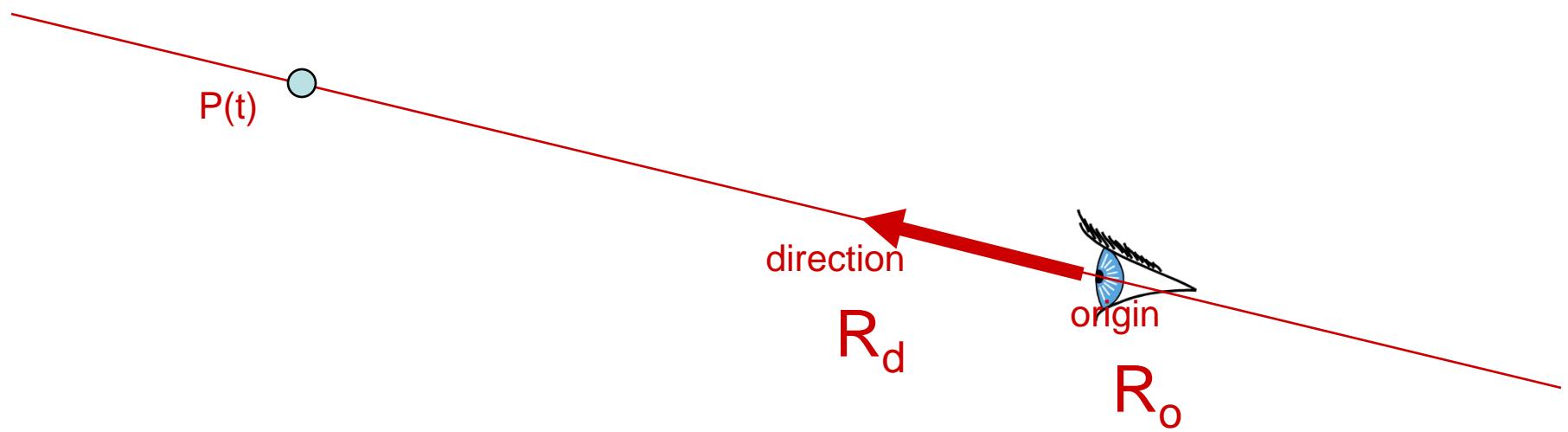
First we will study ray-plane intersection



# Recall: Ray Representation

---

- Parametric line
- $P(t) = R_o + t * R_d$
- Explicit representation



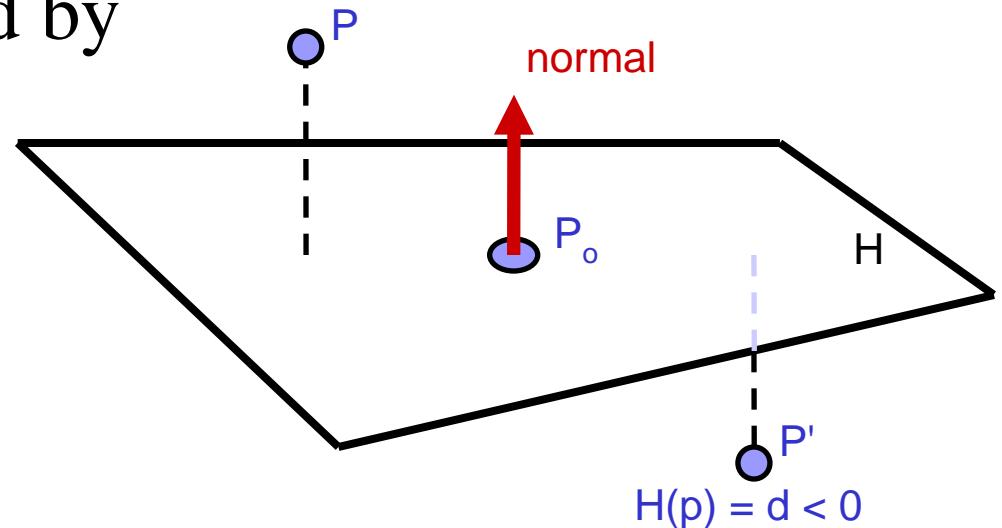
# 3D Plane Representation?

---

- (Infinite) plane defined by

- $P_o = (x_0, y_0, z_0)$
- $n = (A, B, C)$

$$H(p) = d > 0$$



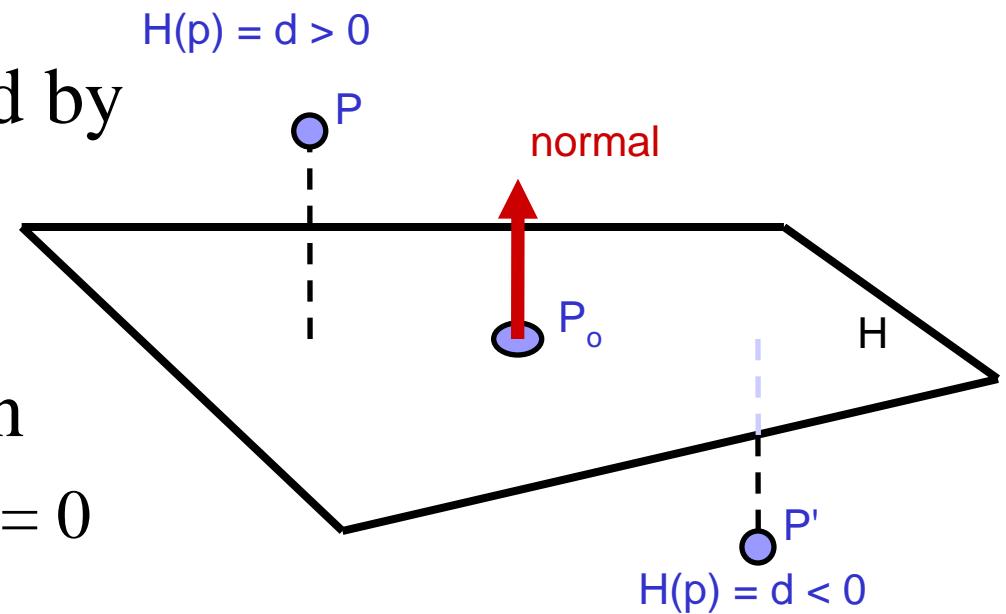
# 3D Plane Representation?

- (Infinite) plane defined by

- $P_o = (x_0, y_0, z_0)$
  - $n = (A, B, C)$

- Implicit plane equation

- $H(P) = Ax + By + Cz + D = 0$   
 $= n \cdot P + D = 0$



# 3D Plane Representation?

- (Infinite) plane defined by

- $P_o = (x_0, y_0, z_0)$
  - $n = (A, B, C)$

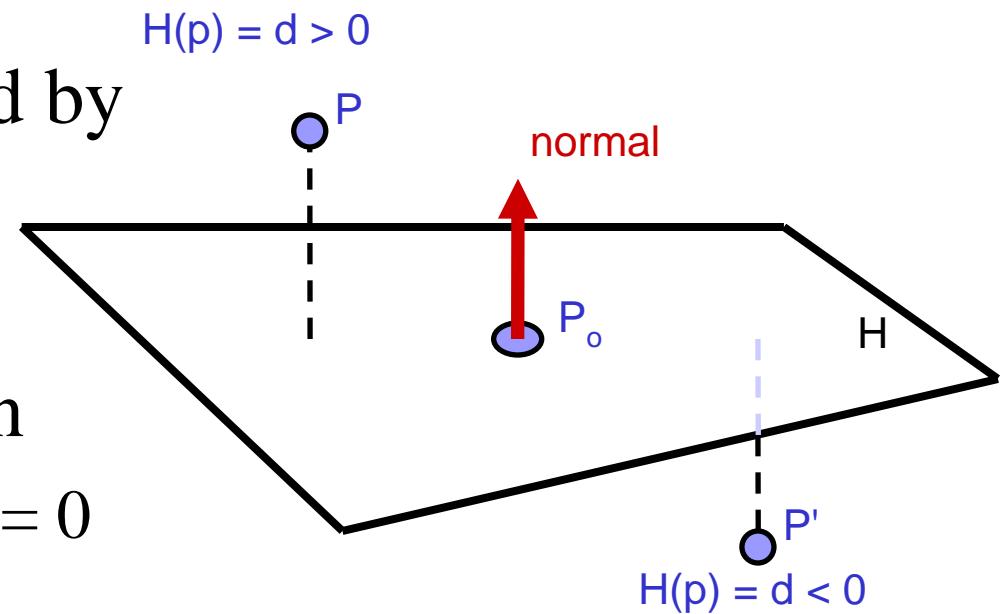
- Implicit plane equation

- $H(P) = Ax + By + Cz + D = 0$   
 $= n \cdot P + D = 0$

- What is  $D$ ?

$$Ax_0 + By_0 + Cz_0 + D = 0 \quad (\text{Point } P_o \text{ must lie on plane})$$

$$\Rightarrow D = -Ax_0 - By_0 - Cz_0$$



# 3D Plane Representation?

- (Infinite) plane defined by

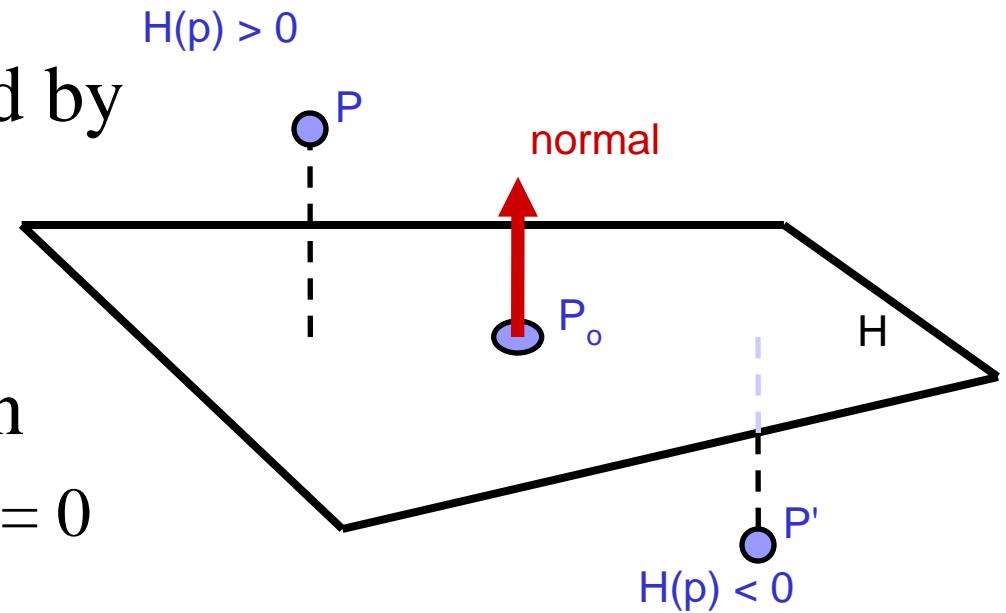
- $P_o = (x_0, y_0, z_0)$
  - $n = (A, B, C)$

- Implicit plane equation

- $H(P) = Ax + By + Cz + D = 0$   
 $= n \cdot P + D = 0$

- Point-Plane distance?

- If  $n$  is normalized,  
distance to plane is  $H(P)$
  - it is a *signed distance*!



# Explicit vs. Implicit?

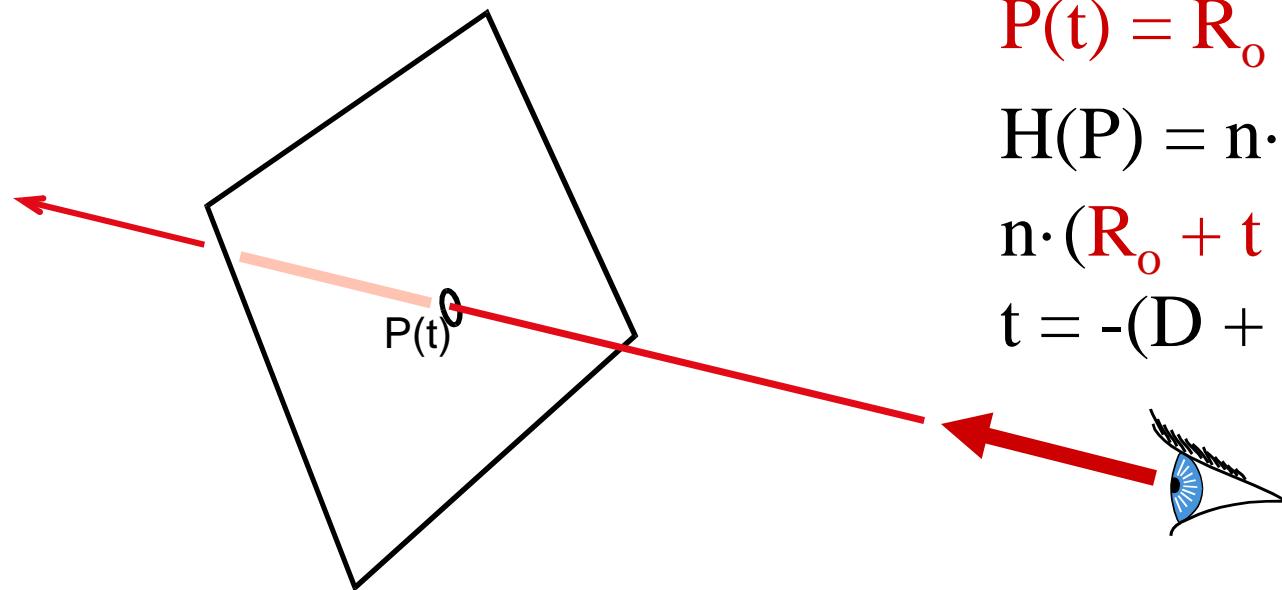
---

- Ray equation is explicit  $P(t) = R_o + t * R_d$ 
  - Parametric
  - Generates points
  - Hard to verify that a point is on the ray
- Plane equation is implicit  $H(P) = n \cdot P + D = 0$ 
  - Solution of an equation
  - Does not generate points
  - Verifies that a point is on the plane
- Exercise: Explicit plane and implicit ray?

# Ray-Plane Intersection

---

- Intersection means both are satisfied
- So, insert explicit equation of ray into implicit equation of plane & solve for t



$$P(t) = R_o + t * R_d$$

$$H(P) = n \cdot P + D = 0$$

$$n \cdot (R_o + t * R_d) + D = 0$$

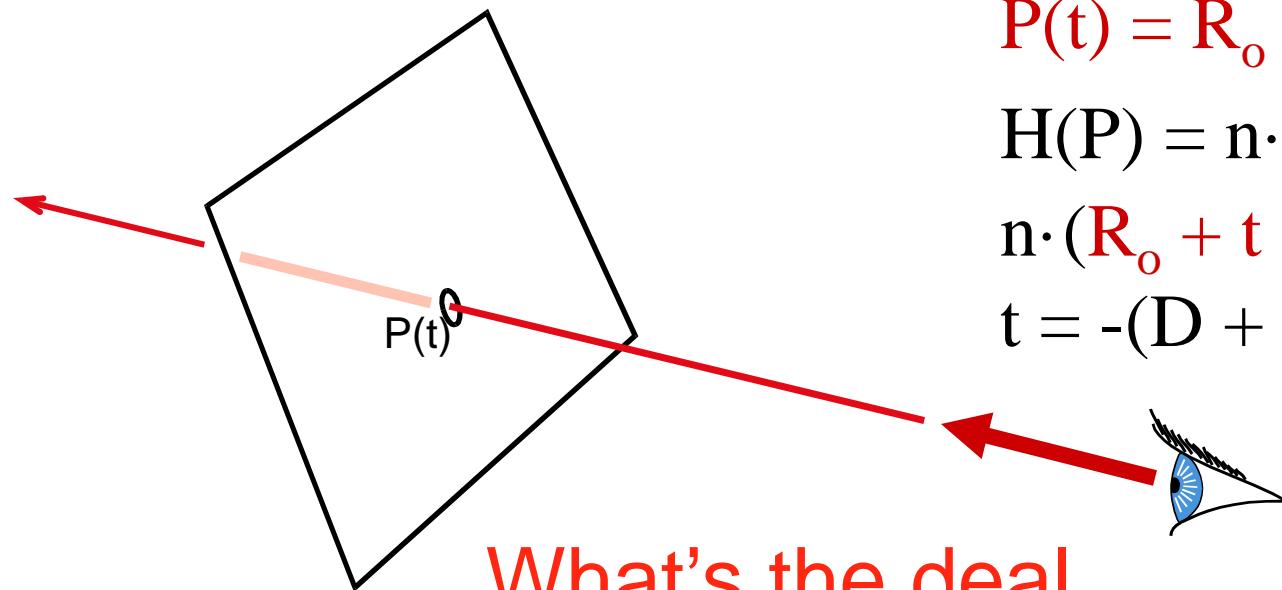
$$t = -(D + n \cdot R_o) / n \cdot R_d$$

**Done!**

# Ray-Plane Intersection

---

- Intersection means both are satisfied
- So, insert explicit equation of ray into implicit equation of plane & solve for t



$$P(t) = R_o + t * R_d$$

$$H(P) = n \cdot P + D = 0$$

$$n \cdot (R_o + t * R_d) + D = 0$$

$$t = -(D + n \cdot R_o) / n \cdot R_d$$

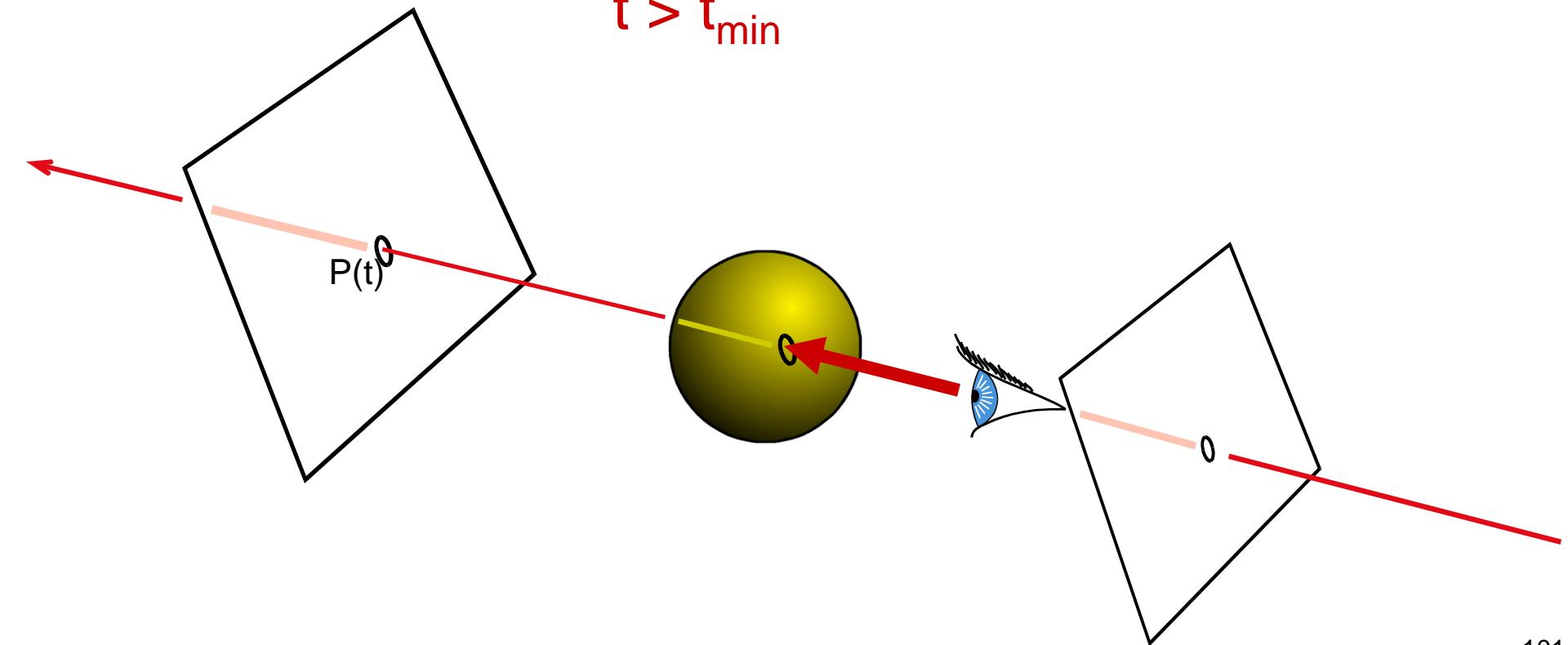
**Done!**

What's the deal  
when  $n \cdot R_d = 0$ ?

# Additional Bookkeeping

---

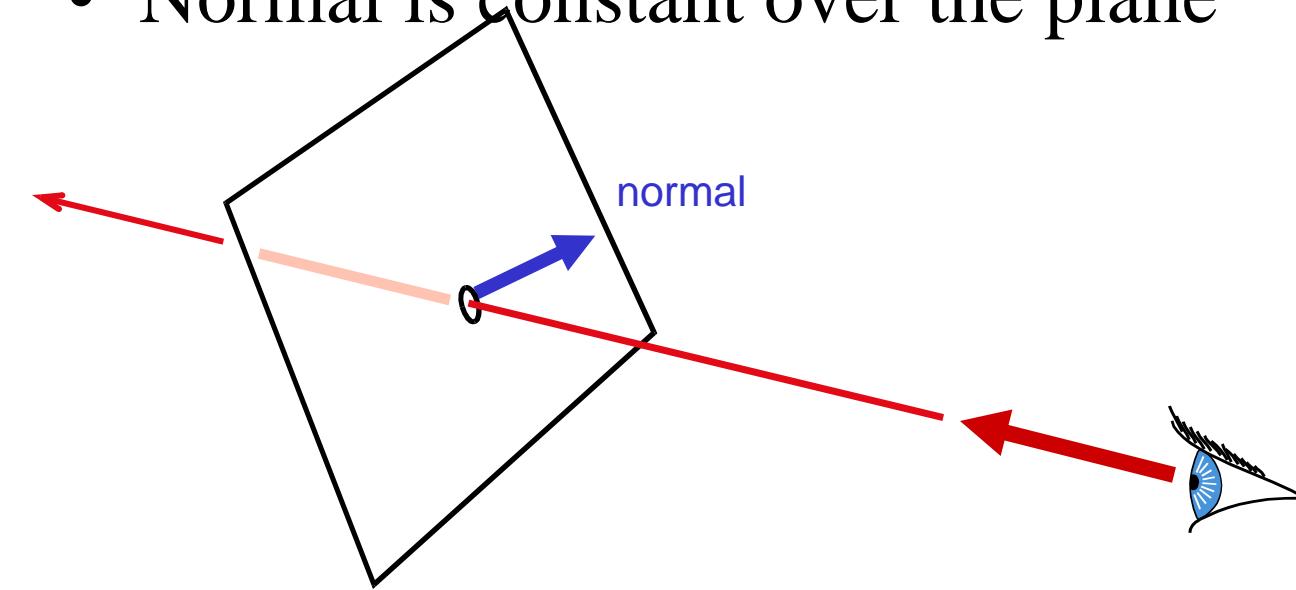
- Verify that intersection is closer than previous  
 $t < t_{\text{current}}$
- Verify that it is not out of range (behind eye)  
 $t > t_{\min}$



# Normal

---

- Also need surface normal for shading
  - (Diffuse: dot product between light direction and normal, clamp to zero)
- Normal is constant over the plane



# Questions?

---

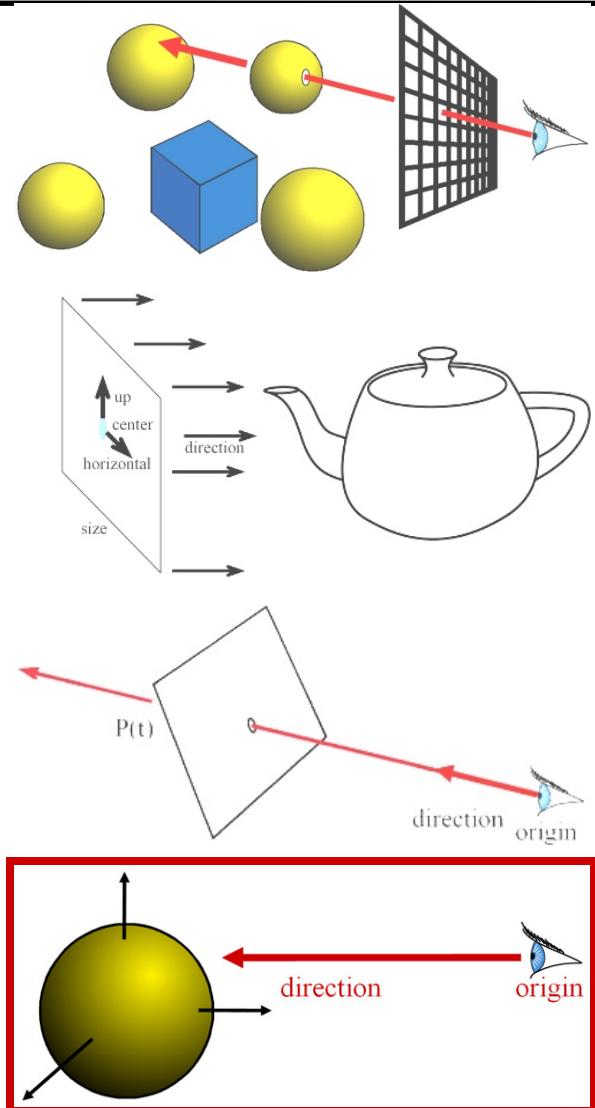


RENDERED USING DALI - HENRIK WANN JENSEN 2000

Image by Henrik Wann Jensen

# Ray Casting

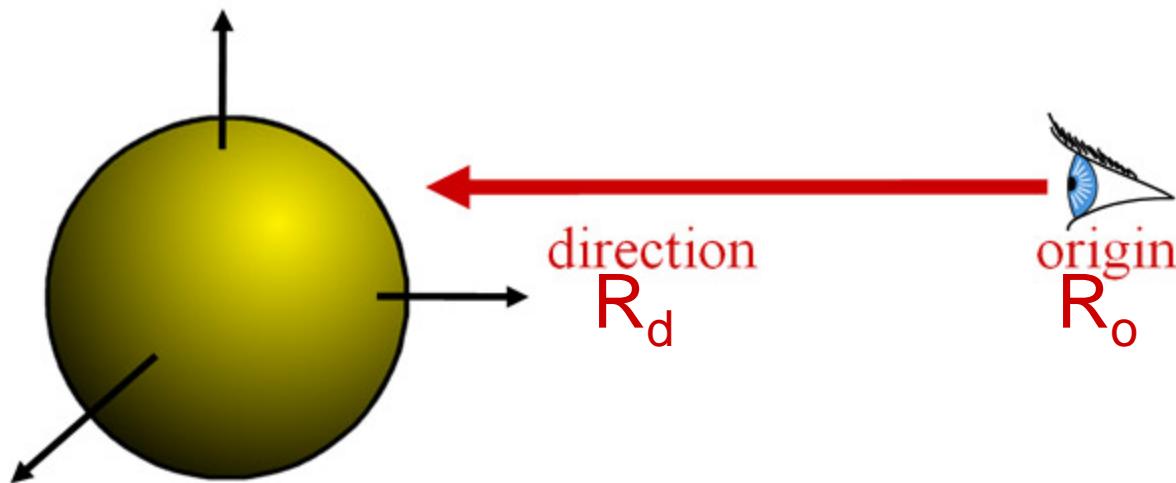
- Ray Casting Basics
- Camera and Ray Generation
- Ray-Plane Intersection
- Ray-Sphere Intersection



# Sphere Representation?

---

- Implicit sphere equation
  - Assume centered at origin (easy to translate)
  - $H(P) = \|P\|^2 - r^2 = P \cdot P - r^2 = 0$



# Ray-Sphere Intersection

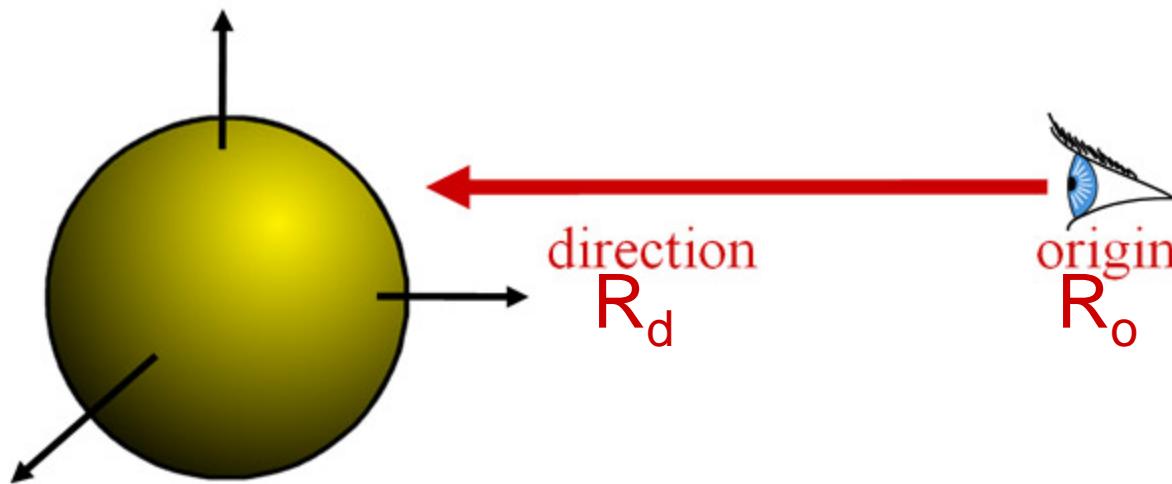
---

- Insert explicit equation of ray into implicit equation of sphere & solve for t

$$P(t) = R_o + t \cdot R_d \quad ; \quad H(P) = P \cdot P - r^2 = 0$$

$$(R_o + t \cdot R_d) \cdot (R_o + t \cdot R_d) - r^2 = 0$$

$$R_d \cdot R_d \cdot t^2 + 2R_d \cdot R_o \cdot t + R_o \cdot R_o - r^2 = 0$$



# Ray-Sphere Intersection

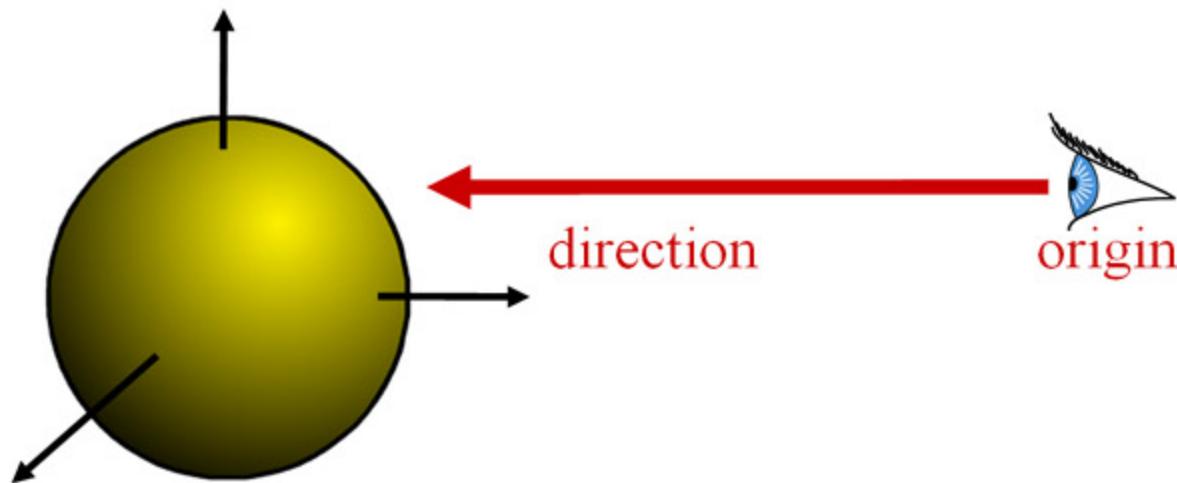
---

- Quadratic:  $at^2 + bt + c = 0$ 
  - $a = 1$  (remember,  $\|R_d\| = 1$ )
  - $b = 2R_d \cdot R_o$
  - $c = R_o \cdot R_o - r^2$
- with discriminant  $d = \sqrt{b^2 - 4ac}$
- and solutions  $t_{\pm} = \frac{-b \pm d}{2a}$

# Ray-Sphere Intersection

---

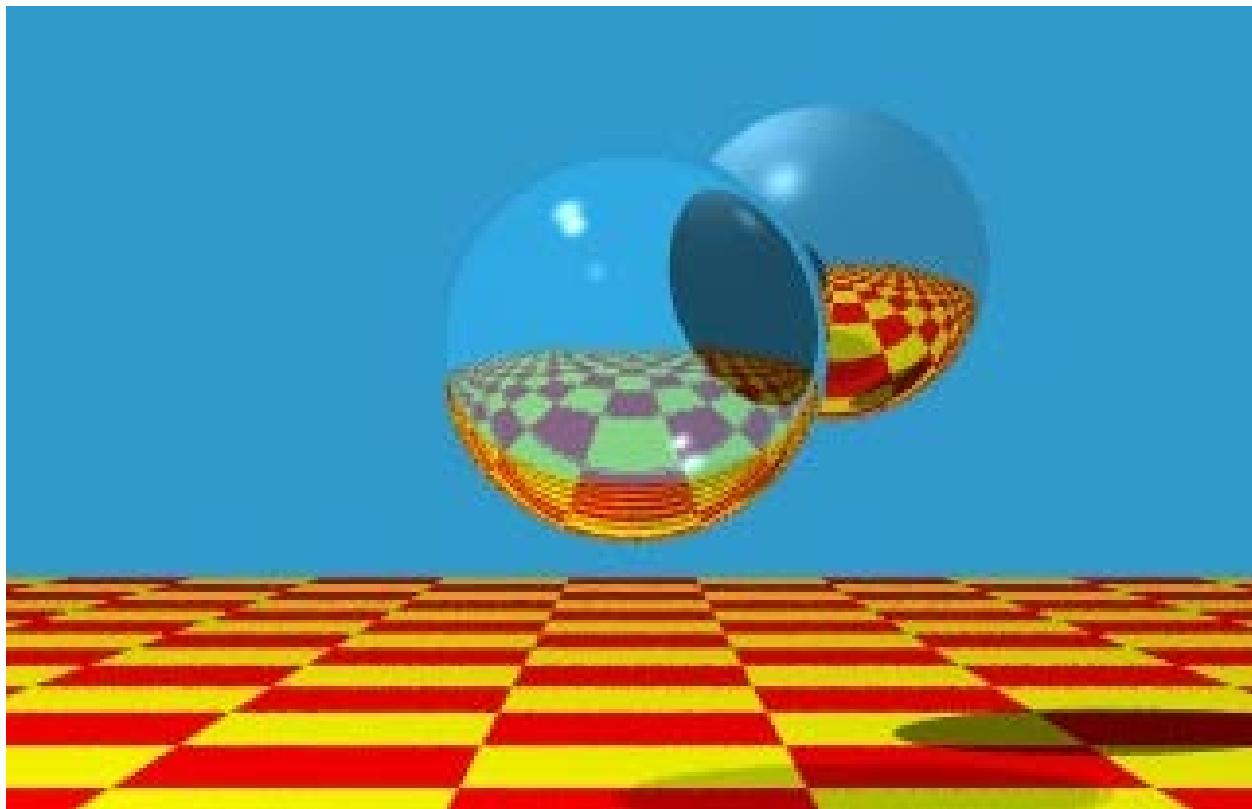
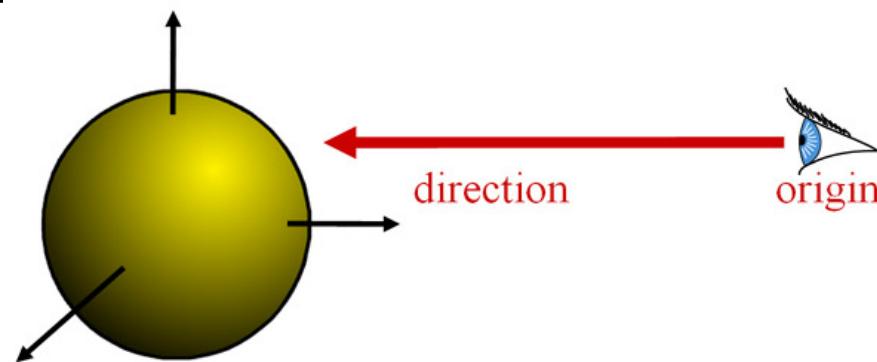
- 3 cases, depending on the sign of  $b^2 - 4ac$
- What do these cases correspond to?
- Which root ( $t+$  or  $t-$ ) should you choose?
  - Closest positive!



# Ray-Sphere Intersection

- It's so easy  
that all  
ray-tracing  
images  
have  
spheres!

:-)

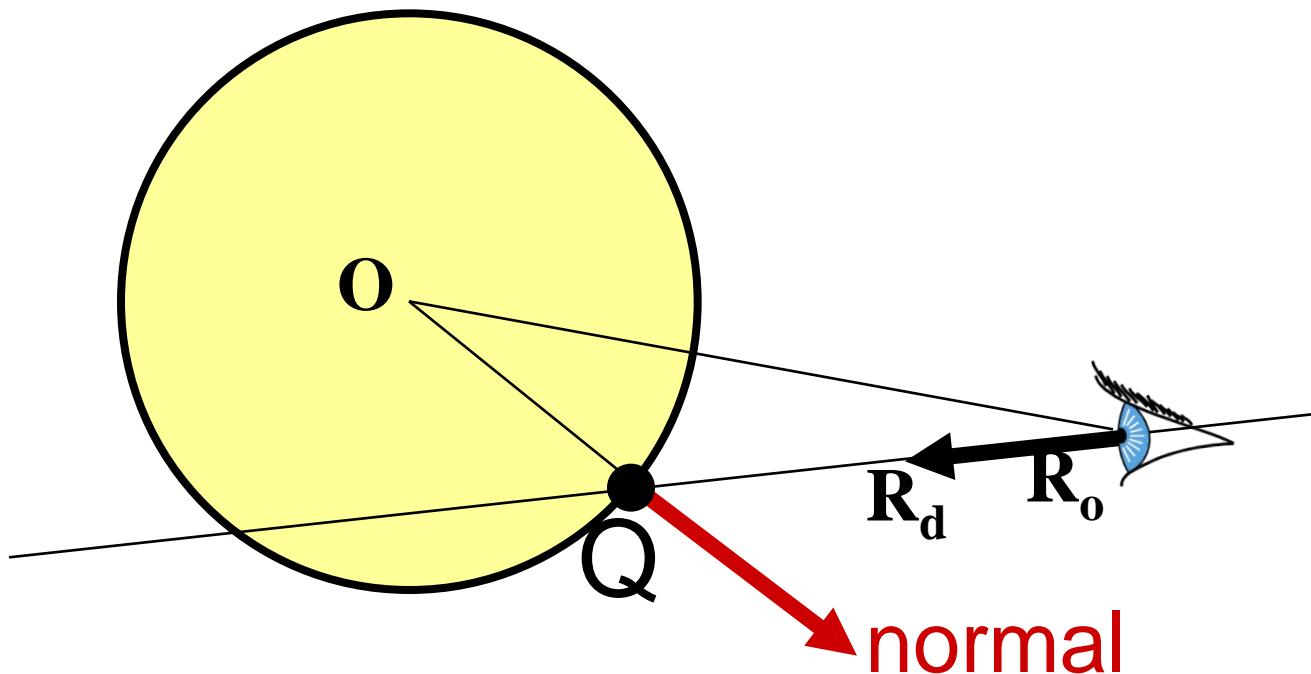


Turner Whitted

# Sphere Normal

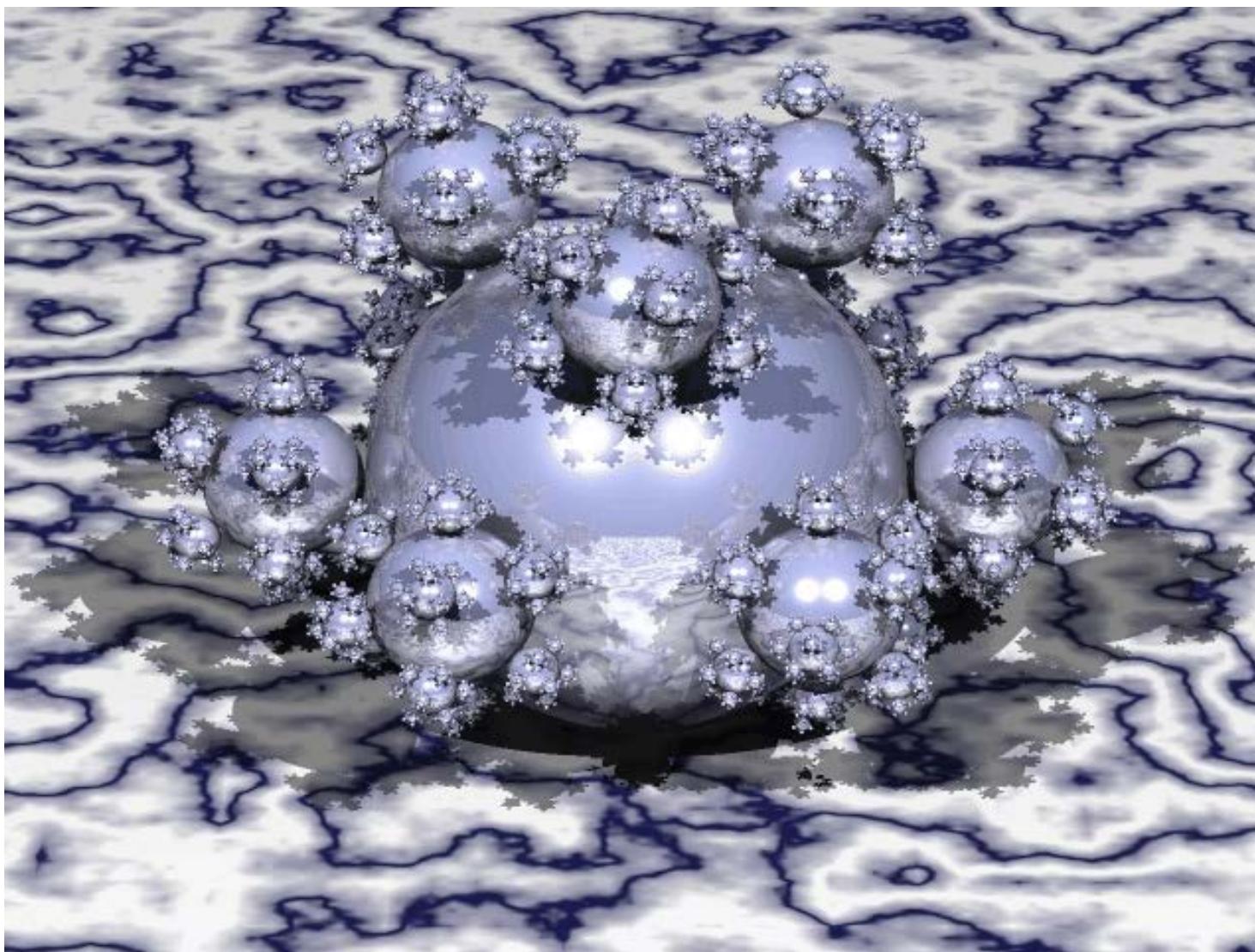
---

- Simply  $\mathbf{Q}/\|\mathbf{Q}\|$ 
  - $\mathbf{Q} = \mathbf{P}(t)$ , intersection point
  - (for spheres centered at origin)



# Questions?

---



# That's All for Today

- But before we talk about the quiz, let's watch a cool video!
- Next time: Ray-triangle intersection, ray tracing

