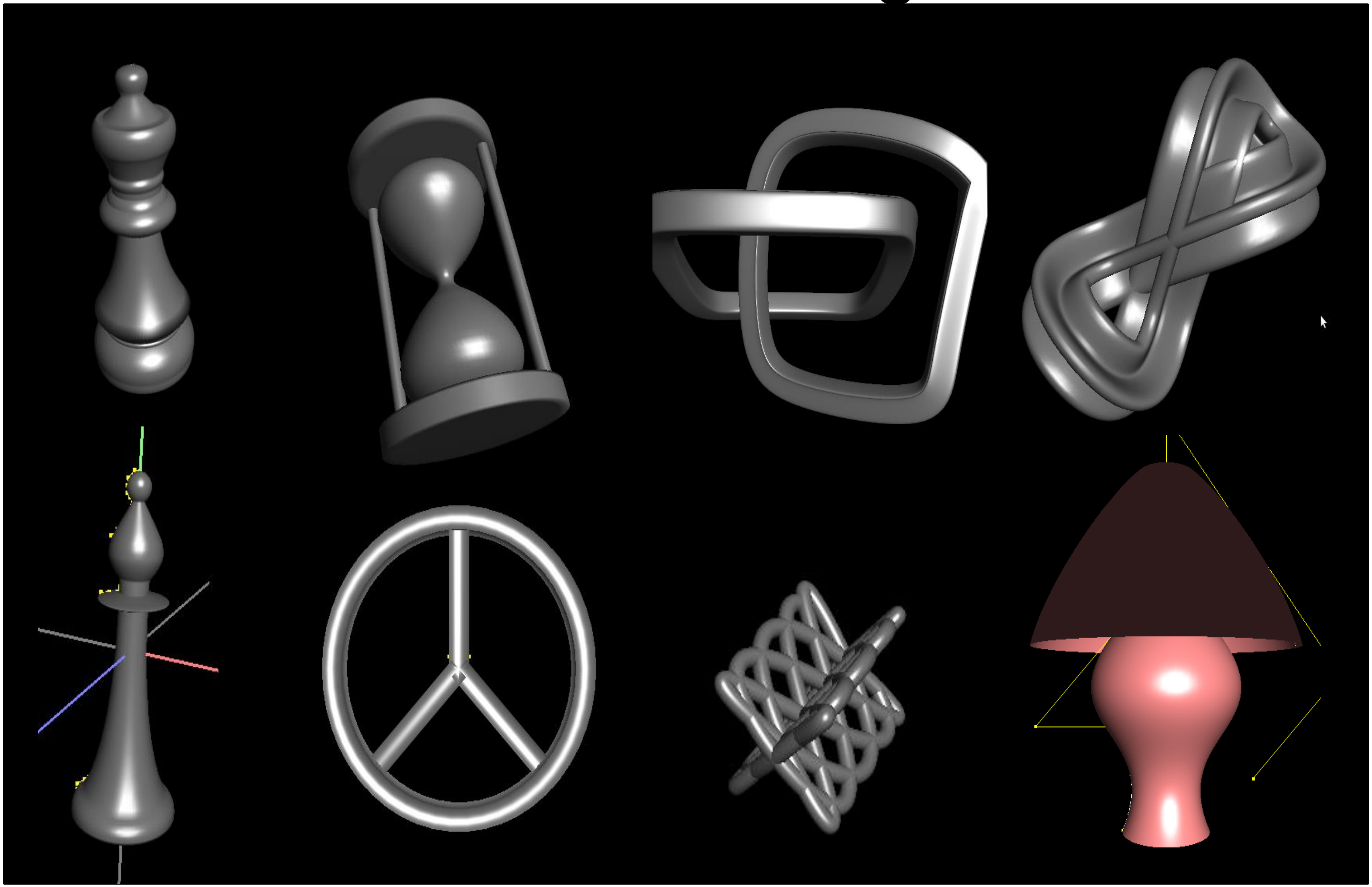
MIT EECS 6.837 Computer Graphics

Part 2 – Rendering

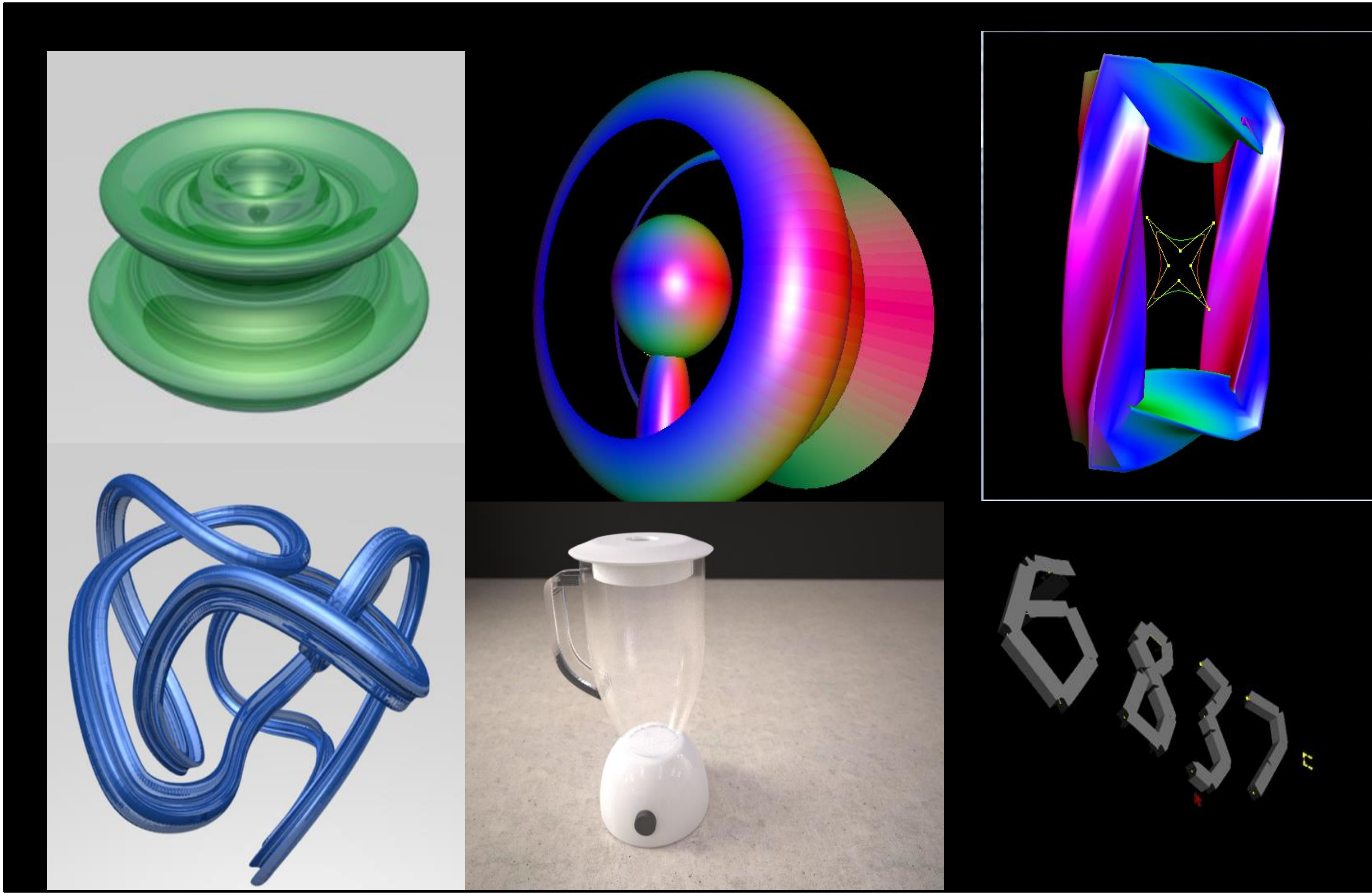
Today: Intro to Rendering, Ray Casting

NVIDIA

Cool Artifacts from Assignment 1



Cool Artifacts from Assignment 1

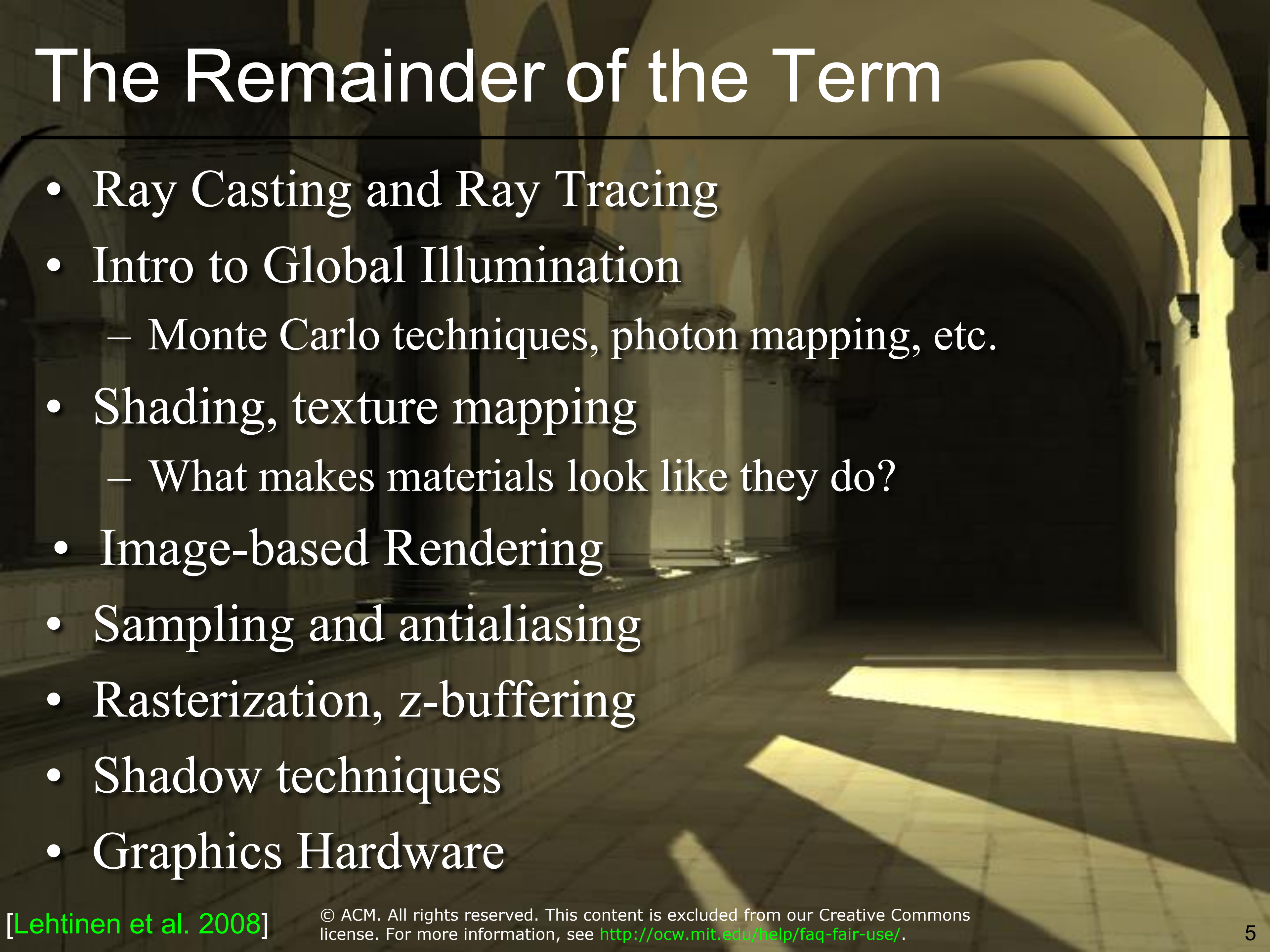


The Story So Far

- Modeling
  - splines, hierarchies, transformations, meshes, etc.
- Animation
  - skinning, ODEs, masses and springs
- **Now we'll to 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

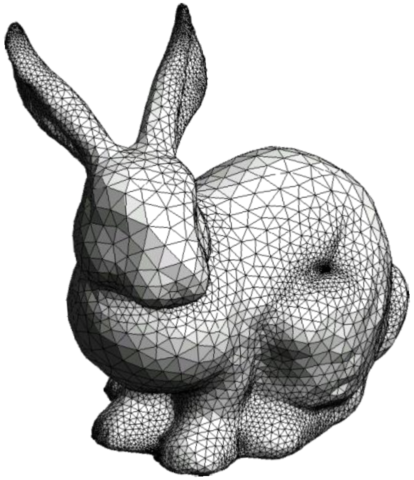


[Lehtinen et al. 2008]

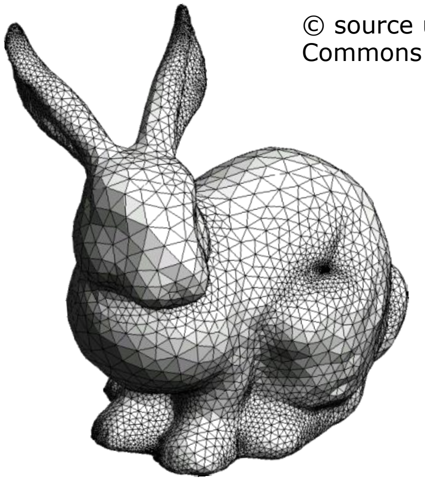
Today

- What does *rendering* mean?

- Basics of ray casting

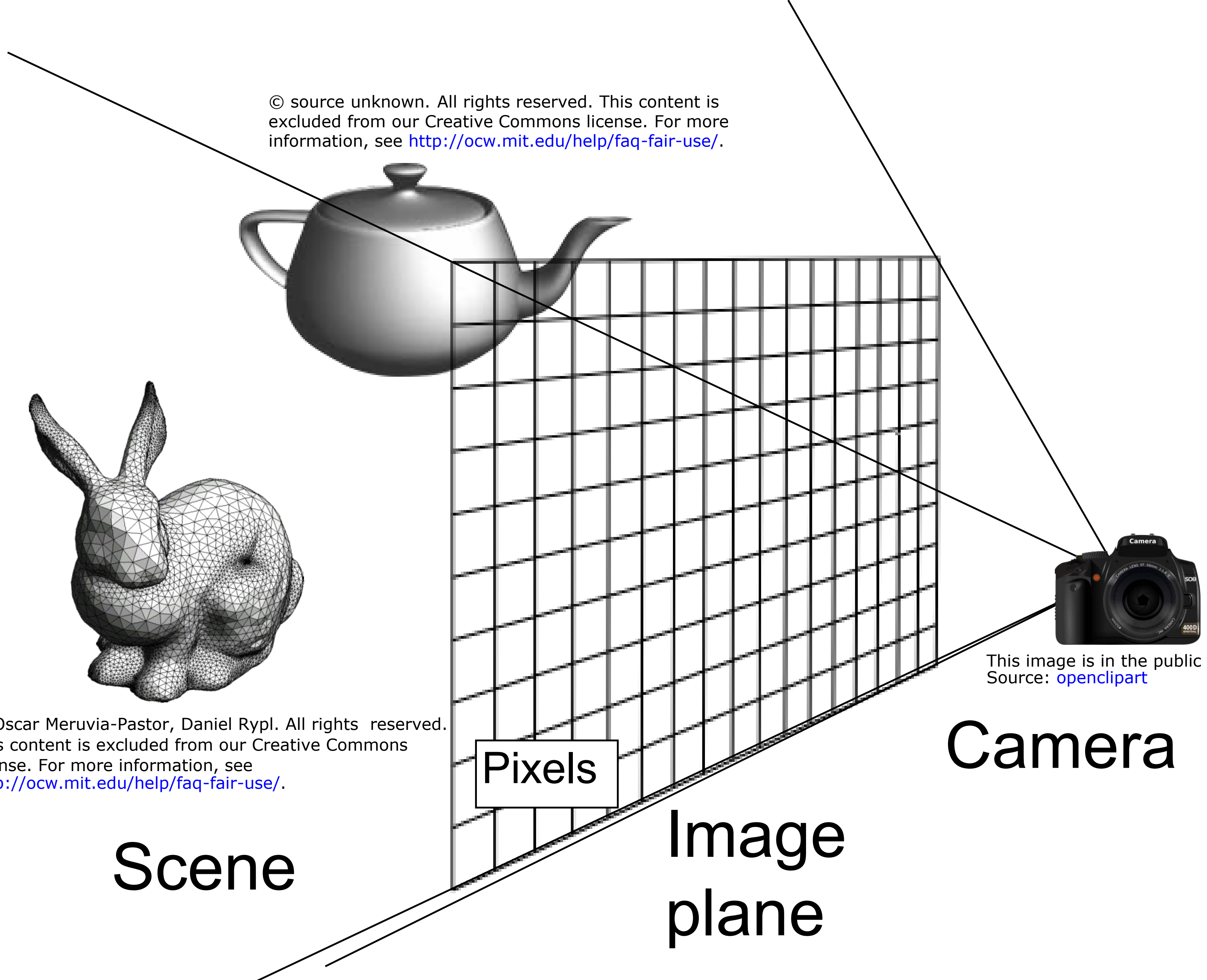


Scene

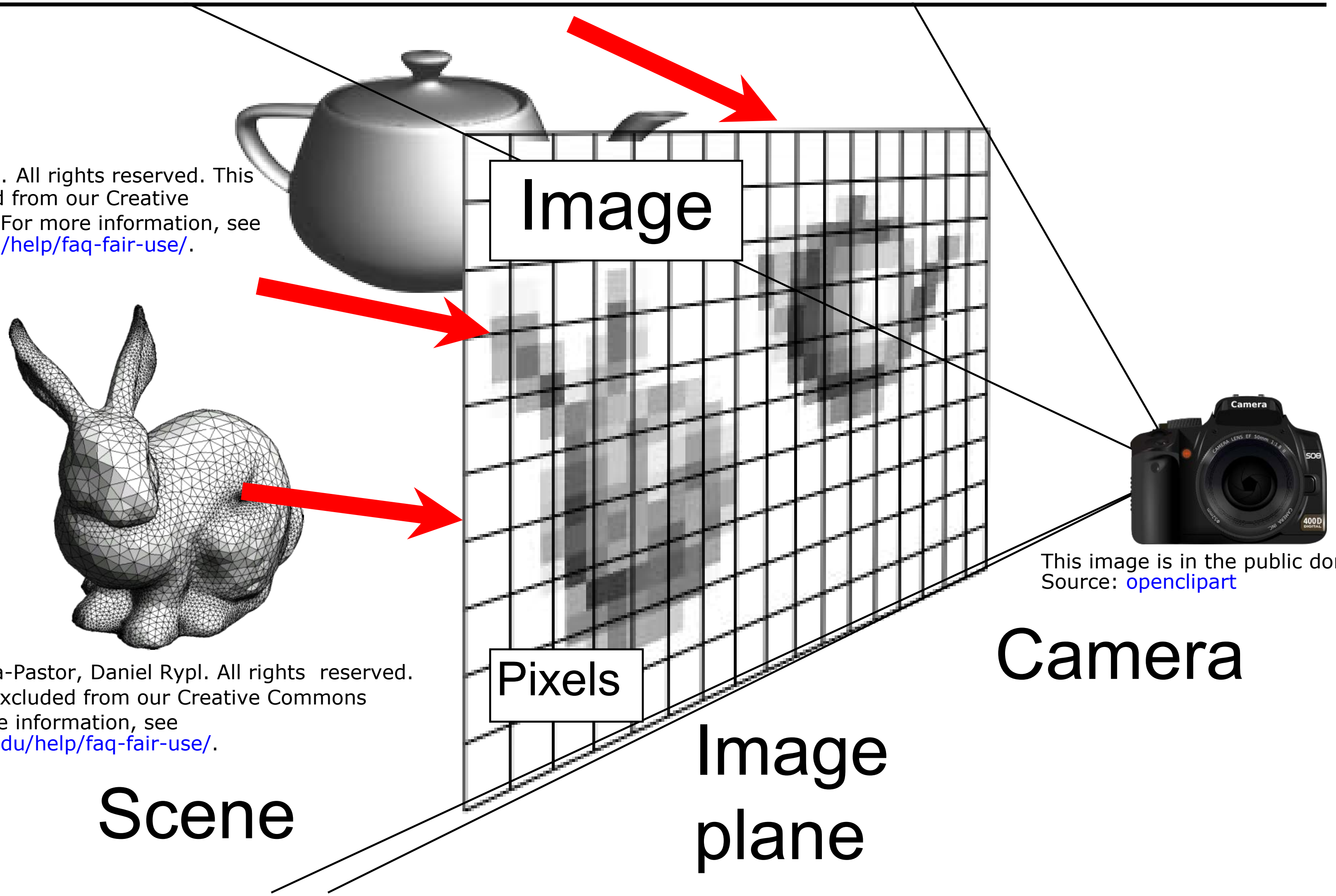


Scene

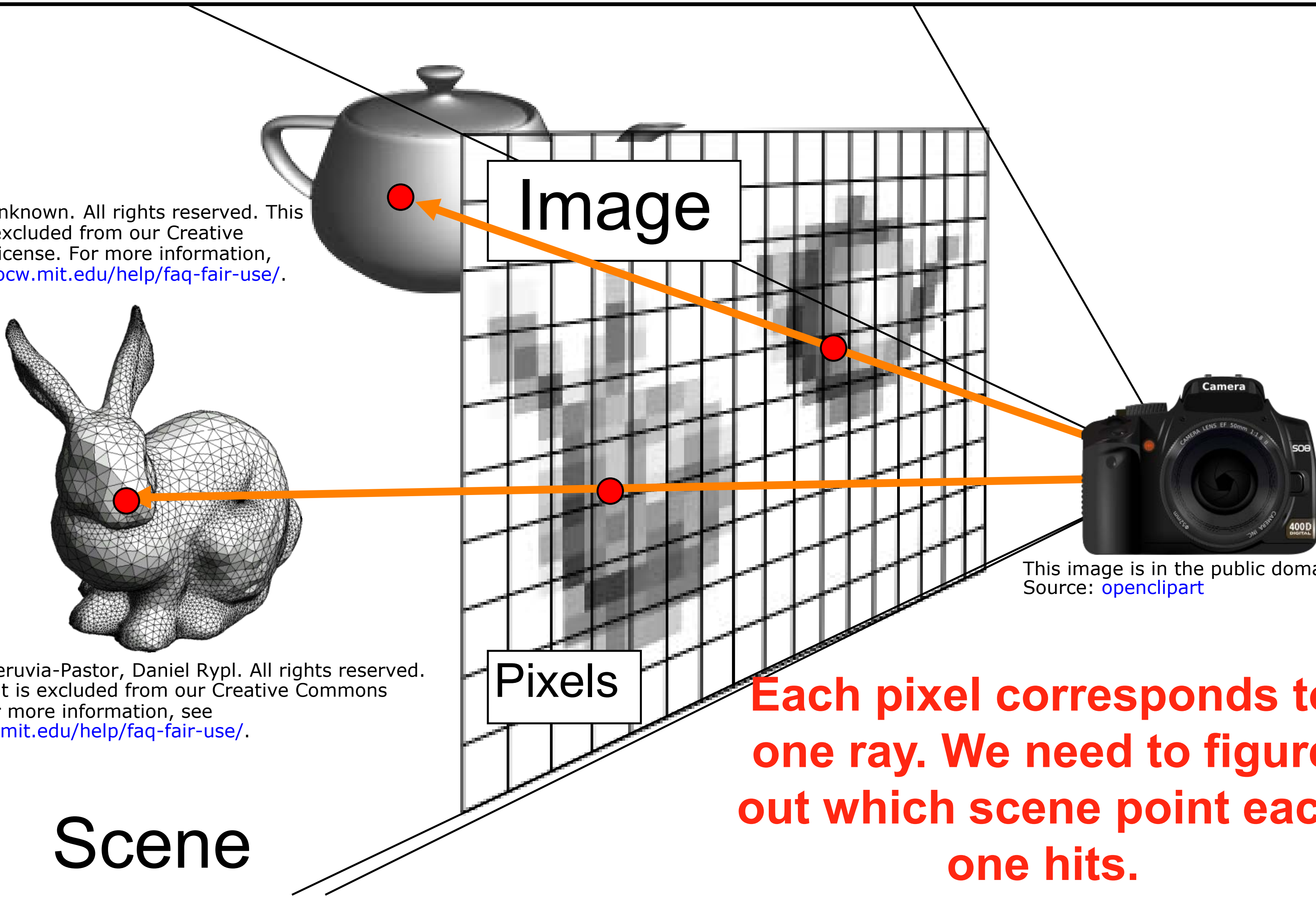
Camera


Pixels
Camera
Image plane
Scene

Rendering = Scene to Image

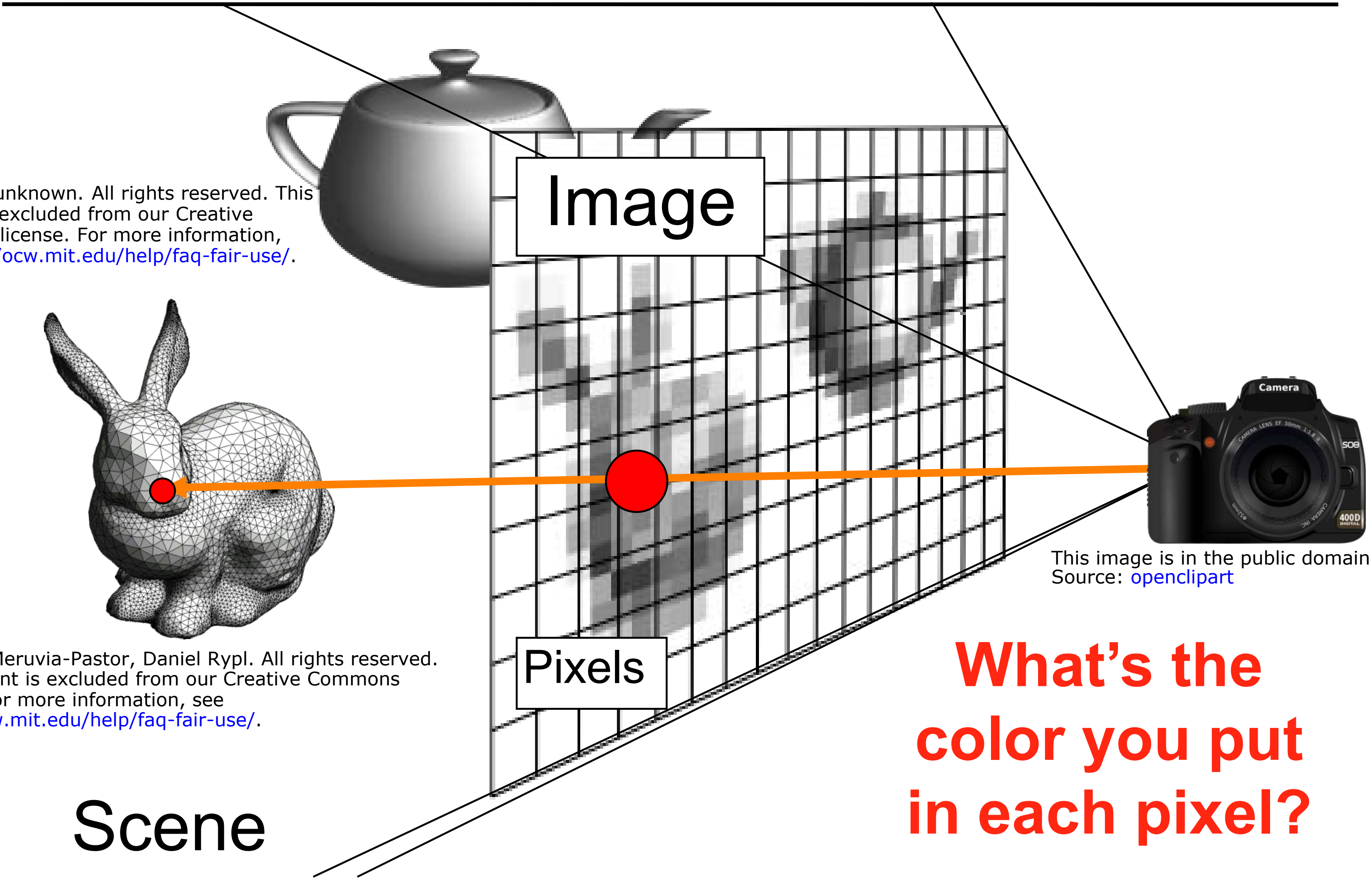


Rendering – Pinhole Camera



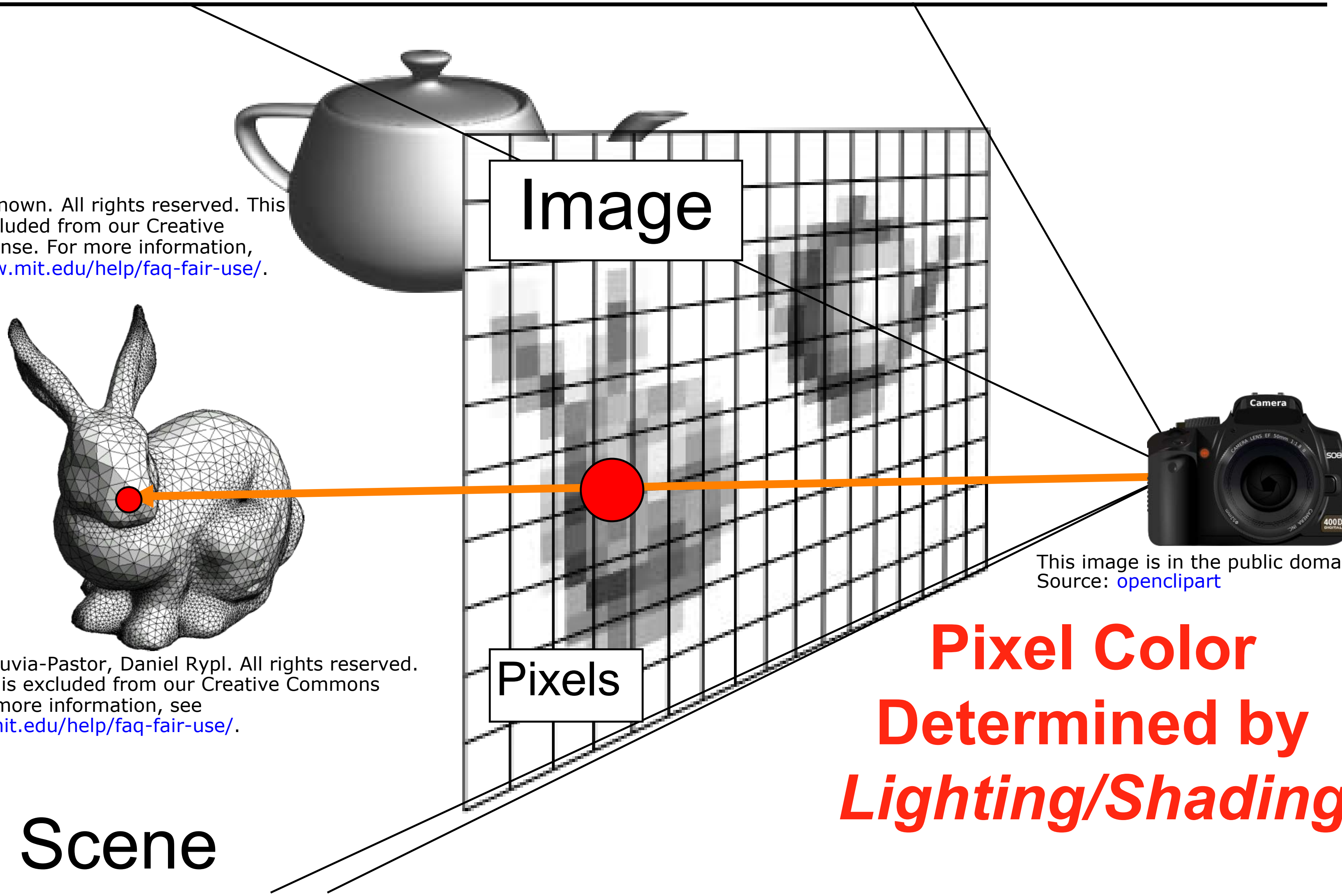
Each pixel corresponds to one ray. We need to figure out which scene point each one hits.

Rendering



What's the color you put in each pixel?

Rendering



Pixel Color Determined by *Lighting/Shading*

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

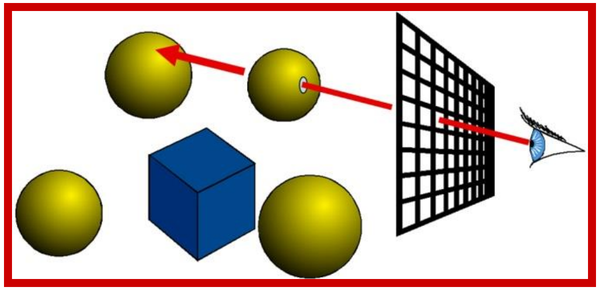
- "Rendering" refers to the entire process that produces color values for pixels
- Pixels correspond to rays; need to figure out the **visible** scene point along each ray
  - Called "hidden surface problem" in older texts
  - "Visibility" is a more modern term
  - Also, we assume (for now) a single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

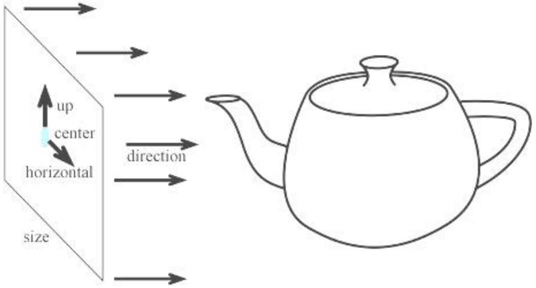
- Note: We are assuming a pinhole camera (for now)

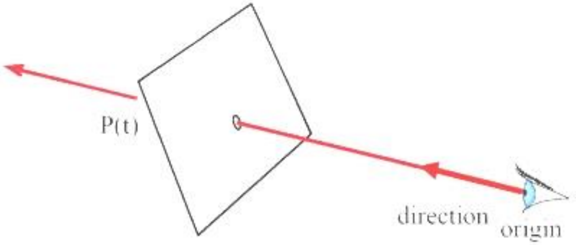
Questions?

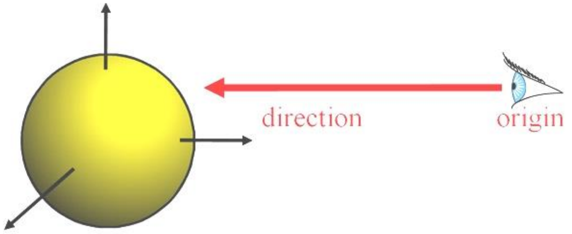
Ray Casting

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



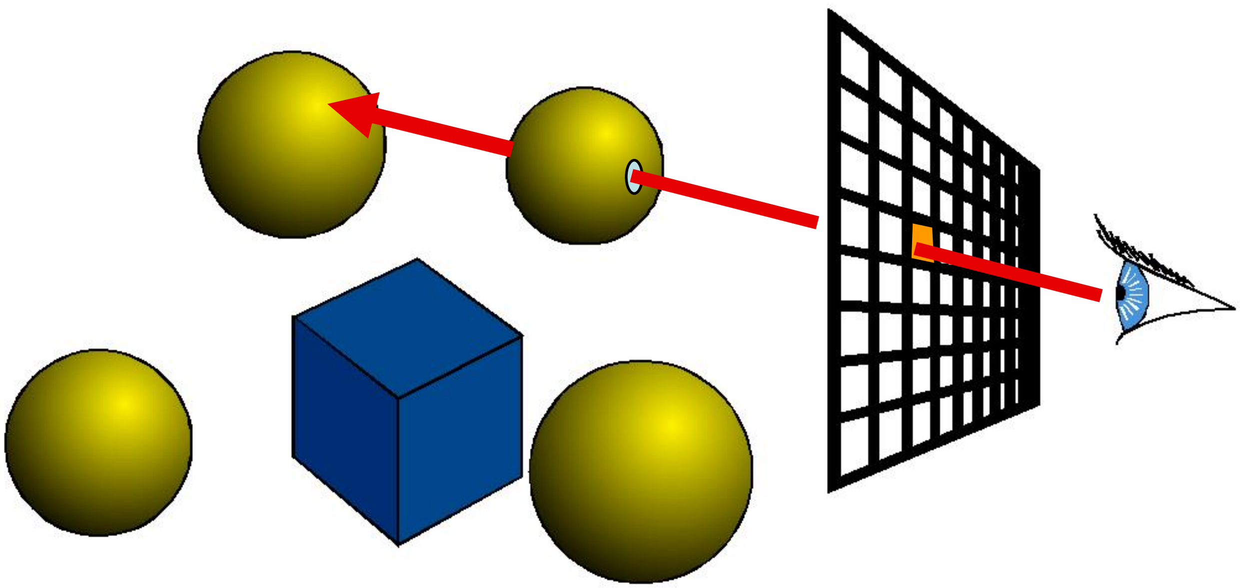






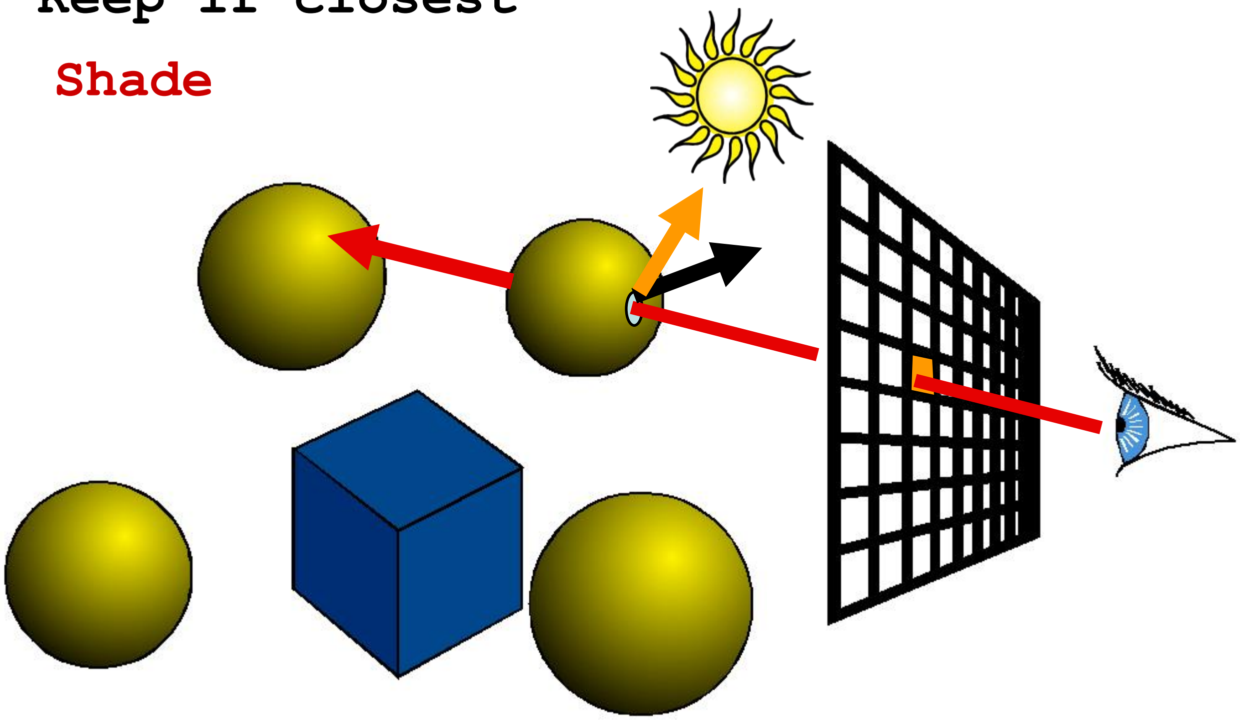
Ray Casting

```
For every pixel
   Construct a ray from the eye
   For every object in the scene
        Find intersection with the ray
        Keep if closest
```



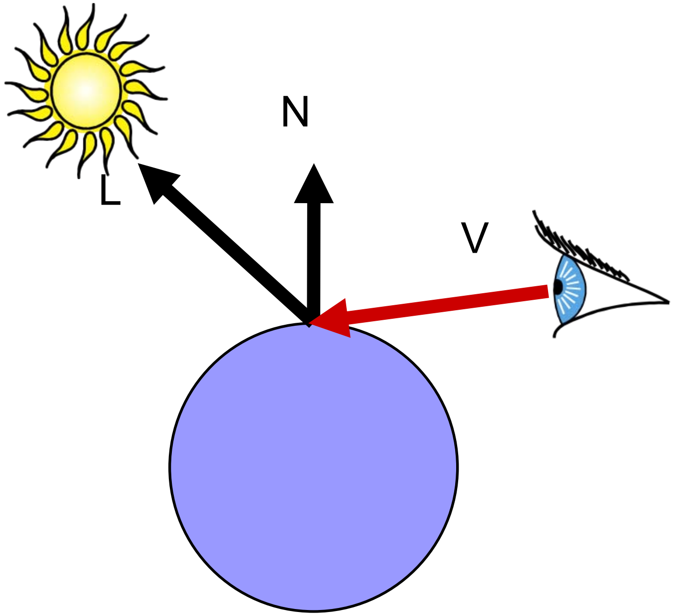
Shading

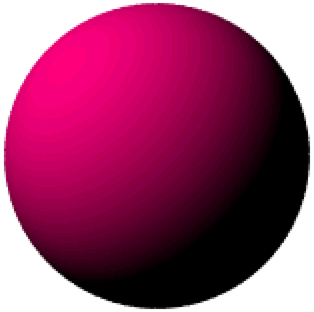
```
For every pixel
   Construct a ray from the eye
   For every object in the scene
        Find intersection with the ray
        Keep if closest
         Shade
```



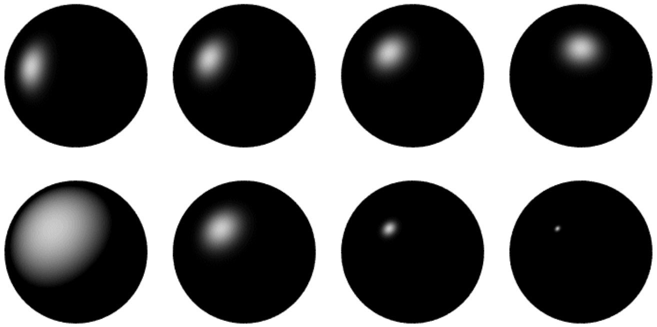
Shading = What Surfaces Look Like

- Surface/Scene Properties
  - surface normal
  - direction to light
  - viewpoint
- Material Properties
  - Diffuse (matte)
  - Specular (shiny)
  - …
- Light properties
  - Position
  - Intensity, ...
- Much more!





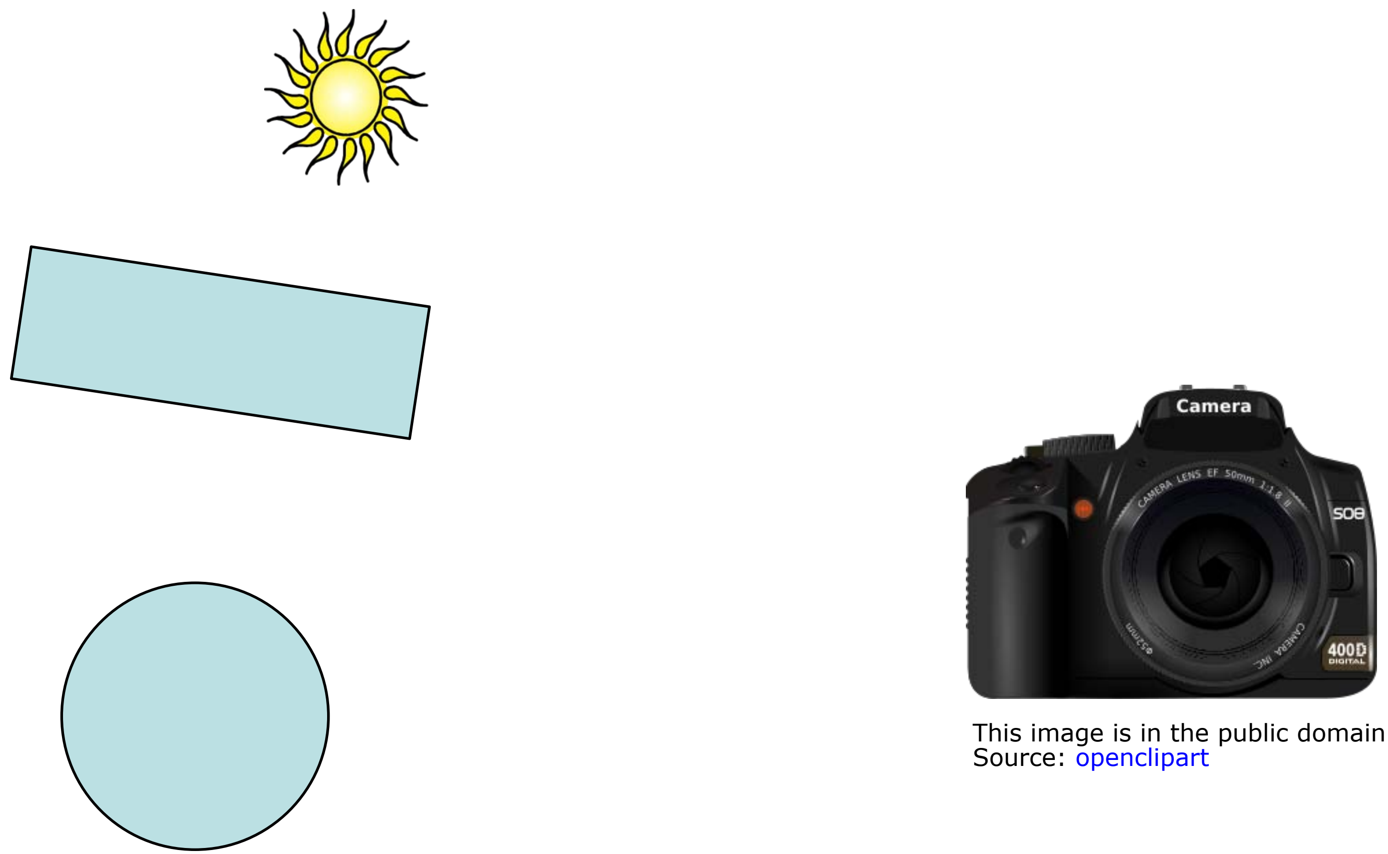
Diffuse sphere



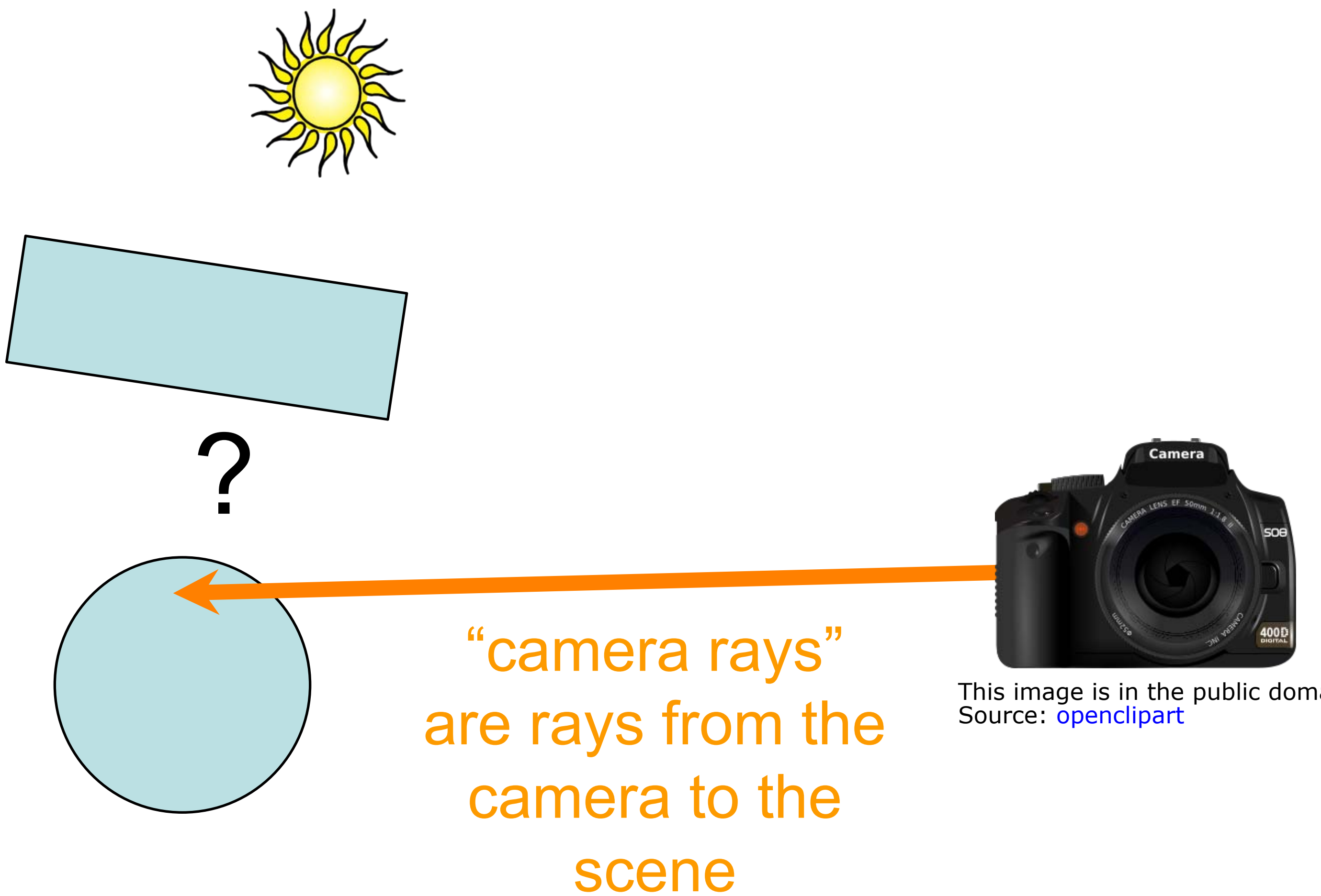
Specular spheres

Ray Casting vs. Ray Tracing

- Let's think about shadows...

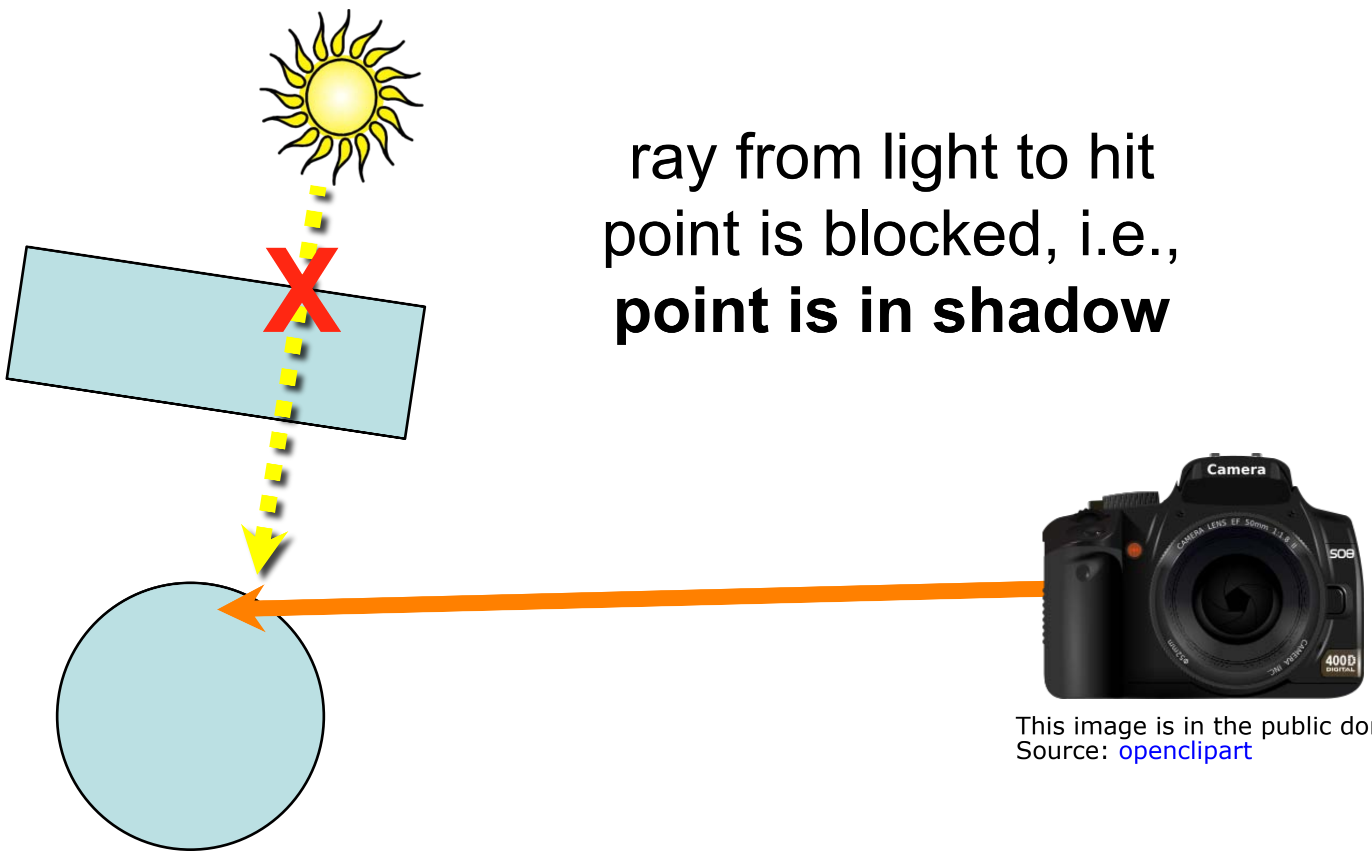


Ray Casting vs. Ray Tracing



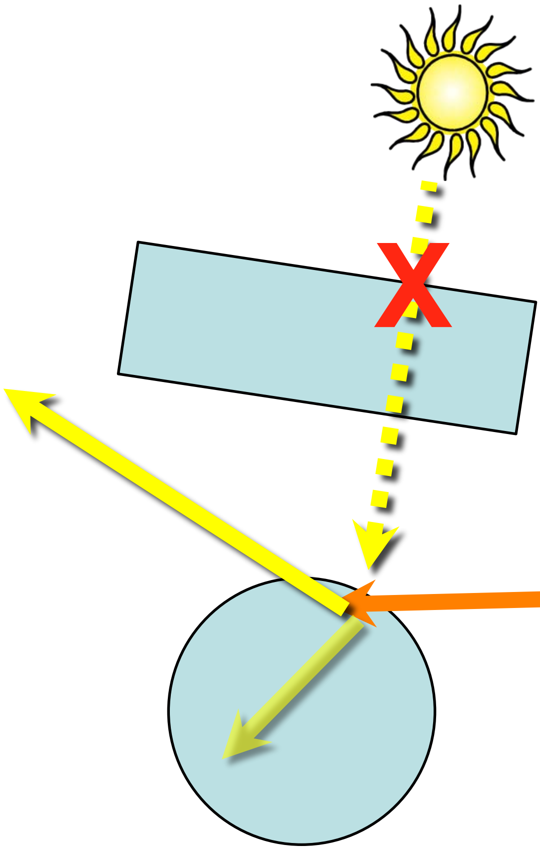
This image is in the public domain.
Source: openclipart

Ray Casting vs. Ray Tracing



Ray Casting vs. Ray Tracing

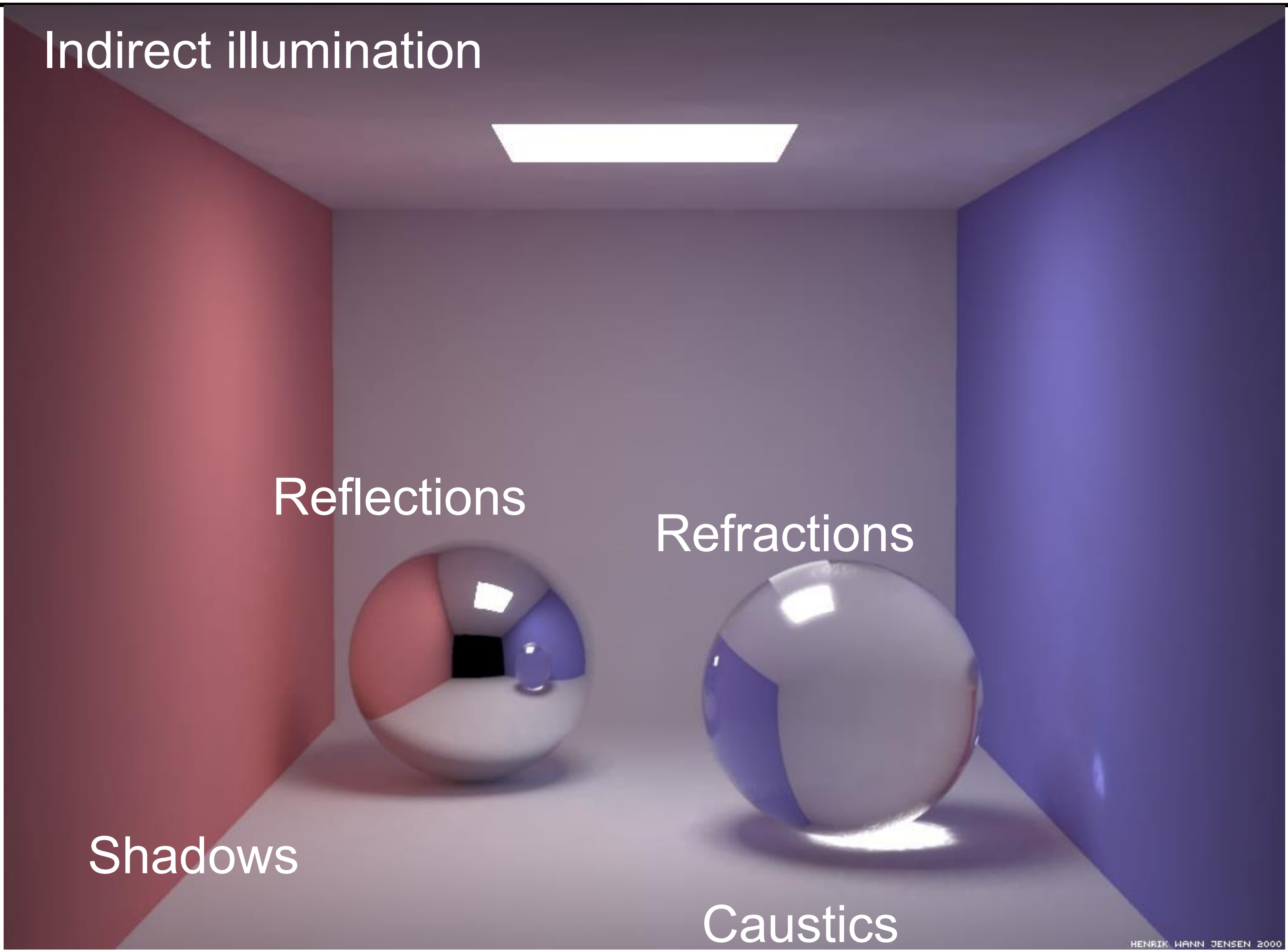
- Ray casting = eye rays only, tracing = also secondary



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

We'll do all this a little later!

Secondary Rays

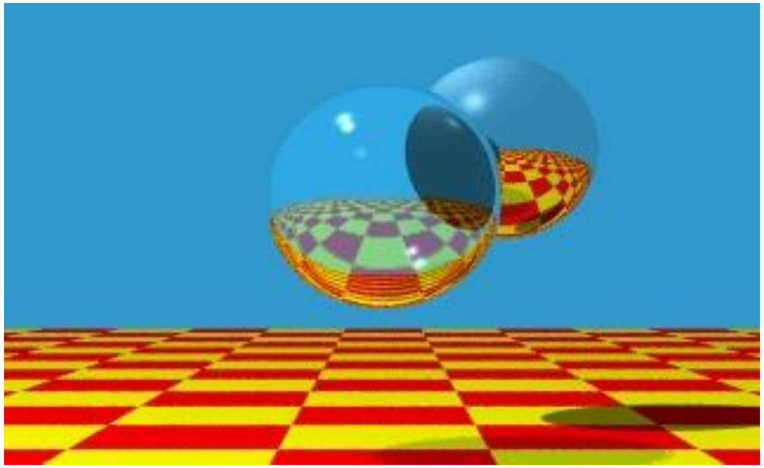


Courtesy of Henrik Wann Jensen. Used with permission.

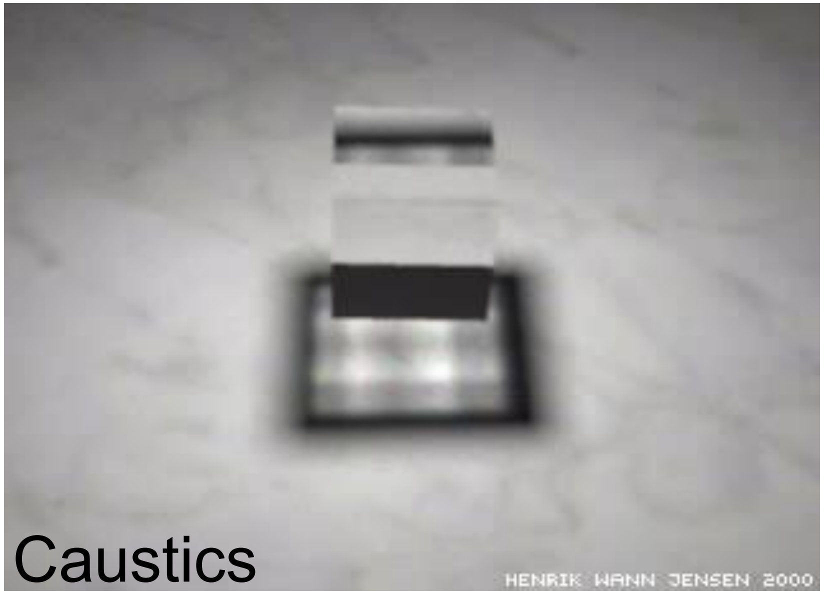
Henrik Wann Jensen

Ray Tracing

Courtesy of Henrik Wann Jensen. Used with permission.



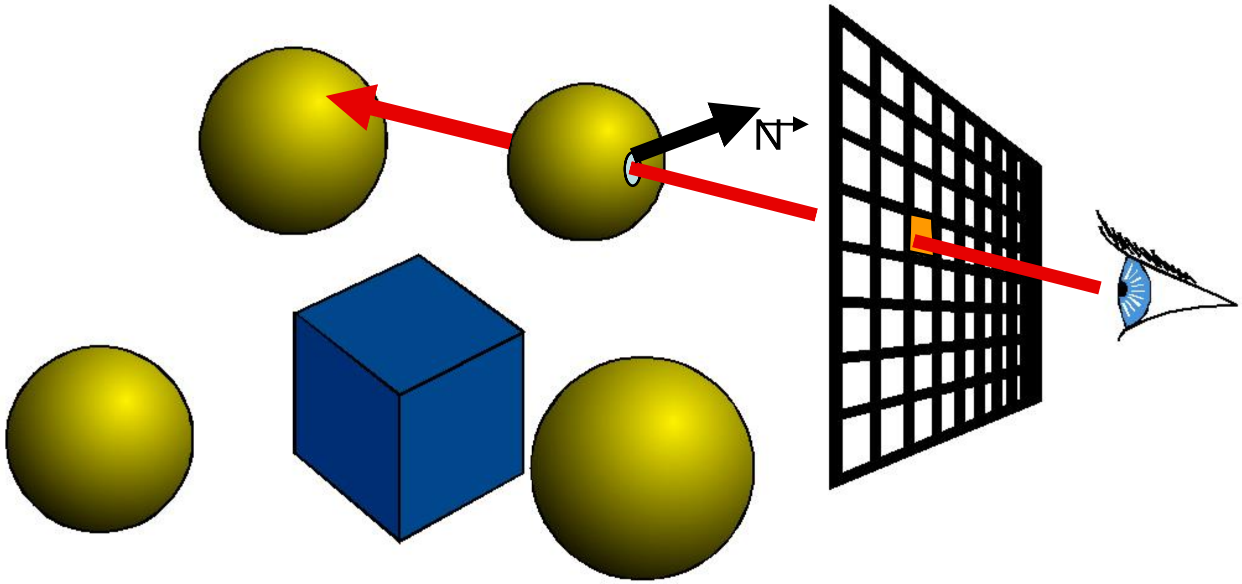
Reflections, refractions



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
```

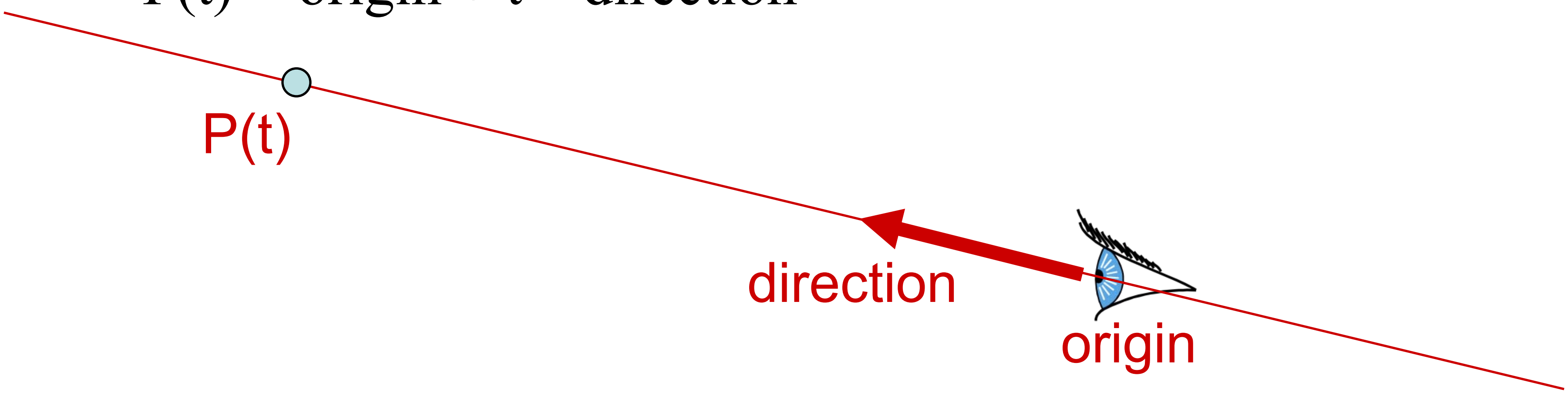


Finding the intersection point and normal is the central part of ray casting

Ray Representation

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

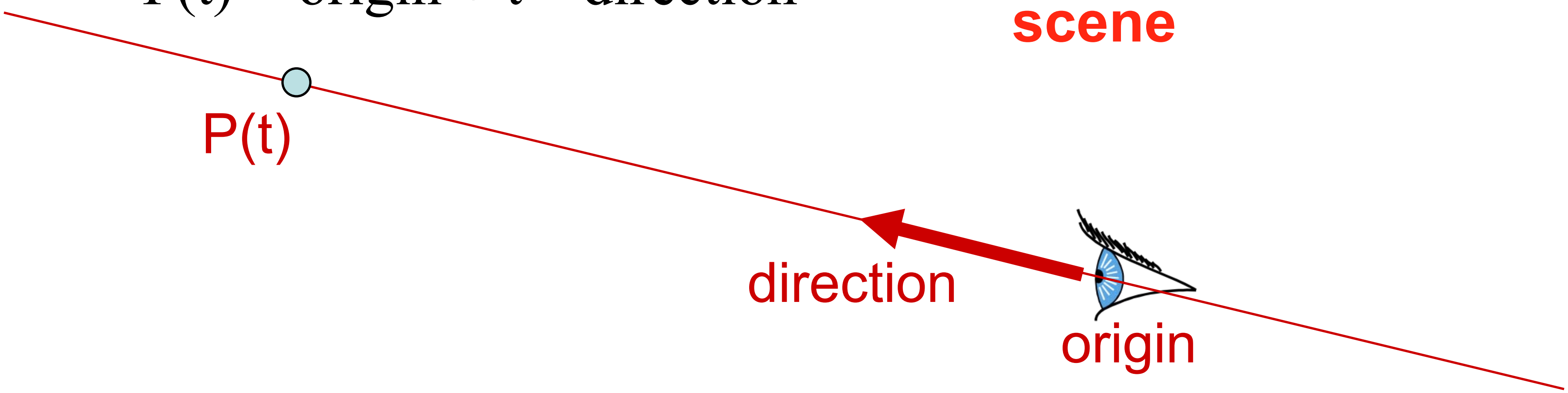
How would you represent a ray?



Ray Representation

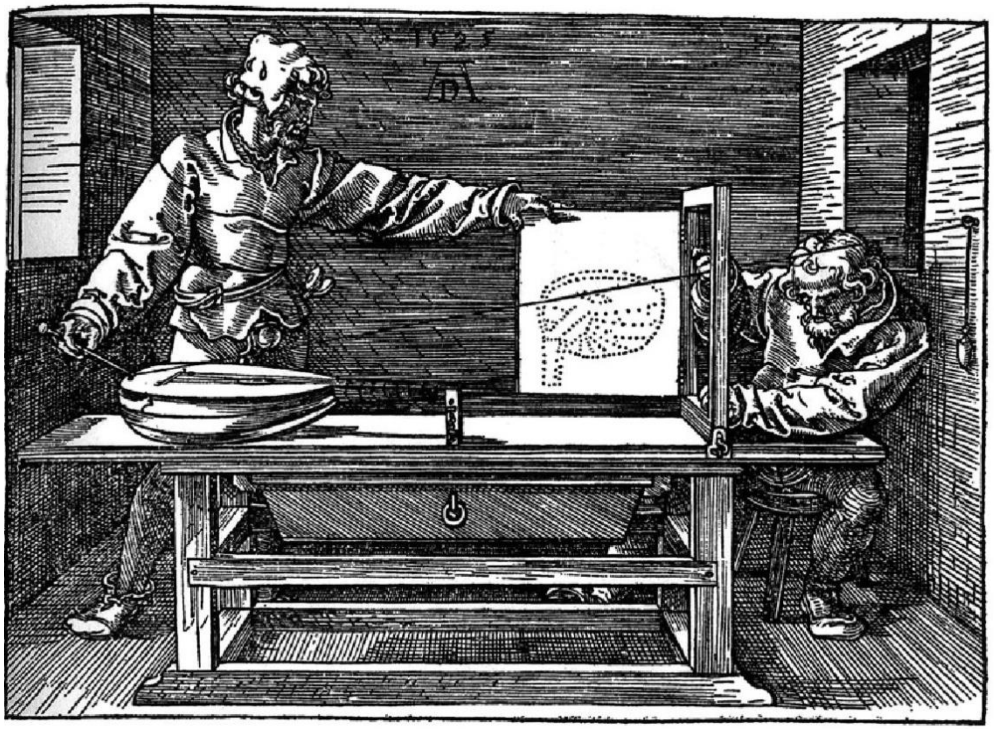
- Origin – Point
- Direction – Vector
  - normalized is better
- Parametric line
  - P(t) = origin + t * direction

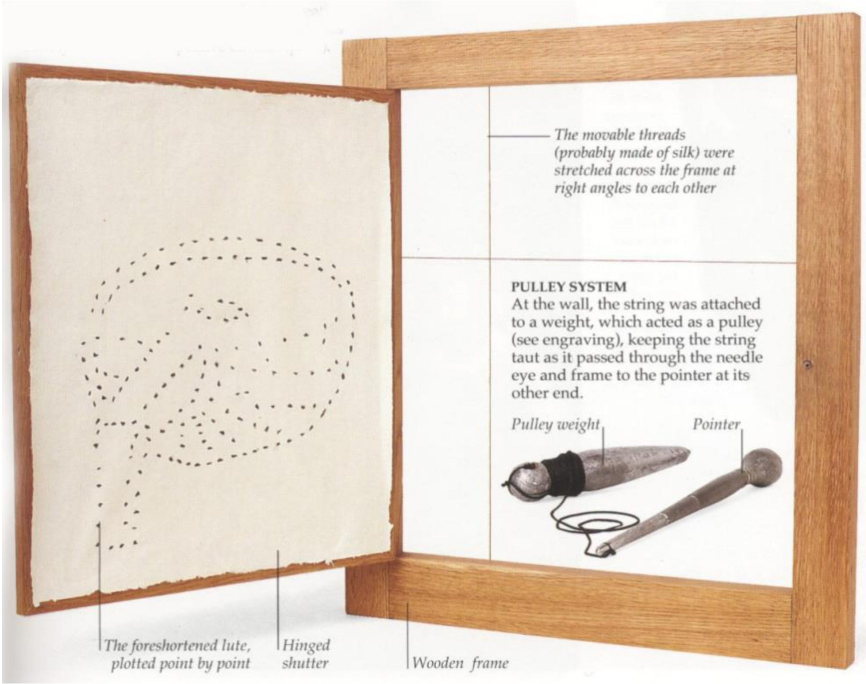
Another way to put the ray casting problem statement:
Find smallest t > 0 such that P(t) lies on a surface in the scene



Dürer’s Ray Casting Machine

- Albrecht Dürer, 16th century



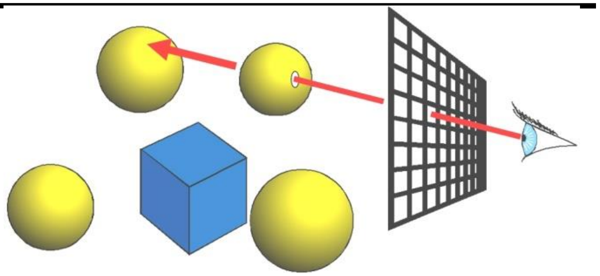


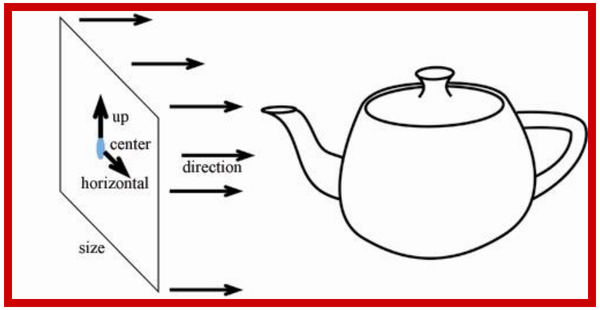
Dürer's Ray Casting Machine

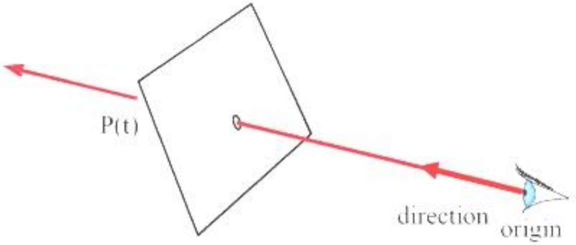
- Albrecht Dürer, 16th century

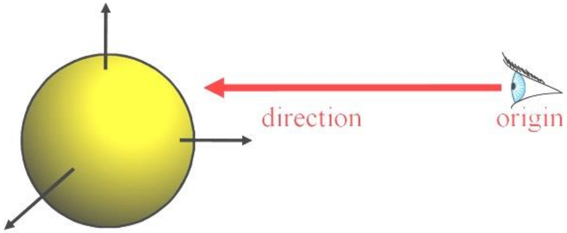
Ray Casting

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



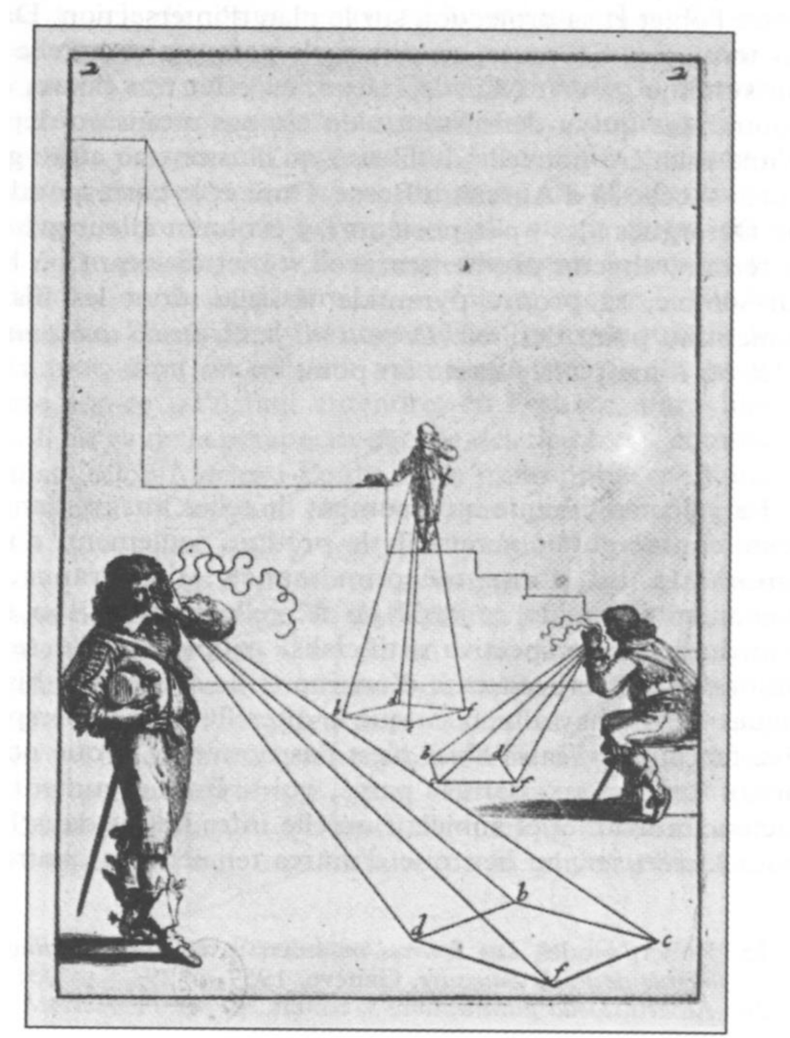






Cameras

```
For every pixel
  Construct a ray from the eye
  For every object in the scene
    Find intersection with ray
      Keep if closest
```

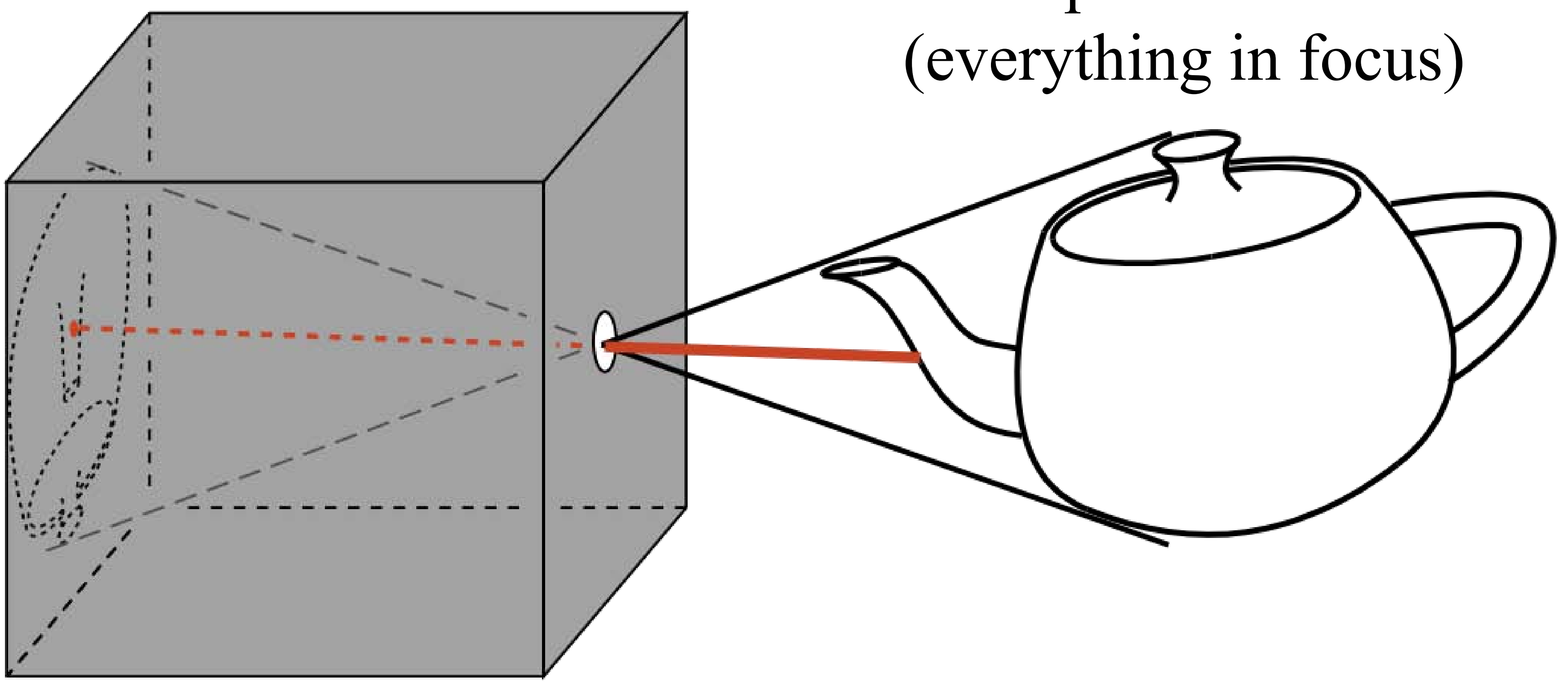


Abraham Bosse, *Les Perspecteurs*. Gravure extraite de la *Manière*

Pinhole Camera

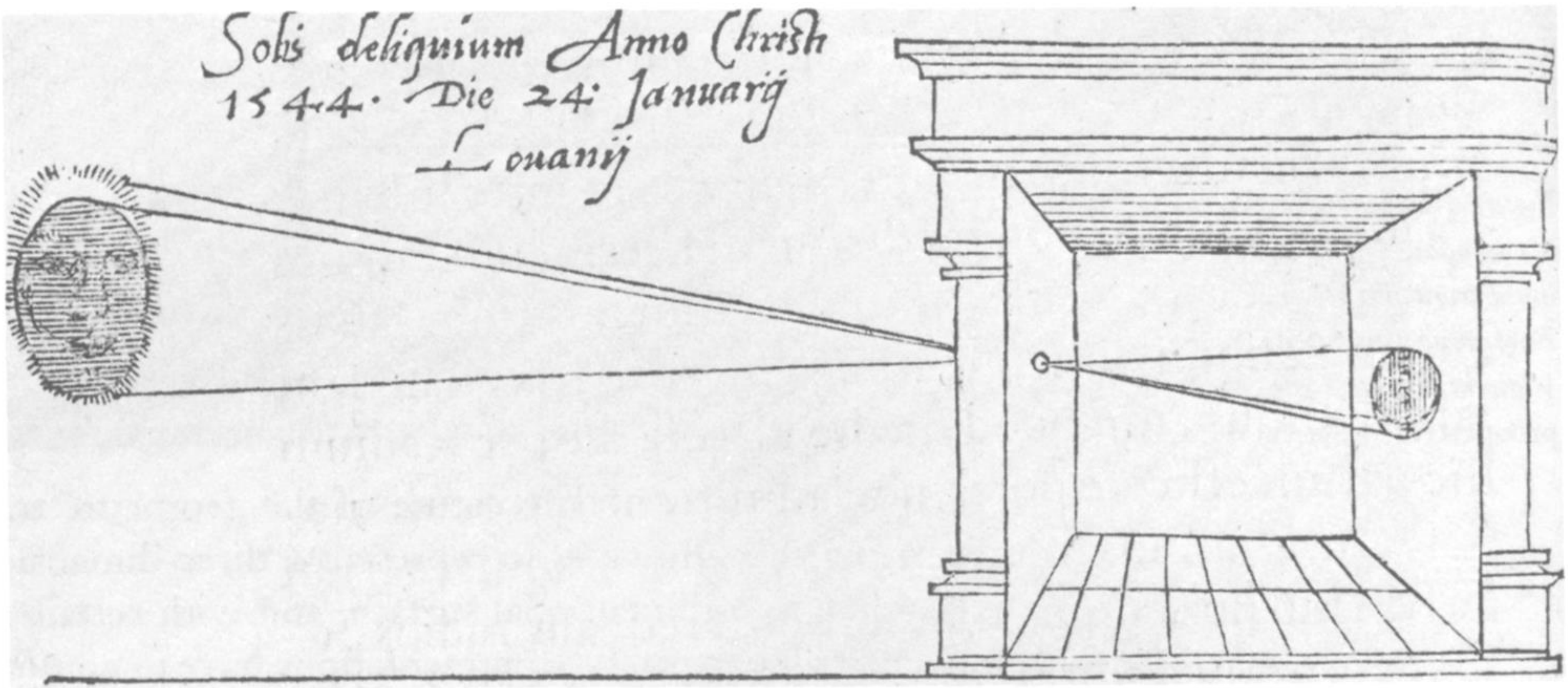
- Box with a tiny hole
- Inverted image
- Similar triangles

- Perfect image if hole infinitely small
- Pure geometric optics
- No depth of field issue (everything in focus)

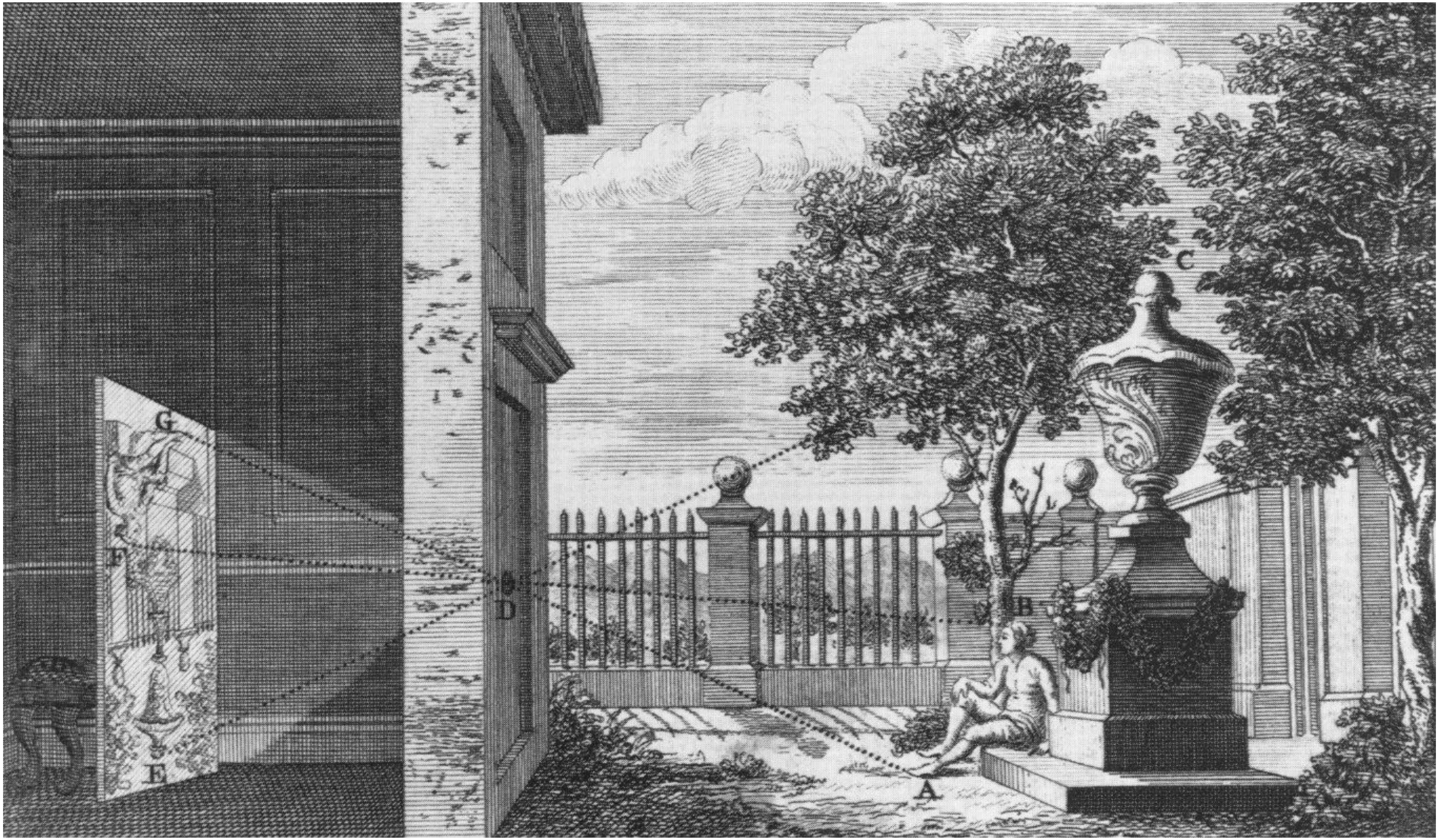


Oldest Illustration

- From Gemma Frisius, 1545



Also Called "Camera Obscura"



Camera Obscura Today

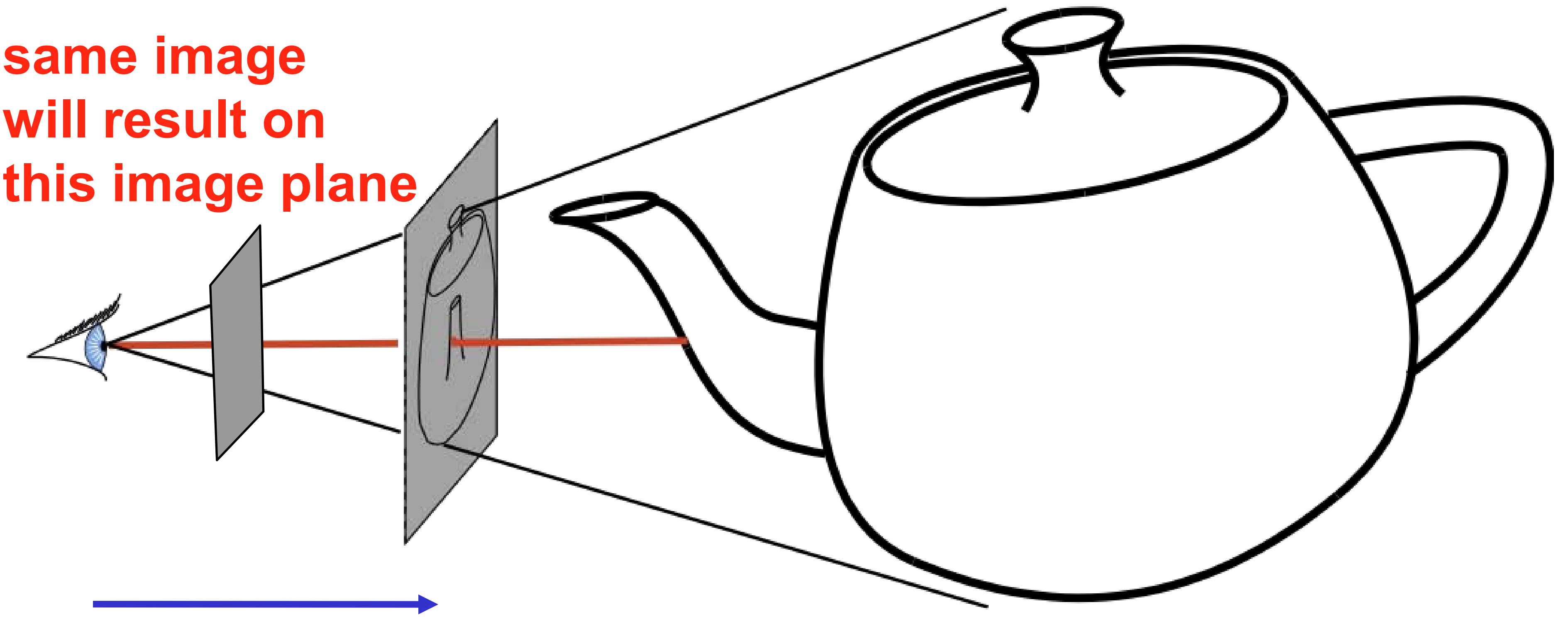
Images removed due to copyright restrictions -- please see
http://www.abelardomorell.net/photography/cameraobsc_01/cameraobsc_17.html
http://www.abelardomorell.net/posts/camera-obscura/
http://www.abelardomorell.net/photography/cameraobsc_49/cameraobsc_63.html
for further details.

Abelardo Morell

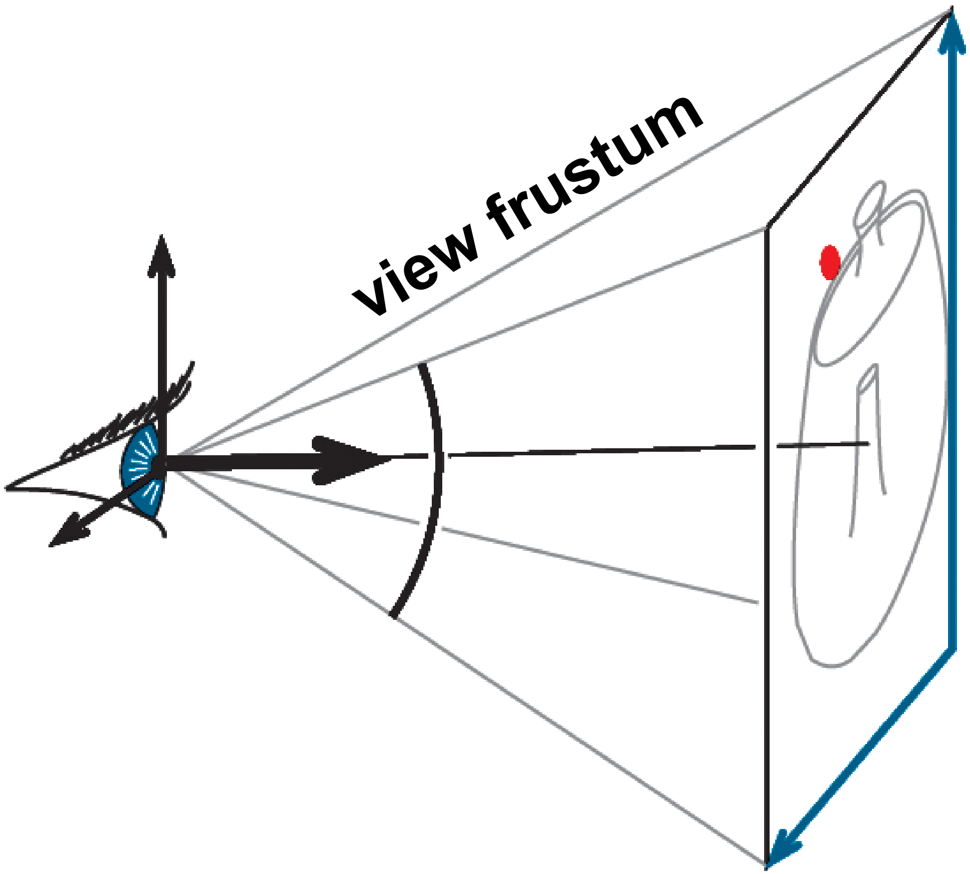
www.abelardomorell.net

Simplified Pinhole Camera

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



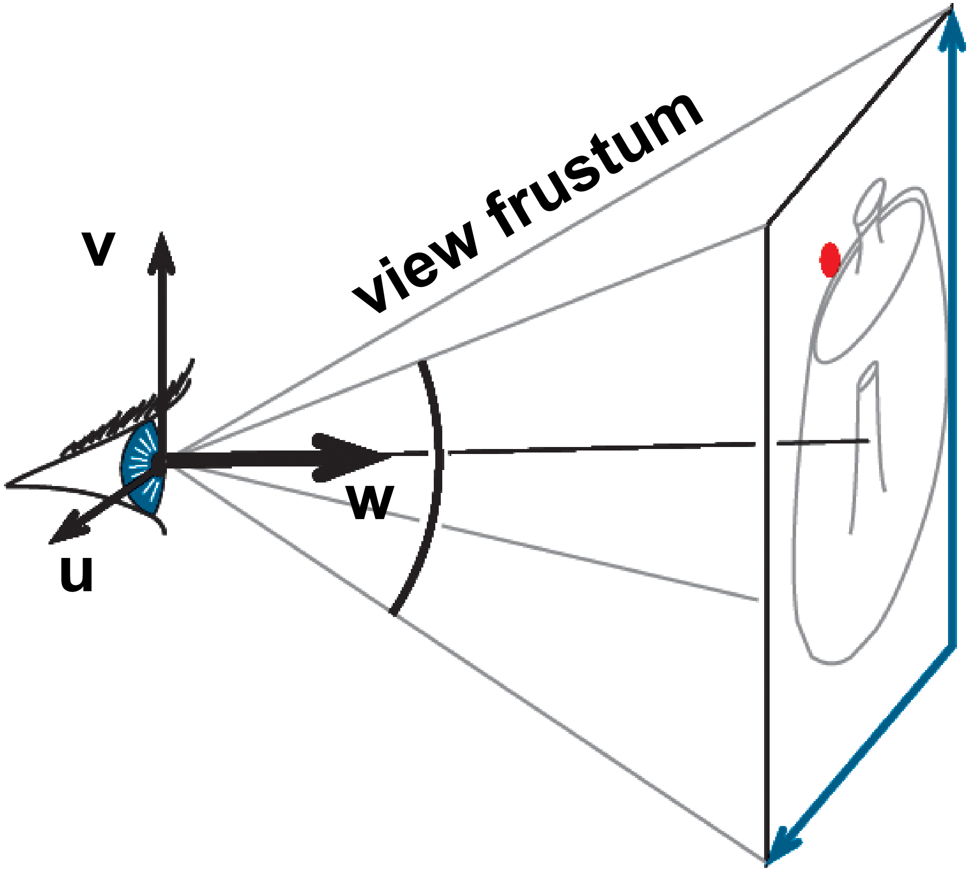
Camera Description?



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

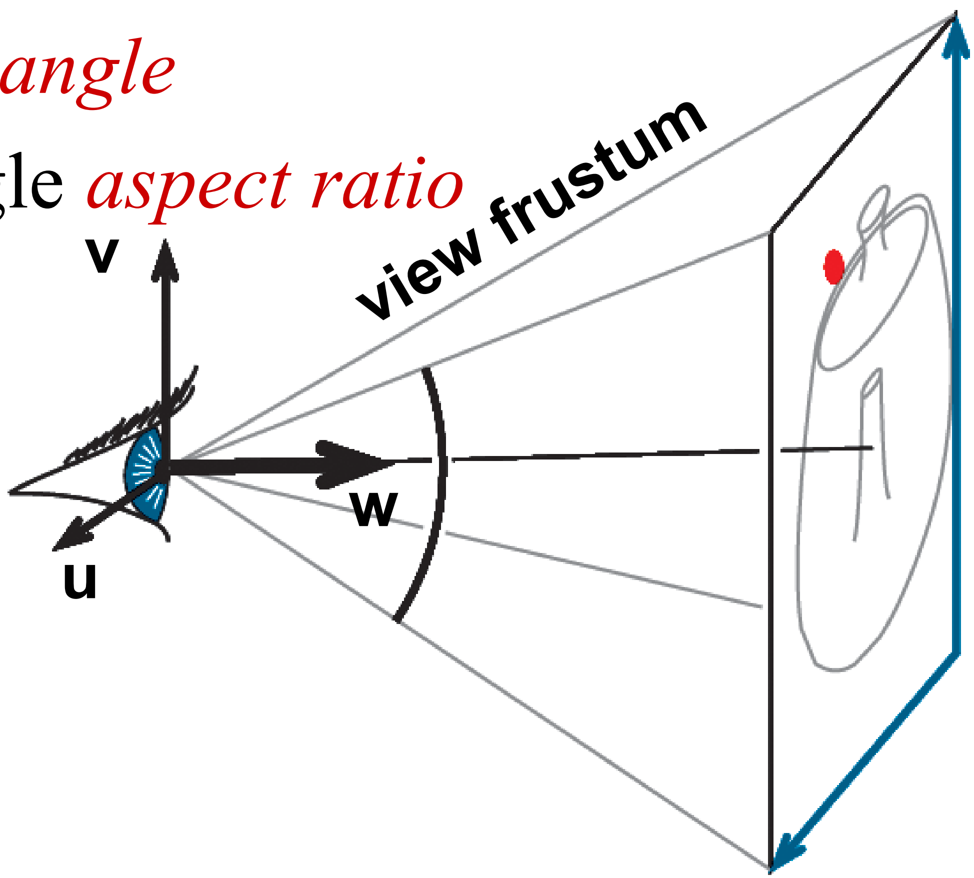
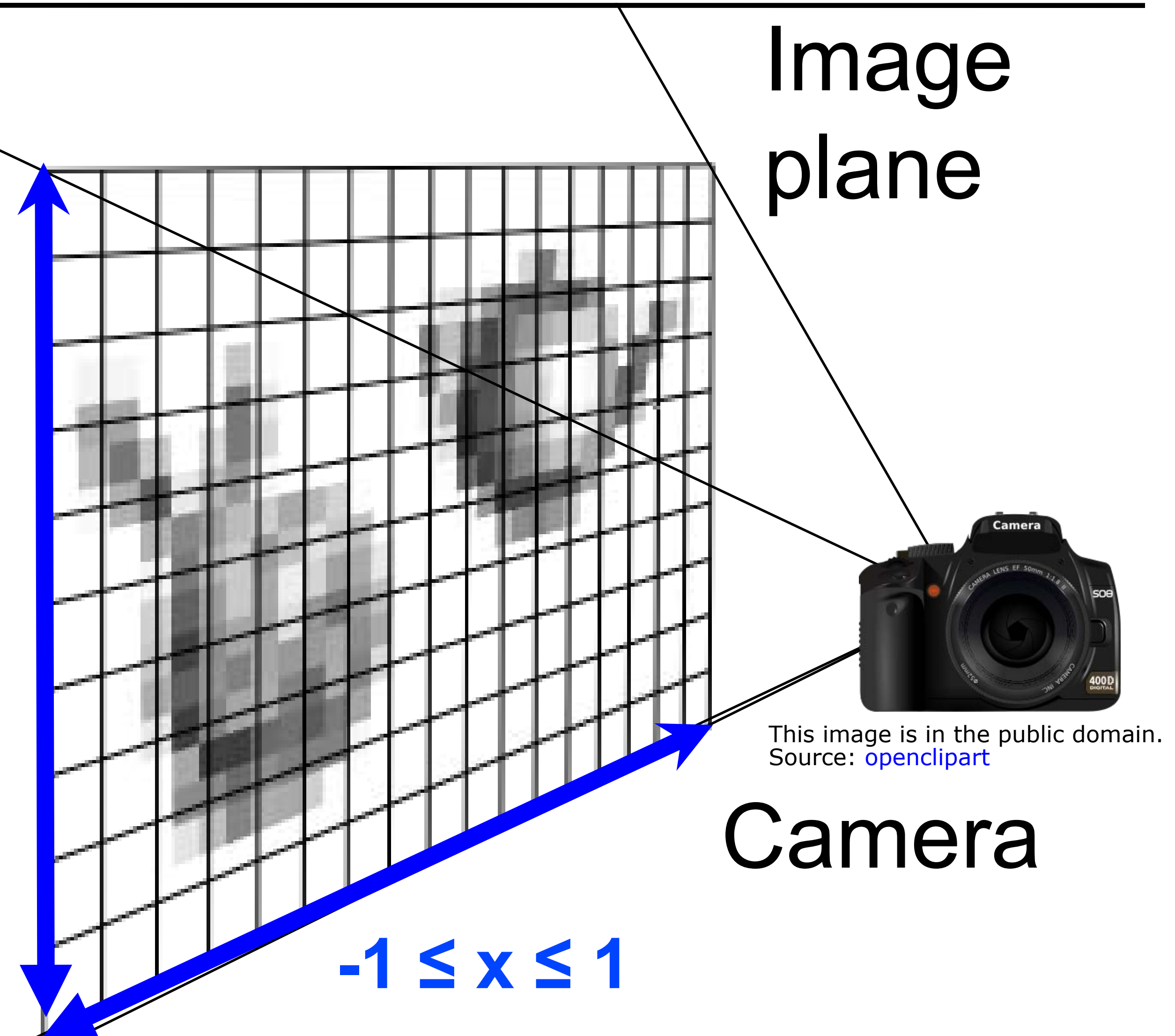
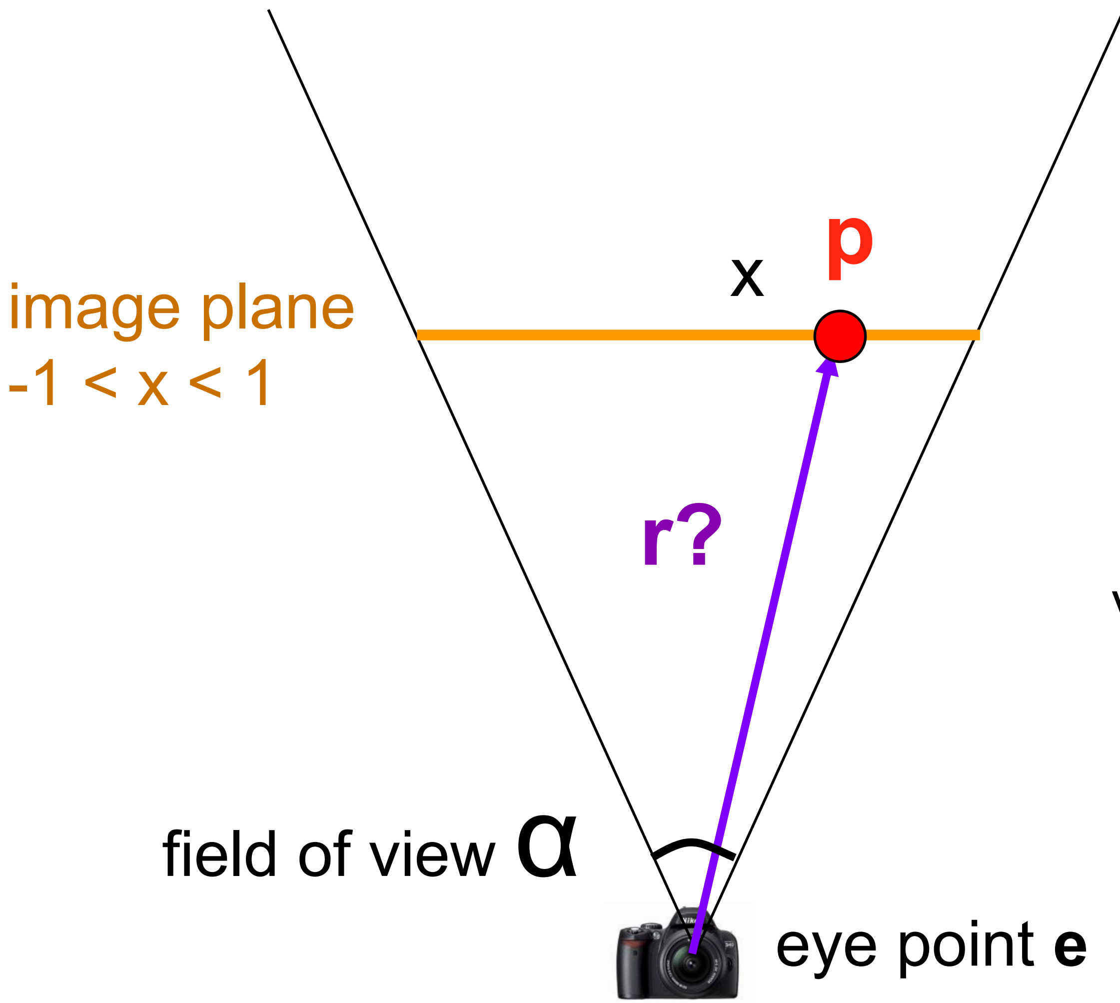


Image Coordinates

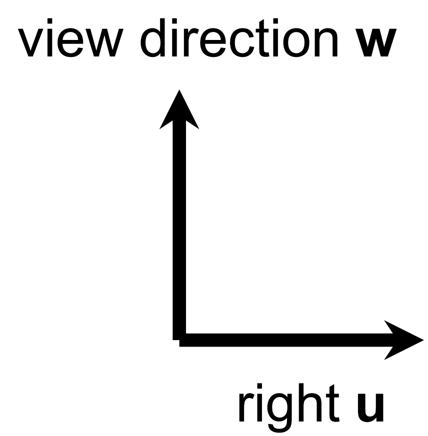


Convenient to define "normalized image coordinates" such that the **x, y coordinates run from -1 to 1 regardless of the dimensions and aspect ratio of the image rectangle.**

Ray Generation in 2D

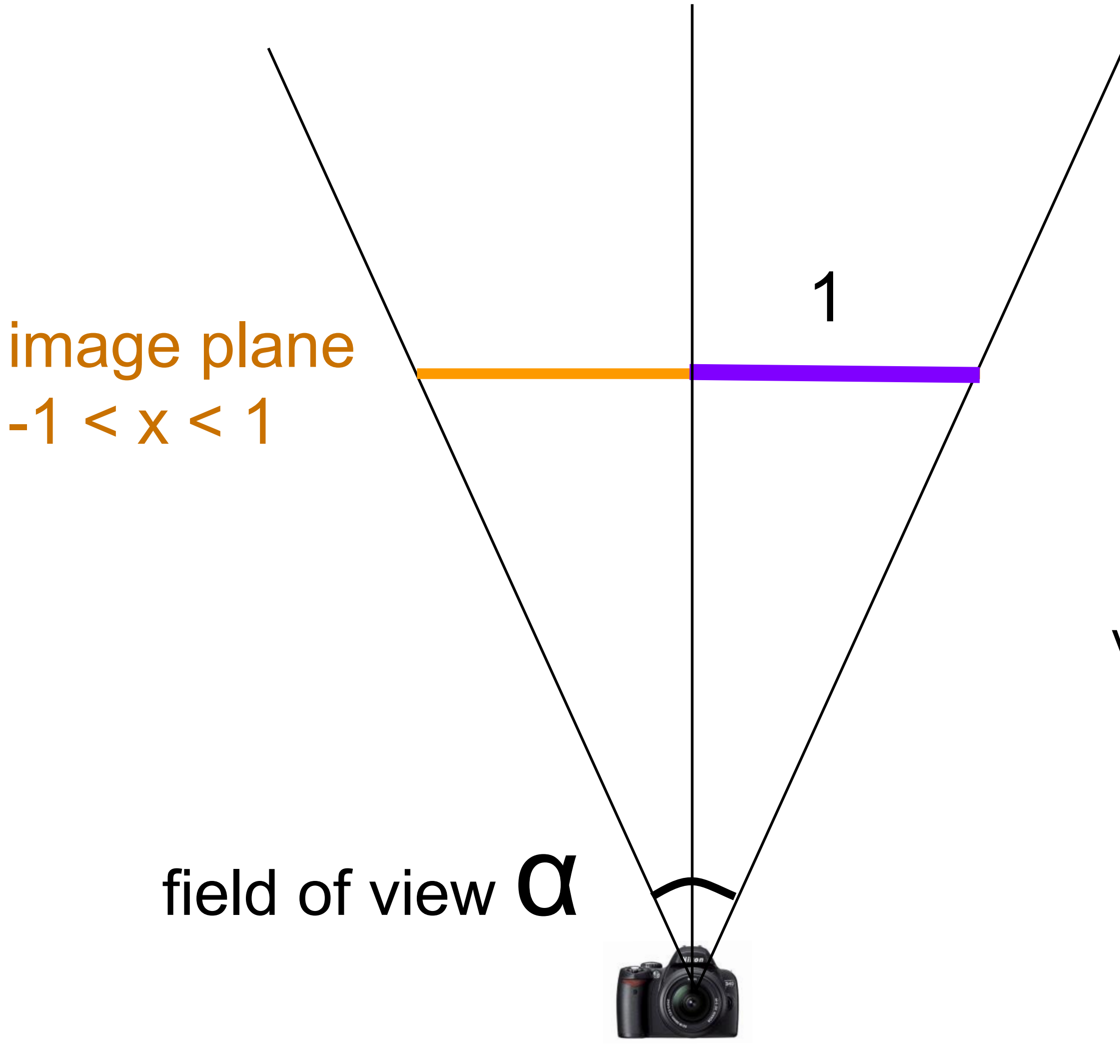


p is point on image plane at coordinate x, we want to know the direction of the ray **r**



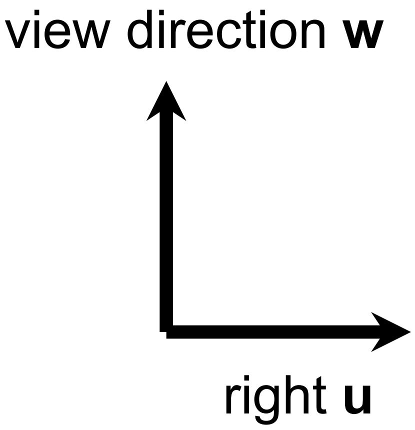
This image is in the public domain. Source: openclipart

Ray Generation in 2D

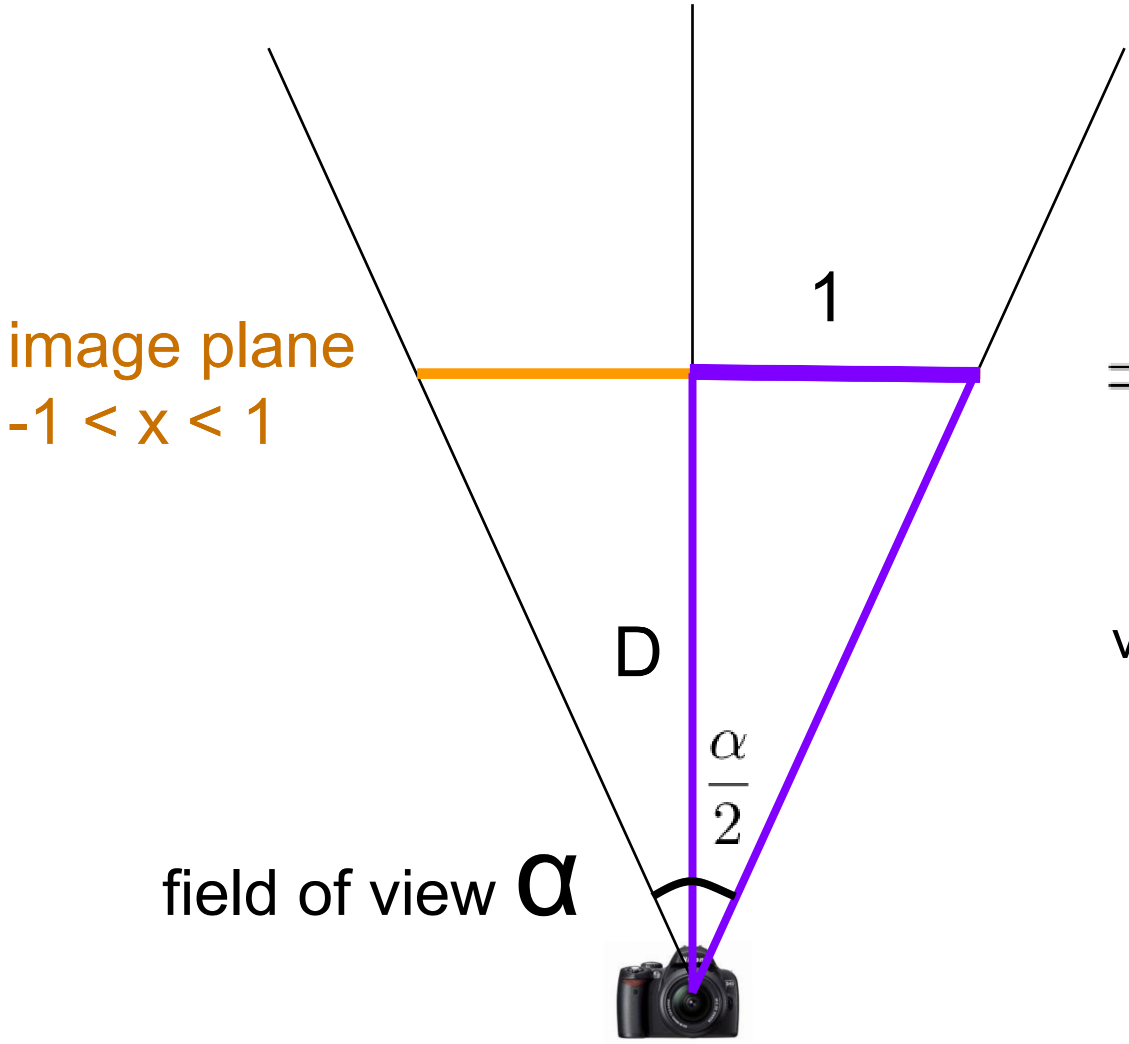


This image is in the public domain. Source: openclipart

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

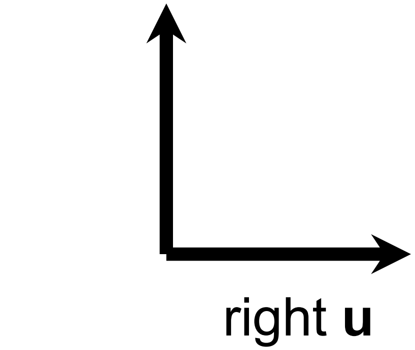


Ray Generation in 2D



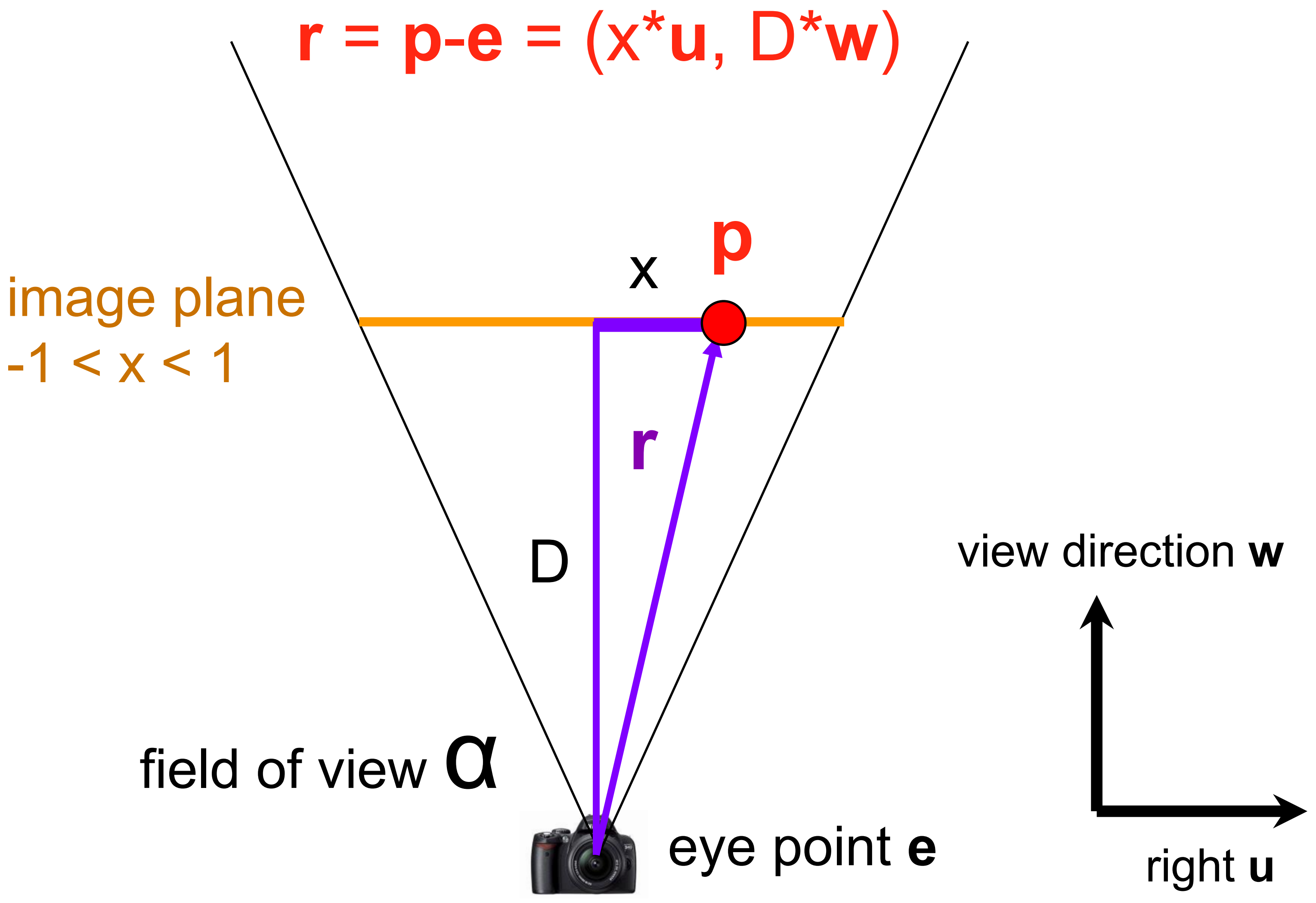
$$\tan\frac{\alpha}{2} = \frac{1}{D}$$

$$\Rightarrow D = \frac{1}{\tan(\alpha/2)}$$

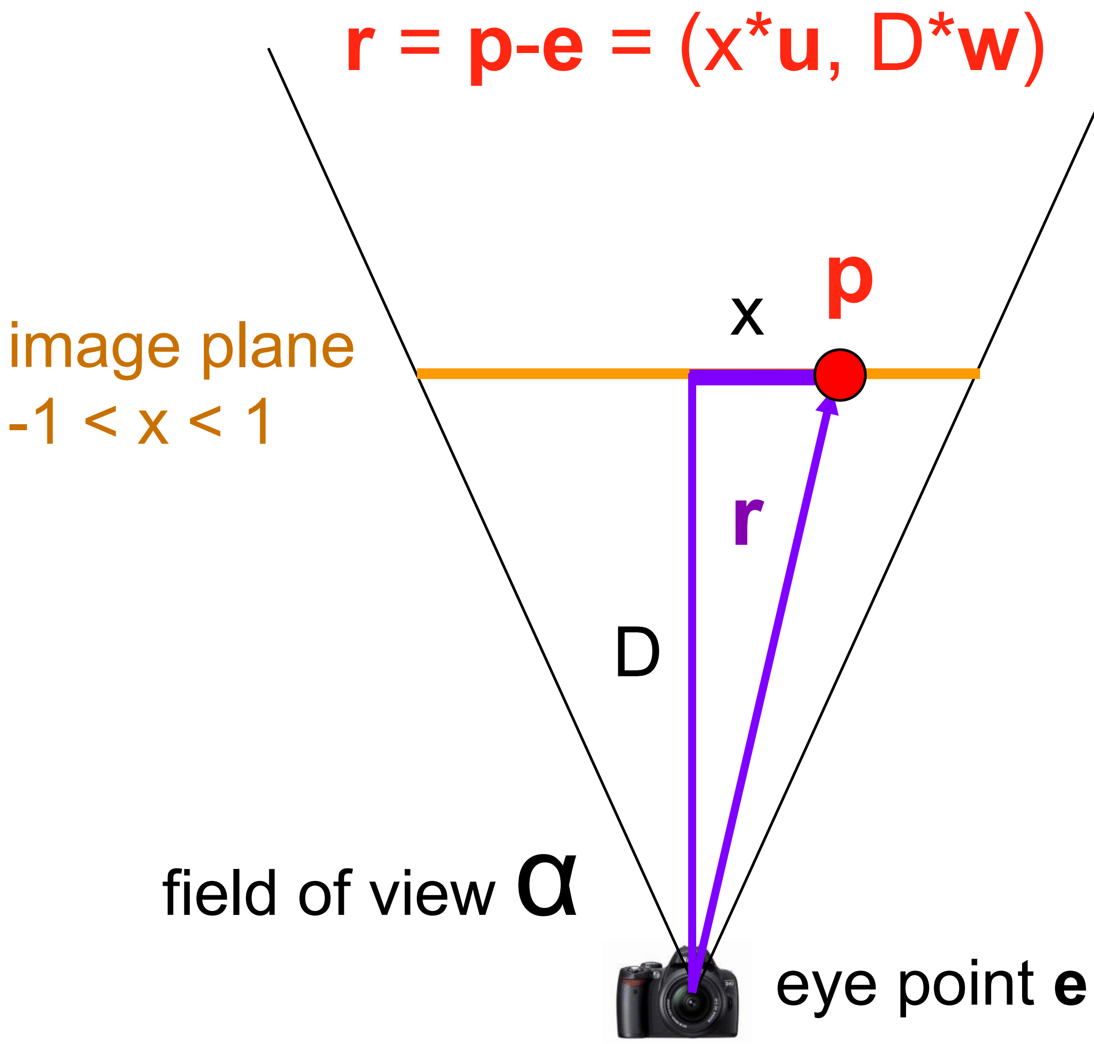


This image is in the public domain. Source: openclipart

Ray Generation in 2D



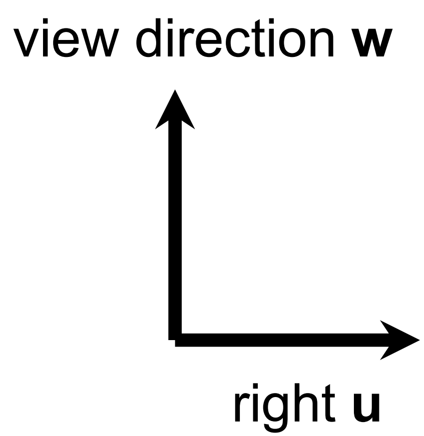
Ray Generation in 2D



then we just normalize **r** to get the ray

$$\boldsymbol{P}(t) = \boldsymbol{e} + t\boldsymbol{d}$$

$$(\boldsymbol{d} = \boldsymbol{r} / ||\boldsymbol{r}||)$$



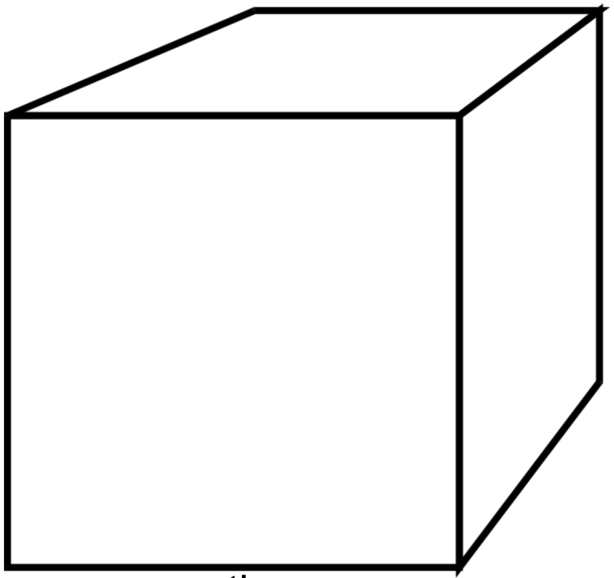
This image is in the public domain. Source: openclipart

That was 2D, 3D is just as simple

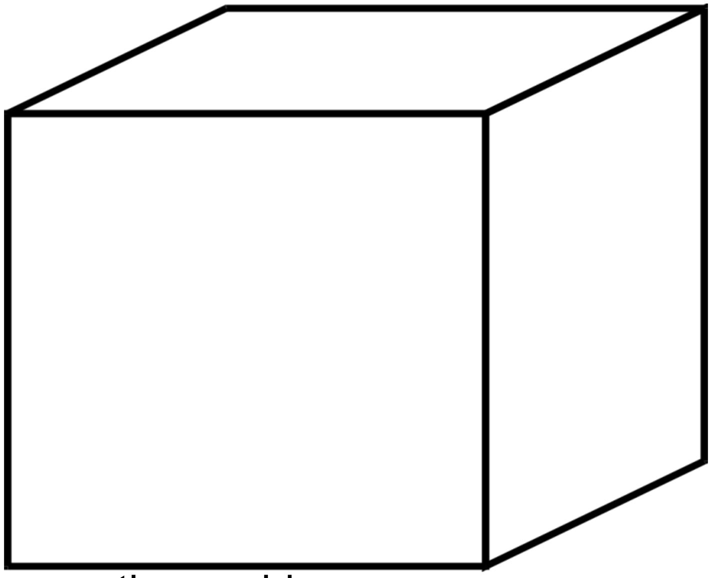
- *y* coordinate is treated just like *x*, except accounting for aspect ratio
  - **r** = (x***u**, aspect*y***v**, D***w**)
  - Again, **u**, **v**, **w** are the basis vectors of the view coordinate system
  - Aspect ratio handles non-square viewports
    - Think of your 16:9 widescreen TV

- 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



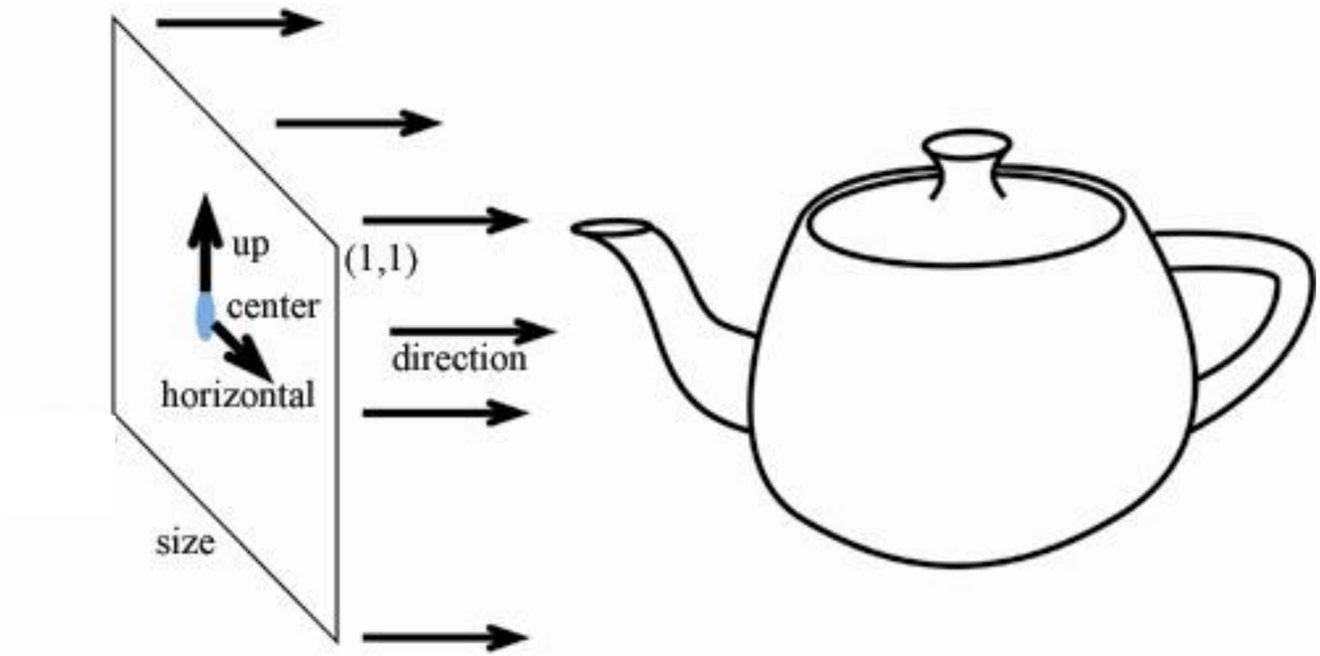
perspective



orthographic

- Parallel projection
- No foreshortening
- No vanishing point

Orthographic Camera



- Ray Generation?
  - Origin = **e** + x*size***u** + y*size***v**
  - Direction is constant: **w**

Other Weird Cameras

- E.g. fish eye, omnimax, parabolic

CAVE Columbia University

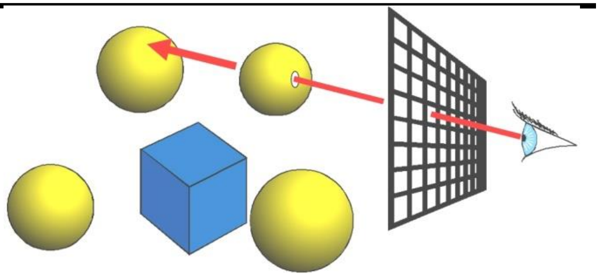
Questions?

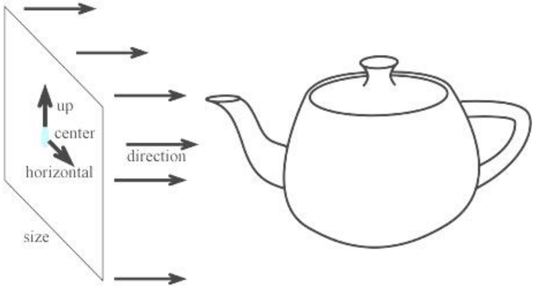
Courtesy of Paul Rademacher. Used with permission.

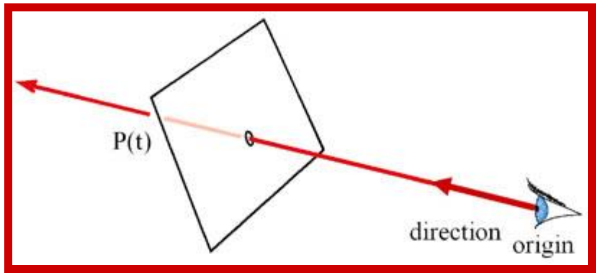
Even Funkier Multiperspective Imaging

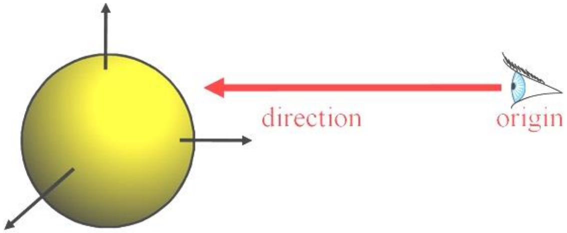
Ray Casting

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





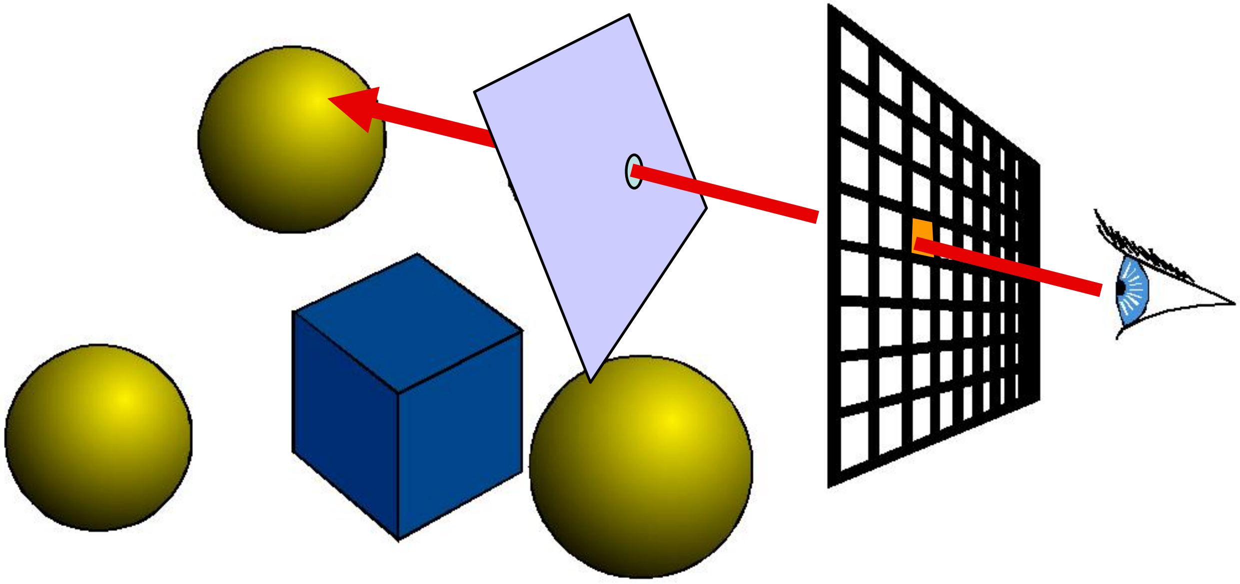




Ray Casting

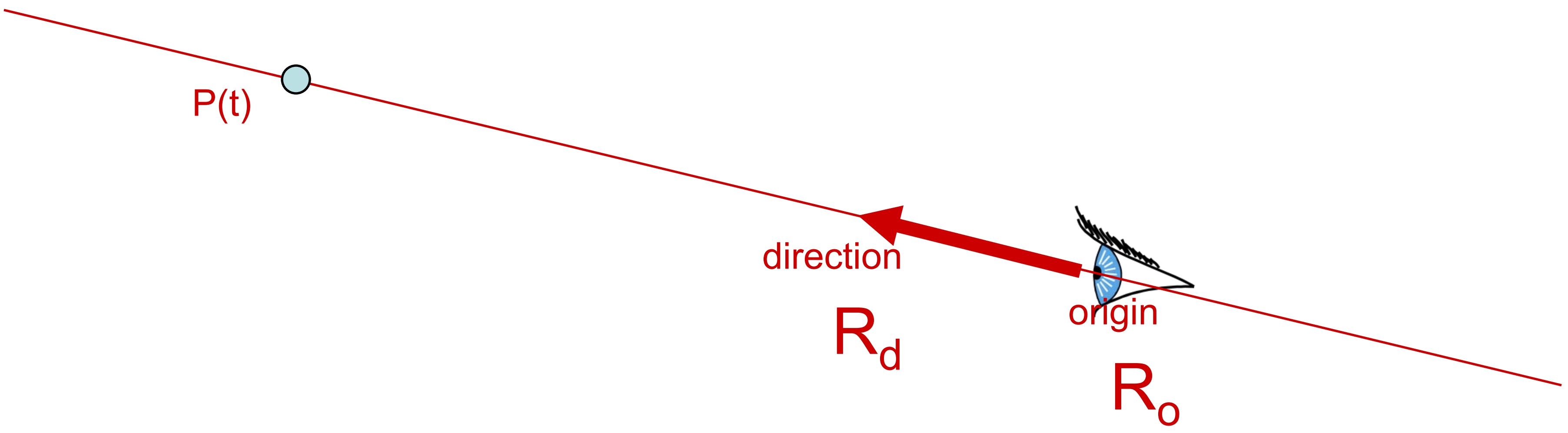
```
For every pixel
   Construct a ray from the eye
   For every object in the scene
       Find intersection with the ray
       Keep if closest
```

First we will study ray-plane intersection



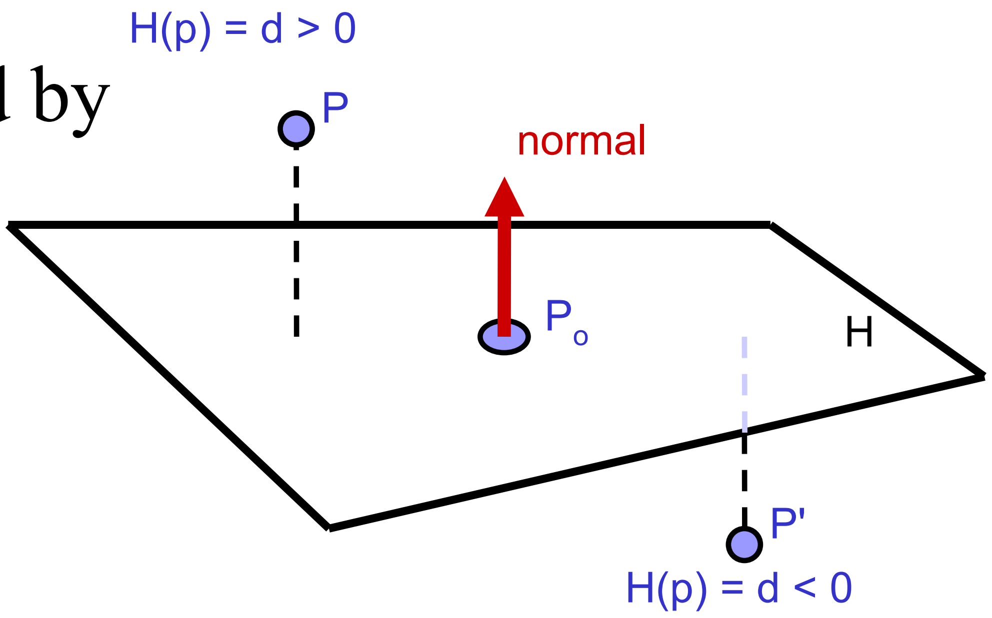
Recall: Ray Representation

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



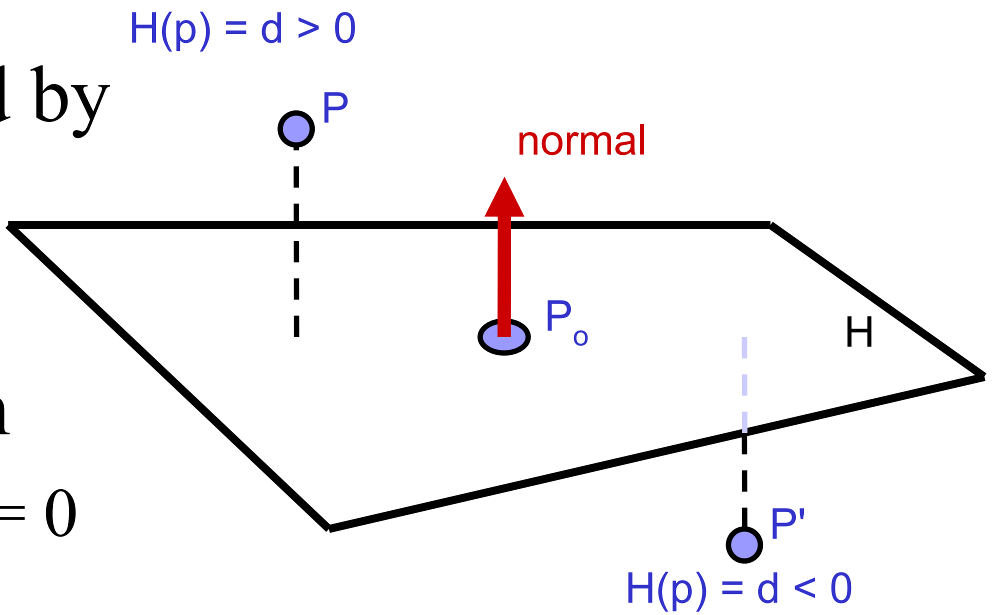
3D Plane Representation?

- (Infinite) plane defined by
  - $P_o = (x_0, y_0, z_0)$
  - $n = (A,B,C)$



3D Plane Representation?

- (Infinite) plane defined by
  - $P_o = (x_0, y_0, z_0)$
  - $n = (A,B,C)$
- Implicit plane equation
  - $H(P) = Ax+By+Cz+D = 0$
    $= n \cdot P + D = 0$

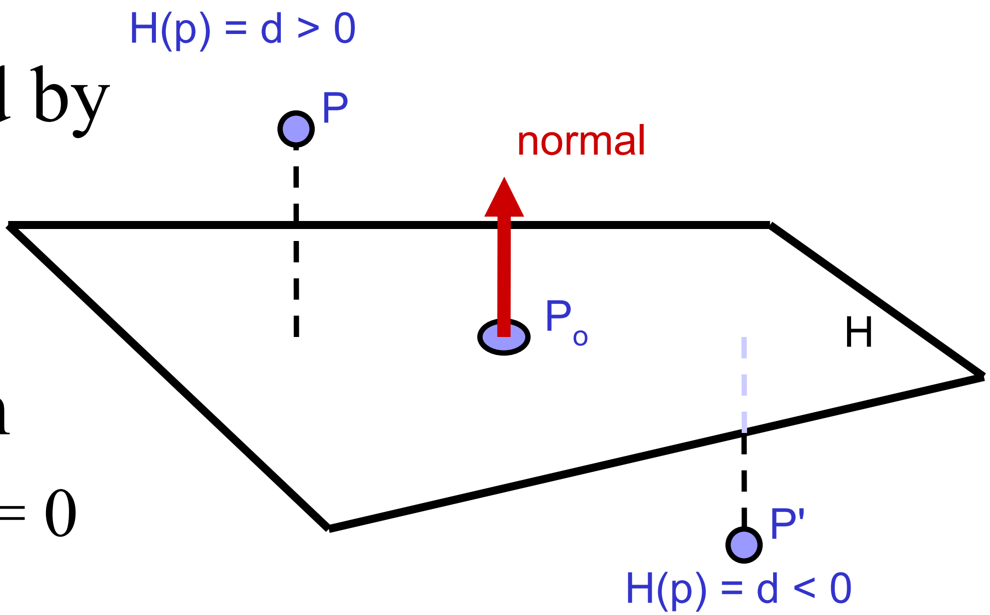


3D Plane Representation?

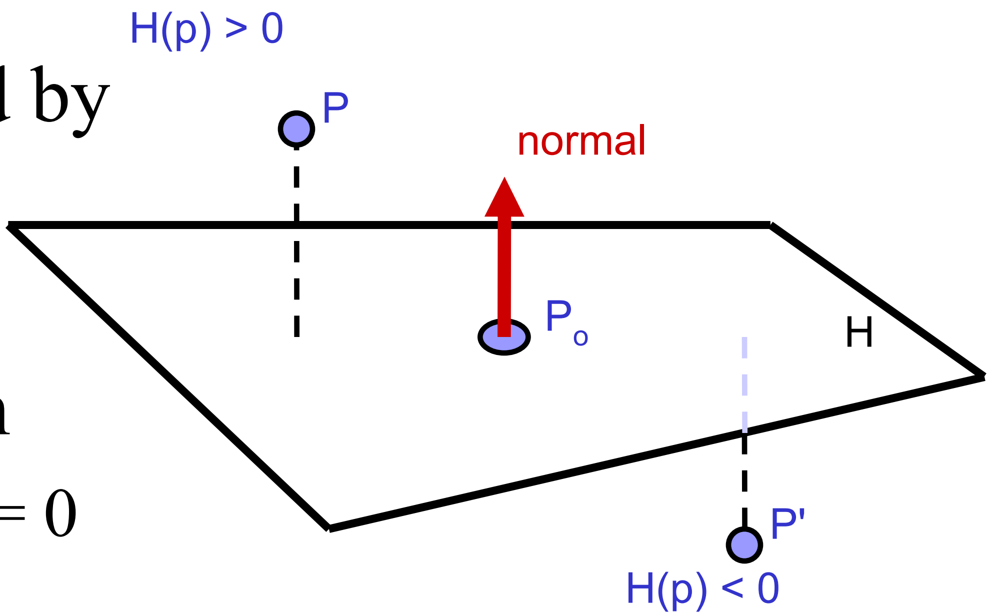
- (Infinite) plane defined by
  - $P_o = (x_0, y_0, z_0)$
  - $n = (A,B,C)$
- Implicit plane equation
  - $H(P) = Ax+By+Cz+D = 0$
    $= n \cdot P + D = 0$
  - What is D?

$$Ax_0 + By_0 + Cz_0 + D = 0$$ (Point P_0 must lie on plane)

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



3D Plane Representation?



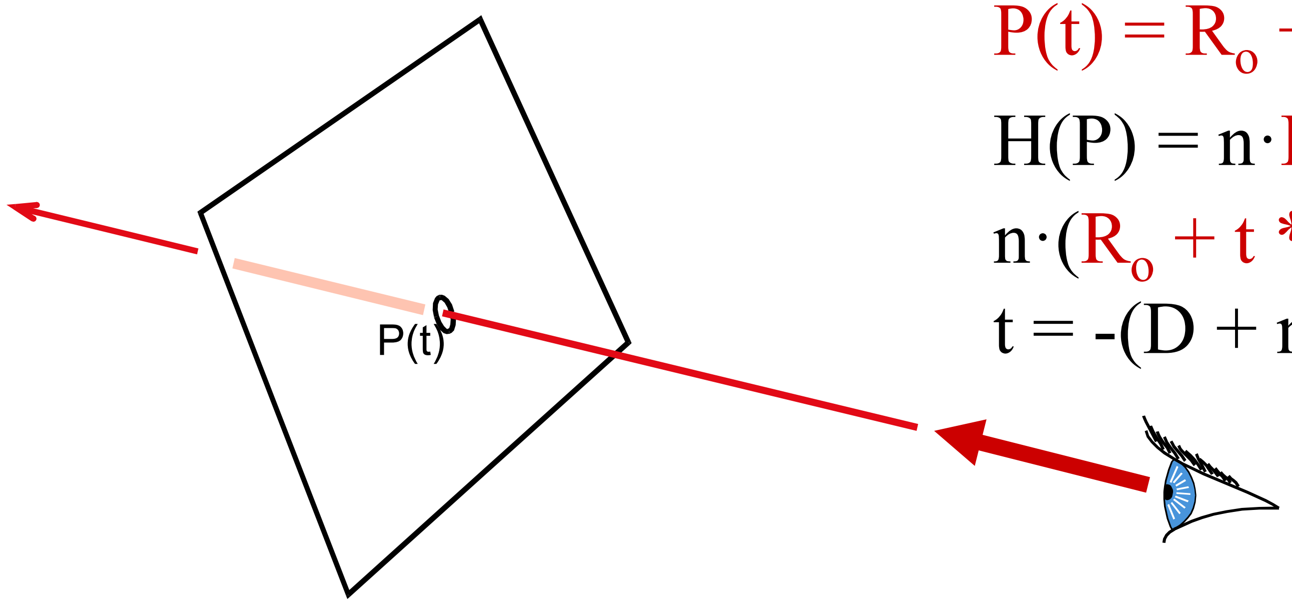
- (Infinite) plane defined by
  - $P_o = (x_0, y_0, z_0)$
  - $n = (A,B,C)$
- Implicit plane equation
  - $H(P) = Ax+By+Cz+D = 0$
    $= n \cdot P + D = 0$
- Point-Plane distance?
  - If n is normalized,
    distance to plane is H(P)
  - it is a ***signed*** *distance*!

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

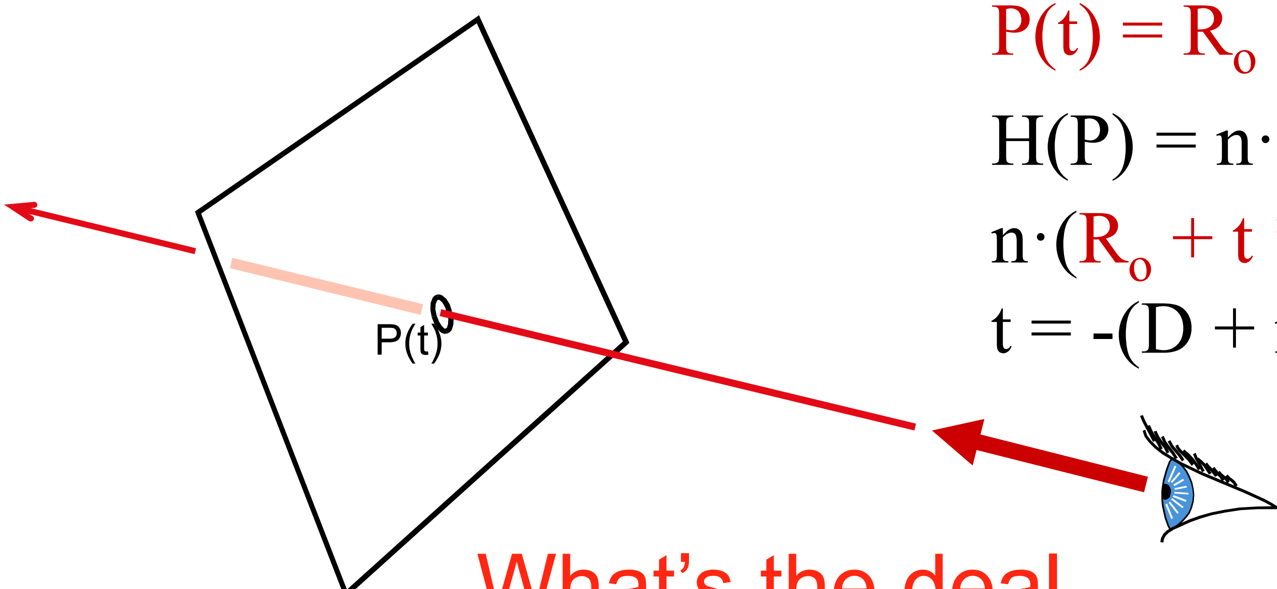
- Intersection means both are satisfied
- So, insert explicit equation of ray into implicit equation of plane & solve for t

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

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

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

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



Done!

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

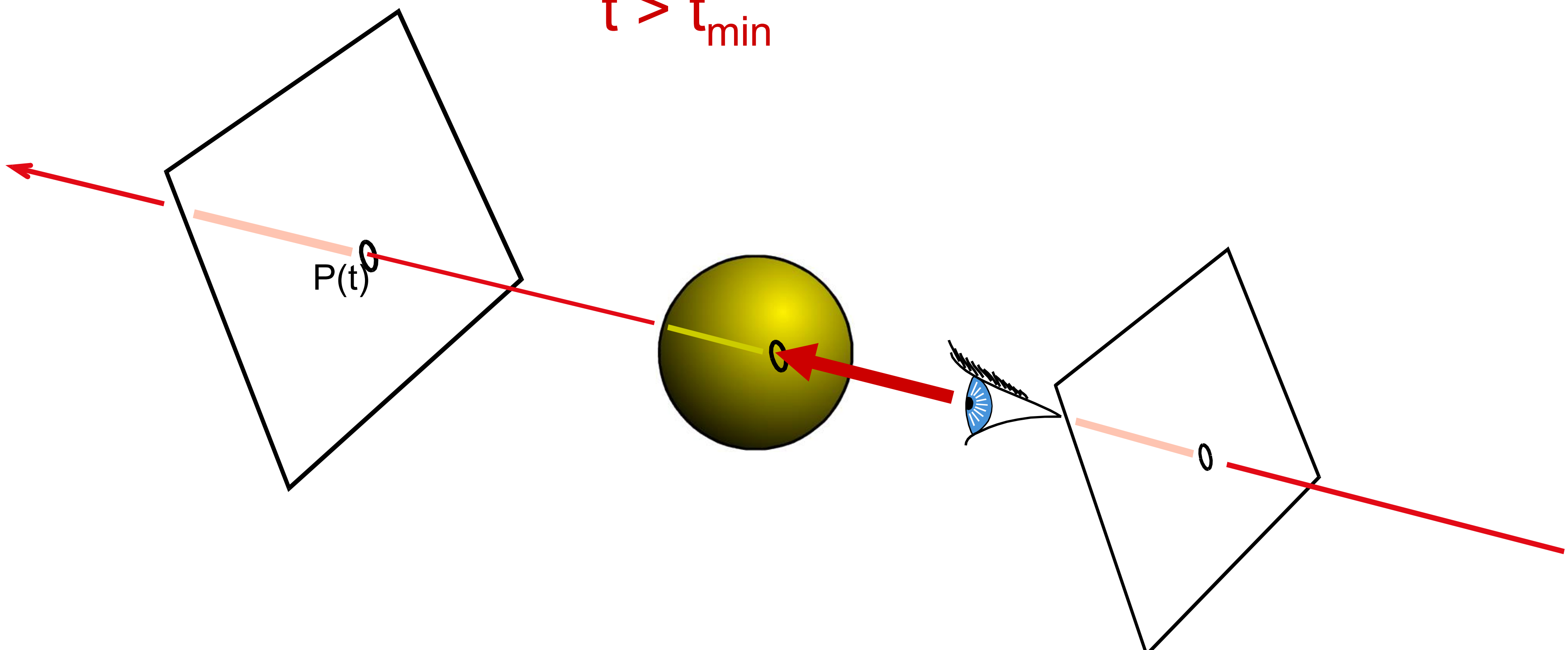
Additional Bookkeeping

- Verify that intersection is closer than previous

  $t < t_{current}$

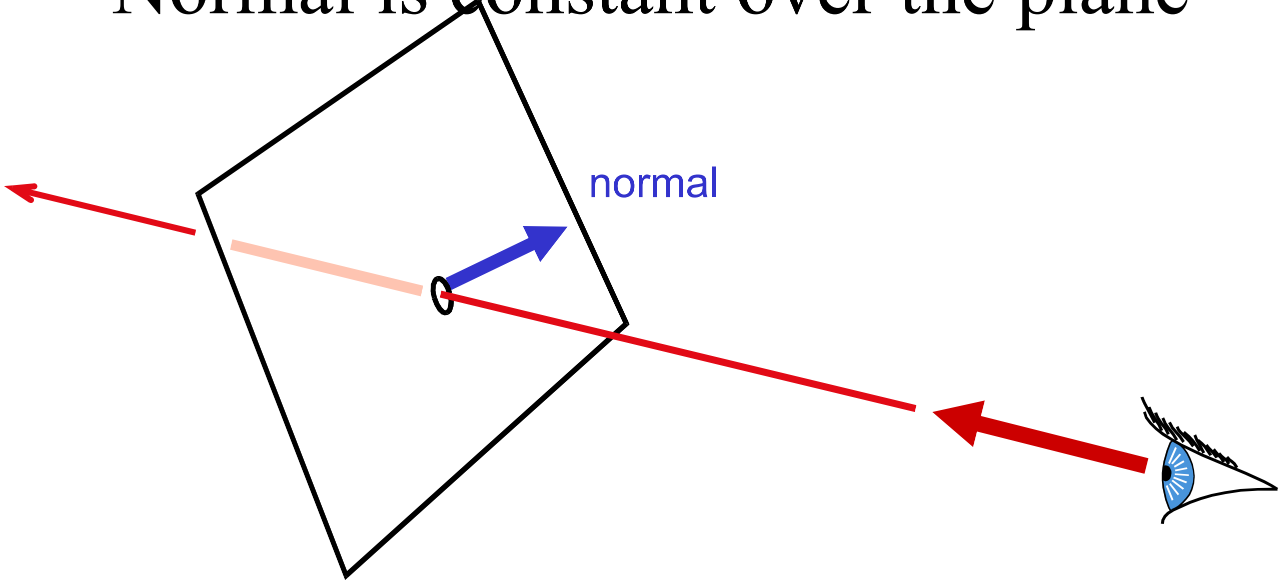
- Verify that it is not out of range (behind eye)

  $t > t_{min}$



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?

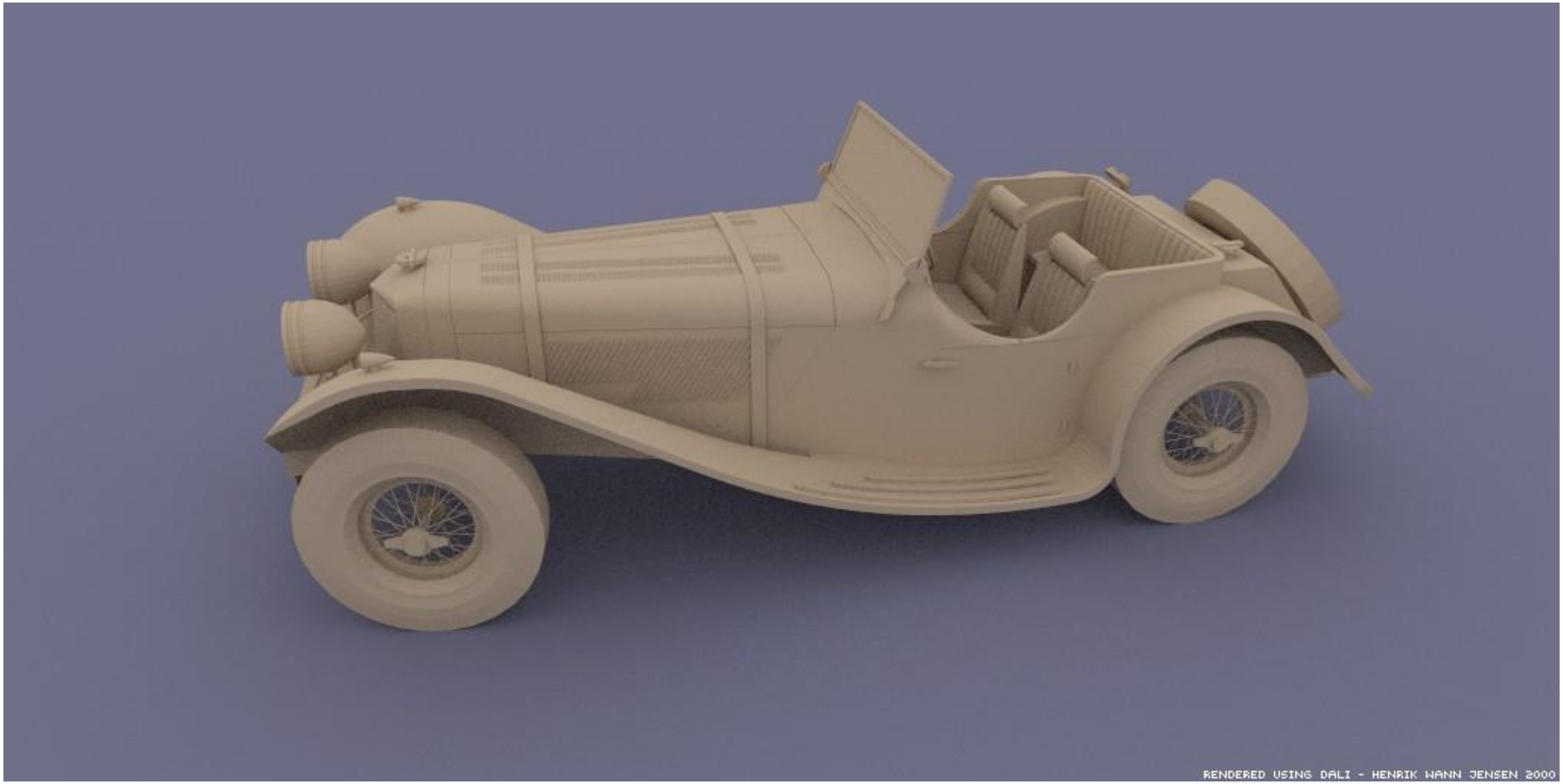
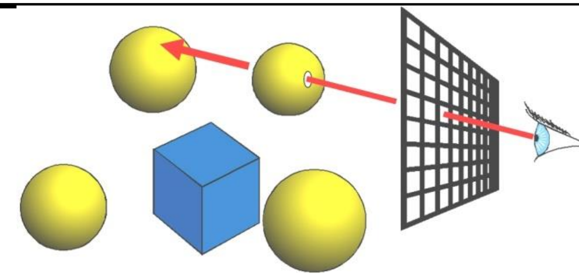


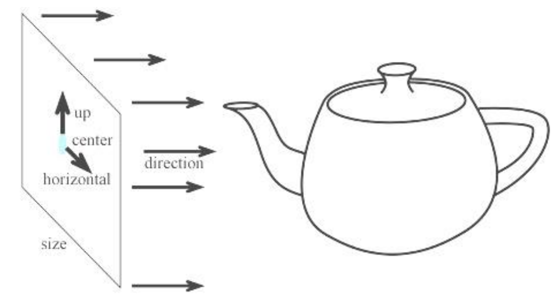
Image by Henrik Wann Jensen

Ray Casting

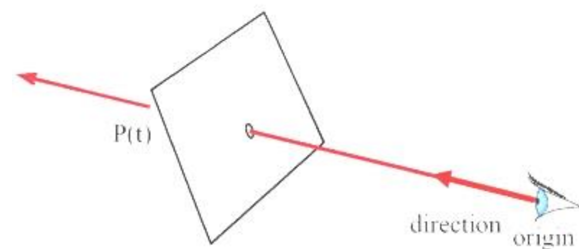
- Ray Casting Basics



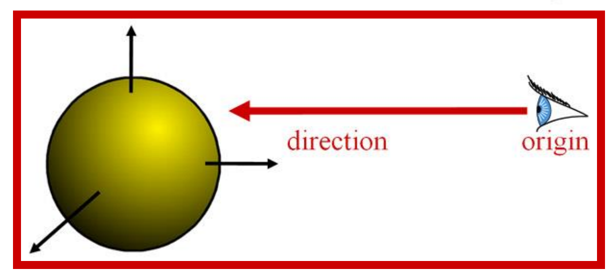
- Camera and Ray Generation



- Ray-Plane Intersection

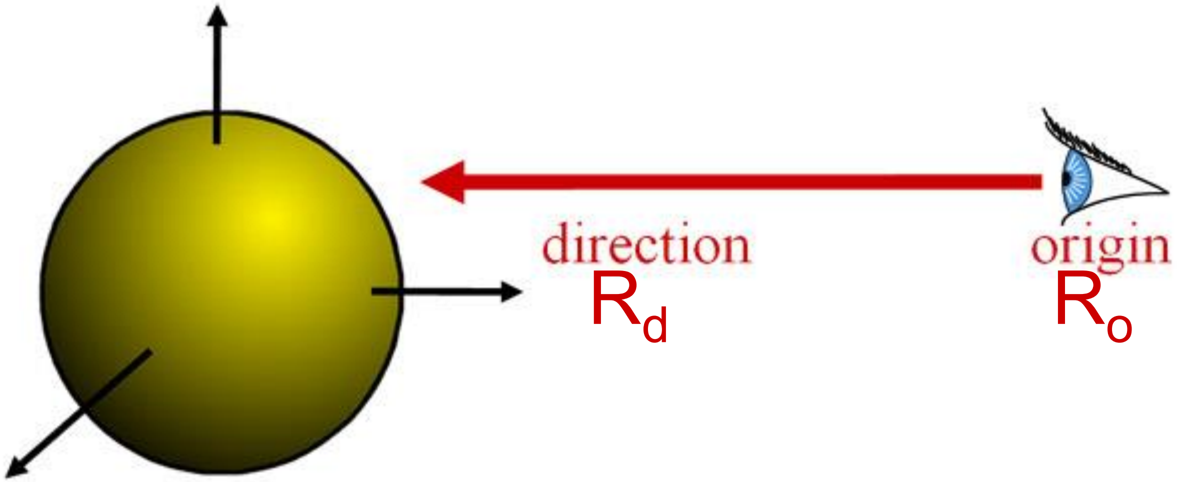


- Ray-Sphere Intersection



Sphere Representation?

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



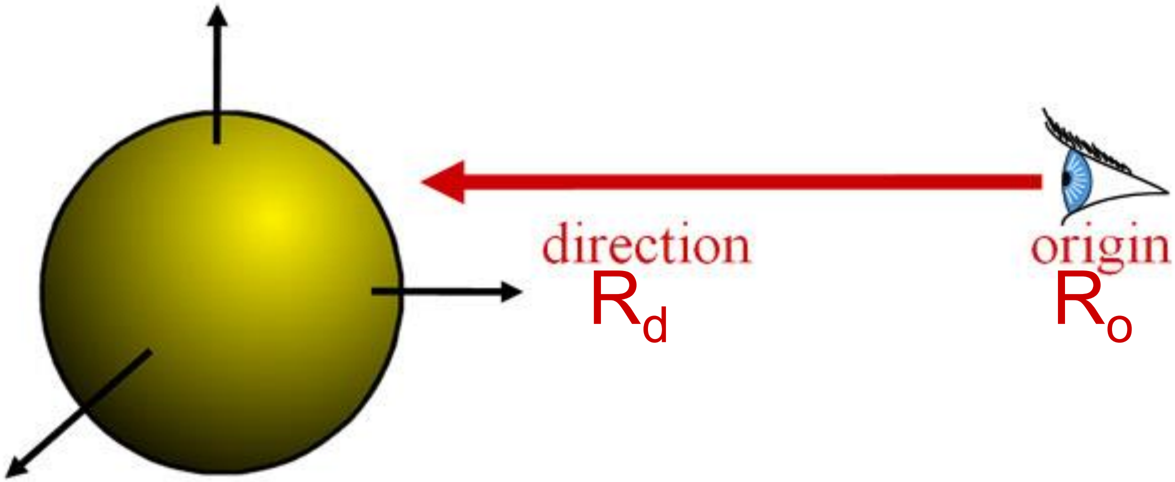
Ray-Sphere Intersection

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

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

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

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

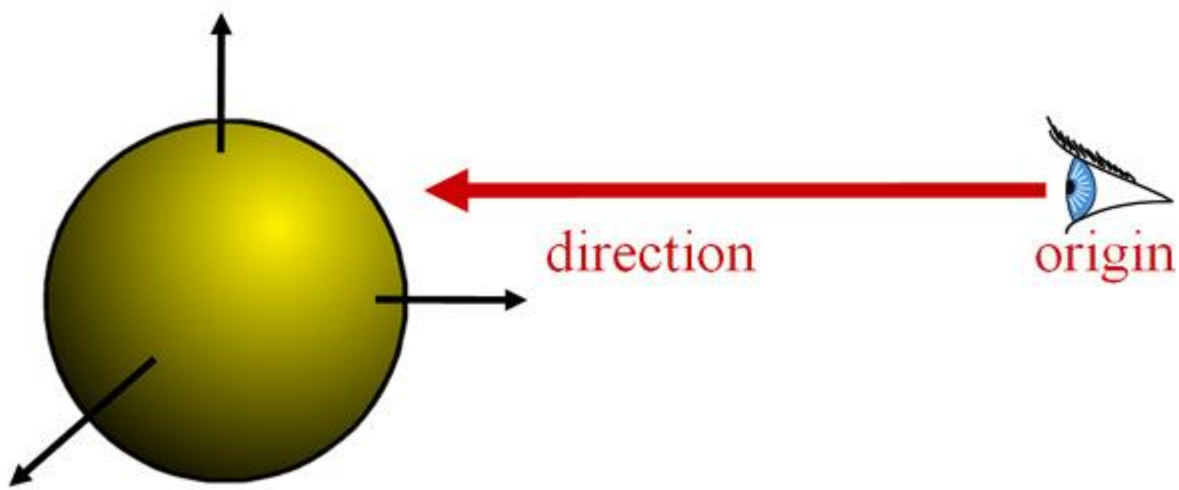


Ray-Sphere Intersection

- 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

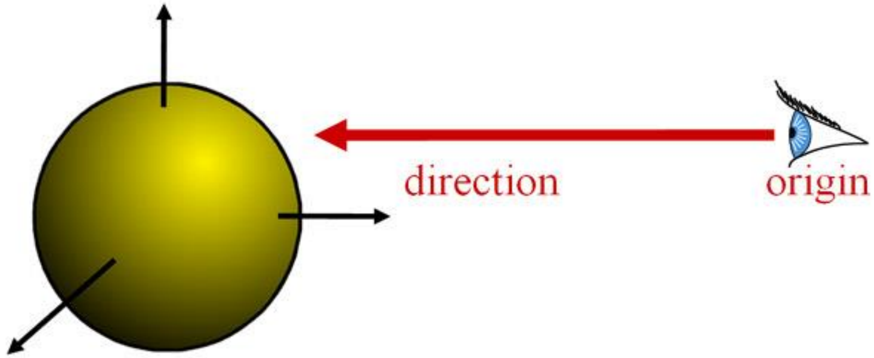
- 3 cases, depending on the sign of $b^2 - 4ac$
- What do these cases correspond to?
- Which root (t+ or t-) should you choose?
  - Closest positive!

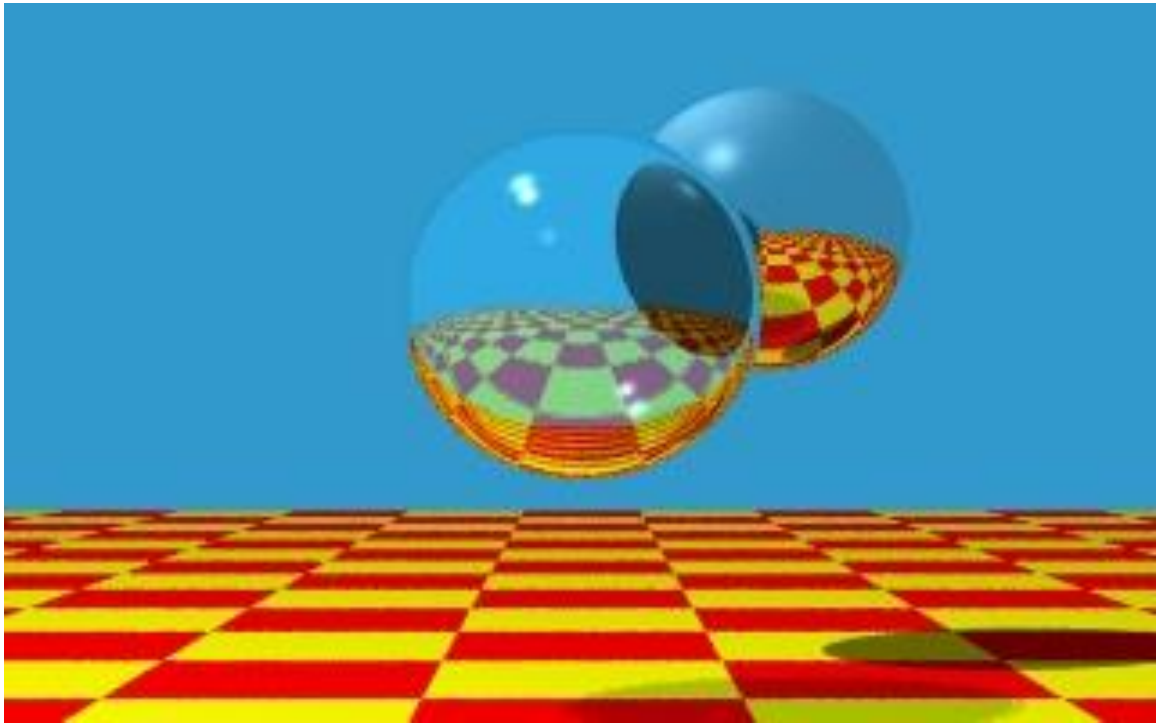


Ray-Sphere Intersection

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

:-)

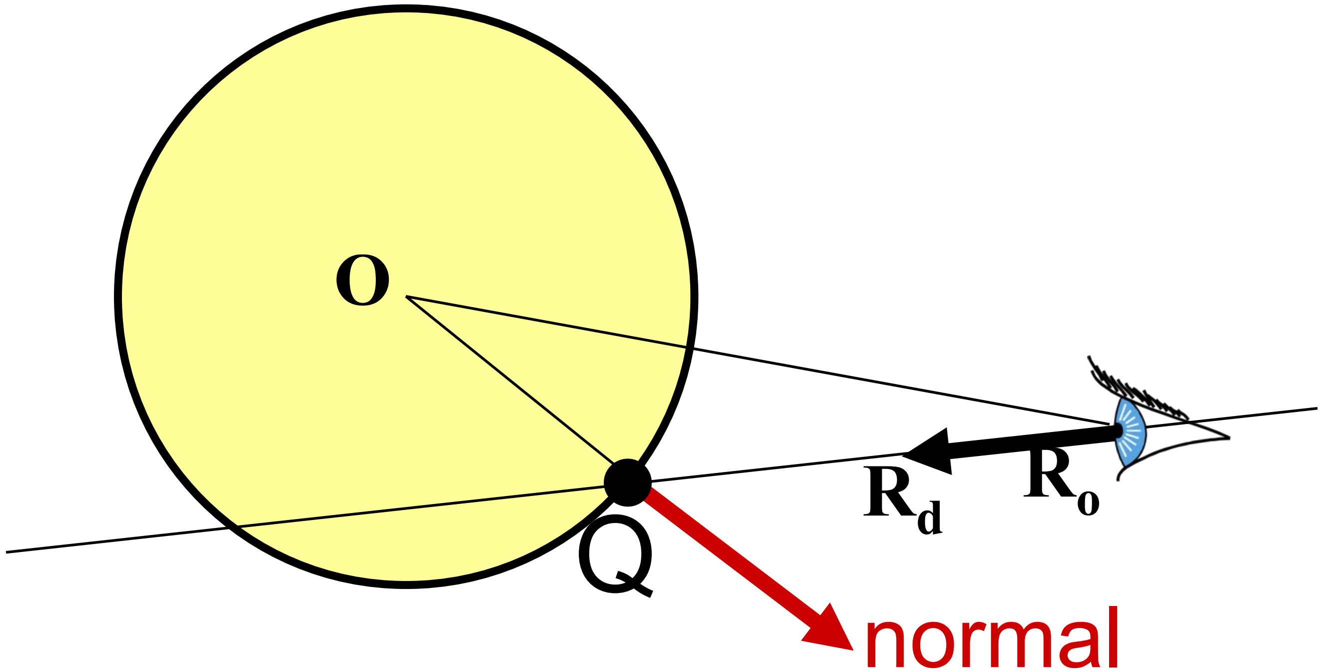




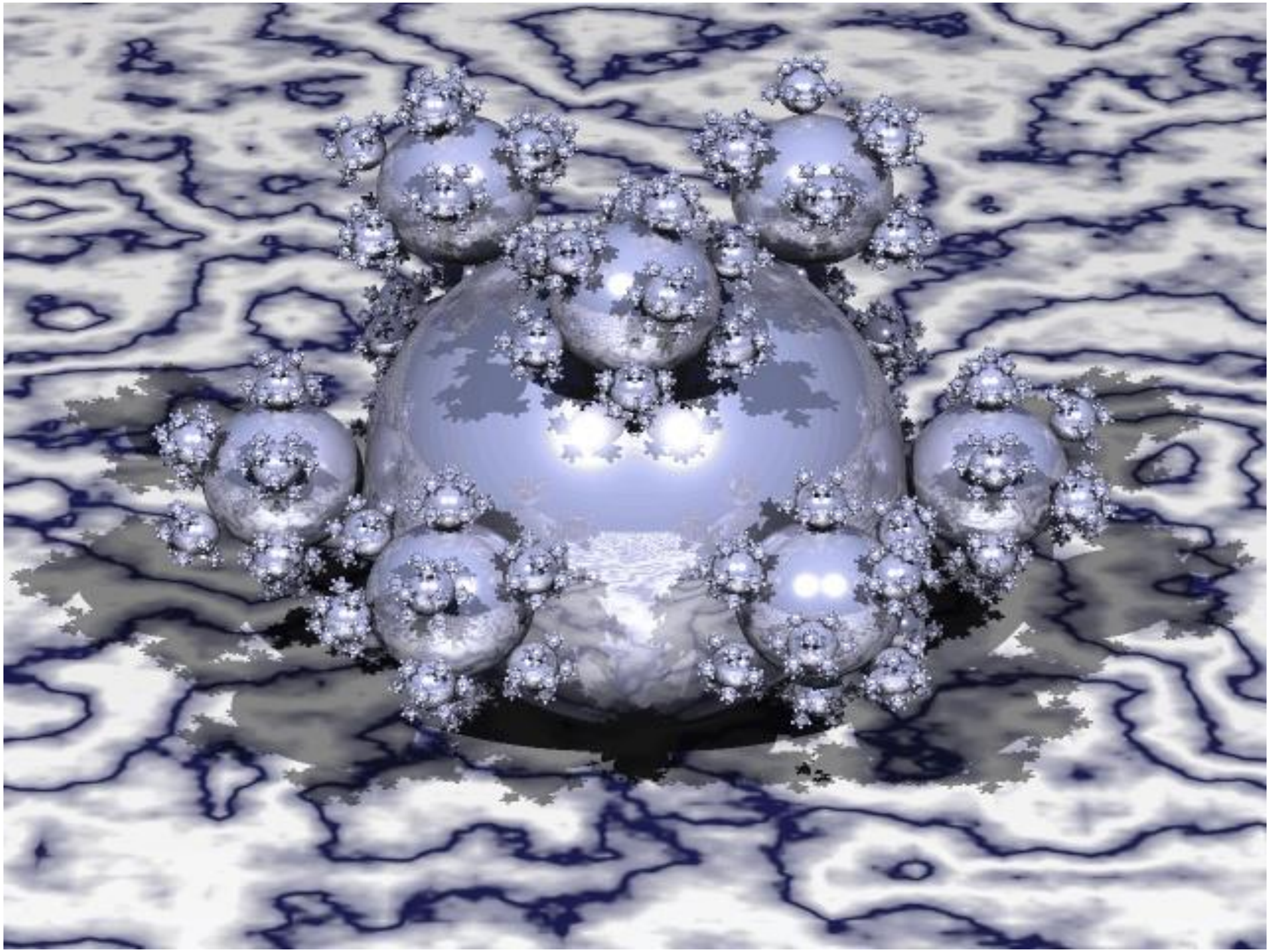
Turner Whitted

Sphere Normal

- Simply Q/||Q||
  - Q = P(t), intersection point
  - (for spheres centered at origin)



Questions?



Courtesy of Henrik Wann Jensen. Used with permission.

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

NVIDIA

6.837 Computer Graphics
Fall 2012