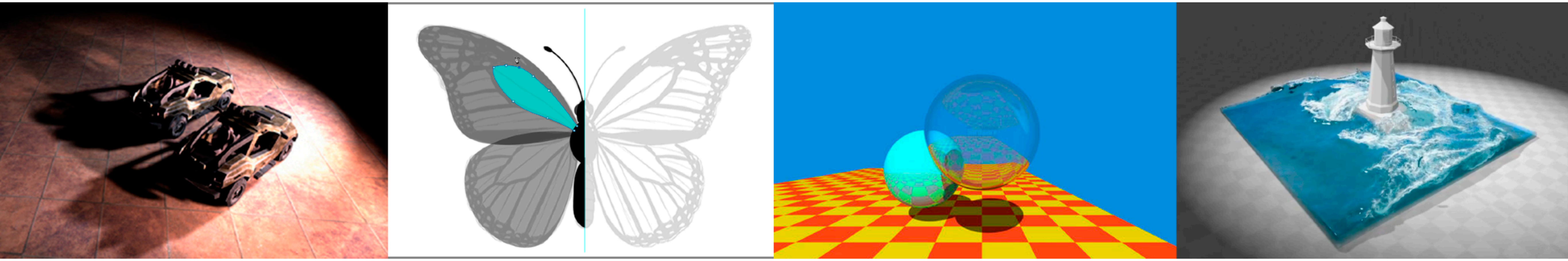
Introduction to Computer Graphics

GAMES101, Lingqi Yan, UC Santa Barbara

Lecture 13:

Ray Tracing 1

(Whitted-Style Ray Tracing)



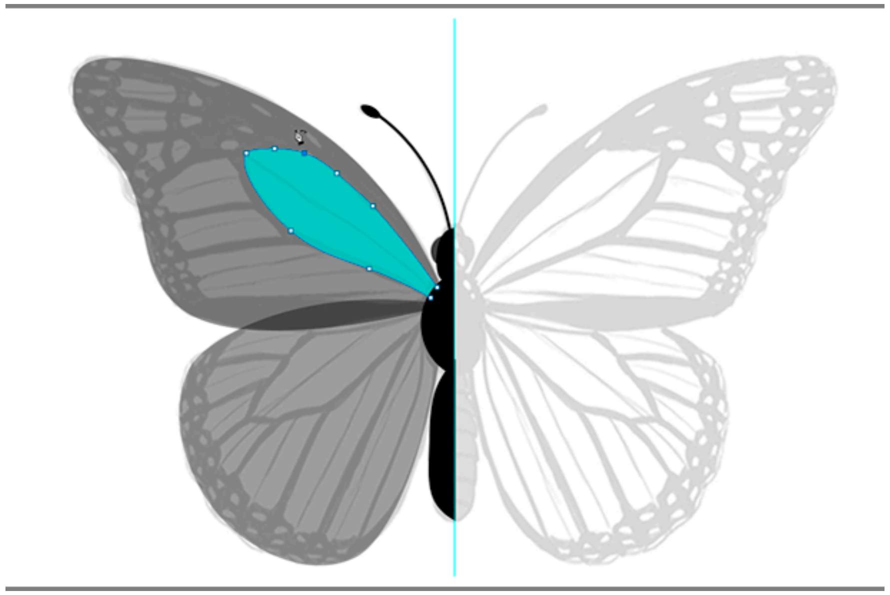
http://www.cs.ucsb.edu/~lingqi/teaching/games101.html

Announcements

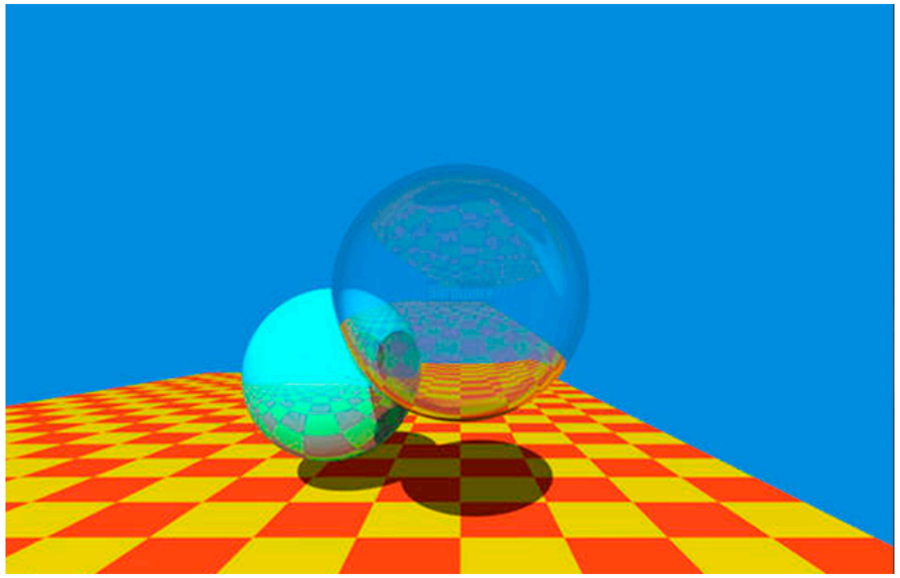
- Homework 4 — 252 submissions so far
- Homework 1 — 80 resubmissions so far
- For universities that use this course
  - Feel free to do so, and if you need scores
  - On your side: give me a name (TA or professor)
    On my side: give you access
    You determine whether to account for resubmissions

Course Roadmap

Rasterization



Geometry



Ray tracing

Animation / simulation

Why Ray Tracing?

- Rasterization couldn't handle global effects well
  - (Soft) shadows
  - And especially when the light bounces more than once

Soft shadows

Glossy reflection

Indirect illumination

Why Ray Tracing?

- Rasterization is fast, but quality is relatively low

Buggy, from PlayerUnknown's Battlegrounds (PC game)

Why Ray Tracing?

- Ray tracing is accurate, but is very slow
  - Rasterization: real-time, ray tracing: offline
  - ~10K CPU core hours to render one frame in production

Zootopia, Disney Animation

Basic Ray-Tracing Algorithm

Light Rays

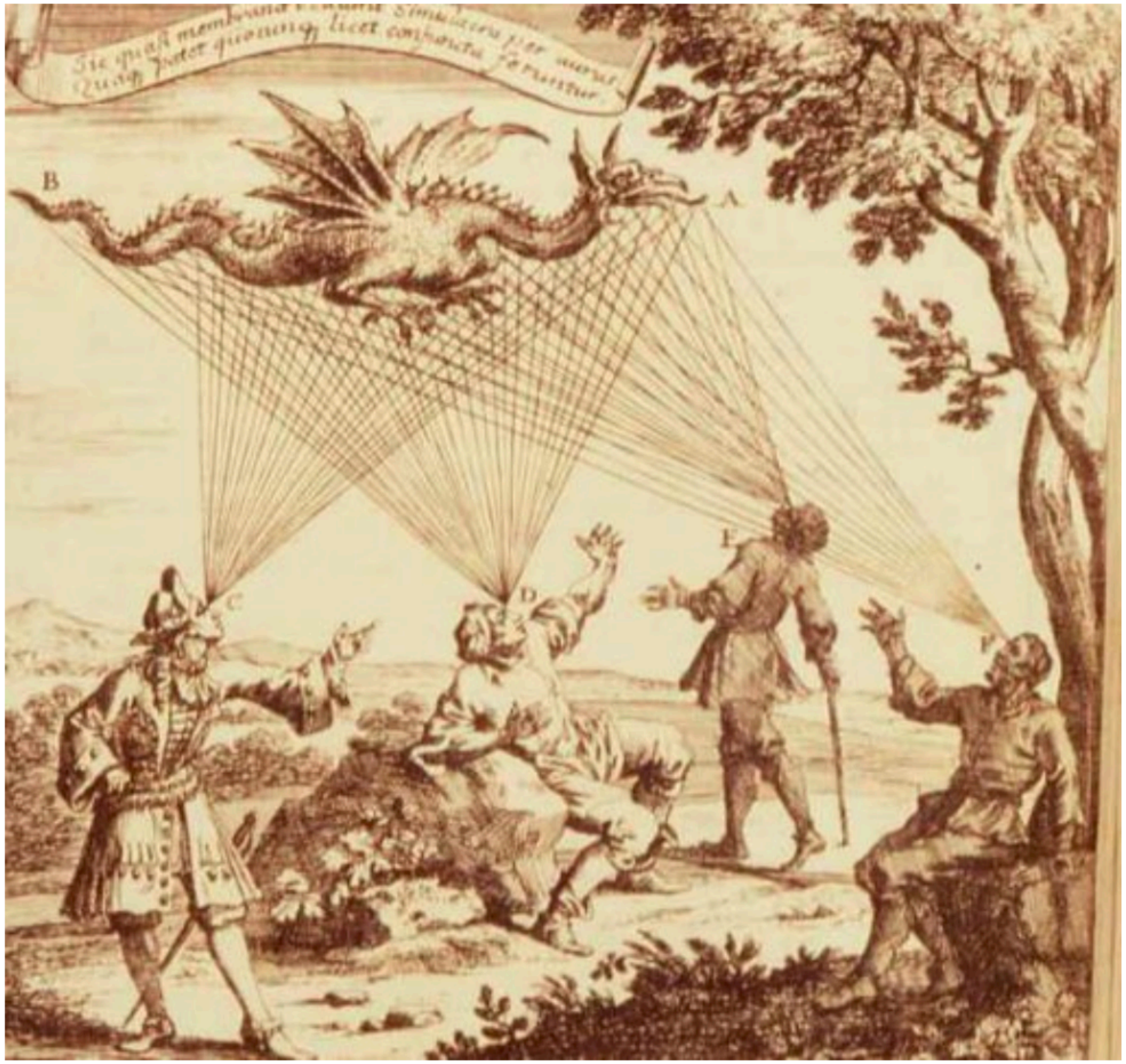
Three ideas about light rays

1. Light travels in straight lines (though this is wrong)
2. Light rays do not “collide” with each other if they cross (though this is still wrong)
3. Light rays travel from the light sources to the eye (but the physics is invariant under path reversal - reciprocity).

“And if you gaze long into an abyss, the abyss also gazes into you.” — Friedrich Wilhelm Nietzsche (translated)

Emission Theory of Vision

“For every complex problem there is an answer that is clear, simple, and wrong.”
-- H. L. Mencken



Eyes send out “feeling rays” into the world

Supported by:
- Empedocles
- Plato
- Euclid (kinda)
- Ptolemy
- …
- 50% of US college students*

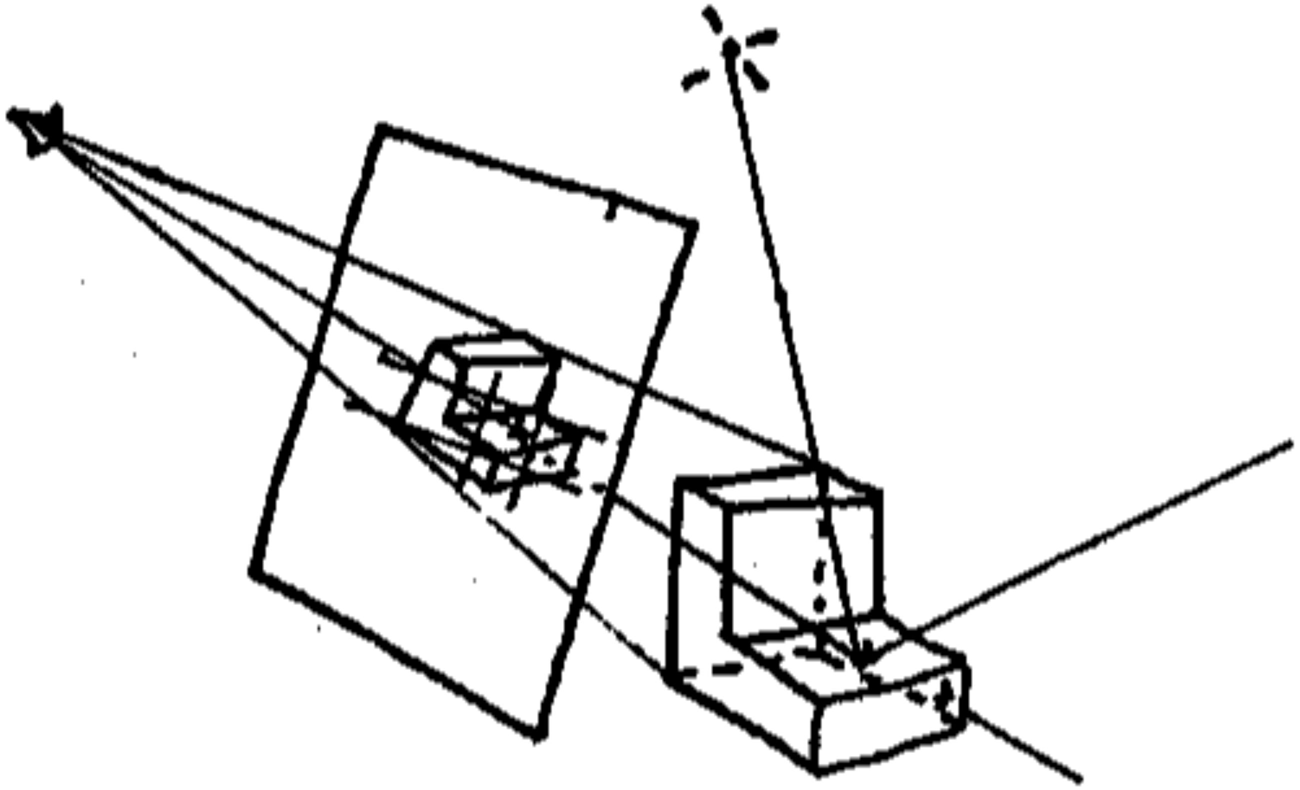
*http://www.ncbi.nlm.nih.gov/pubmed/12094435?dopt=Abstract

Slide courtesy of Prof. Alexei Efros, UC Berkeley

Ray Casting

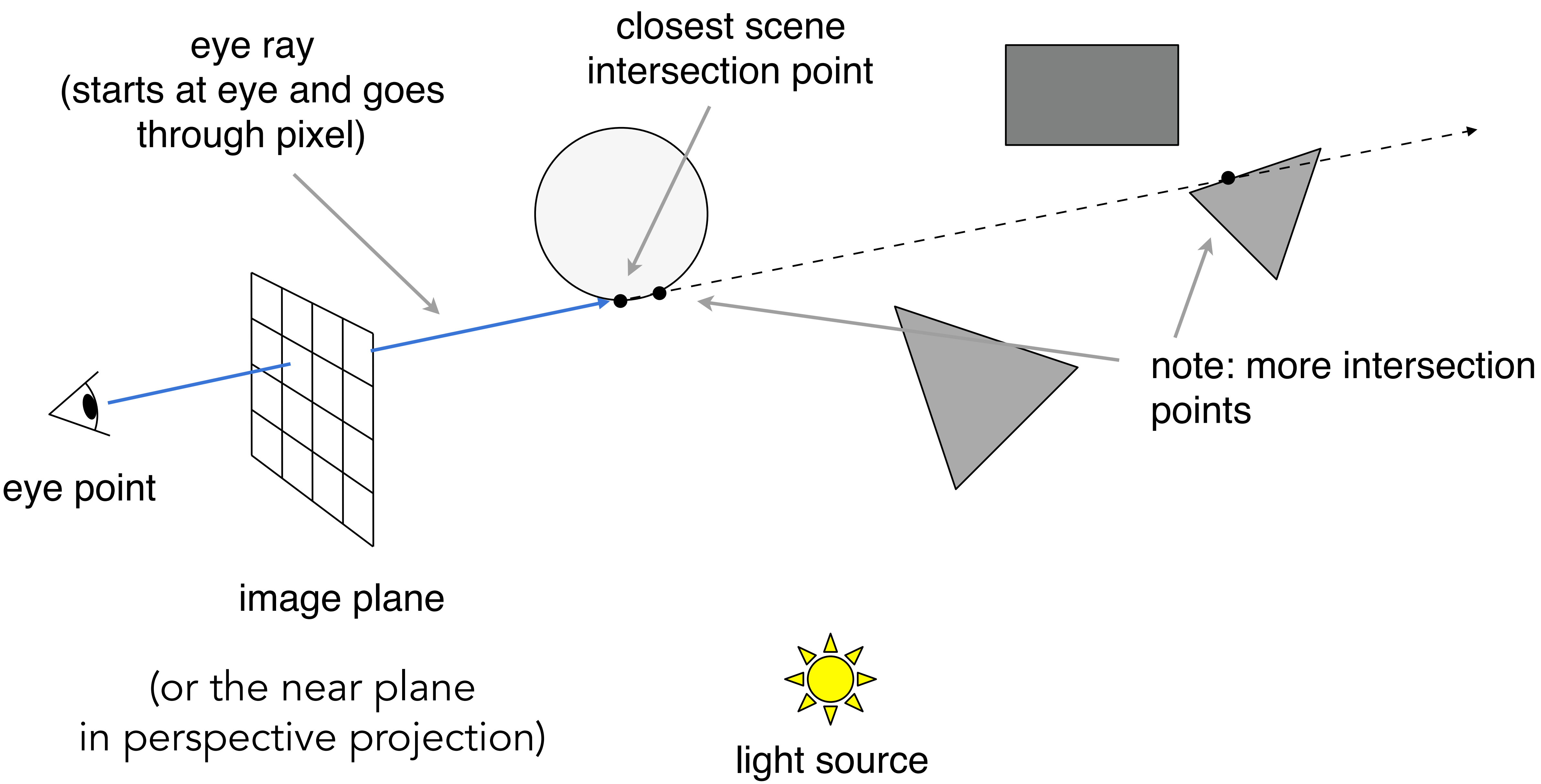
Appel 1968 - Ray casting

1. Generate an image by **casting one ray per pixel**
2. Check for shadows by **sending a ray to the light**



Ray Casting - Generating Eye Rays

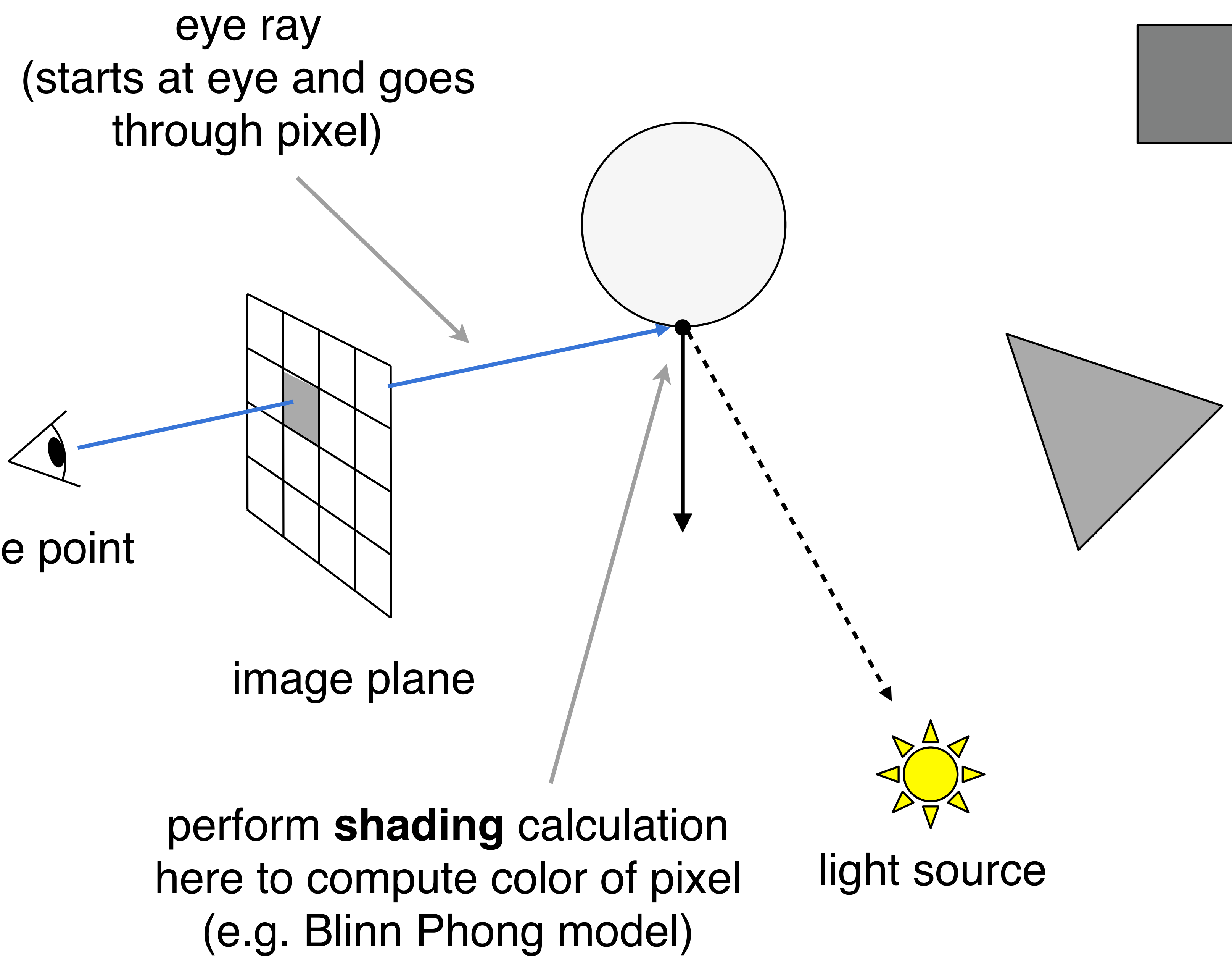
Pinhole Camera Model



(or the near plane
in perspective projection)

Ray Casting - Shading Pixels (Local Only)

Pinhole Camera Model


eye ray
(starts at eye and goes
through pixel)

eye point

image plane

perform **shading** calculation
here to compute color of pixel
(e.g. Blinn Phong model)

light source

Recursive (Whitted-Style) Ray Tracing

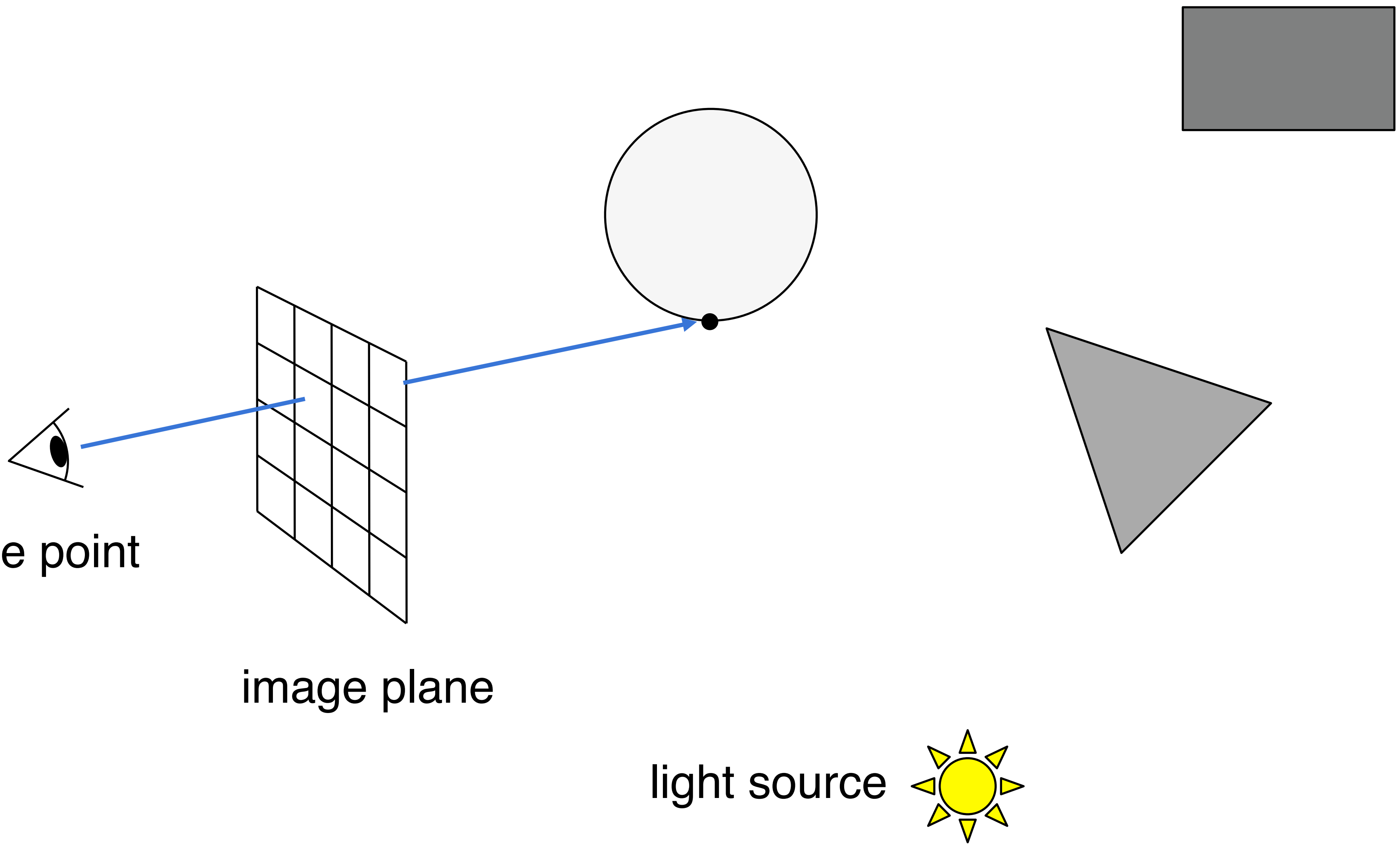
"An improved Illumination model for shaded display"
T. Whitted, CACM 1980

Time:

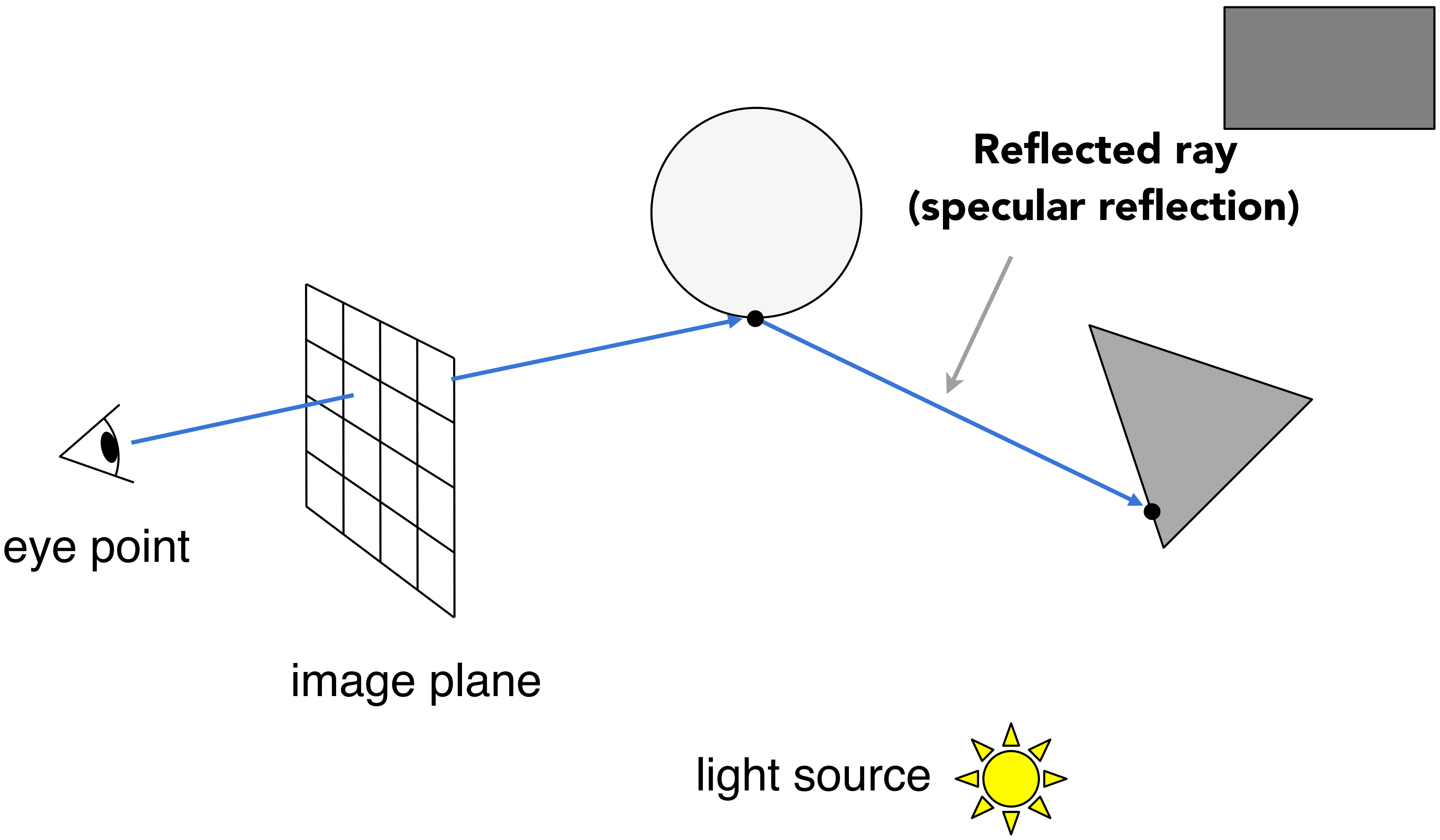
- VAX 11/780 (1979) 74m
- PC (2006) 6s
- GPU (2012) 1/30s

Spheres and Checkerboard, T. Whitted, 1979

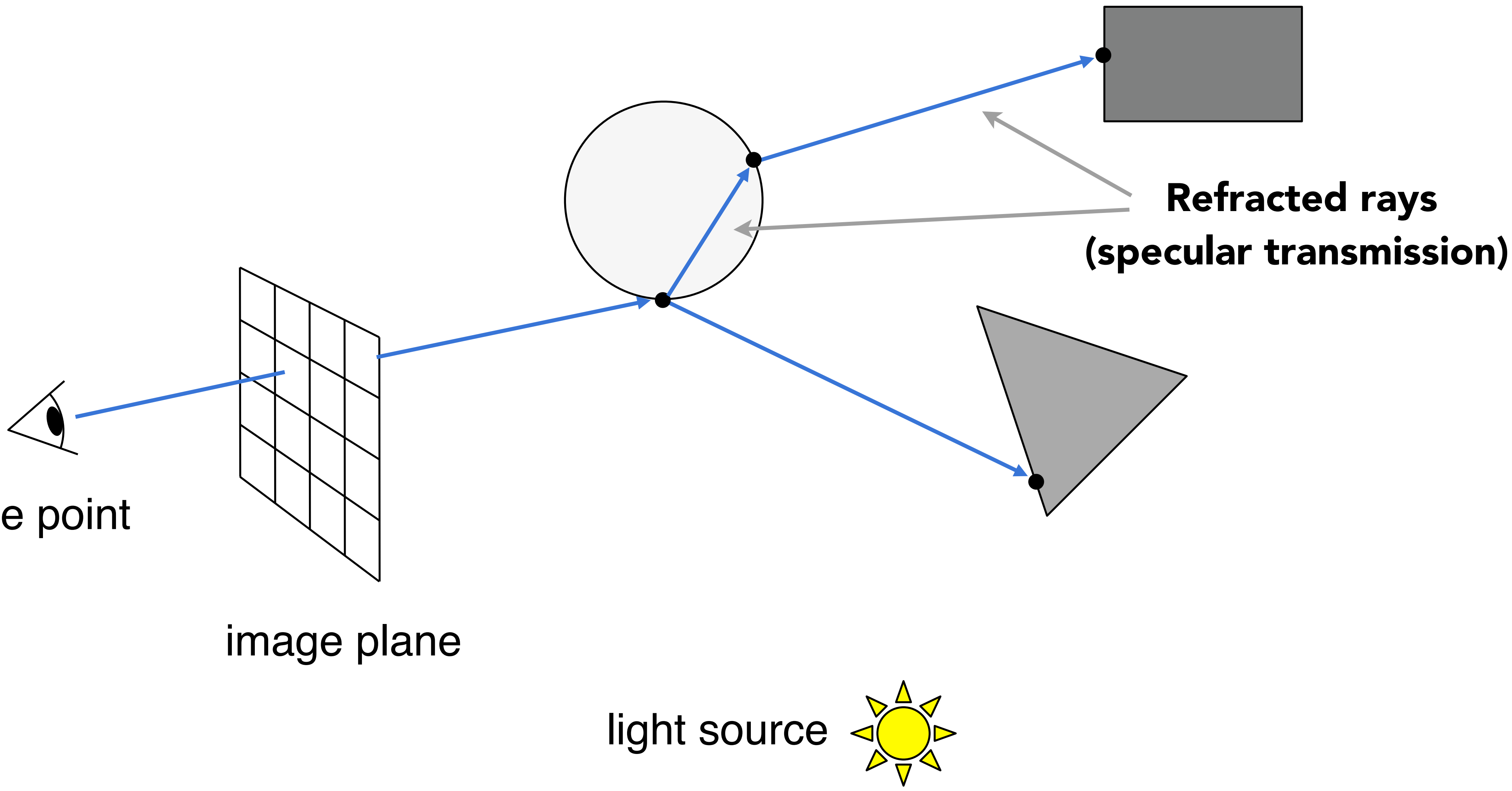
Recursive Ray Tracing



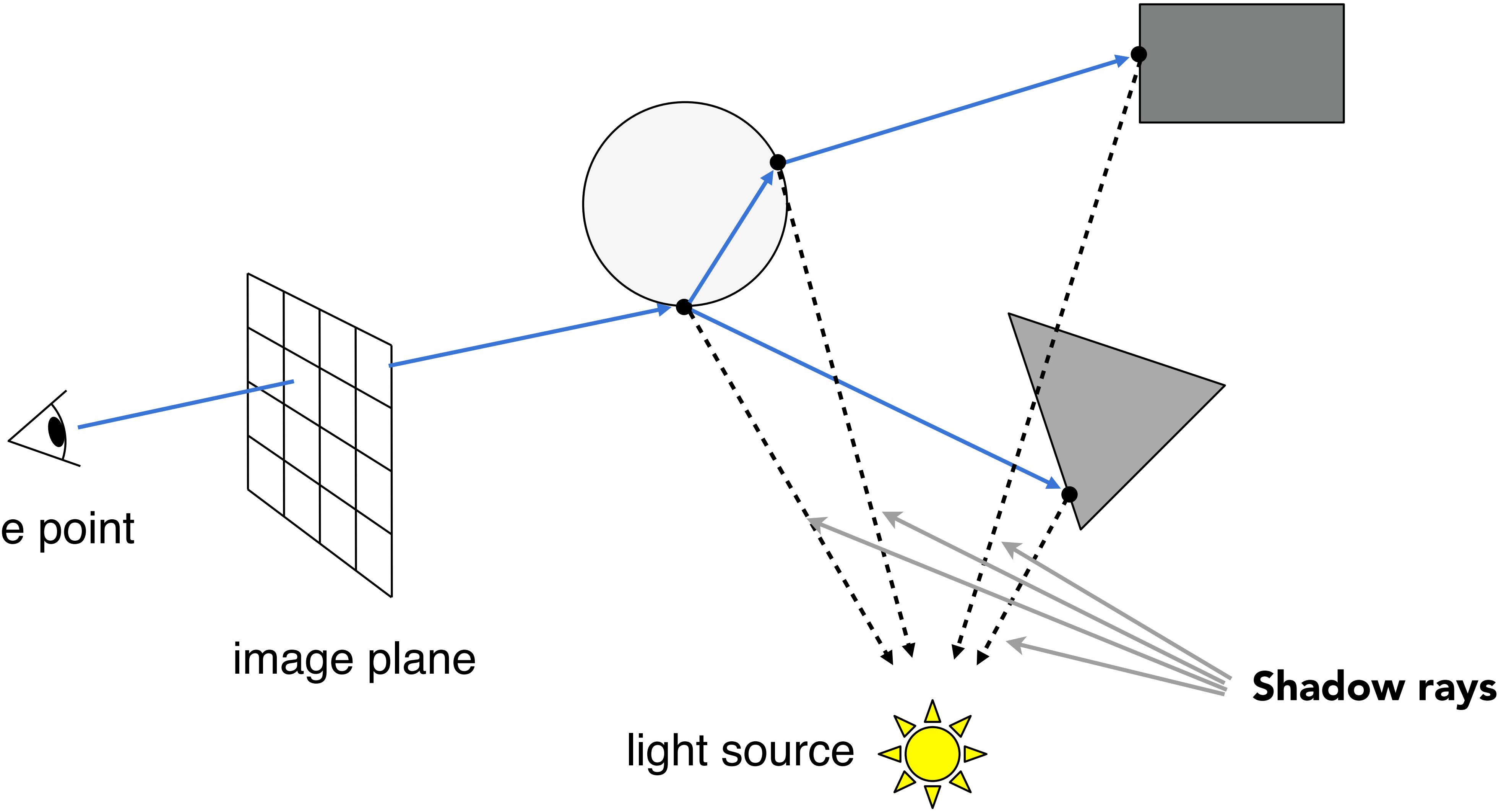
Recursive Ray Tracing



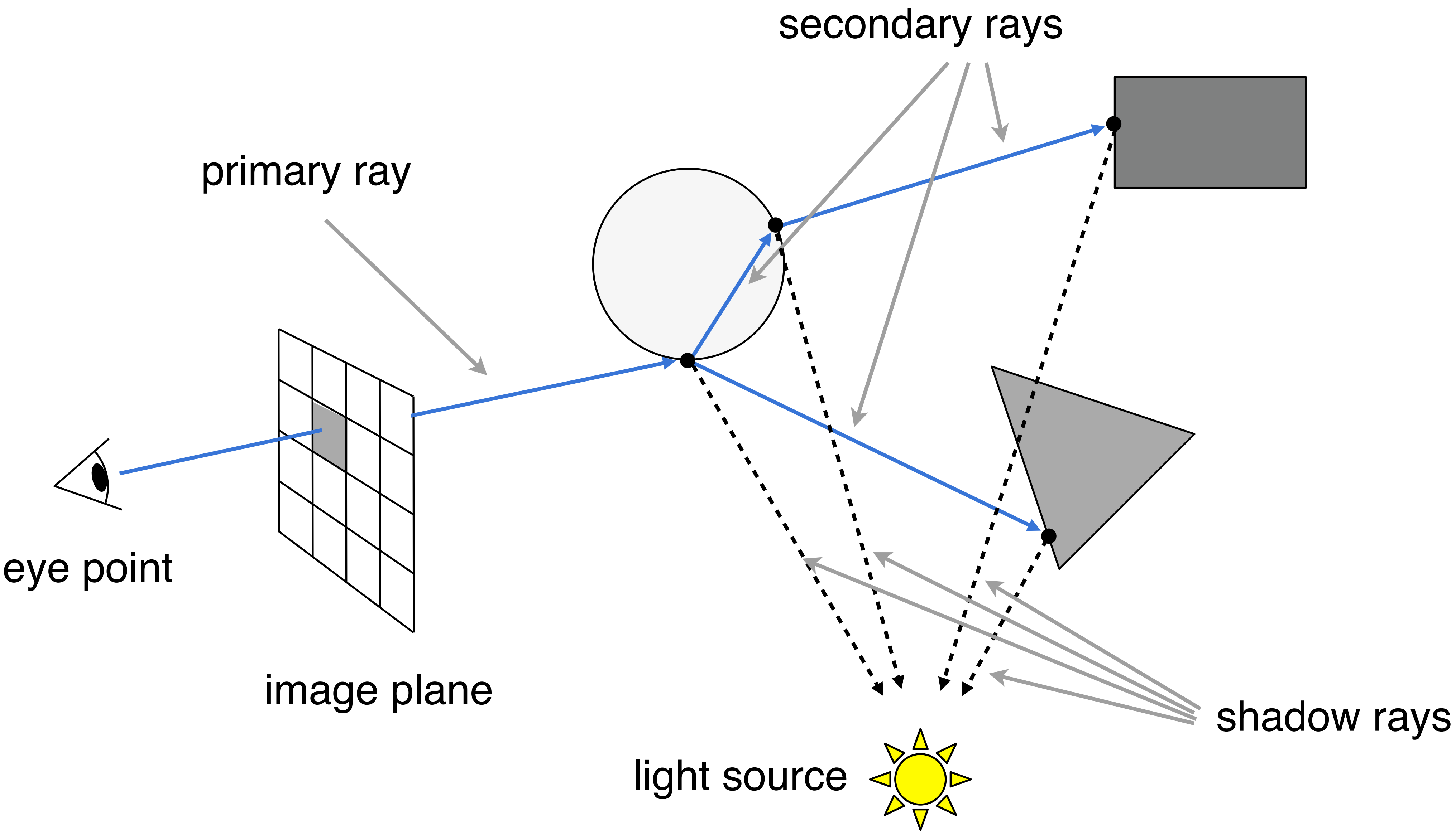
Recursive Ray Tracing



Recursive Ray Tracing



Recursive Ray Tracing



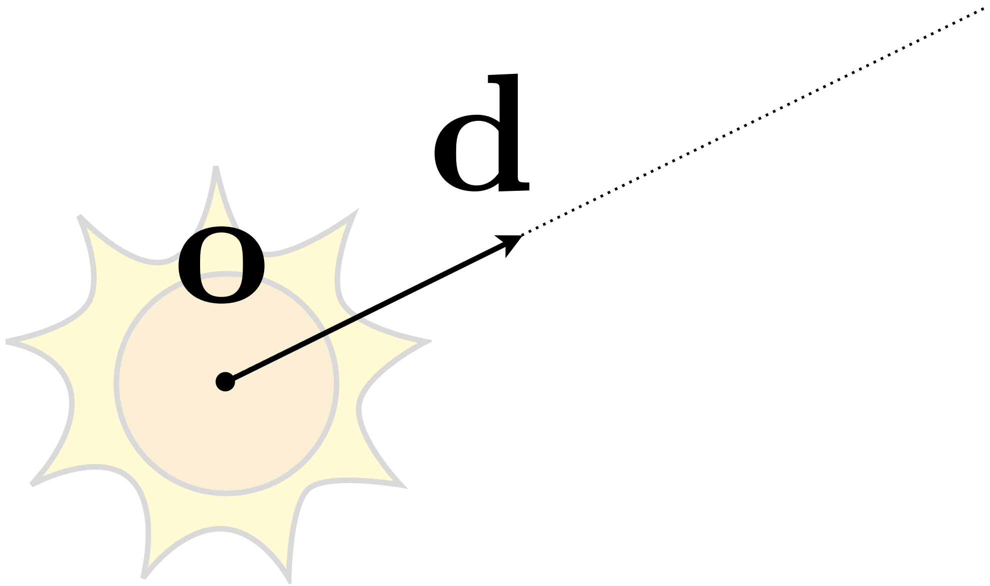
Recursive Ray Tracing

Ray-Surface Intersection

Ray Equation

Ray is defined by its origin and a direction vector

Example:



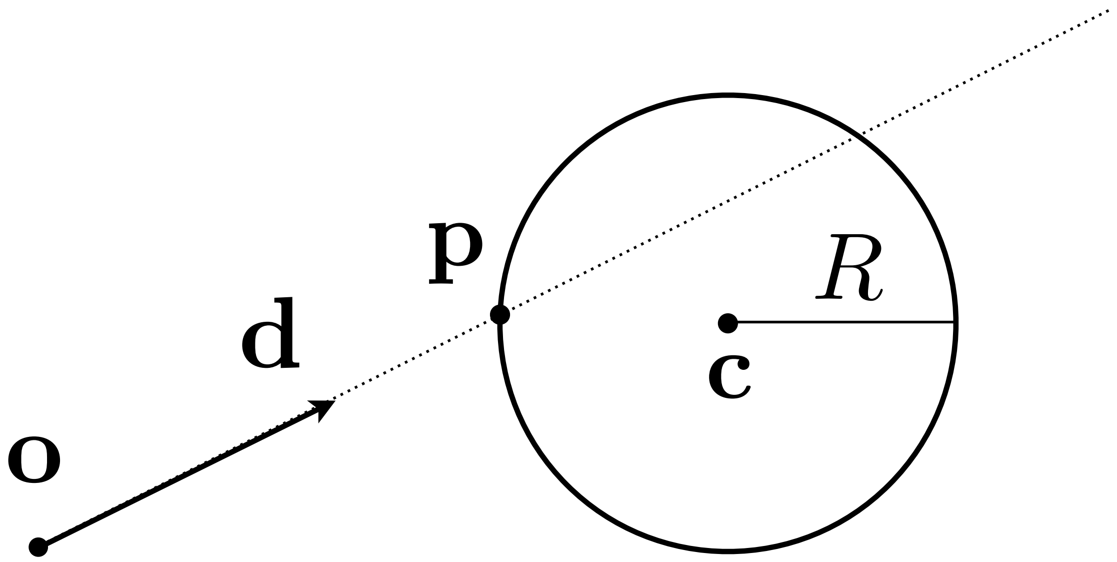
Ray equation:

$$\underset{\text{point along ray}}{\mathbf{r}}(\underset{\text{“time”}}{t}) = \underset{\text{origin}}{\mathbf{o}} + t\underset{\text{(normalized) direction}}{\mathbf{d}} \qquad 0 \le t < \infty$$

Ray Intersection With Sphere

Ray: $\mathbf{r}(t) = \mathbf{o} + t\,\mathbf{d}, \;\; 0 \leq t < \infty$

Sphere: $\mathbf{p} : (\mathbf{p} - \mathbf{c})^2 - R^2 = 0$



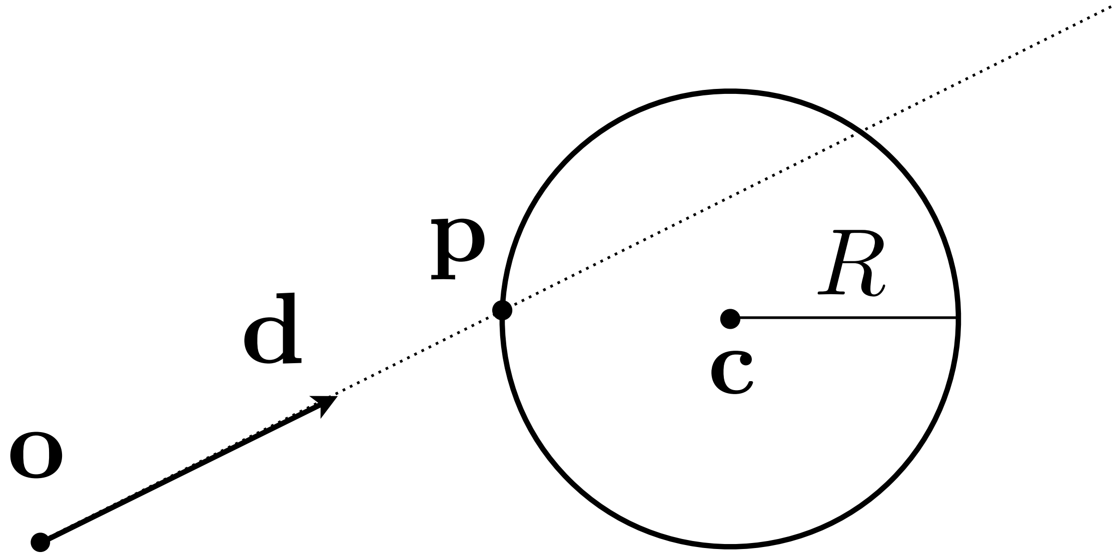
What is an intersection?

The intersection p must satisfy both ray equation and sphere equation

Solve for intersection:

$$(\mathbf{o} + t\,\mathbf{d} - \mathbf{c})^2 - R^2 = 0$$

Ray Intersection With Sphere



Solve for intersection:

$$(\mathbf{o} + t\,\mathbf{d} - \mathbf{c})^2 - R^2 = 0$$

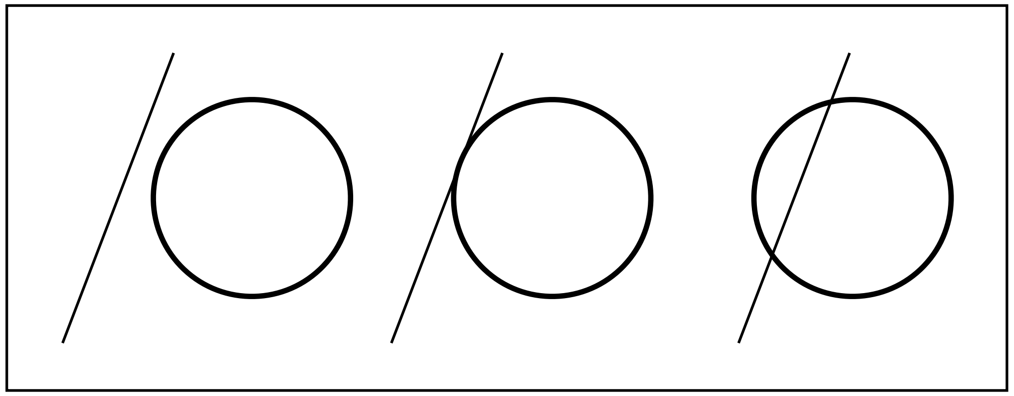
$$a\,t^2 + b\,t + c = 0, \text{ where}$$

$$a = \mathbf{d} \cdot \mathbf{d}$$

$$b = 2(\mathbf{o} - \mathbf{c}) \cdot \mathbf{d}$$

$$c = (\mathbf{o} - \mathbf{c}) \cdot (\mathbf{o} - \mathbf{c}) - R^2$$

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$



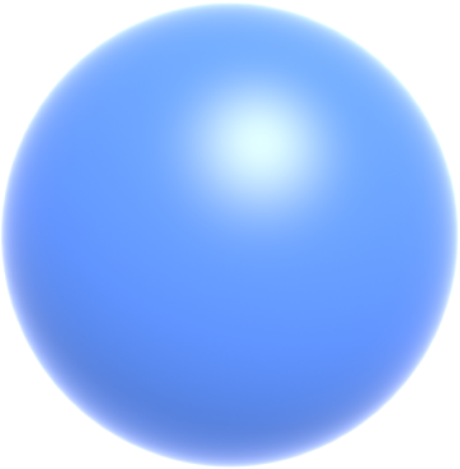
Ray Intersection With Implicit Surface

Ray: $\mathbf{r}(t) = \mathbf{o} + t\,\mathbf{d},\ 0 \leq t < \infty$

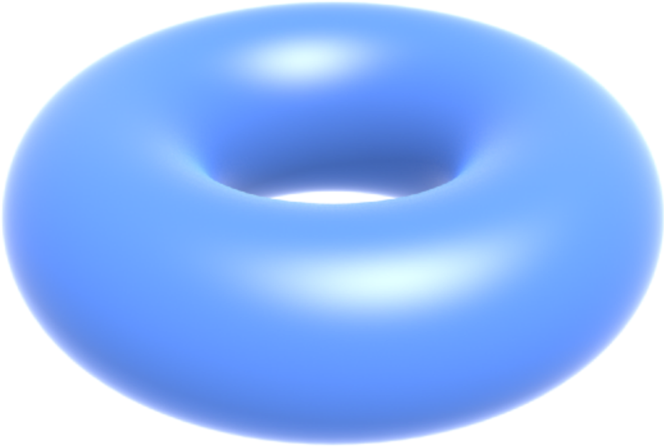
General implicit surface: $\mathbf{p} : f(\mathbf{p}) = 0$

Substitute ray equation: $f(\mathbf{o} + t\,\mathbf{d}) = 0$

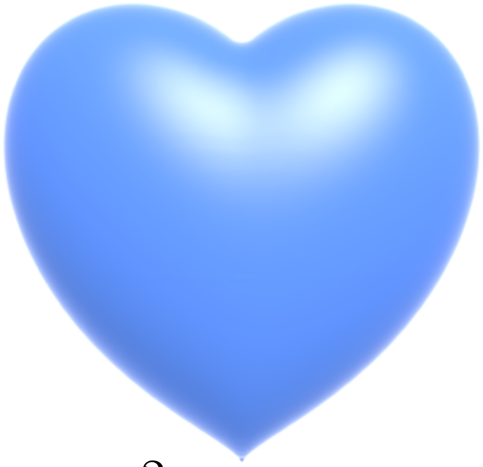
Solve for **real, positive** roots



$x^2+y^2+z^2=1$

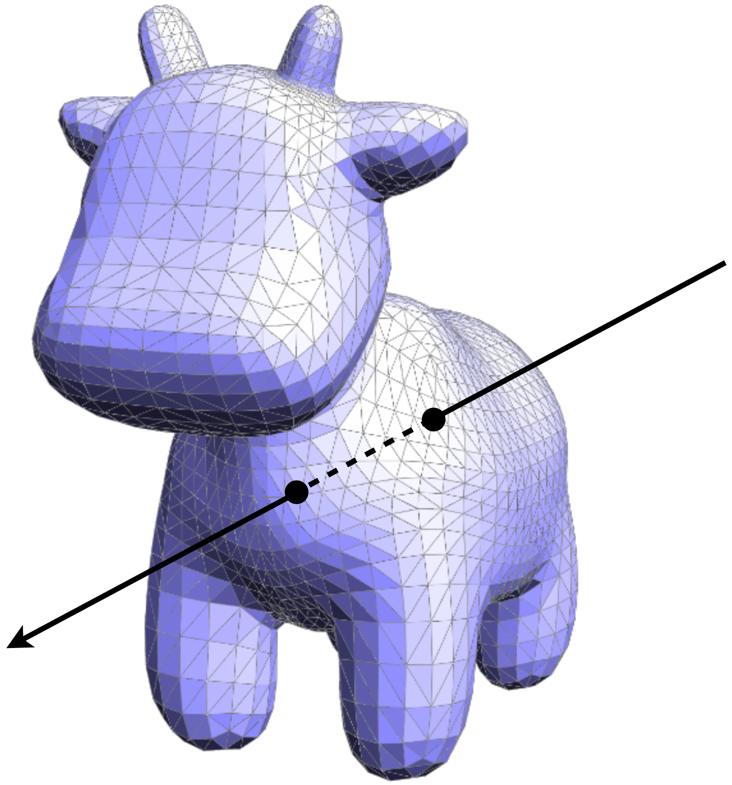


$(R - \sqrt{x^2+y^2})^2 + z^2 = r^2$



$$(x^2 + \frac{9y^2}{4} + z^2 - 1)^3 = x^2 z^3 + \frac{9y^2 z^3}{80}$$

Ray Intersection With Triangle Mesh



Why?

- Rendering: visibility, shadows, lighting …
- Geometry: inside/outside test

How to compute?

Let's break this down:

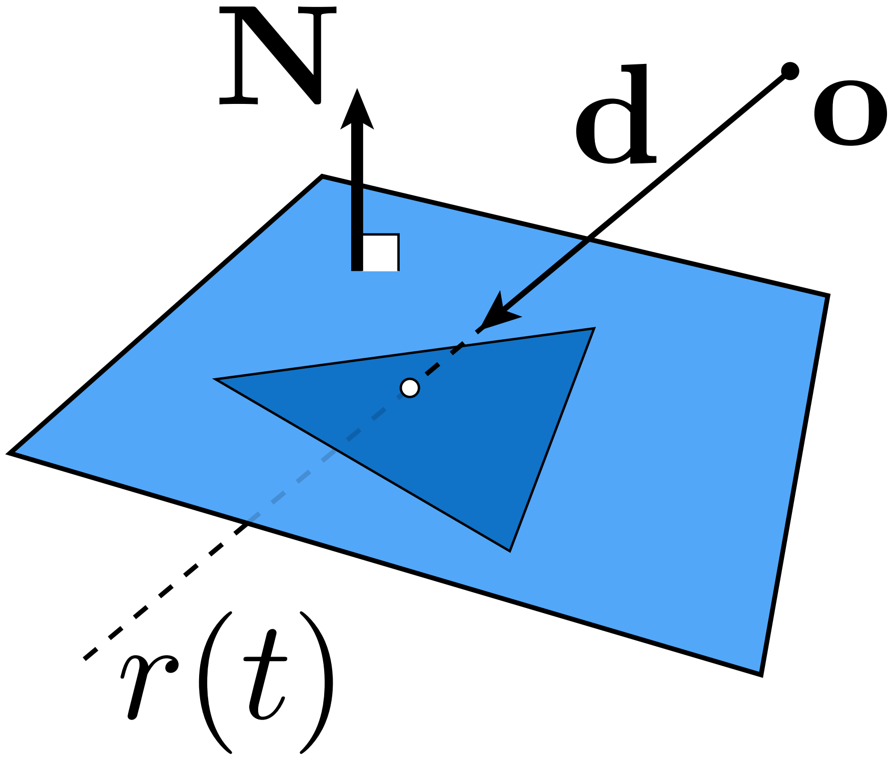
- Simple idea: just intersect ray with each triangle
- Simple, but slow (acceleration?)
- Note: can have 0, 1 intersections (ignoring multiple intersections)

Ray Intersection With Triangle

Triangle is in a plane

- Ray-plane intersection
- Test if hit point is inside triangle

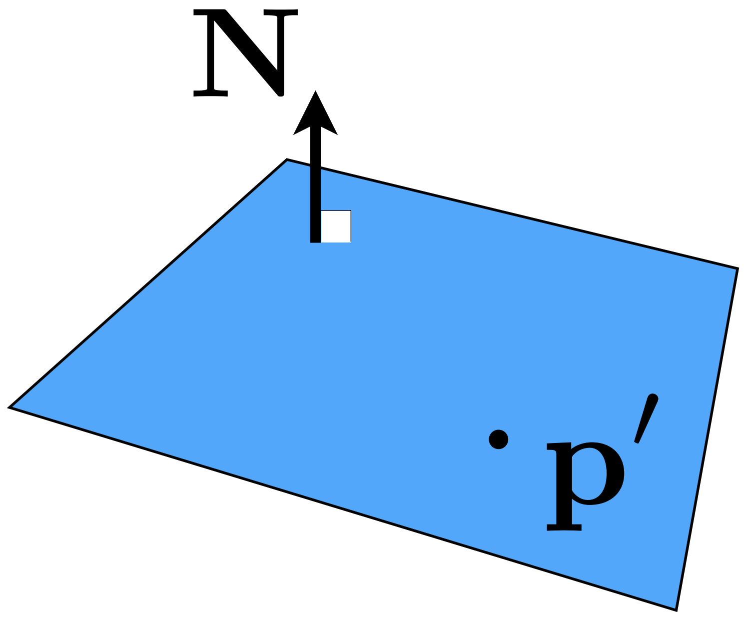
Many ways to optimize…



Plane Equation

Plane is defined by normal vector and a point on plane

Example:

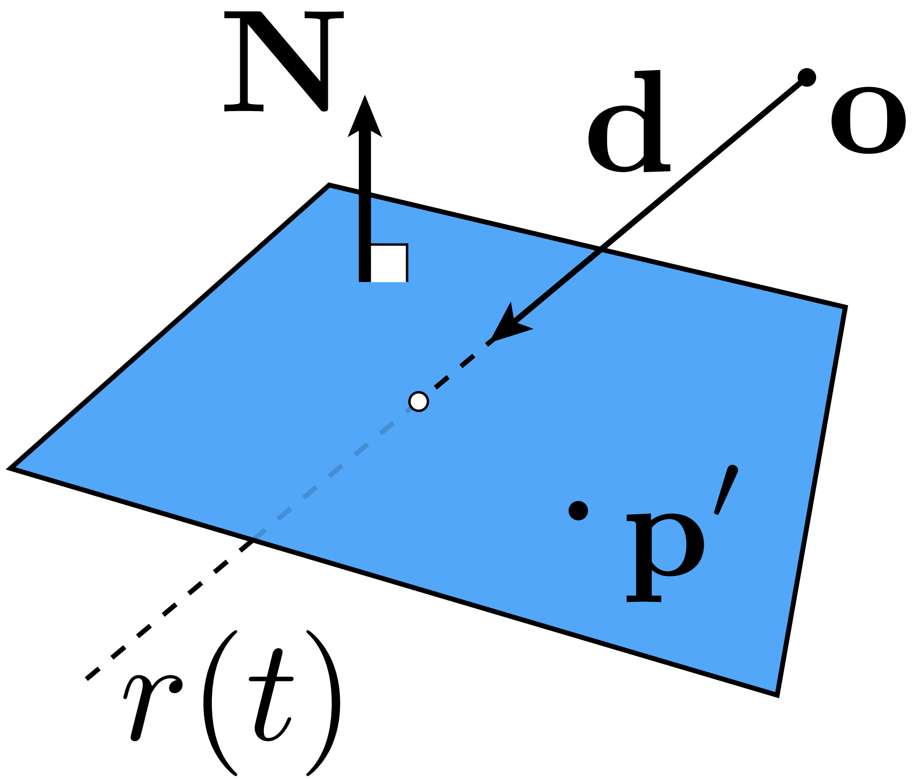


Plane Equation (if p satisfies it, then p is on the plane):

$$\underset{\overset{\uparrow}{\text{all points on plane}}}{\mathbf{p}} : (\mathbf{p} - \underset{\overset{\uparrow}{\text{one point on plane}}}{\mathbf{p}'}) \cdot \underset{\overset{\uparrow}{\text{normal vector}}}{\mathbf{N}} = 0$$

$$ax + by + cz + d = 0$$

Ray Intersection With Plane



Ray equation:

$$\mathbf{r}(t) = \mathbf{o} + t\,\mathbf{d},\ \ 0 \le t < \infty$$

Plane equation:

$$\mathbf{p} : (\mathbf{p} - \mathbf{p}') \cdot \mathbf{N} = 0$$

Solve for intersection

$$\text{Set } \mathbf{p} = \mathbf{r}(t) \text{ and solve for } t$$

$$(\mathbf{p} - \mathbf{p}') \cdot \mathbf{N} = (\mathbf{o} + t\,\mathbf{d} - \mathbf{p}') \cdot \mathbf{N} = 0$$

$$t = \frac{(\mathbf{p}' - \mathbf{o}) \cdot \mathbf{N}}{\mathbf{d} \cdot \mathbf{N}}$$

Check: $0 \le t < \infty$

Möller Trumbore Algorithm

A faster approach, giving barycentric coordinate directly

Derivation in the discussion section!

$$\vec{\mathbf{O}} + t\vec{\mathbf{D}} = (1 - b_1 - b_2)\vec{\mathbf{P}}_0 + b_1\vec{\mathbf{P}}_1 + b_2\vec{\mathbf{P}}_2$$

$$\begin{bmatrix} t \\ b_1 \\ b_2 \end{bmatrix} = \frac{1}{\vec{\mathbf{S}}_1 \bullet \vec{\mathbf{E}}_1} \begin{bmatrix} \vec{\mathbf{S}}_2 \bullet \vec{\mathbf{E}}_2 \\ \vec{\mathbf{S}}_1 \bullet \vec{\mathbf{S}} \\ \vec{\mathbf{S}}_2 \bullet \vec{\mathbf{D}} \end{bmatrix}$$

Where:

$$\vec{\mathbf{E}}_1 = \vec{\mathbf{P}}_1 - \vec{\mathbf{P}}_0$$
$$\vec{\mathbf{E}}_2 = \vec{\mathbf{P}}_2 - \vec{\mathbf{P}}_0$$
$$\vec{\mathbf{S}} = \vec{\mathbf{O}} - \vec{\mathbf{P}}_0$$
$$\vec{\mathbf{S}}_1 = \vec{\mathbf{D}} \times \vec{\mathbf{E}}_2$$
$$\vec{\mathbf{S}}_2 = \vec{\mathbf{S}} \times \vec{\mathbf{E}}_1$$

Cost = (1 div, 27 mul, 17 add)

Recall: How to determine if the “intersection” is inside the triangle?

Hint:
(1-b1-b2), b1, b2 are barycentric coordinates!

Accelerating Ray-Surface Intersection

Ray Tracing – Performance Challenges

Simple ray-scene intersection

- Exhaustively test ray-intersection with **every triangle**
- Find the closest hit (i.e. minimum t)

Problem:

- Naive algorithm = #pixels ⨉ # traingles (⨉ #bounces)
- Very slow!

For generality, we use the term objects instead of triangles later (but doesn't necessarily mean entire objects)

Ray Tracing – Performance Challenges


Jun Yan, Tracy Renderer

San Miguel Scene, 10.7M triangles

Ray Tracing – Performance Challenges

Deussen et al; Pharr & Humphreys, PBRT

Plant Ecosystem, 20M triangles

Bounding Volumes

Bounding Volumes

Quick way to avoid intersections: bound complex object with a simple volume

- Object is fully contained in the volume
- If it doesn’t hit the volume, it doesn’t hit the object
- So test BVol first, then test object if it hits

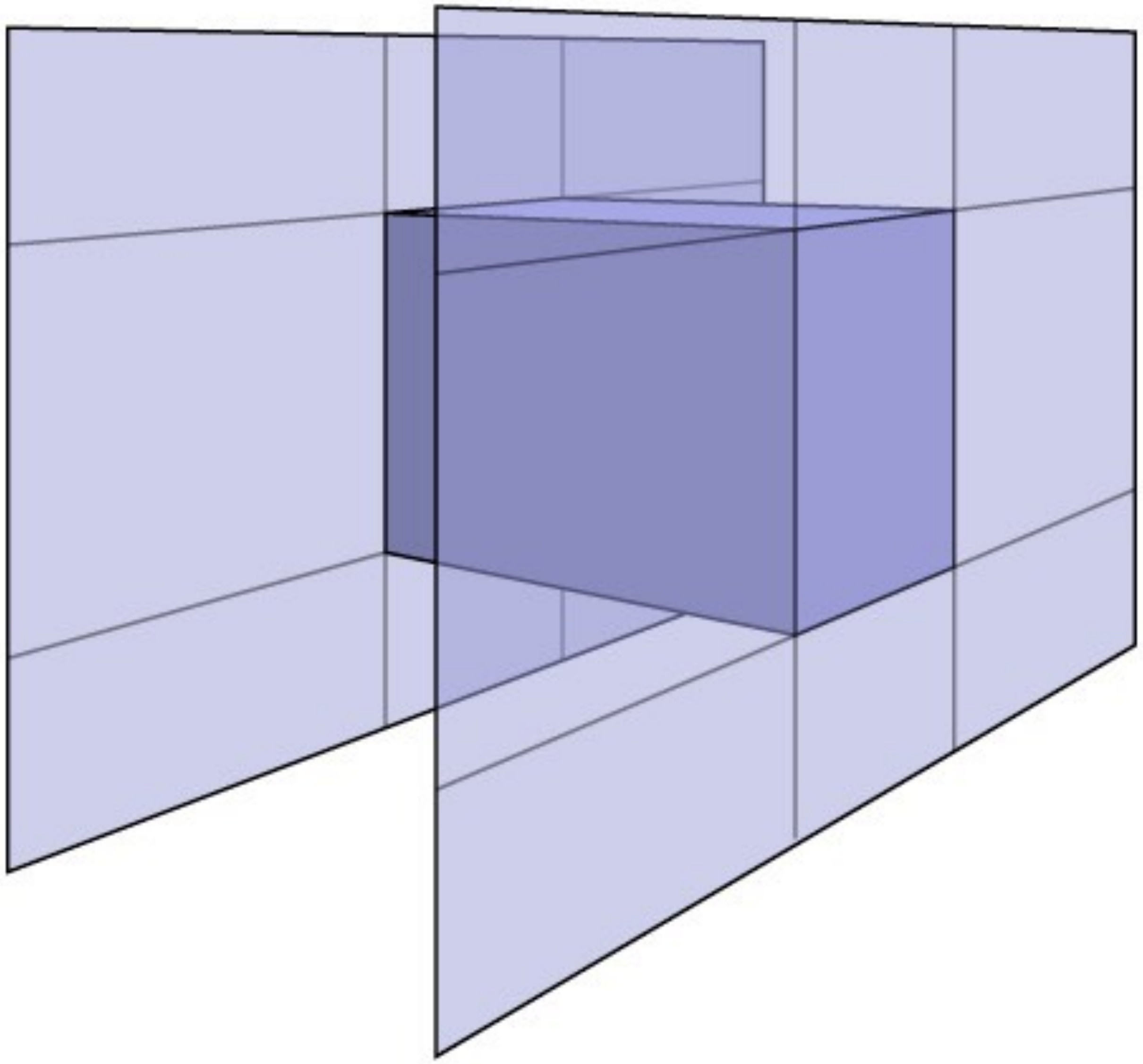
Ray-Intersection With Box

Understanding: **box is the intersection of 3 pairs of slabs**

Specifically:

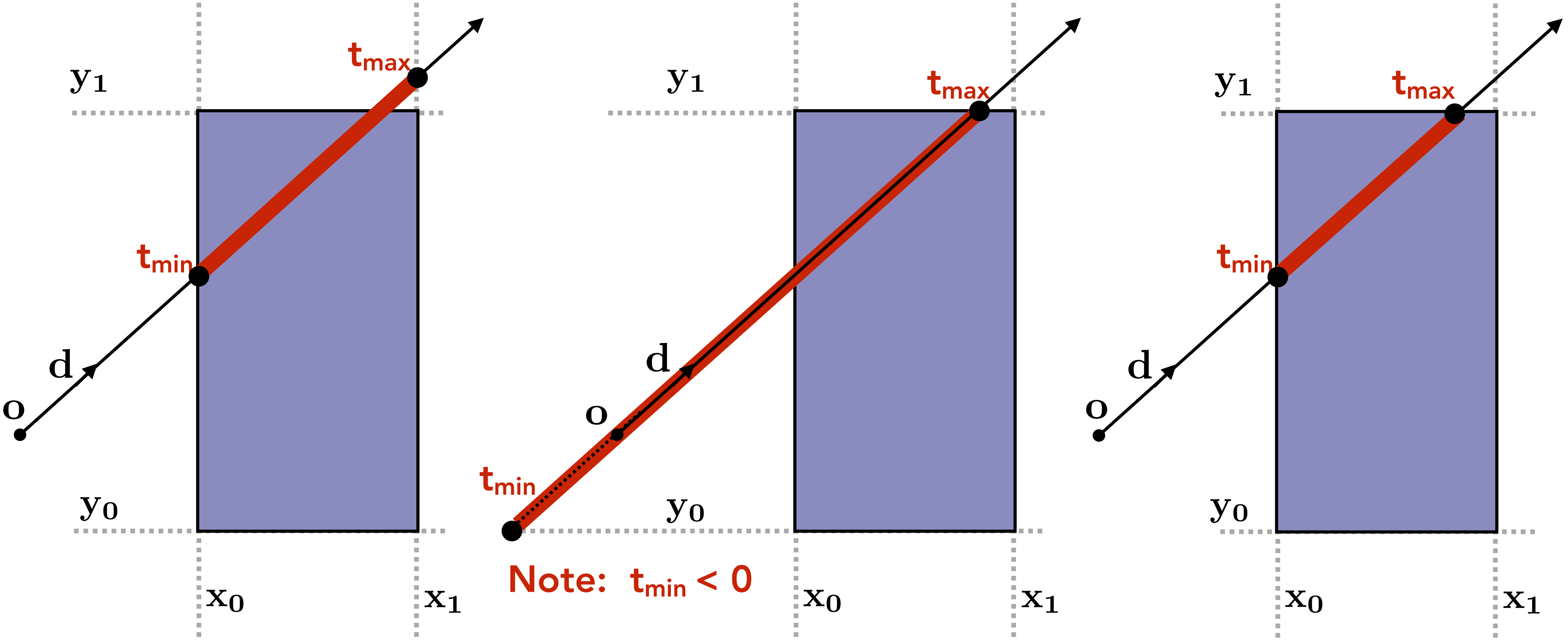
We often use an **Axis-Aligned Bounding Box (AABB)** (轴对齐包围盒)

i.e. any side of the BB is along either x, y, or z axis



Ray Intersection with Axis-Aligned Box

2D example; 3D is the same! Compute intersections with slabs and take intersection of t_{min}/t_{max} intervals



Intersections with *x* planes **Intersections with *y* planes** **Final intersection result**

How do we know when the ray intersects the box?

Ray Intersection with Axis-Aligned Box

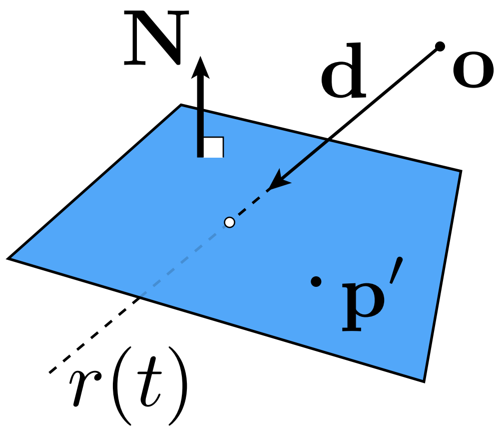
- Recall: a box (3D) = three pairs of infinitely large slabs
- Key ideas
  - The ray enters the box **only when** it enters all pairs of slabs
  - The ray exits the box **as long as** it exits any pair of slabs
- For each pair, calculate the t_{min} and t_{max} (negative is fine)
- For the 3D box, $t_{enter} =$ **max**$\{t_{min}\}$, $t_{exit} =$ **min**$\{t_{max}\}$
- If $t_{enter} < t_{exit}$, we know the ray **stays a while** in the box (so they must intersect!) (not done yet, see the next slide)

Ray Intersection with Axis-Aligned Box

- However, ray is not a line
  - Should check whether t is negative for physical correctness!
- What if $t_{exit} < 0$?
  - The box is “behind” the ray — no intersection!
- What if $t_{exit} >= 0$ and $t_{enter} < 0$?
  - The ray’s origin is inside the box — have intersection!
- In summary, ray and AABB intersect iff
  - $t_{enter} < t_{exit}$ && $t_{exit} >= 0$

Why Axis-Aligned?

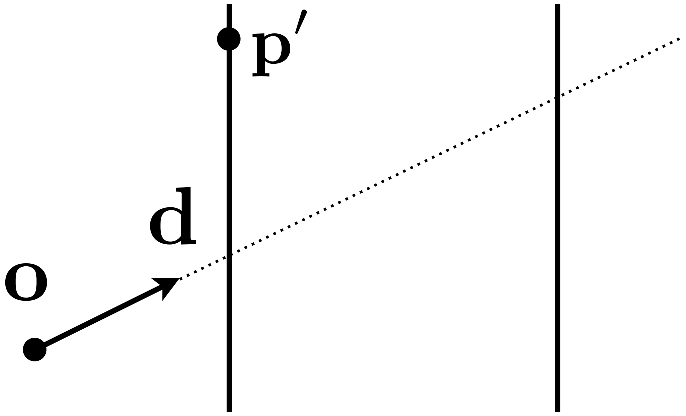
General



$$t = \frac{(\mathbf{p}' - \mathbf{o}) \cdot \mathbf{N}}{\mathbf{d} \cdot \mathbf{N}}$$

3 subtractions, 6 multiplies, 1 division

Slabs perpendicular to x-axis



$$t = \frac{\mathbf{p}'_x - \mathbf{o}_x}{\mathbf{d}_x}$$

1 subtraction, 1 division

Thank you!

(And thank Prof. Ravi Ramamoorthi and Prof. Ren Ng for many of the slides!)