Lecture 9:

Intro to Ray-Tracing

Computer Graphics and Imaging
UC Berkeley CS184/284A

Towards Photorealistic Rendering

Credit: Bertrand Benoit. “Sweet Feast,” 2009. [Blender /VRay]

Discussion: What Do You See?

3 min, 3 people, 3 observations

- Look closely, curiously, and write down 3 visual features you want to know how to compute

Credit: Bertrand Benoit. "Sweet Feast," 2009. [Blender /VRay]

- https://cgsociety.org/c/featured/6hgf/sweet-feast

Discussion: What Do You See?

Your observations

- Different focal points — front of photo focused, back is blurred
- Transparency in the glass, tea
- Diffraction? Light spreading in the highlight, maybe around edges of shadows
- Refraction through drink in the back
- Light interacts with non-reflective materials, like the napkin — physical shape and attributes of the cloth
- Sponge cake — like a volume, so soft
- How light reflects light in the custard; bumps in the food
- The mug behind is reflecting the back of the mug in front that none of us can see
- Caramel — light diffusing through this; through the grapes too

Image credit: Bertrand Benoit. “Sweet Feast,” 2009. [Blender /VRay]

Course Roadmap

Rasterization Pipeline

Core Concepts
- Sampling
- Antialiasing
- Transforms

Geometric Modeling

Core Concepts
- Splines, Bezier Curves
- Topological Mesh Representations
- Subdivision, Geometry Processing

Lighting & Materials

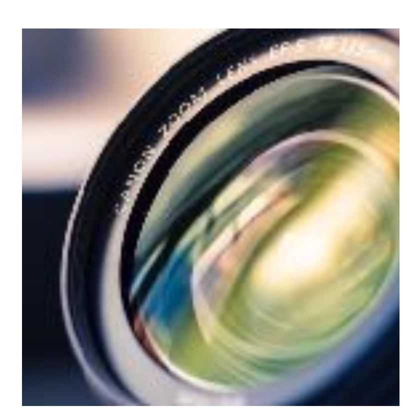
Core Concepts
- Measuring Light
- Unbiased Integral Estimation
- Light Transport & Materials

Cameras & Imaging

Rasterization
Transforms & Projection
Texture Mapping
Visibility, Shading, Overall Pipeline

Intro to Geometry
Curves and Surfaces
Geometry Processing
Ray-Tracing & Acceleration Today

Radiometry & Photometry
Monte Carlo Integration
Global Illumination & Path Tracing
Material Modeling

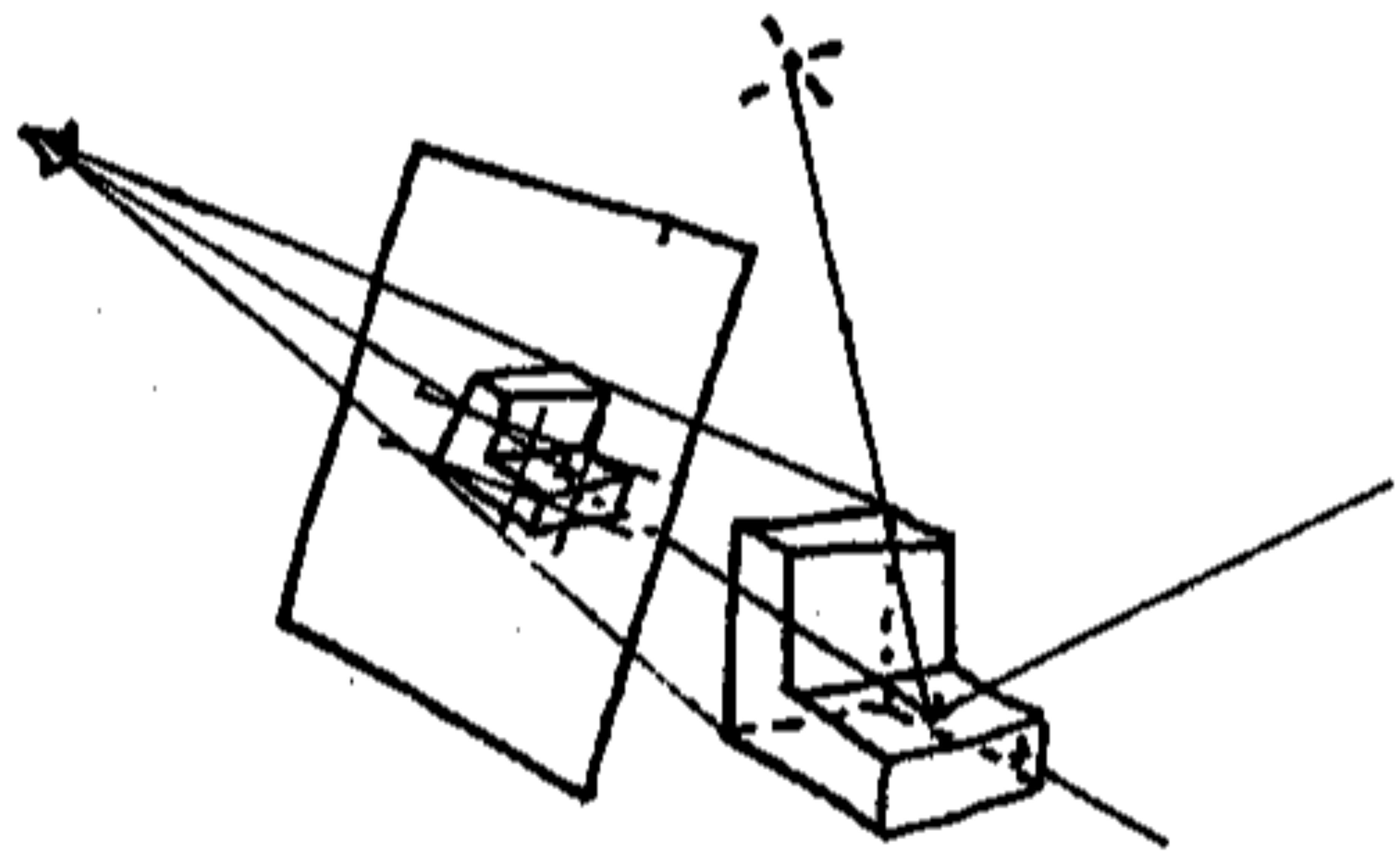


Basic Ray-Tracing Algorithm

Ray Casting

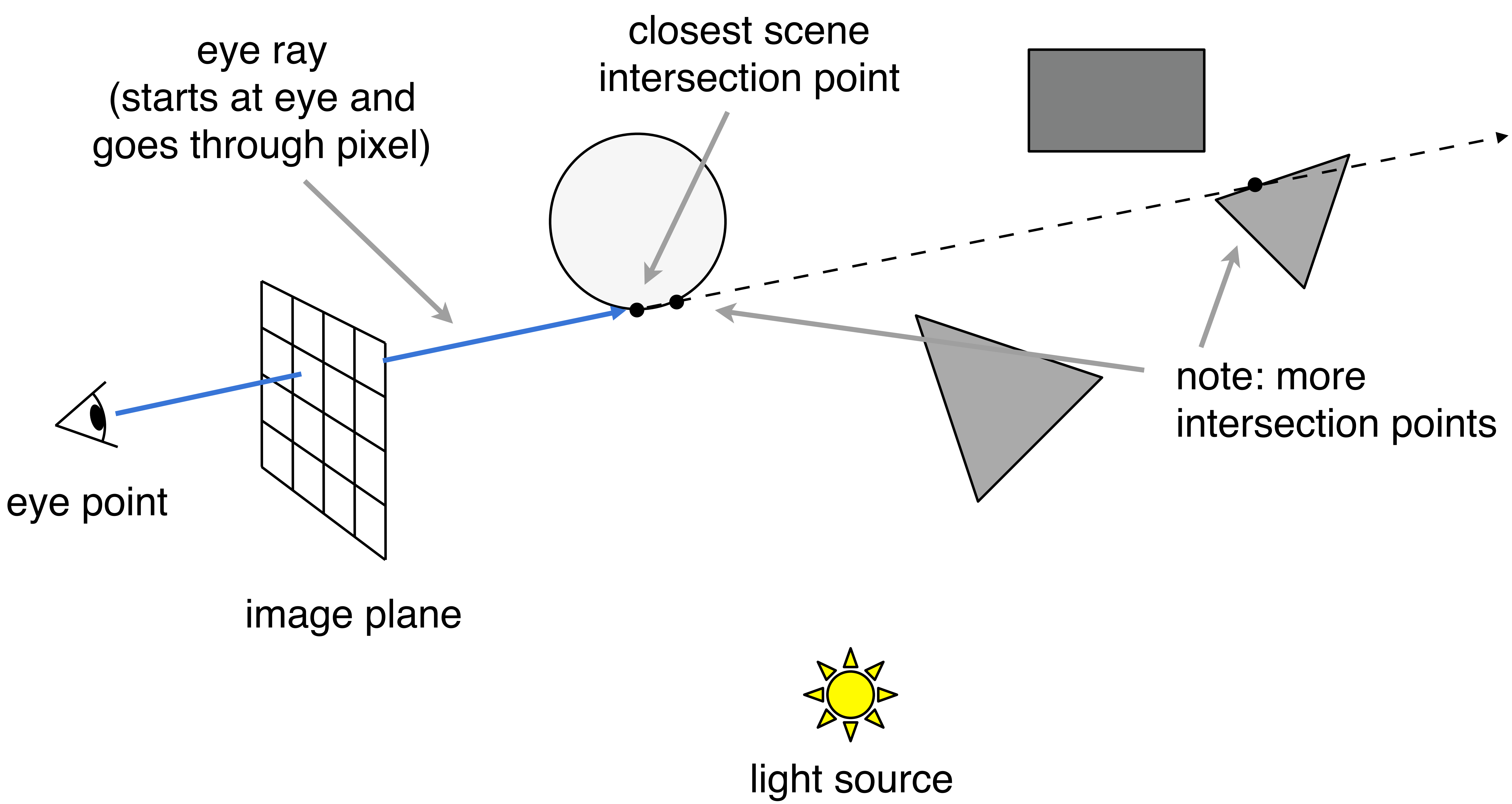
Appel 1968 - Ray casting

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



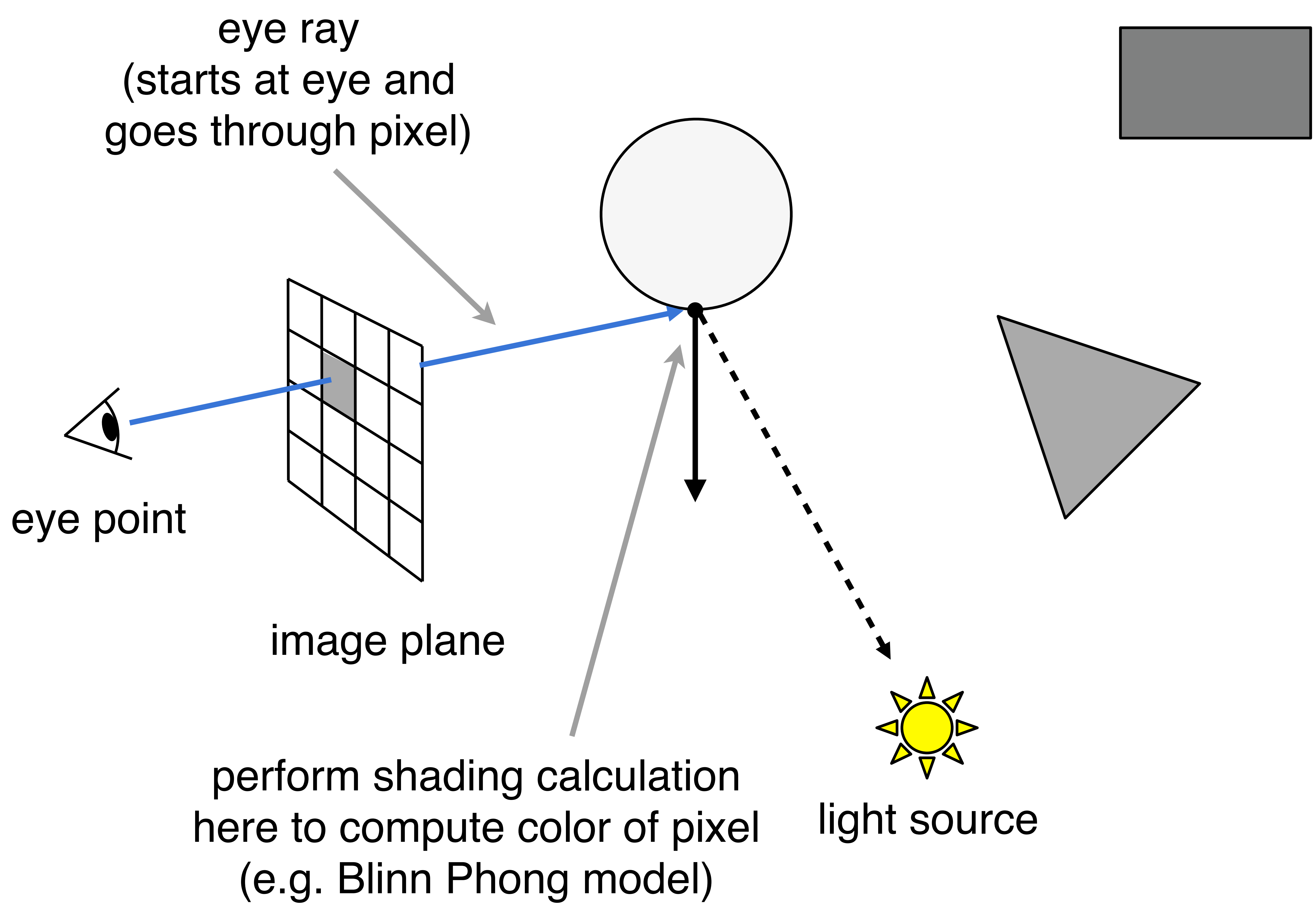
Ray Casting - Generating Eye Rays

Pinhole Camera Model



Ray Casting - Shading Pixels (Local Only)

Pinhole Camera Model

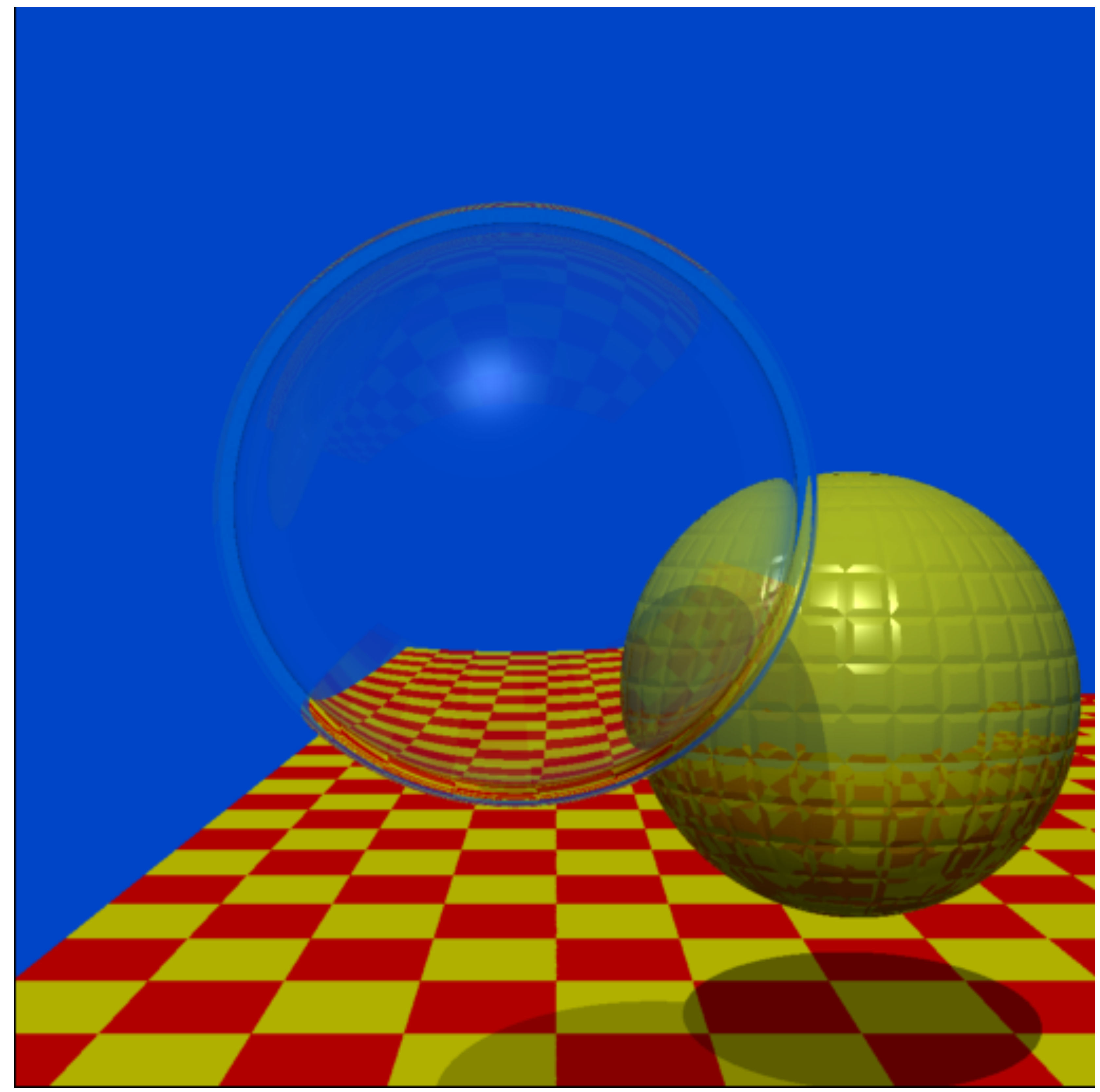


Recursive Ray Tracing

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

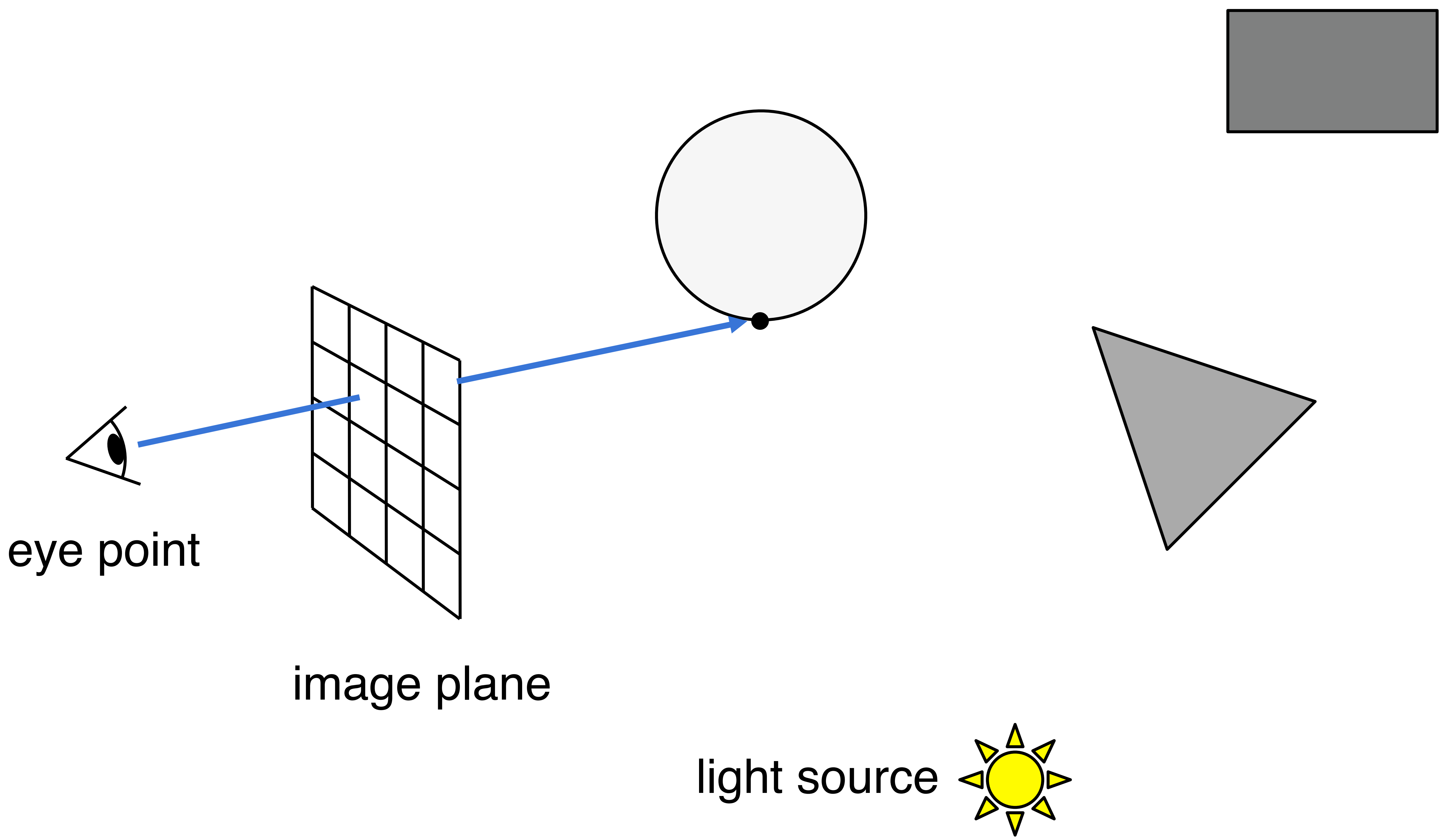
Time:

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

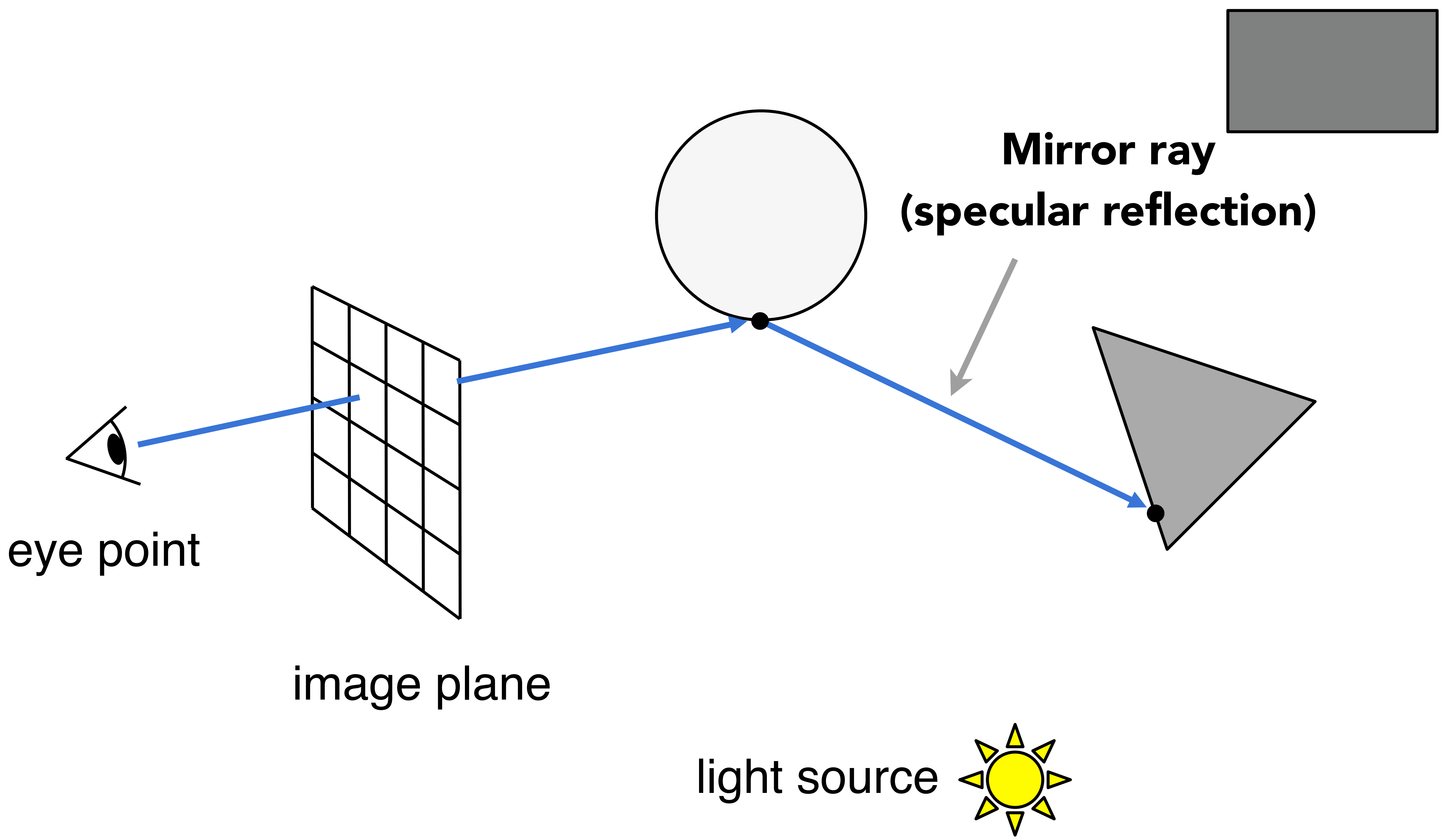


Spheres and Checkerboard, T. Whitted, 1979

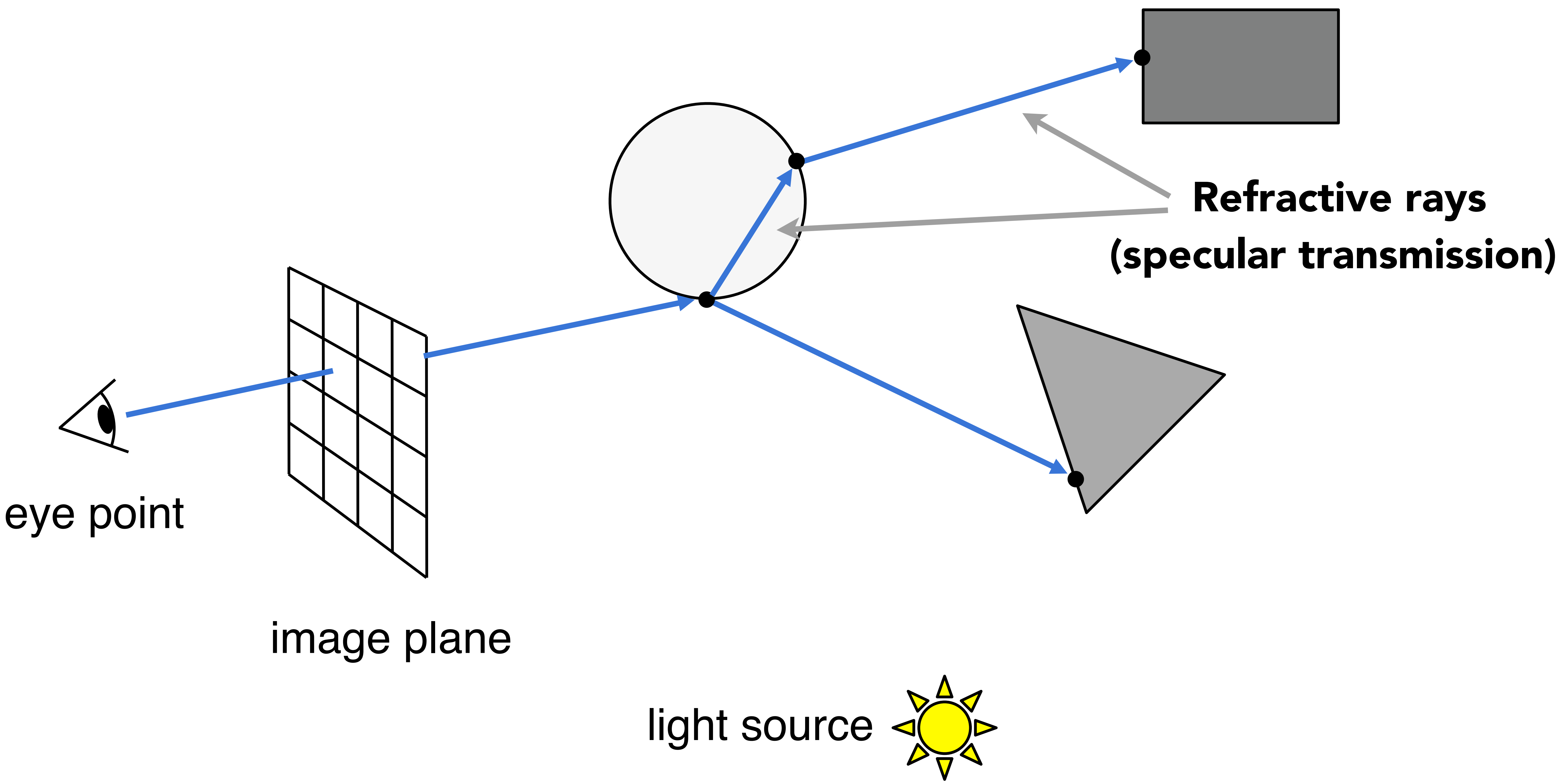
Recursive Ray Tracing



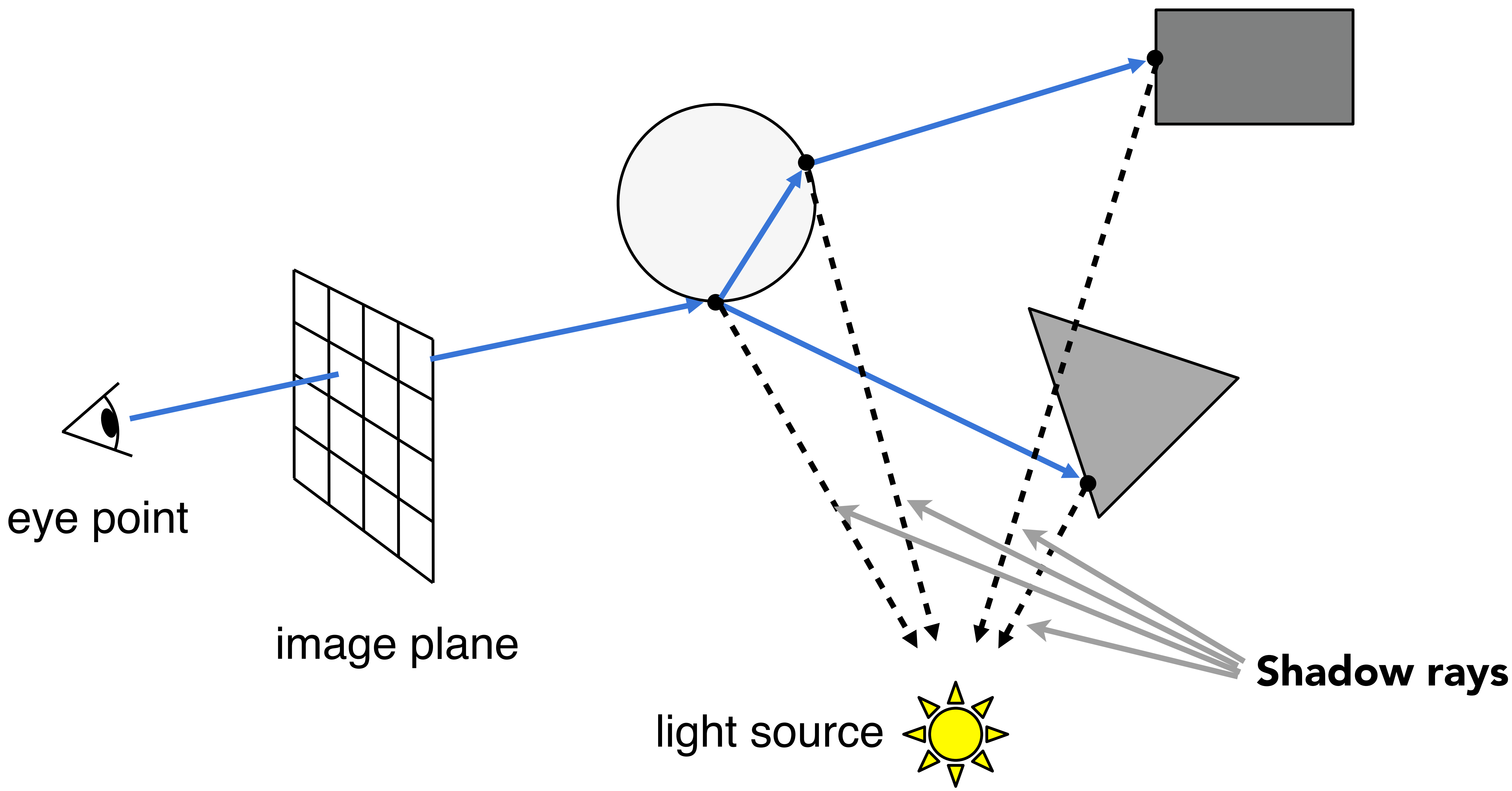
Recursive Ray Tracing



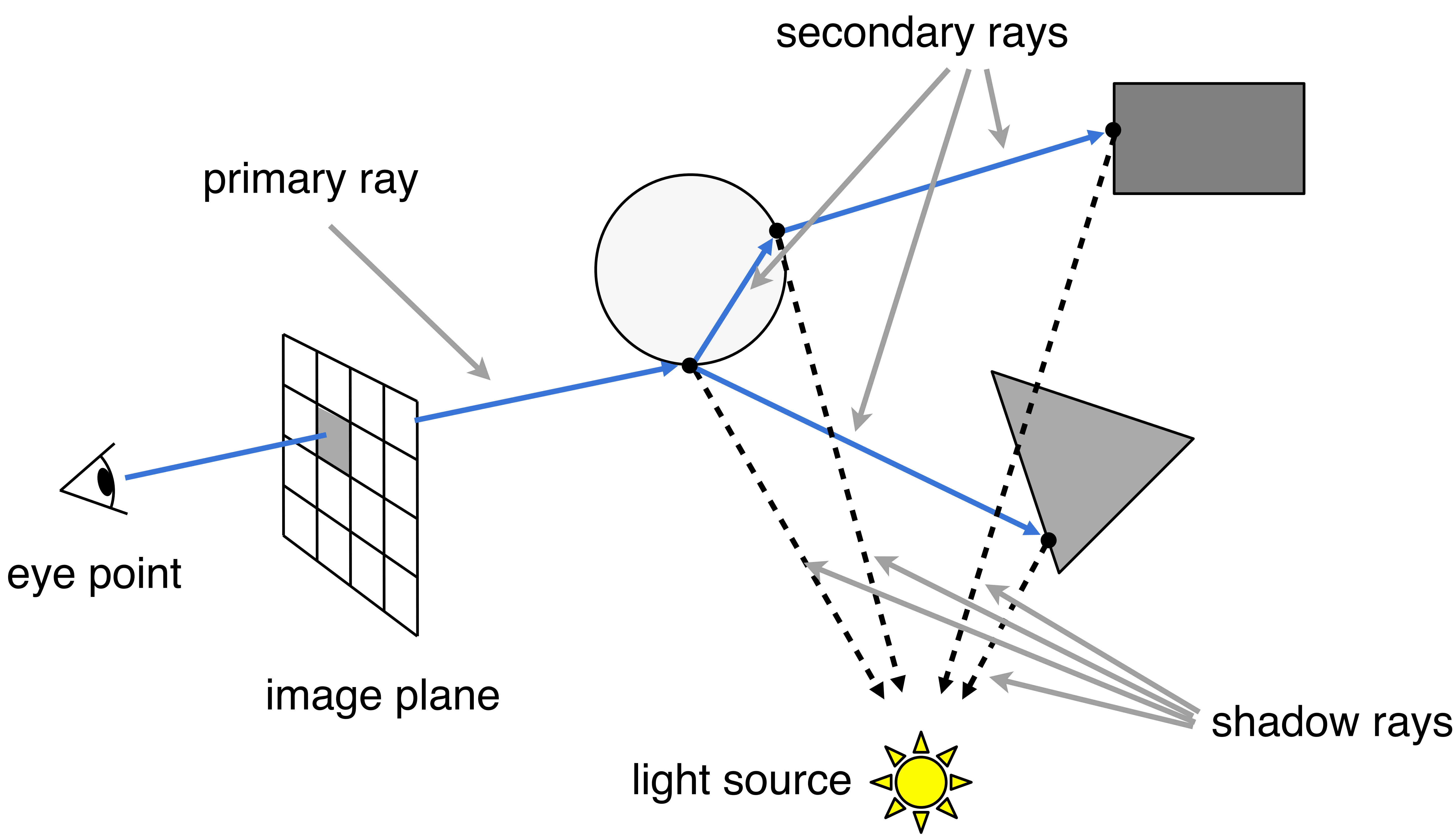
Recursive Ray Tracing



Recursive Ray Tracing



Recursive Ray Tracing



- Trace secondary rays recursively until hit a non-specular surface (or max desired levels of recursion)
- At each hit point, trace shadow rays to test light visibility (no contribution if blocked)
- Final pixel color is weighted sum of contributions along rays, as shown
- Gives more sophisticated effects (e.g. specular reflection, refraction, shadows), but we will go much further to derive a physically-based illumination model

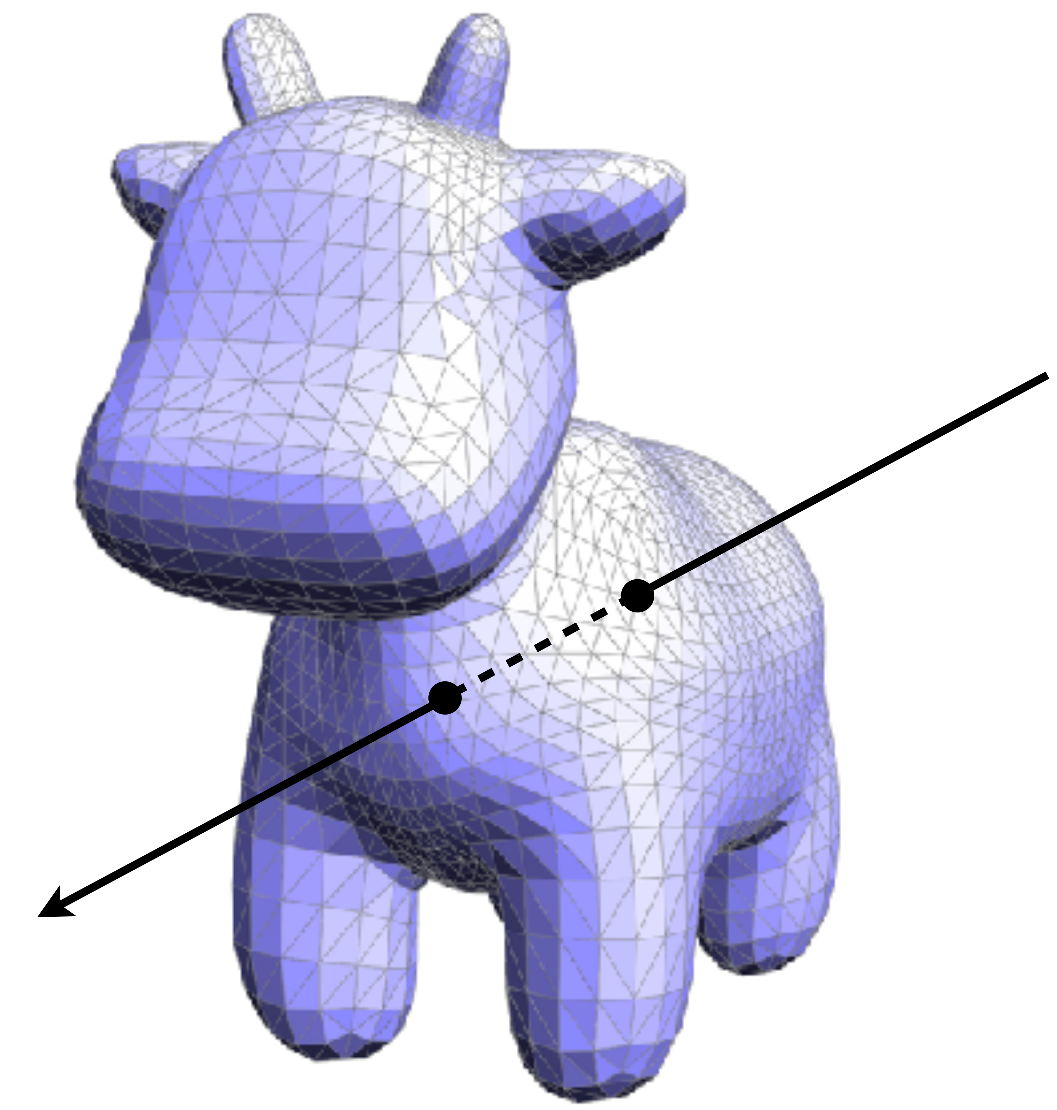
Recursive Ray Tracing

Ray-Surface Intersection

Ray Intersection With Triangle Mesh

Why?

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



How to compute?

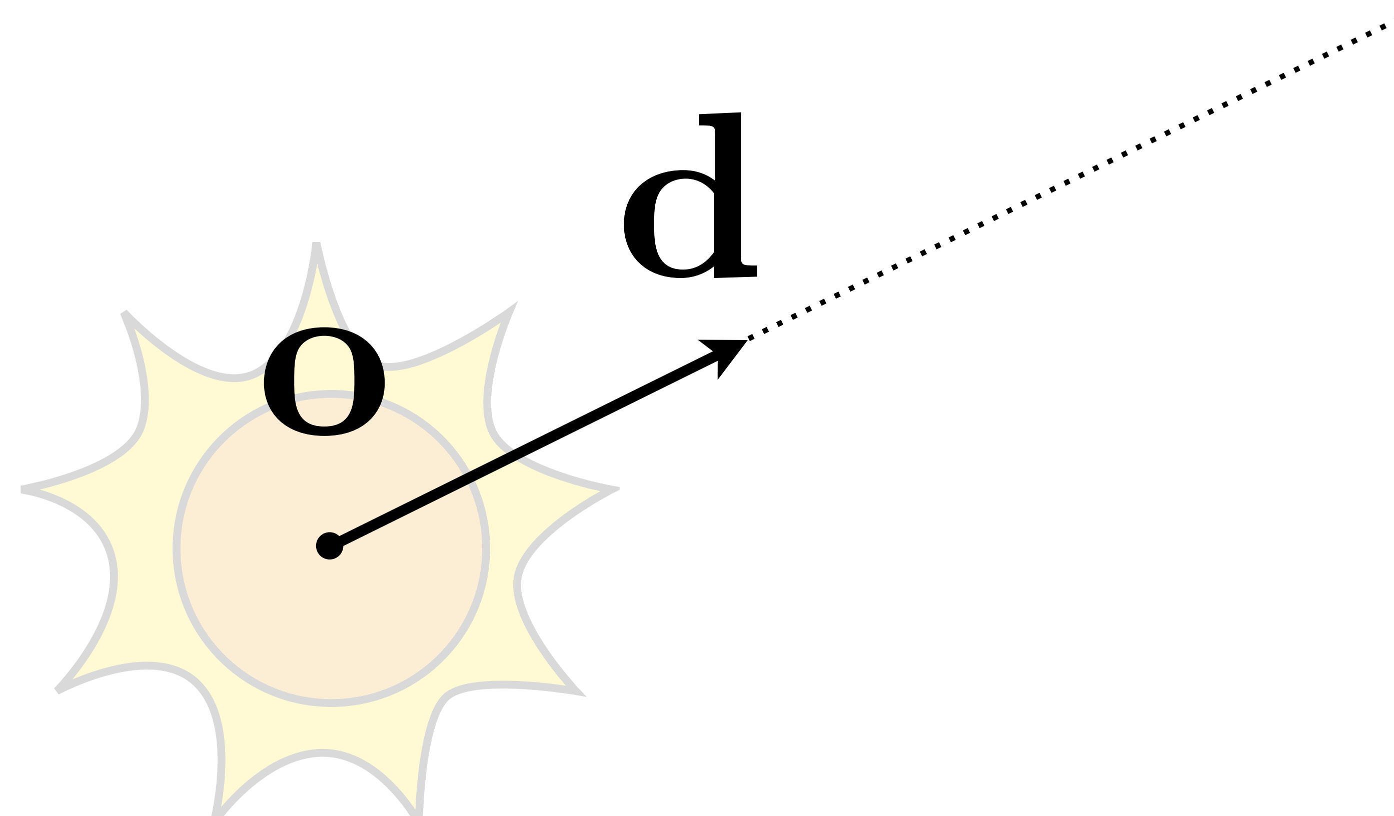
Let’s break this down:

- Simple idea: just intersect ray with each triangle
- Simple, but slow (accelerate next time)
- Note: can have 0, 1 or multiple intersections

Ray Equation

Ray is defined by its origin and a direction vector

Example:



Ray equation:

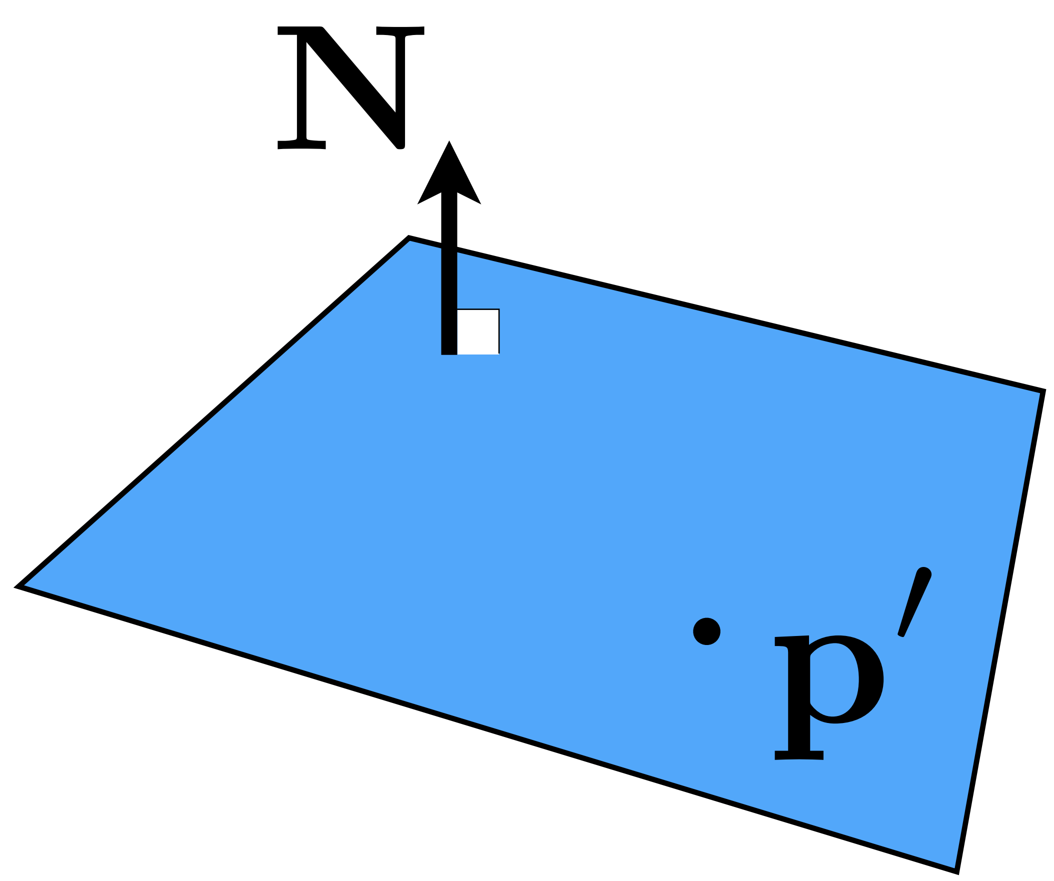
$$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d} \qquad 0 \leq t < \infty$$

point along ray "time" origin unit direction

Plane Equation

Plane is defined by normal vector and a point on plane

Example:

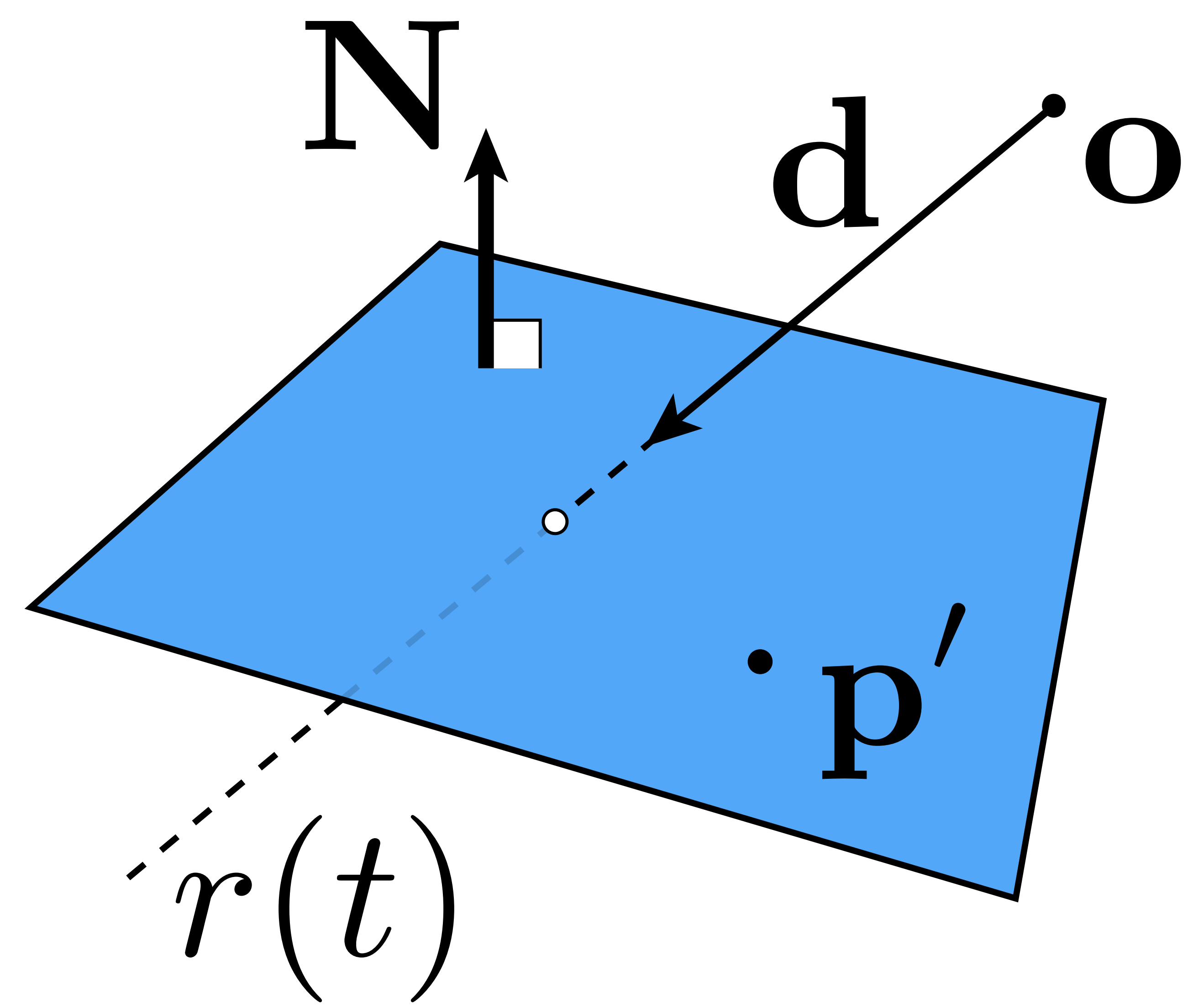


Plane Equation:

$$\mathbf{p} : (\mathbf{p} - \mathbf{p}') \cdot \mathbf{N} = 0 \qquad\qquad ax + by + cz + d = 0$$

all points on plane   any point   normal vector

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

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

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

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

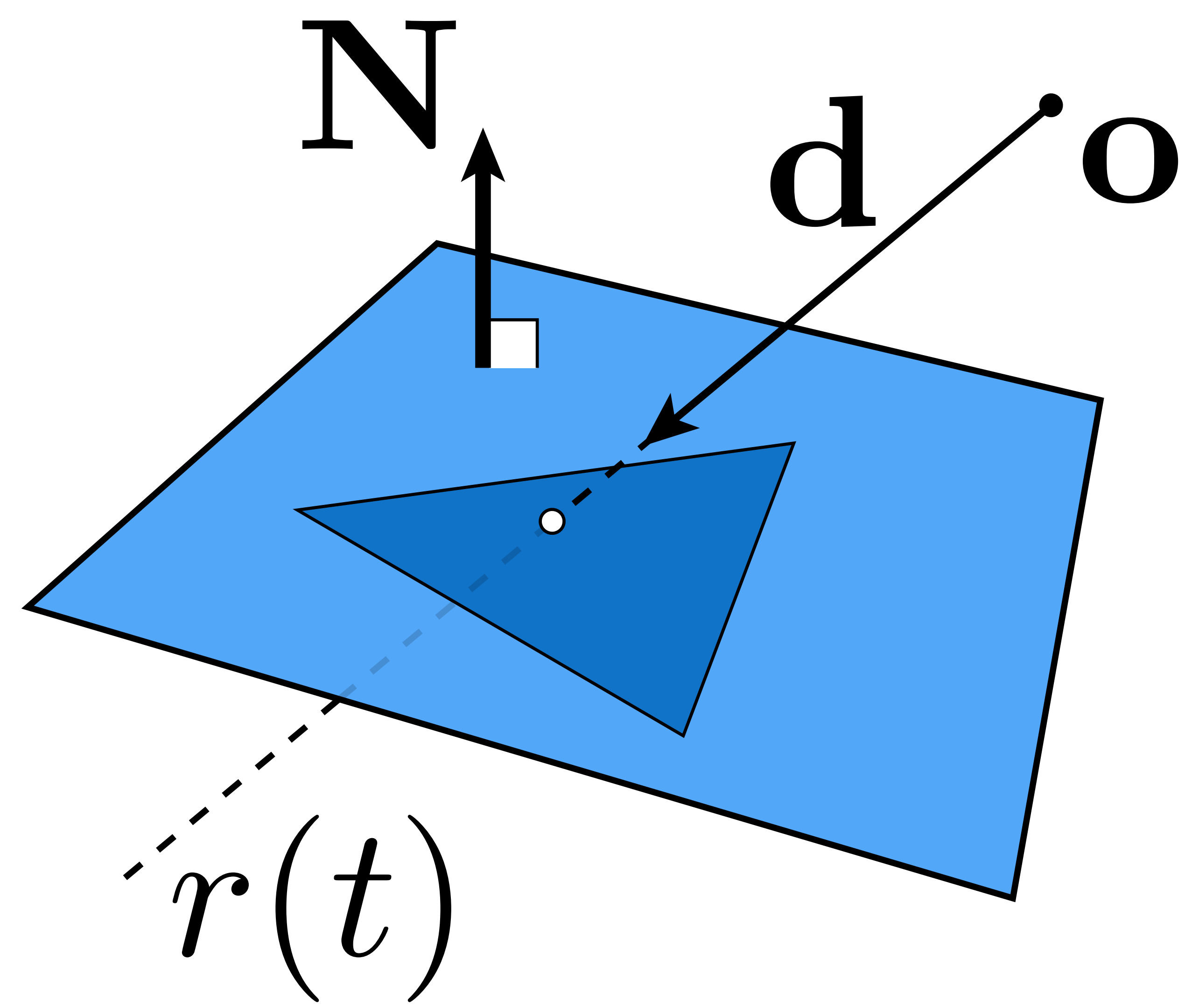
Check: $0 \le t < \infty$

Ray Intersection With Triangle

Triangle is in a plane

- Ray-plane intersection
- Test if hit point is inside triangle (Assignment 1!)

Many ways to optimize…



Can Optimize: e.g. Möller Trumbore Algorithm

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

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

Where:

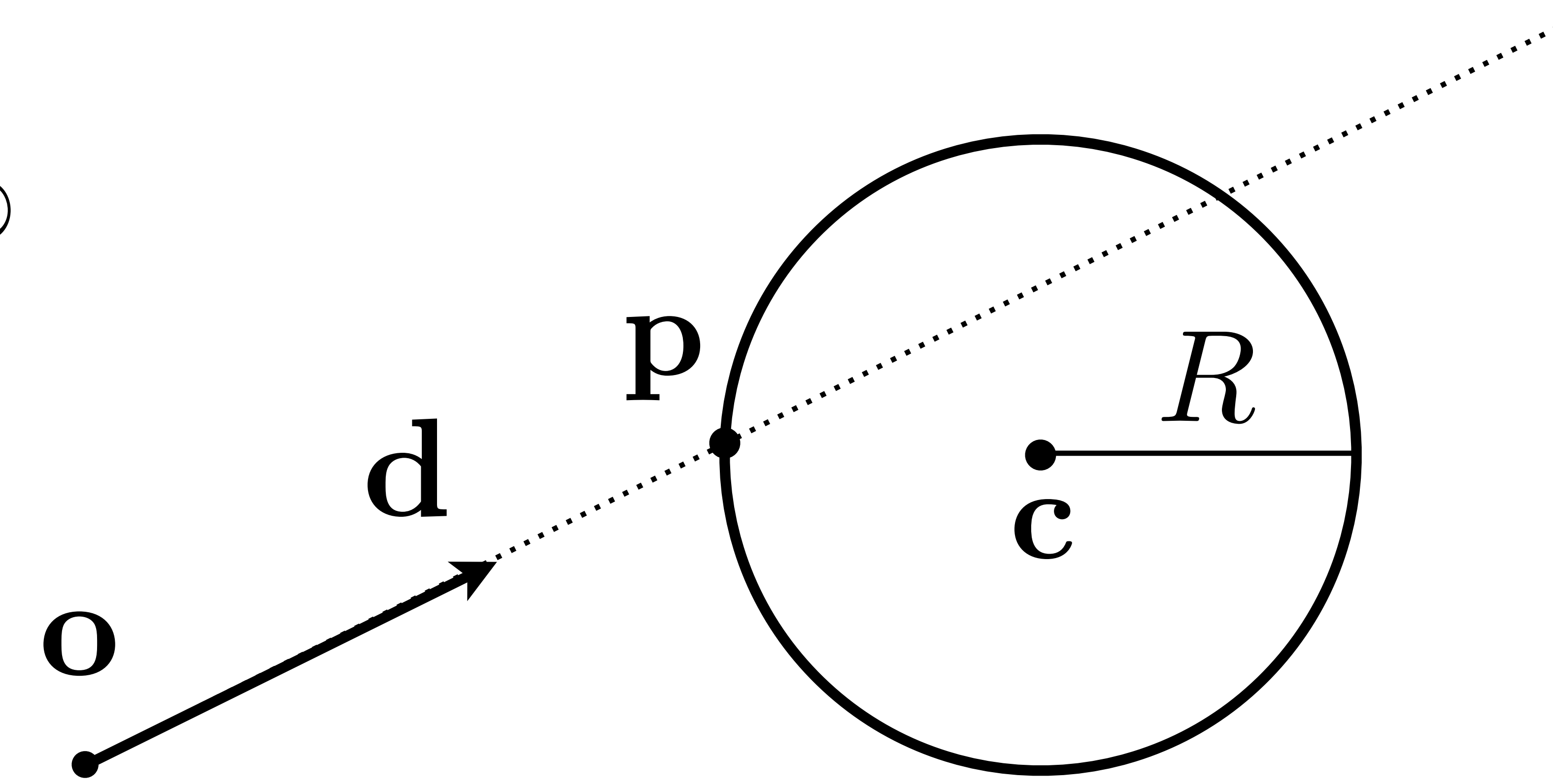
$$\vec{\mathbf{E}}_1 = \vec{\mathbf{P}}_1 - \vec{\mathbf{P}}_0$$
$$\vec{\mathbf{E}}_2 = \vec{\mathbf{P}}_2 - \vec{\mathbf{P}}_0$$
$$\vec{\mathbf{S}} = \vec{\mathbf{O}} - \vec{\mathbf{P}}_0$$
$$\vec{\mathbf{S}}_1 = \vec{\mathbf{D}} \times \vec{\mathbf{E}}_2$$
$$\vec{\mathbf{S}}_2 = \vec{\mathbf{S}} \times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

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

Ray Intersection With Sphere

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

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



Solve for intersection:

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

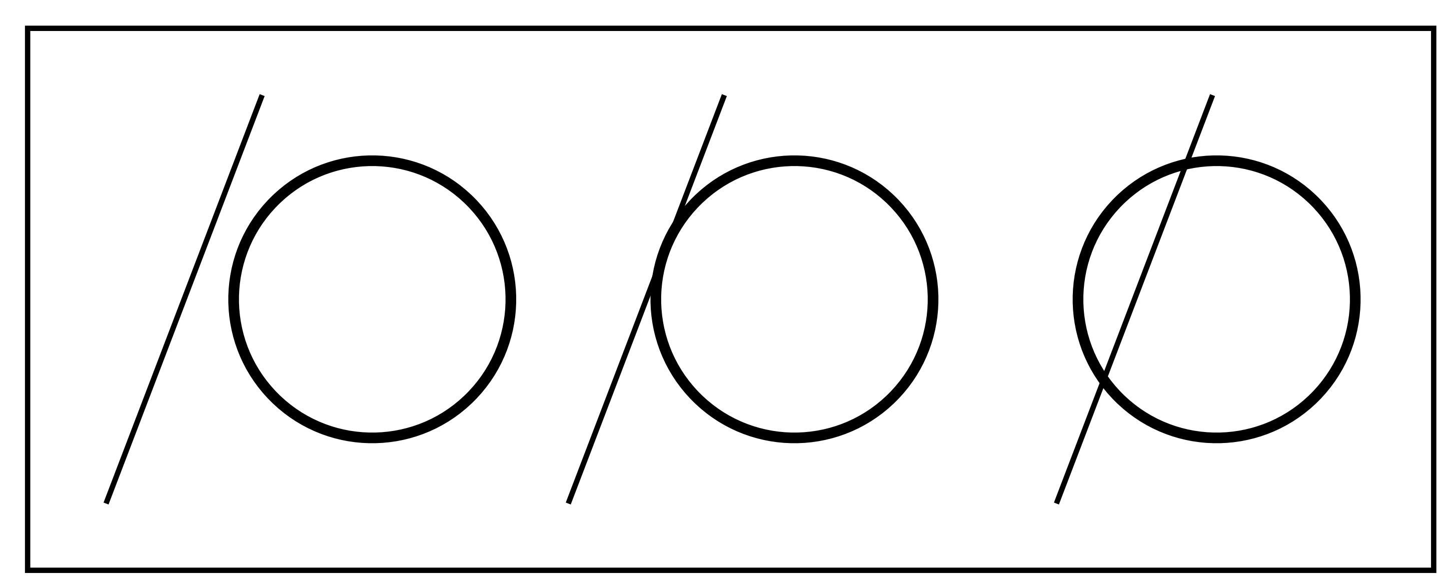
$$a\,t^2 + b\,t + c = 0, \text{ where}$$

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

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

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

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



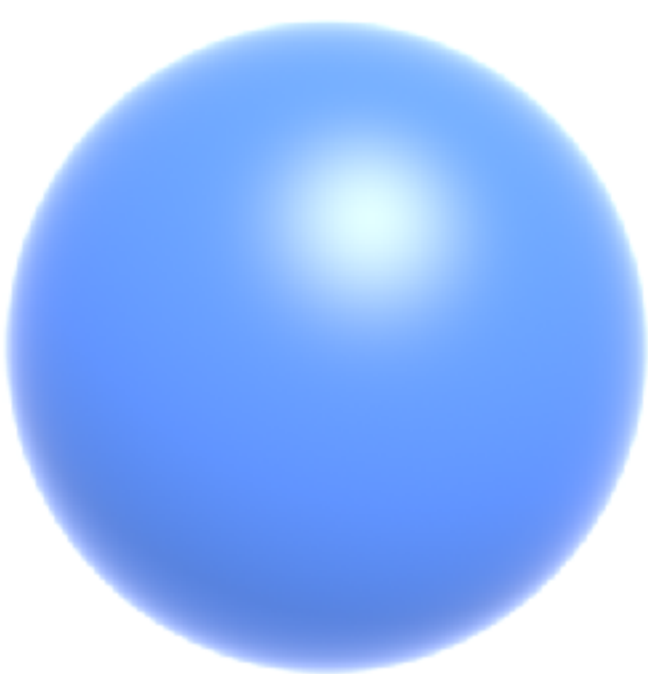
Ray Intersection With Implicit Surface

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

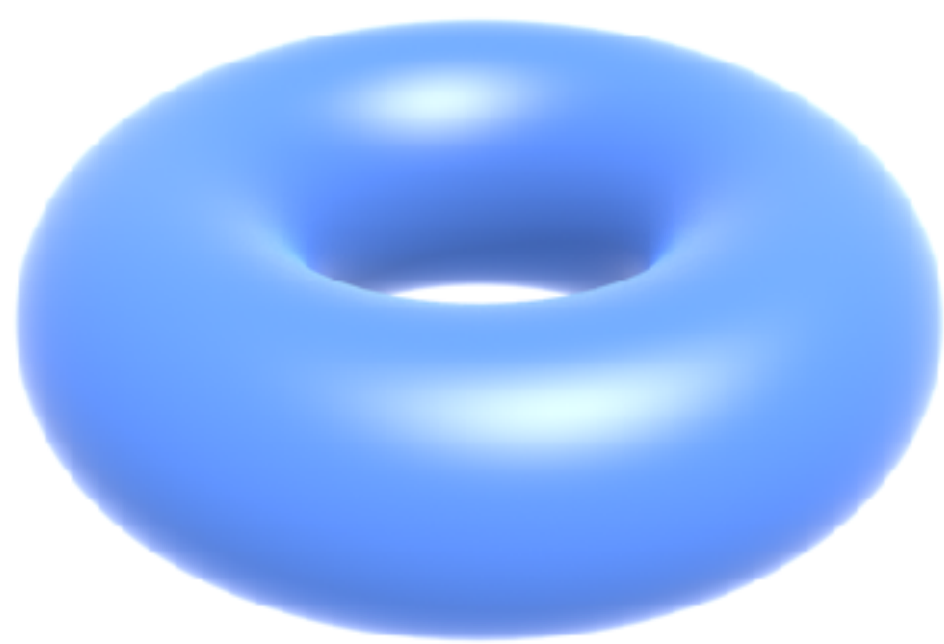
General implicit surface: $\mathbf{p} : f(\mathbf{p}) = 0$

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

Solve for real, positive roots



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



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

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

Accelerating Ray-Surface Intersection

Ray Tracing – Performance Challenges

Simple ray-scene intersection

- Exhaustively test ray-intersection with every object

Problem:

- Exhaustive algorithm = #pixels $\times$ #objects
- Very slow!

Ray Tracing – Performance Challenges

San Miguel Scene, 10.7M triangles

Ray Tracing – Performance Challenges

Plant Ecosystem, 20M triangles

Discussion: Accelerating Ray-Scene Intersection

~1 million pixels, ~20 million triangles

Deussen et al; Pharr & Humphreys, PBRT

In pairs, brainstorm accelerations, small or big ideas.

Write down 3-4 ideas.

Discussion: Accelerating Ray-Scene Intersection

Brainstorm 3 or 4 accelerations, small or big ideas.

- Heuristics for which triangle to look at first
- Ignore triangles behind the camera
- Get lazy with small triangles and blur faraway regions
- Copy and paste computation if many similar parts of trees
- Inspired by quick sort, try to bound these are the only triangles in this area — and maybe do so recursively
- If image is sparse, apply compressive sensing somehow?
- Quad tree to hierarchically organize the scene
- “Parenting” objects - idea of hierarchical representation of scene
- Stop recursion after a few steps for non-detailed areas
- Reverse the ray — start from the camera — why?
- Perform blocking to improve cache performance
- Selective ray-tracing — optimize for specific visual effects trace only the relevant rays from those surfaces

Bounding Volumes

Bounding Volumes

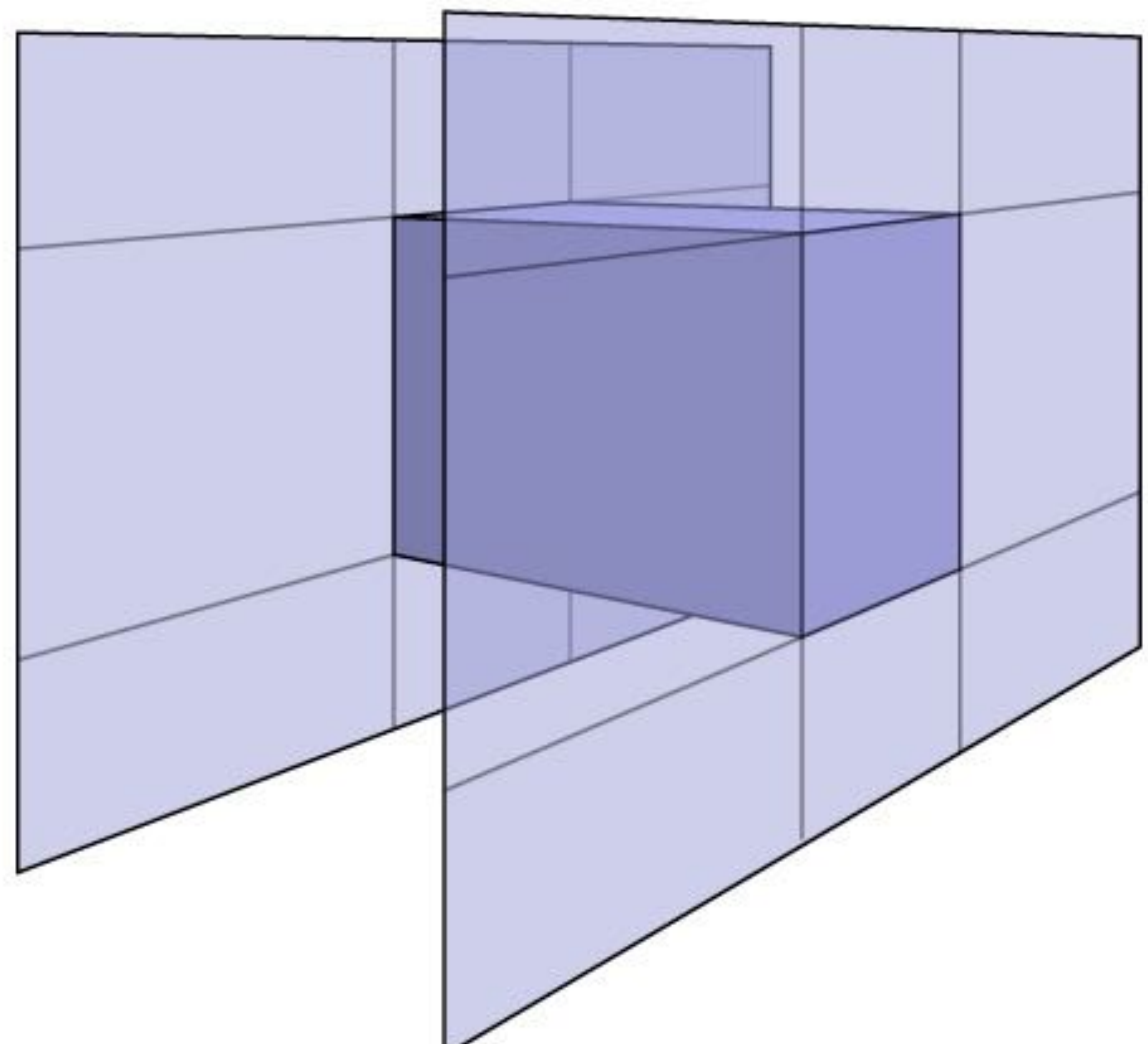
Quick way to avoid intersections: bound complex object with a simple volume

- 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

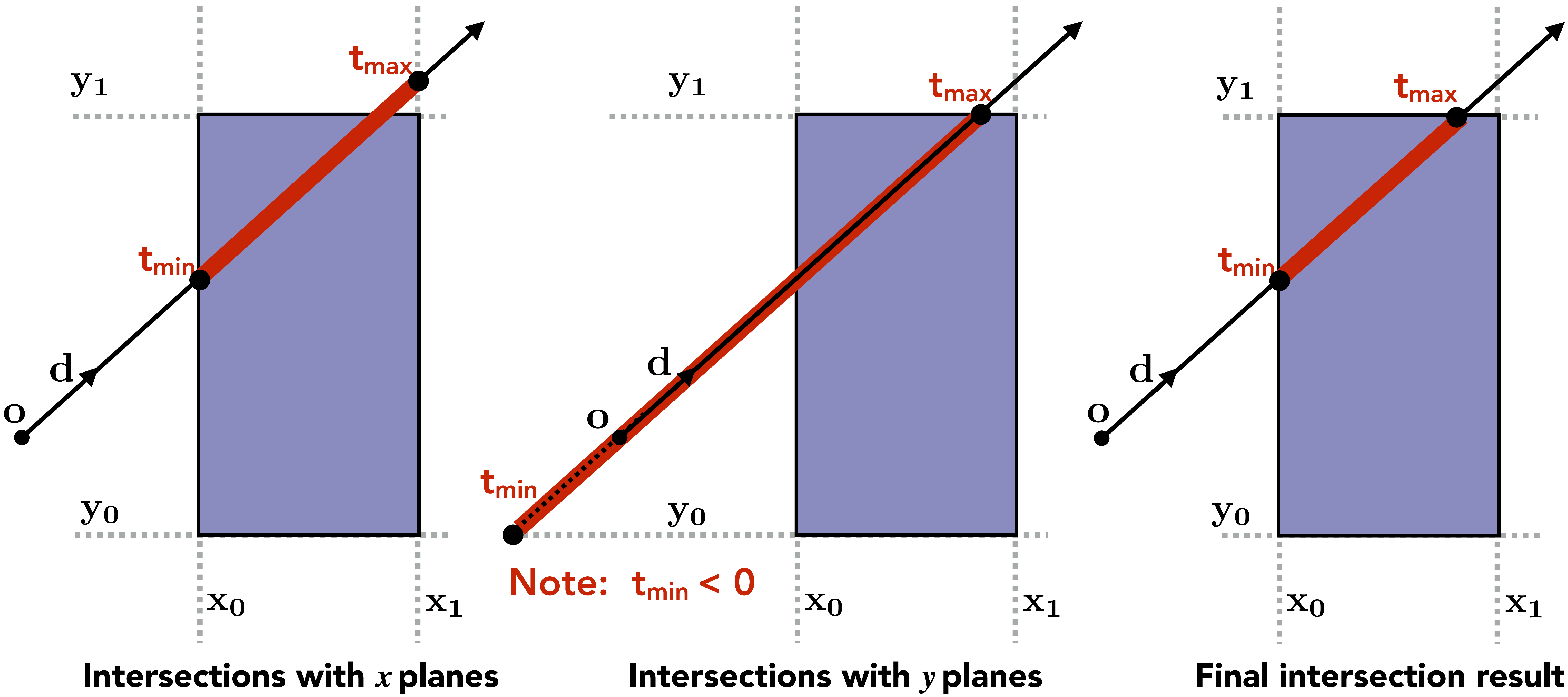
Could intersect with 6 faces individually

Better way: box is the intersection of 3 slabs



Ray Intersection with Axis-Aligned Box

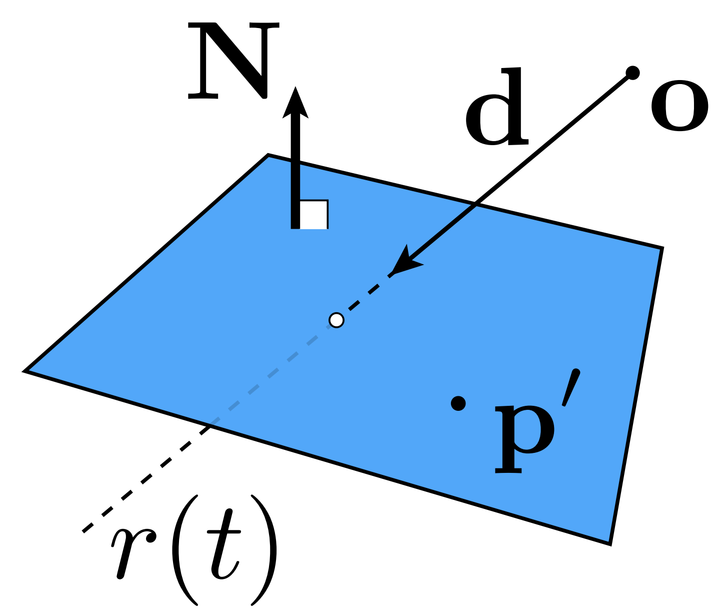
2D example; 3D is the same! Compute intersections with slabs and take intersection of t_{min}/t_{max} intervals



How do we know when the ray misses the box?

Optimize Ray-Plane Intersection For Axis-Aligned Planes?

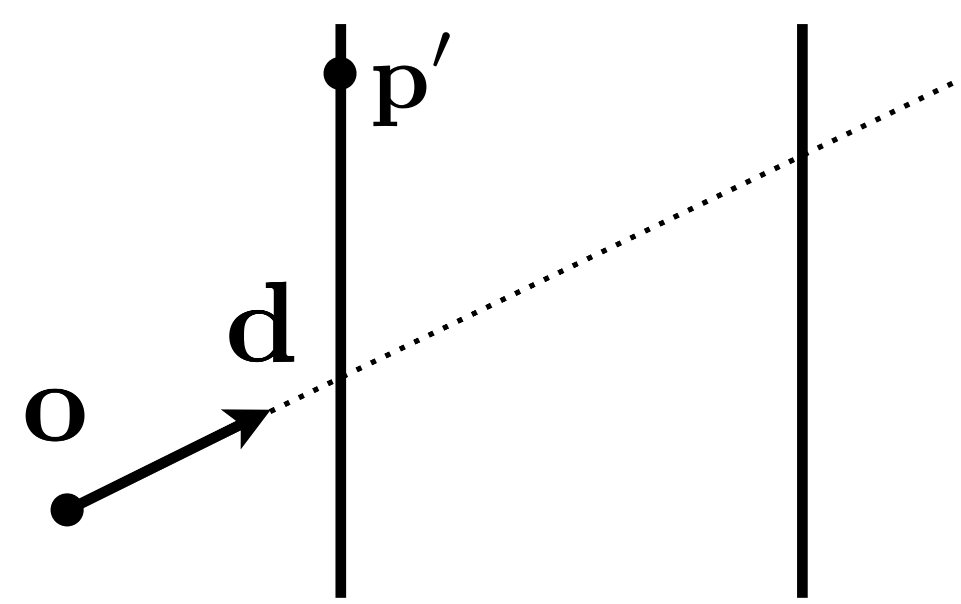
General



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

3 subtractions, 6 multiplies, 1 division

Perpendicular to x-axis



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

1 subtraction, 1 division

To Be Continued

Acknowledgments

Thanks to Pat Hanrahan, Kayvon Fatahalian, Mark Pauly and Steve Marschner for lecture resources.