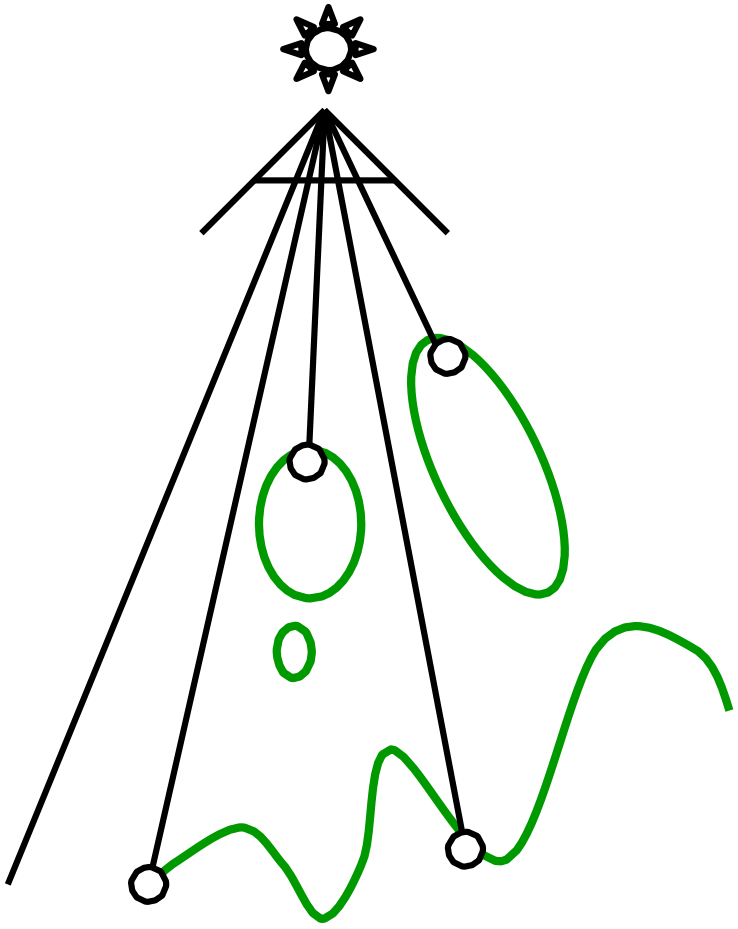
Shadow Mapping

- An Image-space Algorithm
  - no knowledge of scene's geometry during shadow computation
  - must deal with aliasing artifacts

- Key idea:
  - the points NOT in shadow must be seen both **by the light** and **by the camera**

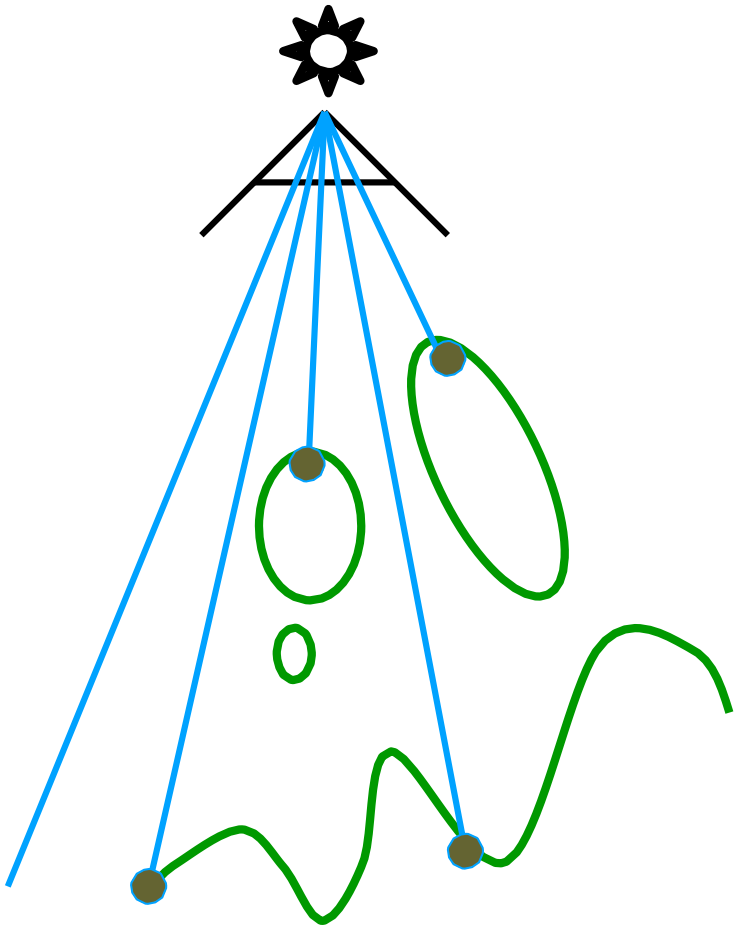
Pass 1: Render from Light

- Depth image from light source



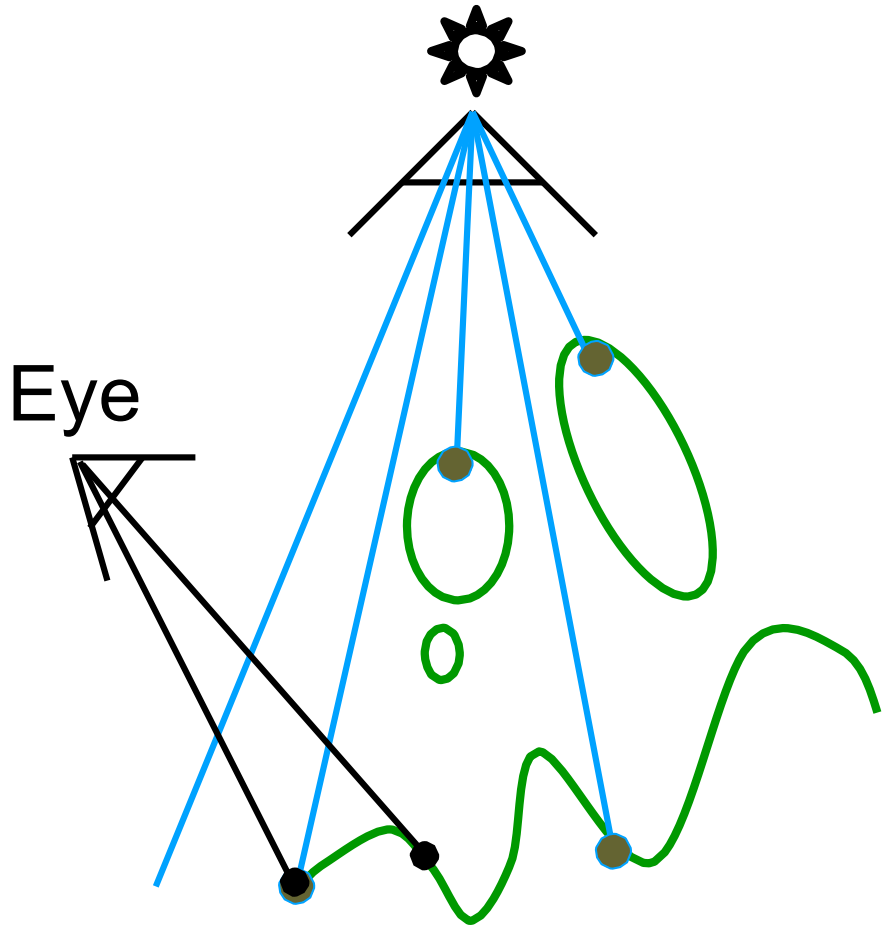
Pass 1: Render from Light

- Depth image from light source



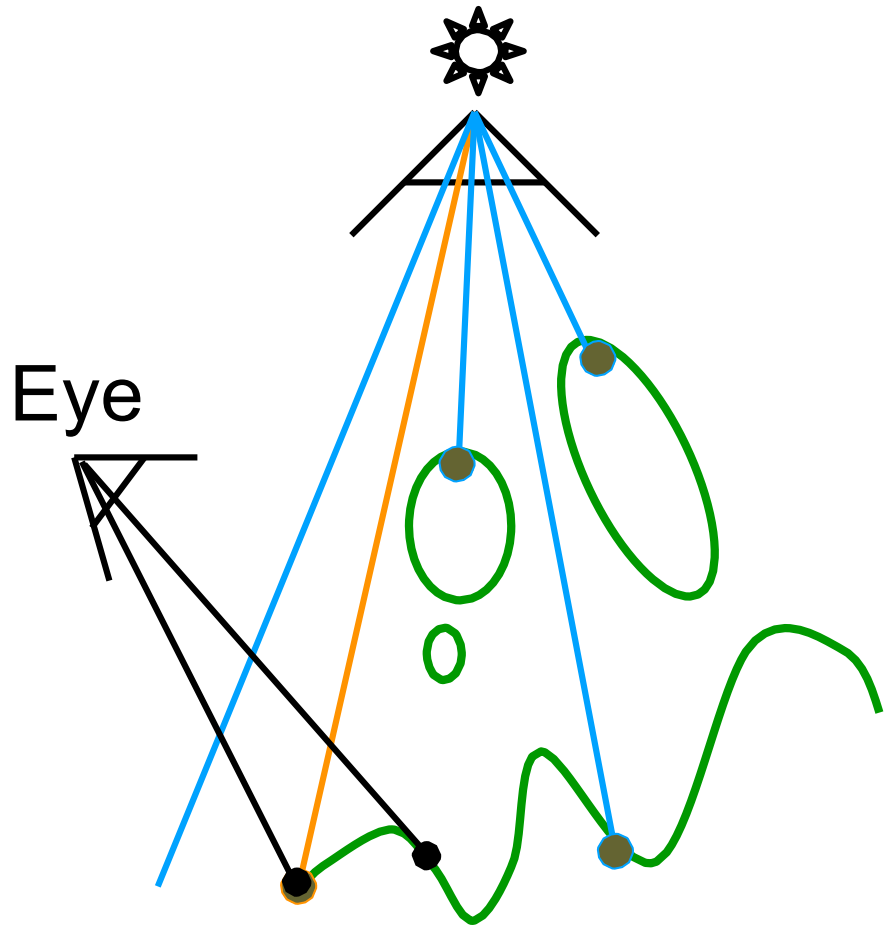
Pass 2A: Render from Eye

- Standard image (with depth) from eye



Pass 2B: Project to light

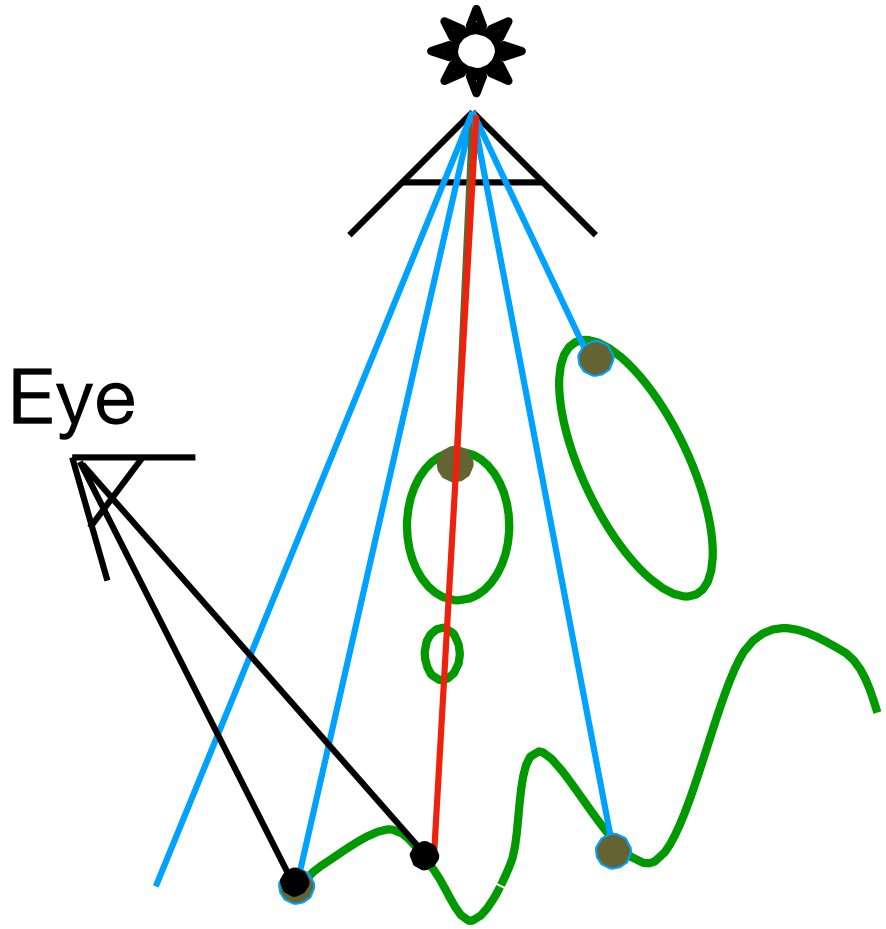
- Project visible points in eye view back to light source



(Reprojected) depths match for light and eye. VISIBLE

Pass 2B: Project to light

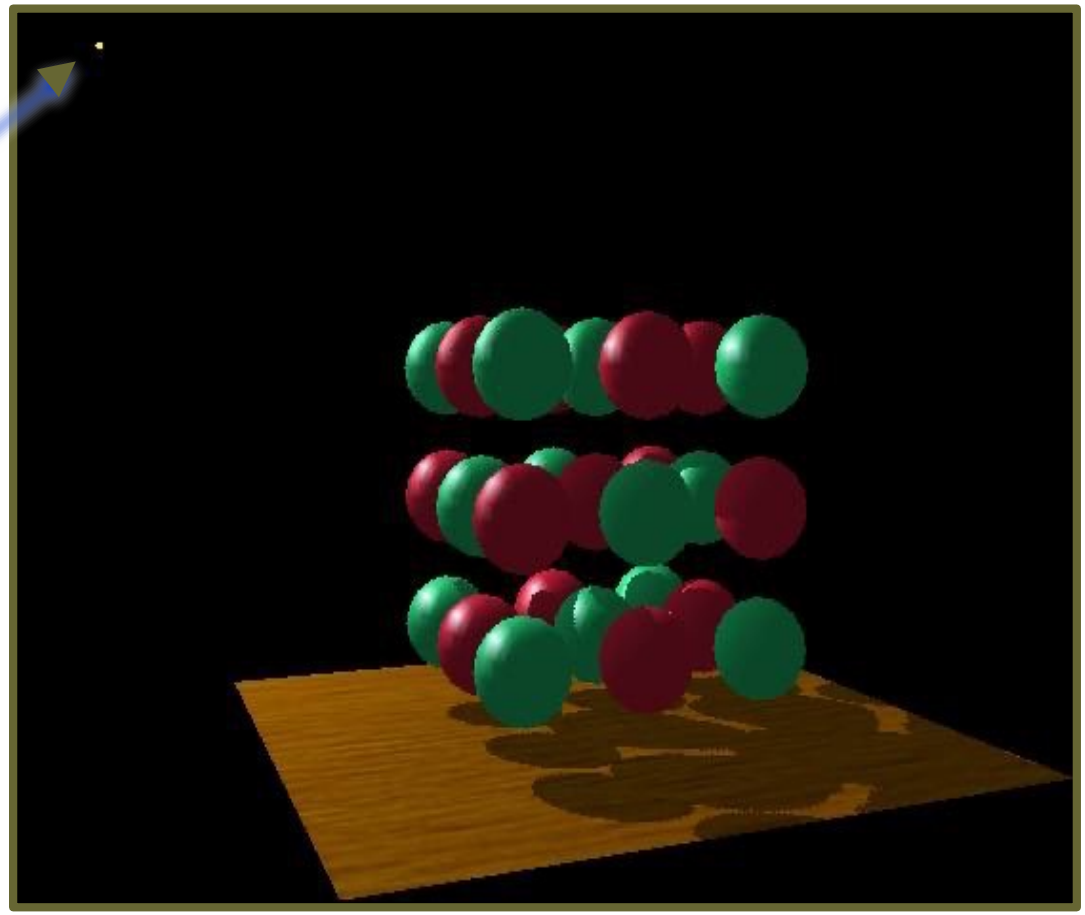
- Project visible points in eye view back to light source



(Reprojected) depths from light and eye are not the same. BLOCKED!!

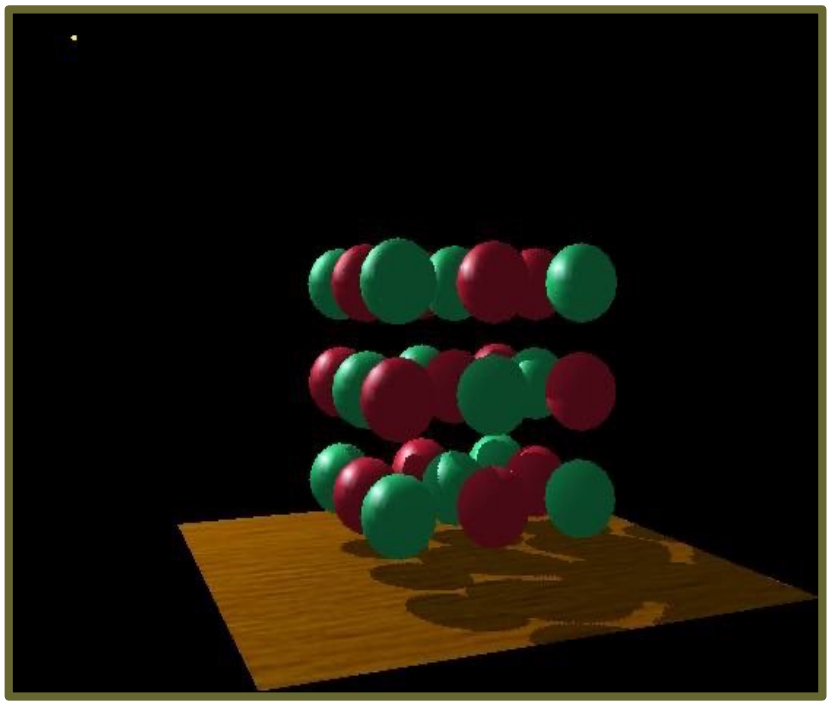
Visualizing Shadow Mapping

- A fairly complex scene with shadows

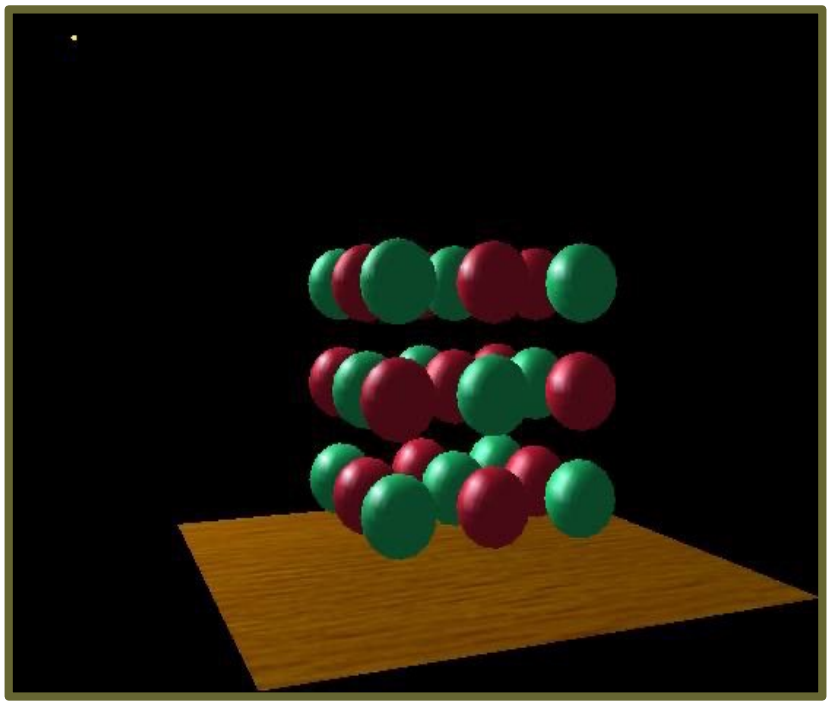


Visualizing Shadow Mapping

- Compare with and without shadows



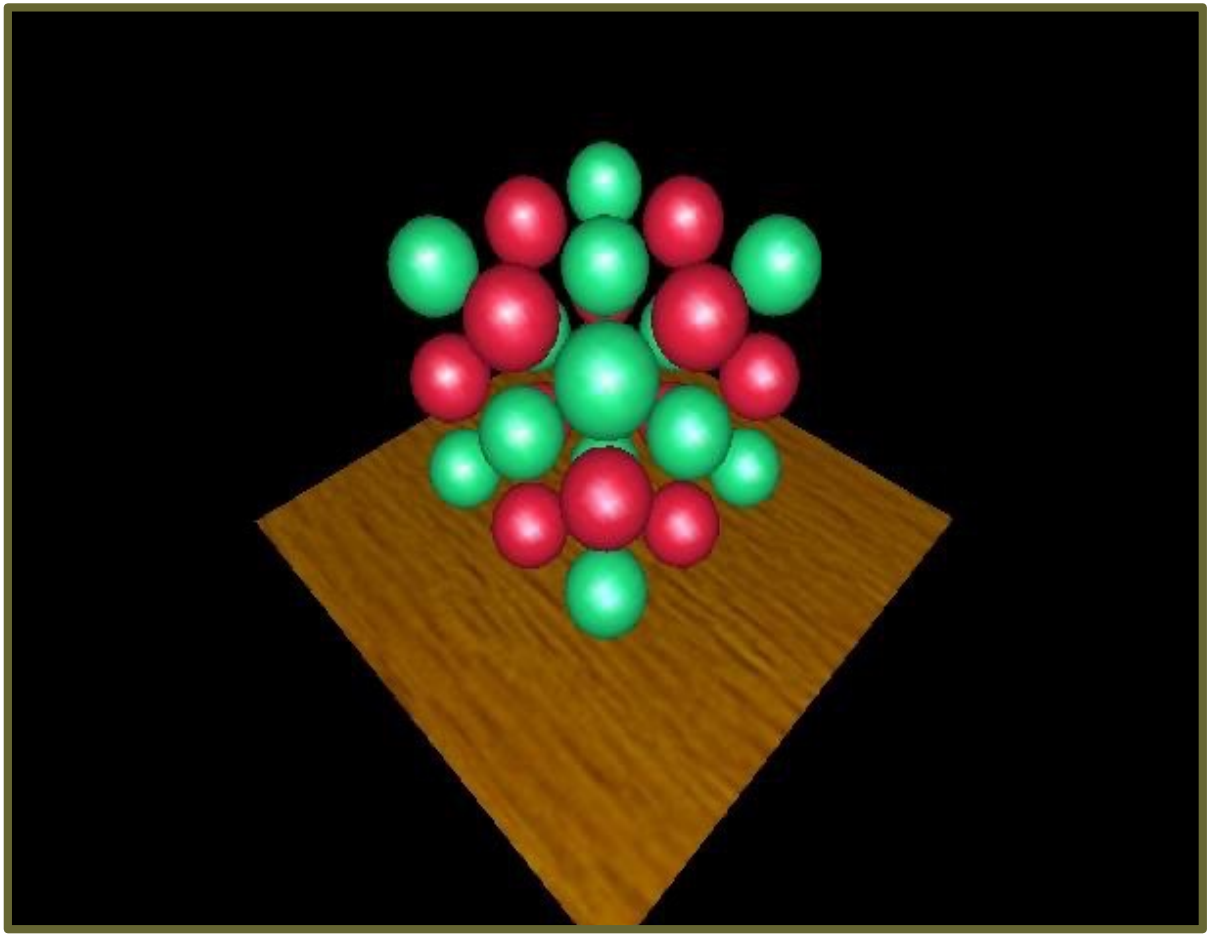
with shadows

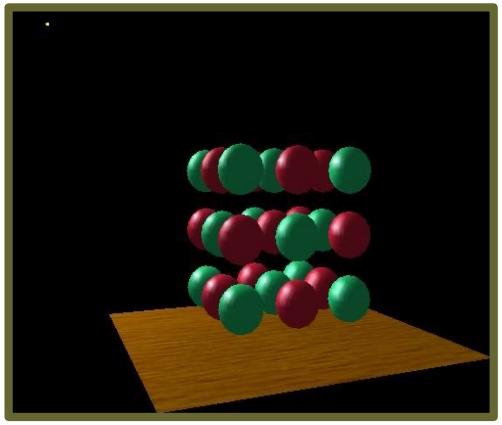


without shadows

Visualizing Shadow Mapping

- The scene from the light's point-of-view

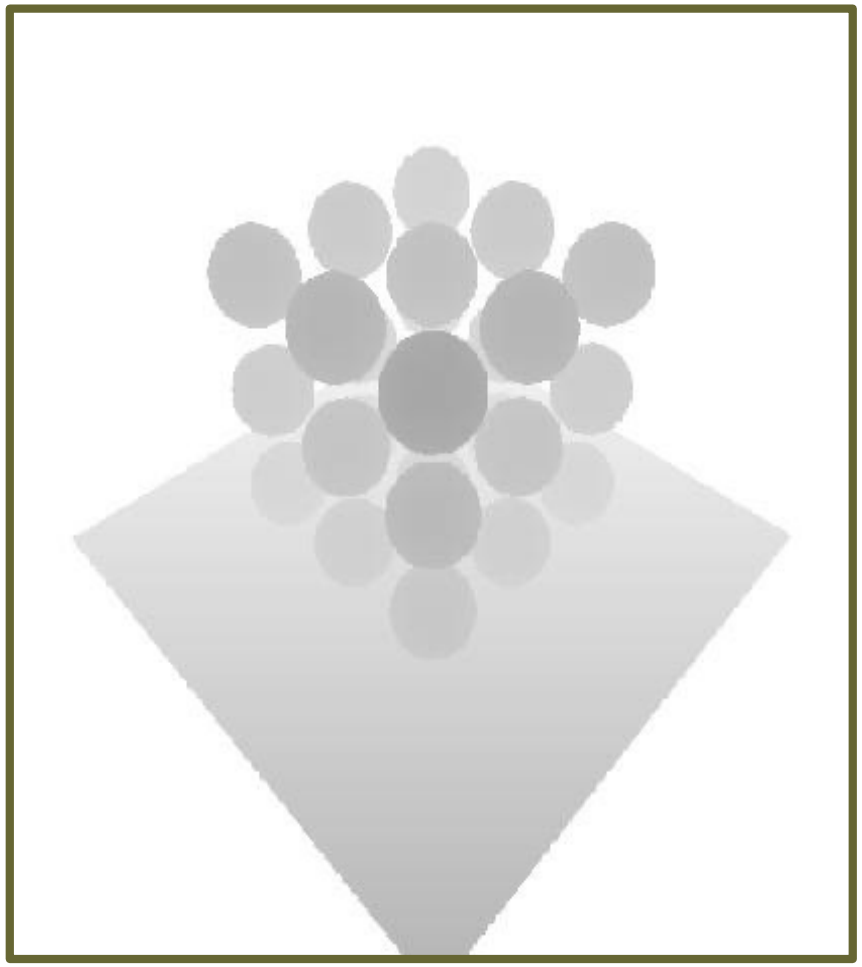


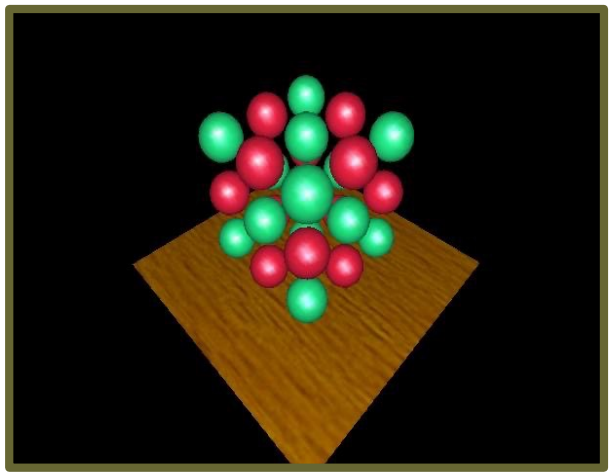


FYI: from the eye's point-of-view again

Visualizing Shadow Mapping

- The depth buffer from the light's point-of-view



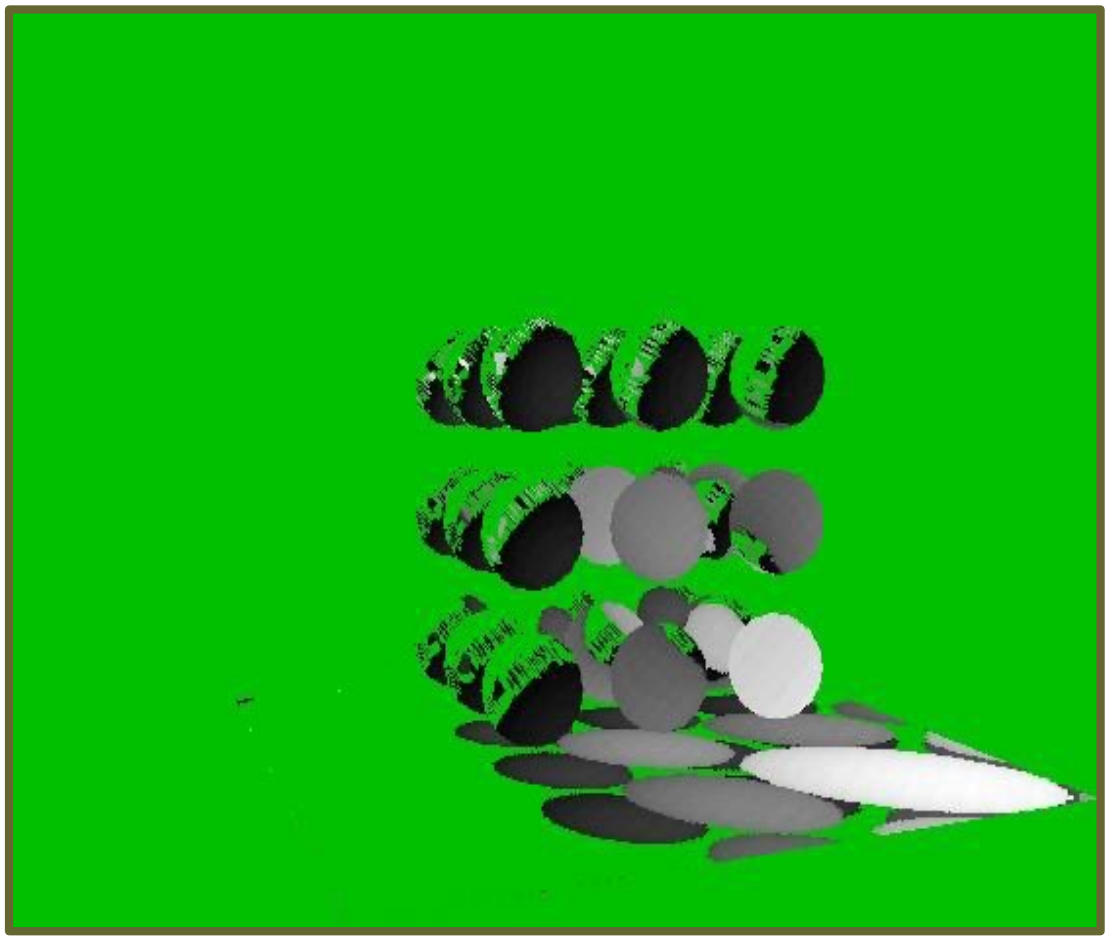


FYI: from the light's point-of-view again

Visualizing Shadow Mapping

- Comparing Dist(light, shading point) with shadow map

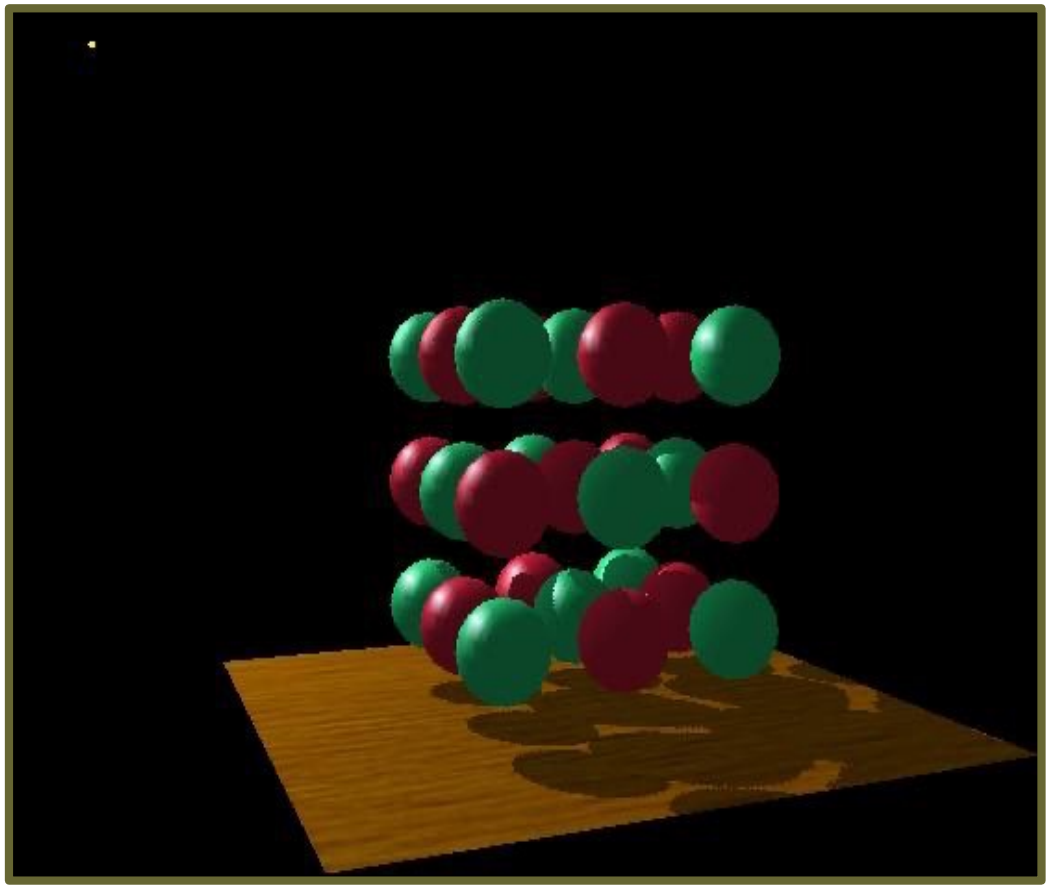
Green is where the distance(light, shading point) ≈ depth on the shadow map



Non-green is where shadows should be

Visualizing Shadow Mapping

- Scene with shadows



Shadow Mapping

- Well known rendering technique
  - Basic shadowing technique for early animations (Toy Story, etc.) and in EVERY 3D video game

Zelda: Breath of the Wild

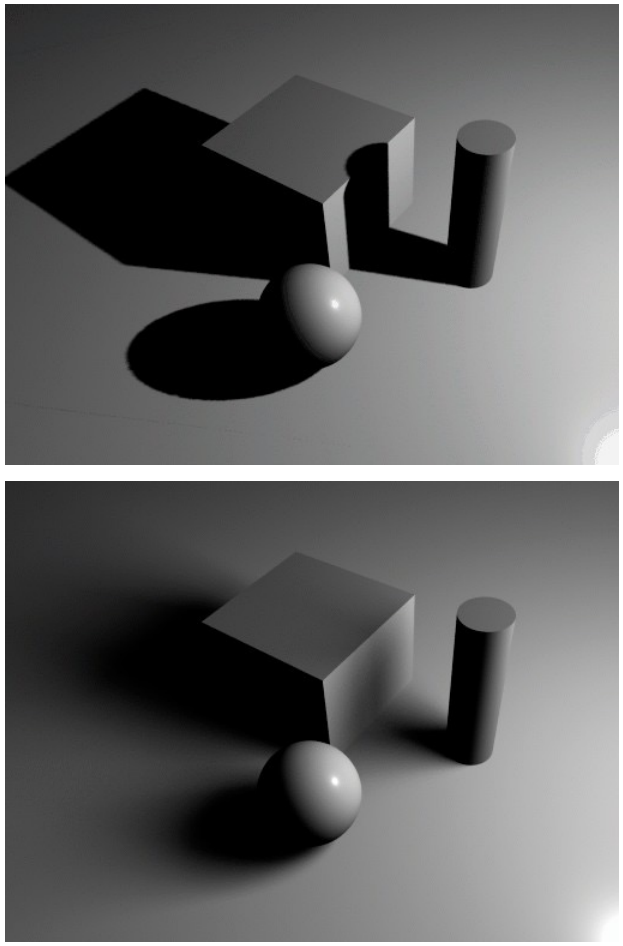
Super Mario Odyssey

Problems with shadow maps

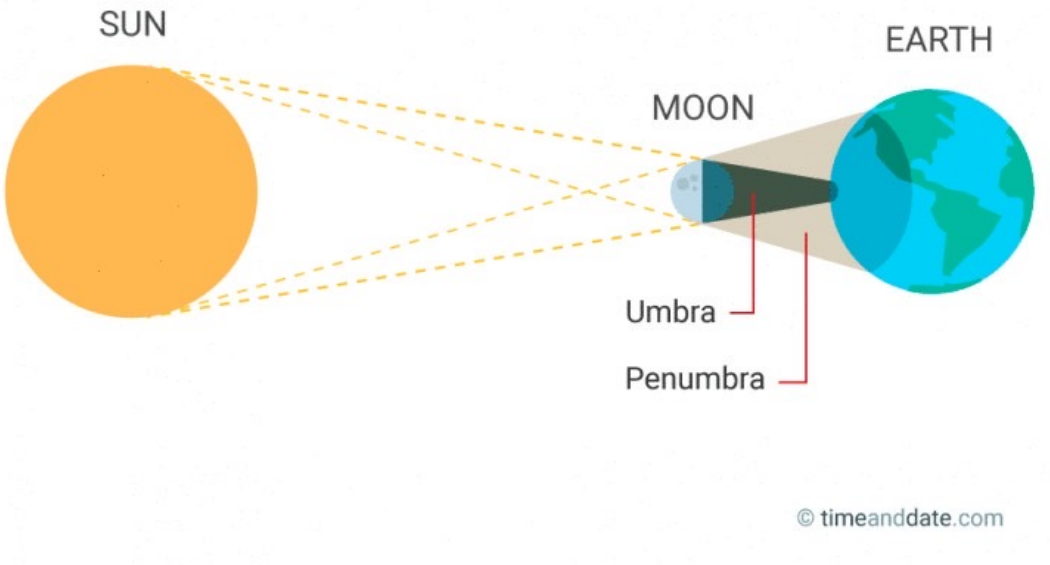
- Hard shadows (point lights only)
- Quality depends on shadow map resolution (general problem with image-based techniques)
- Involves equality comparison of floating point depth values means issues of scale, bias, tolerance

Problems with shadow maps

- Hard shadows vs. soft shadows



[RenderMan]



[https://www.timeanddate.com/eclipse/umbra-shadow.html]

Ray Tracing I

(Whitted-Style Ray Tracing)

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

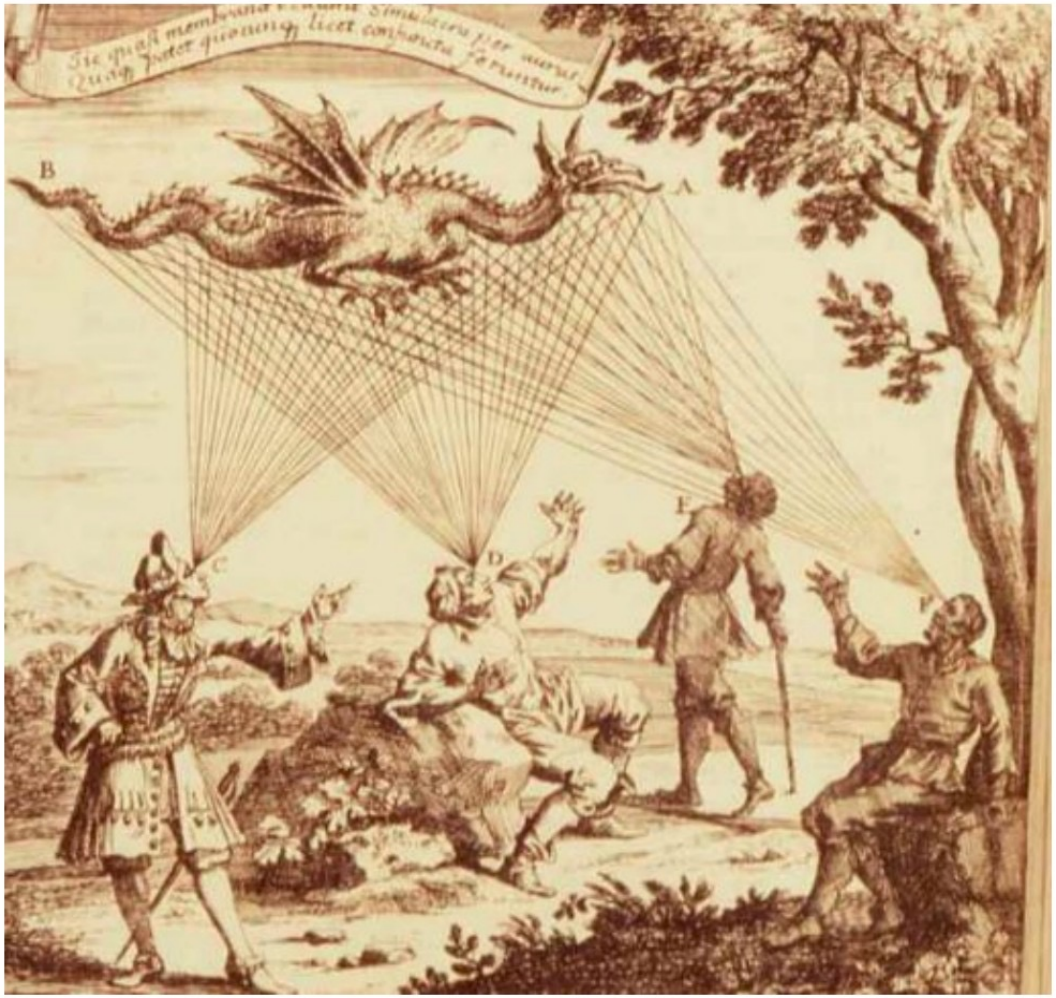
Three ideas about light rays

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*

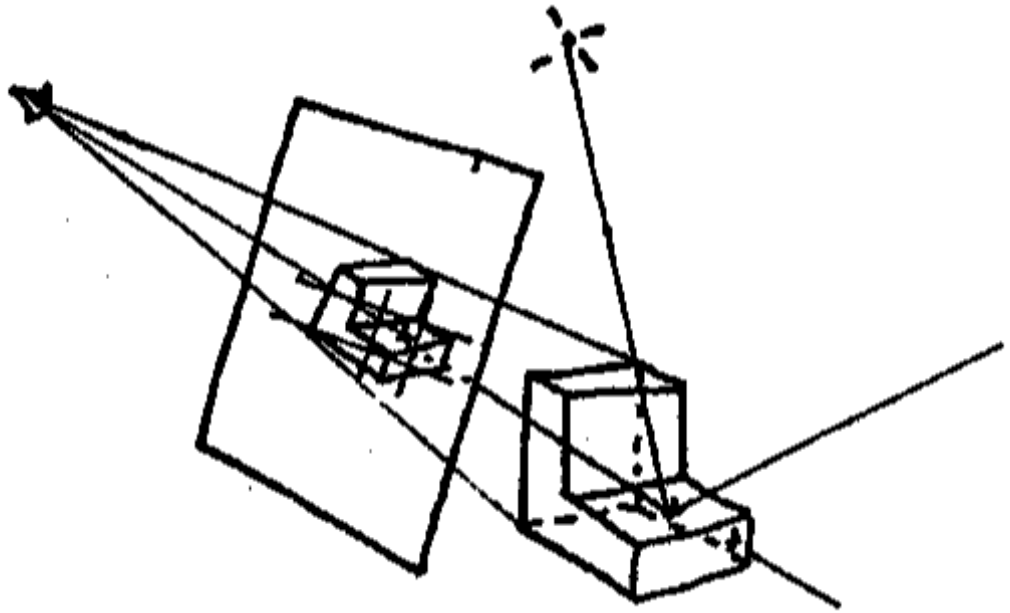
*http://www.ncbi.nlm.nih.gov/pubmed/12094435?dopt=Abstract

[Slide courtesy of Prof. Alexei Efros, UC Berkeley]

Ray Casting

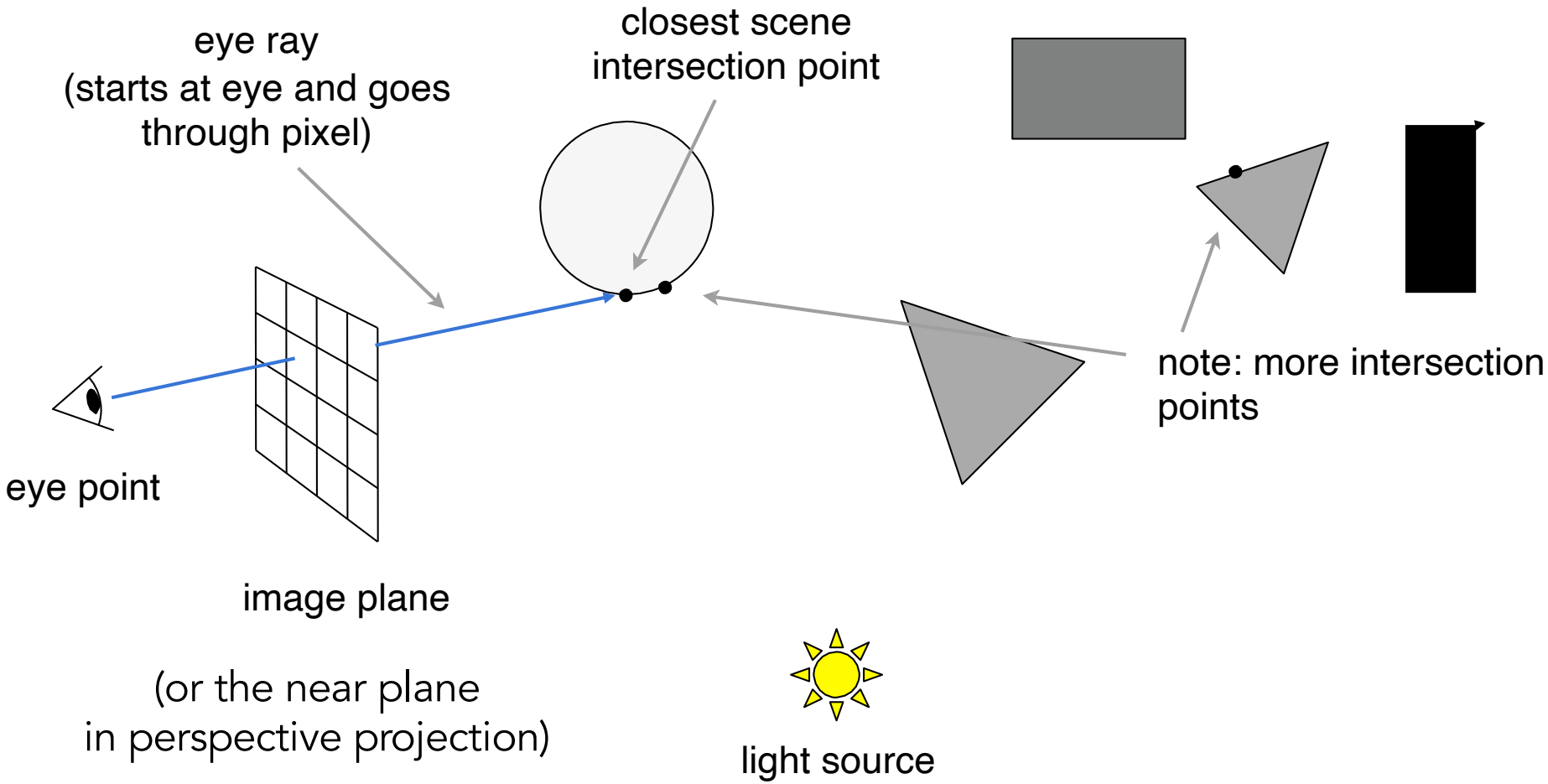
Appel 1968 - Ray casting

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



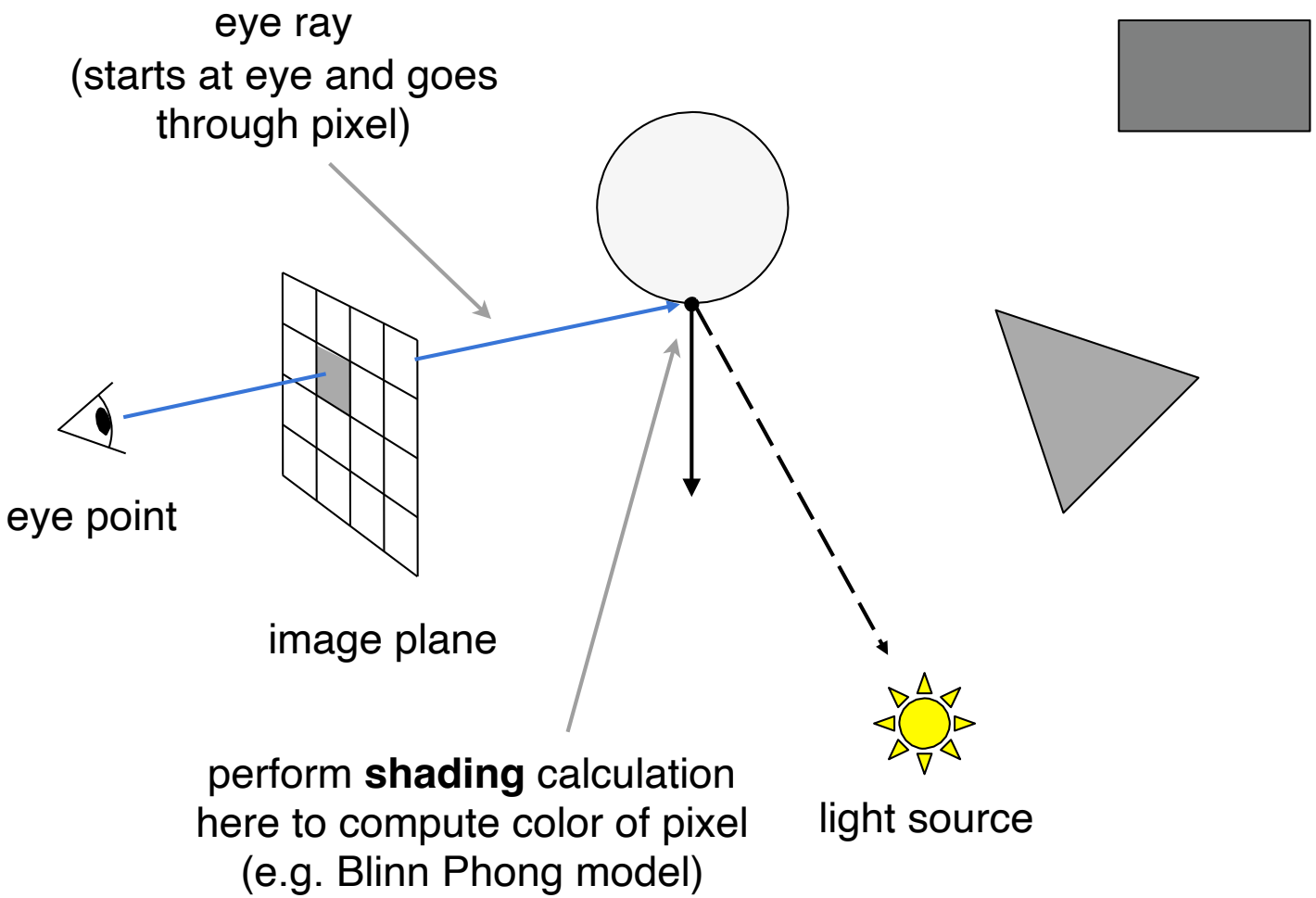
Ray Casting - Generating Eye Rays

Pinhole Camera Model



Ray Casting - Shading Pixels (Local Only)

Pinhole Camera Model

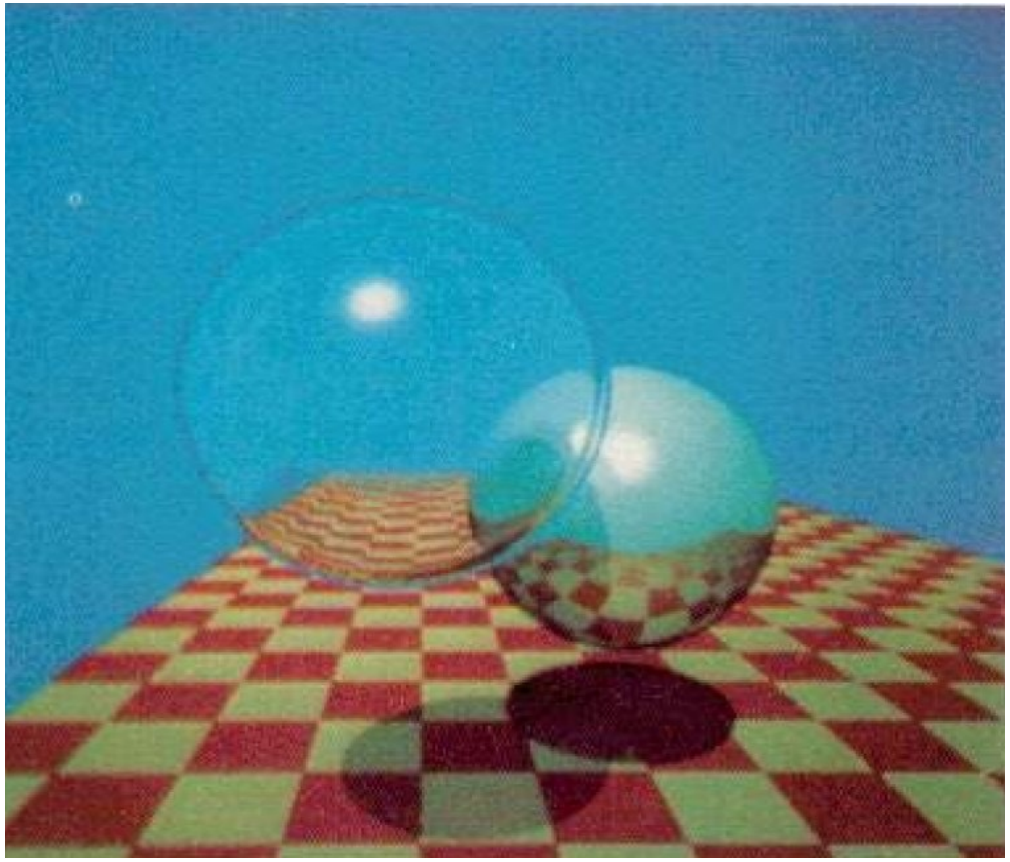


Recursive (Whitted-Style) Ray Tracing

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

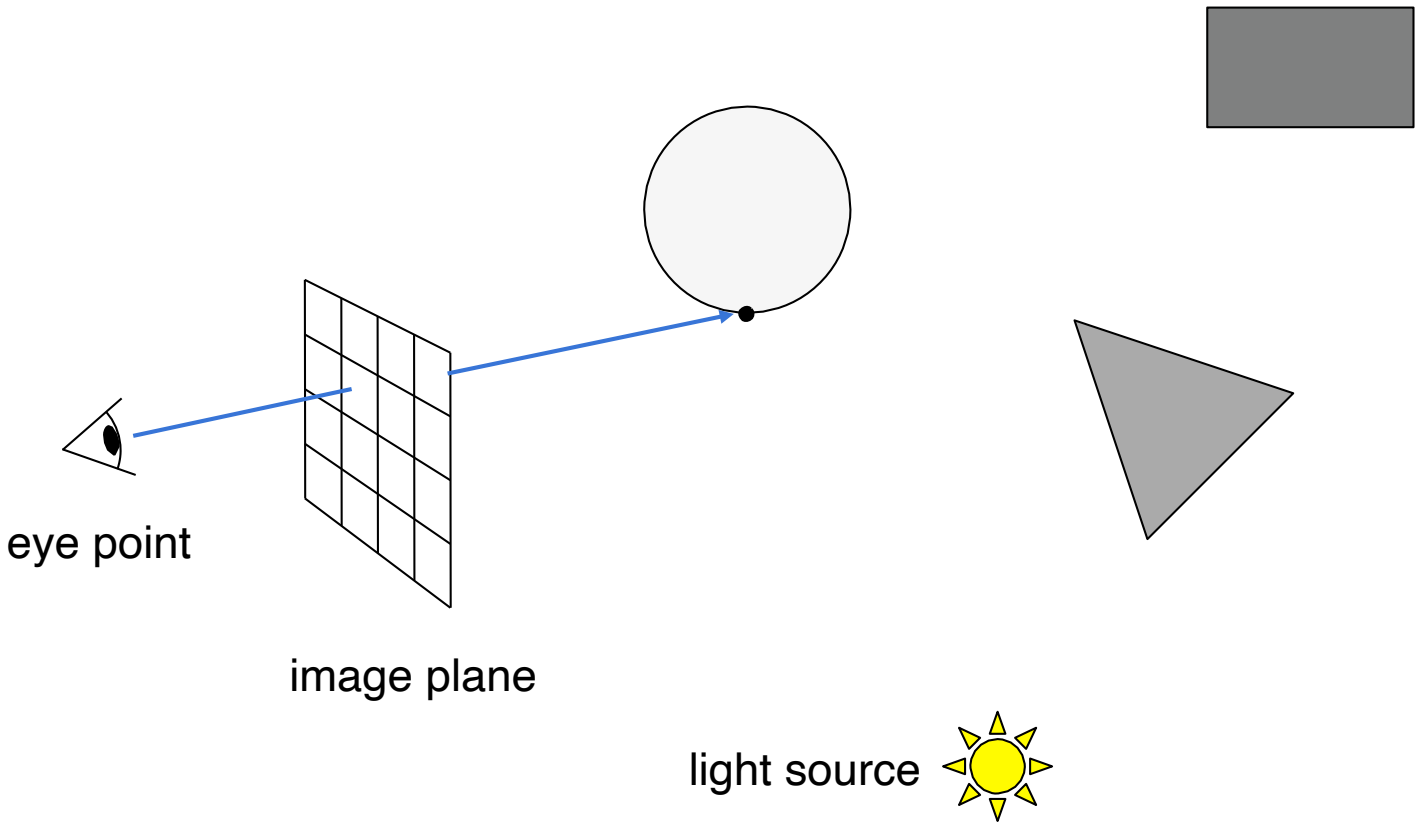
Time:

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

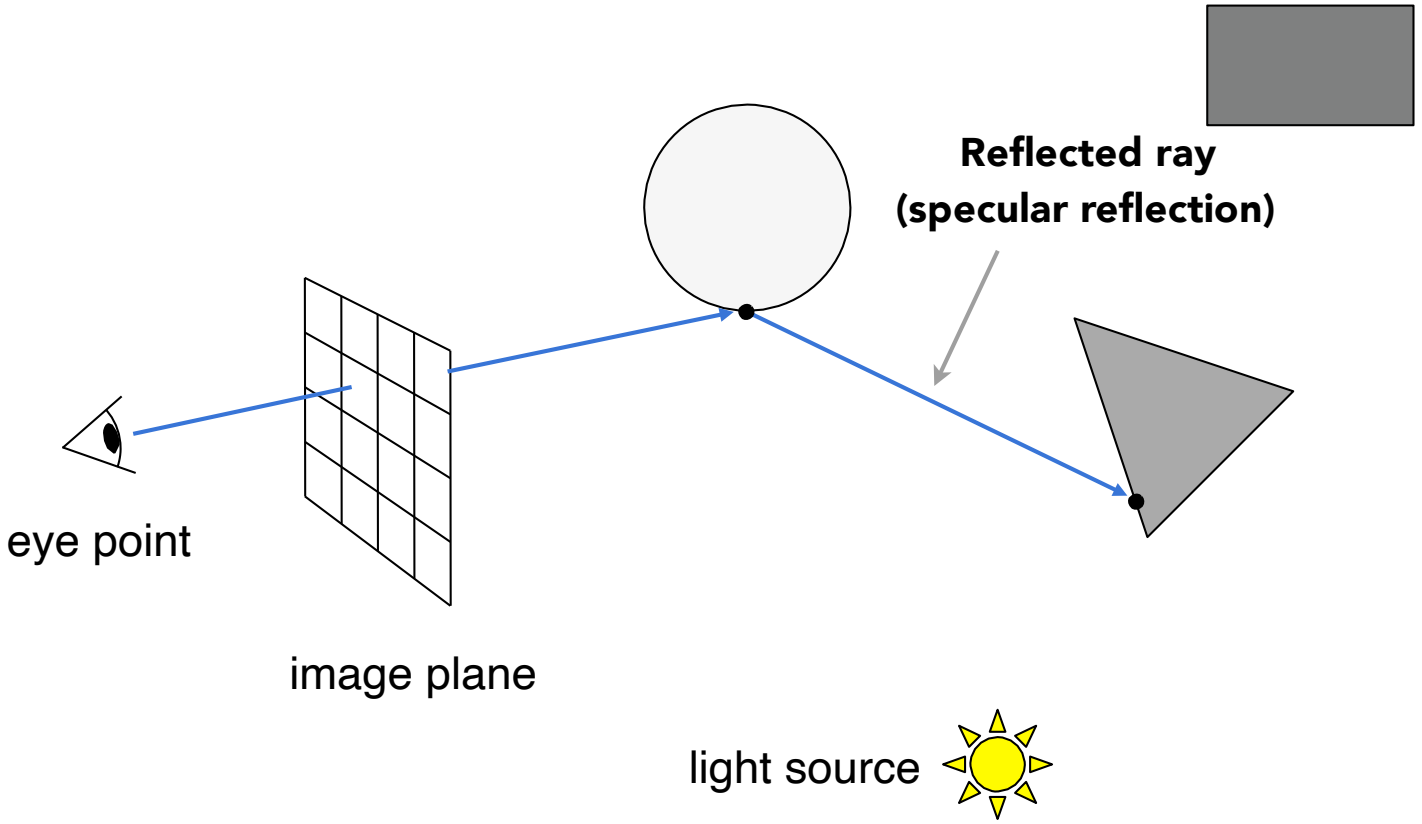


Spheres and Checkerboard, T. Whitted, 1979

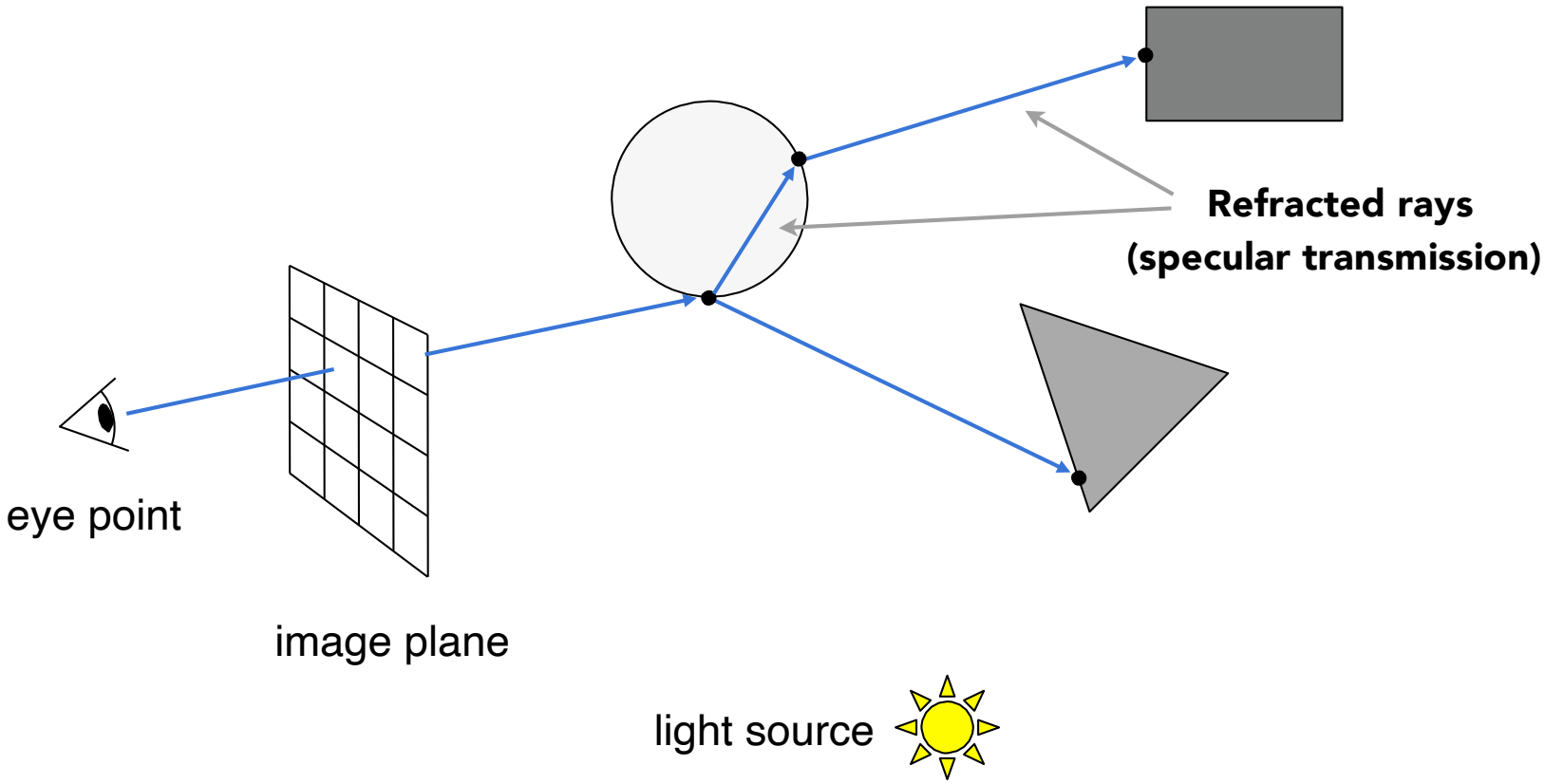
Recursive Ray Tracing



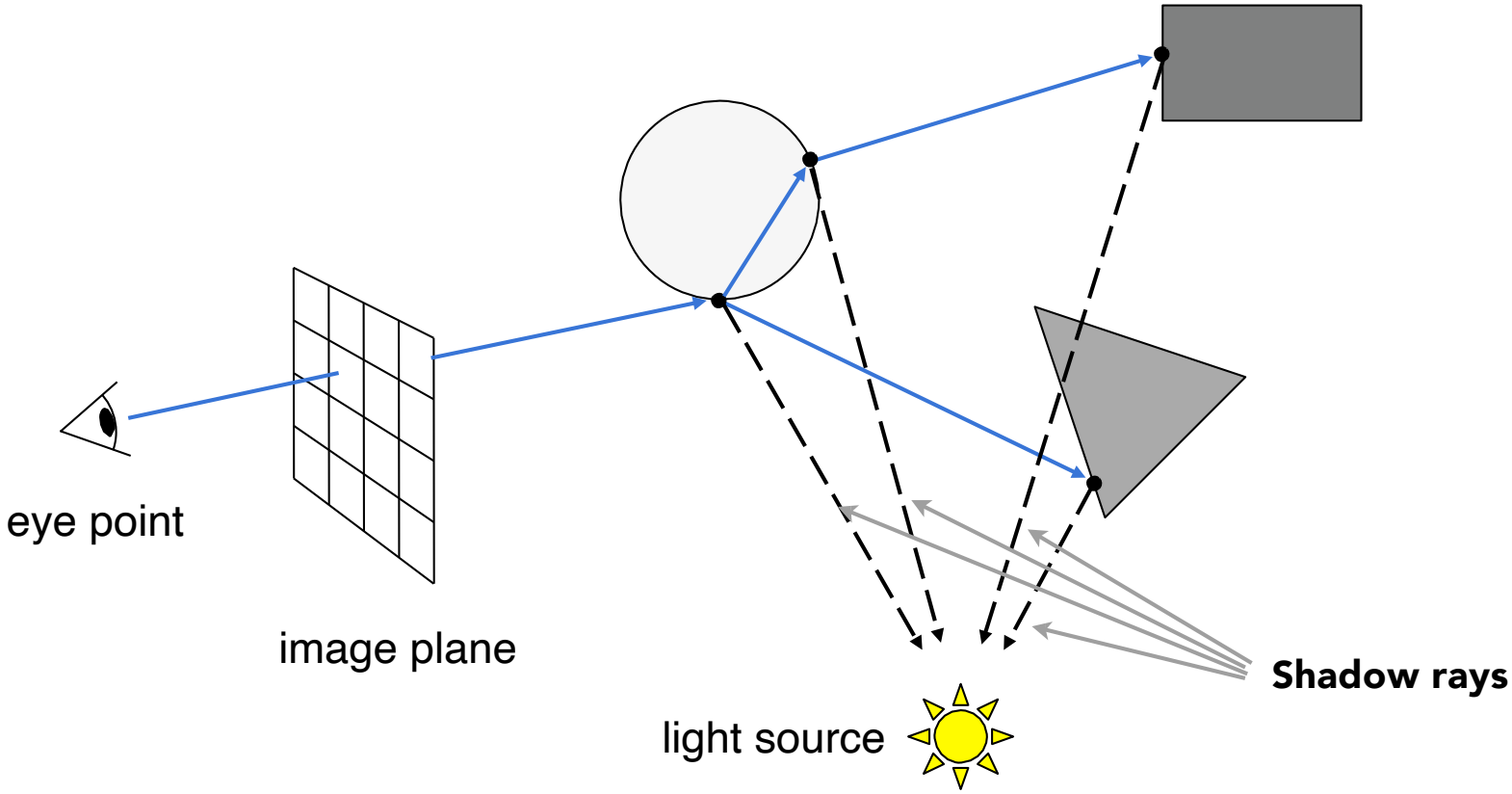
Recursive Ray Tracing



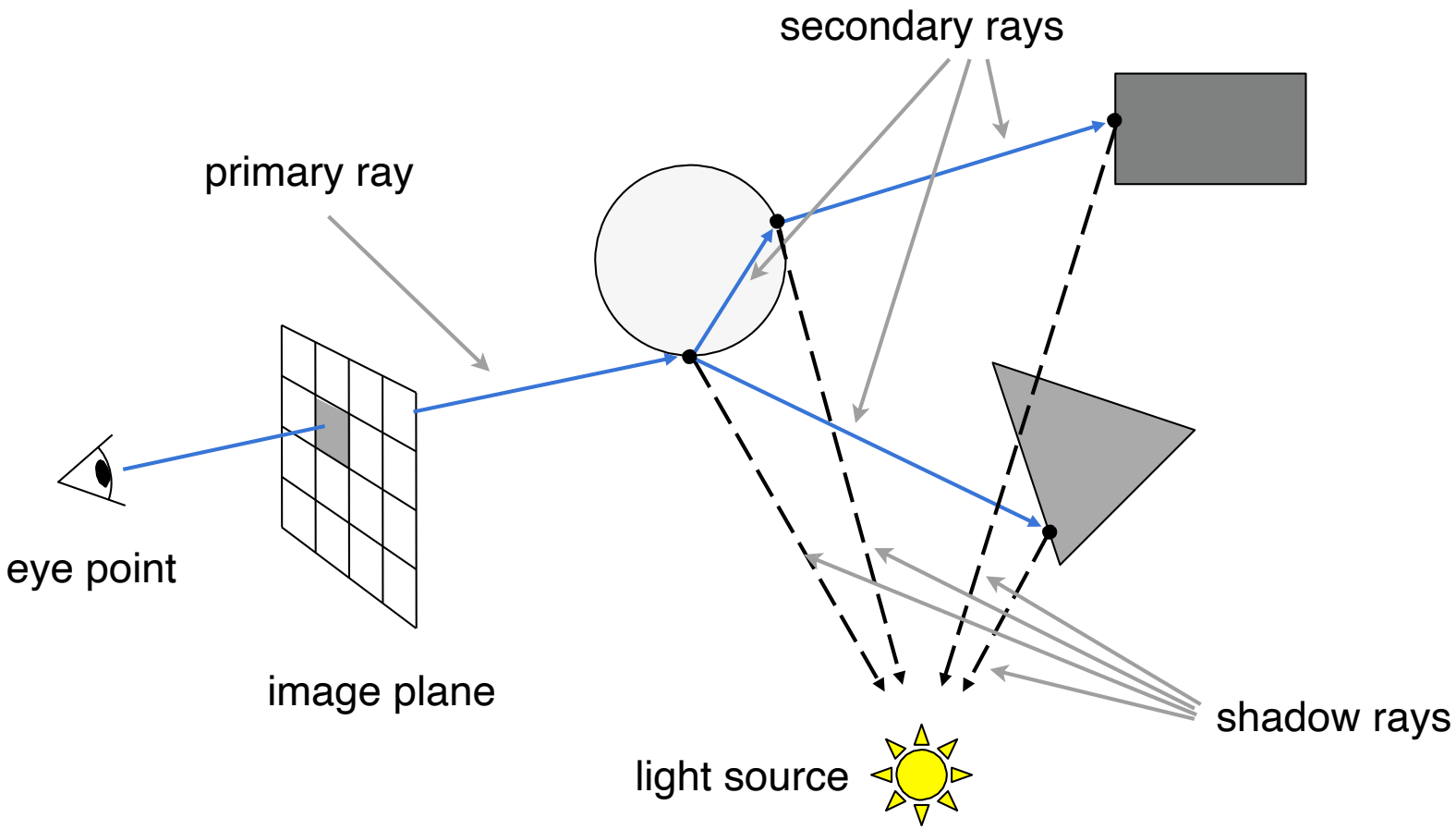
Recursive Ray Tracing



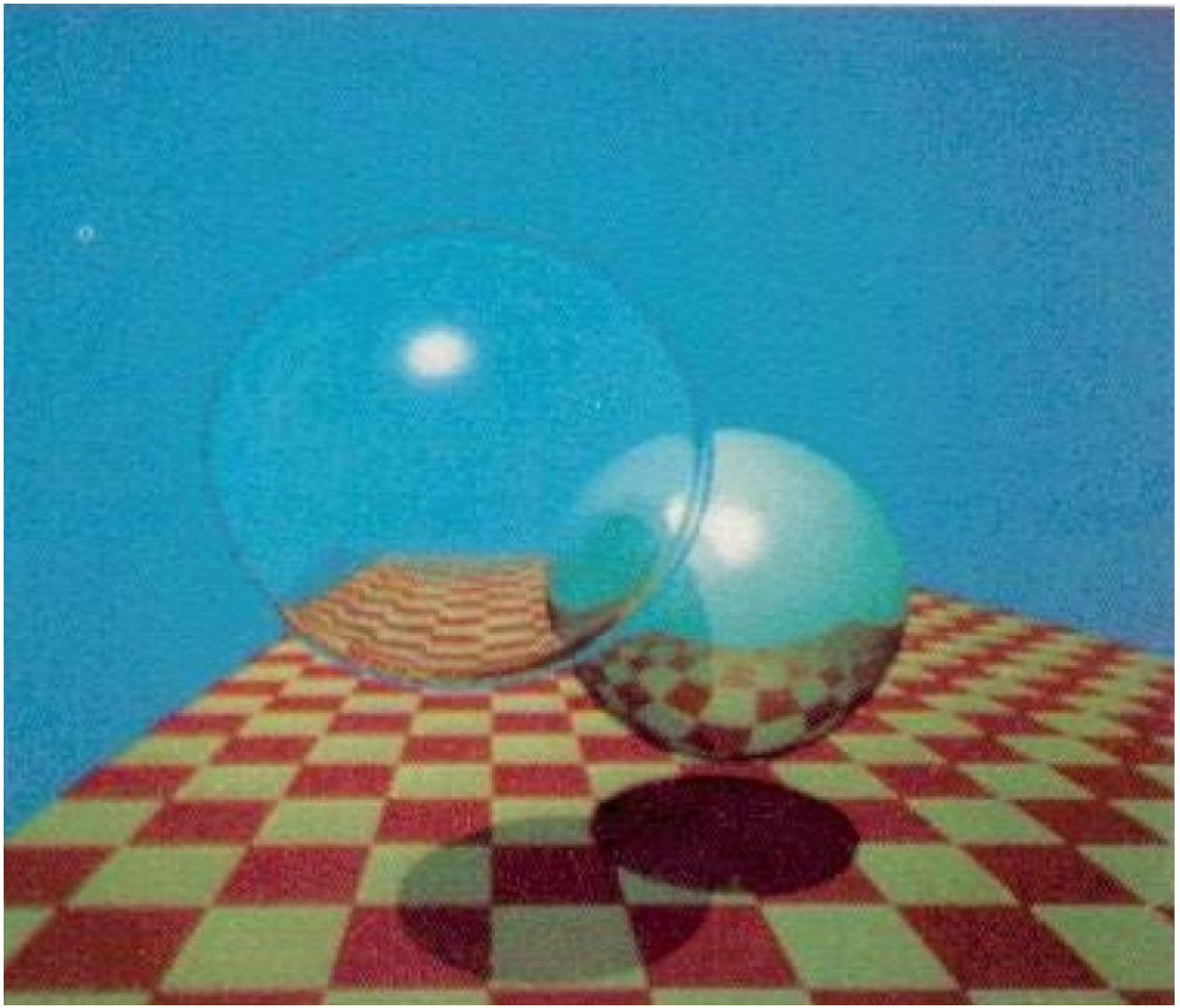
Recursive Ray Tracing



Recursive Ray Tracing



Recursive Ray Tracing

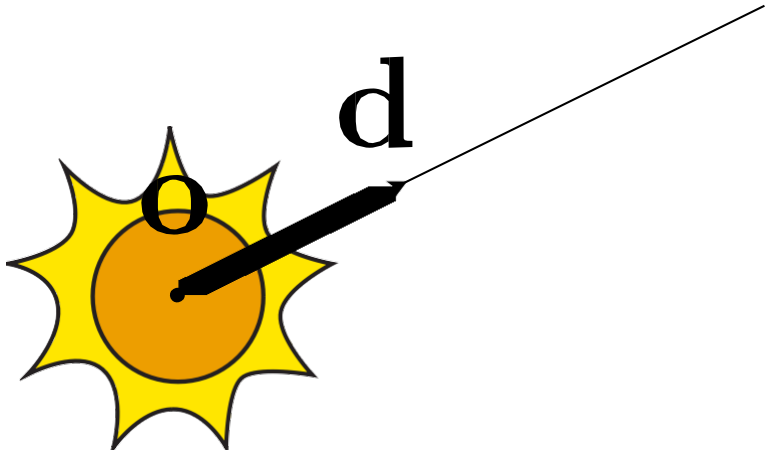


Ray-Surface Intersection

Ray Equation

Ray is defined by its origin and a direction vector

Example:



Ray equation:

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

point along ray — "time" — origin — (normalized) direction

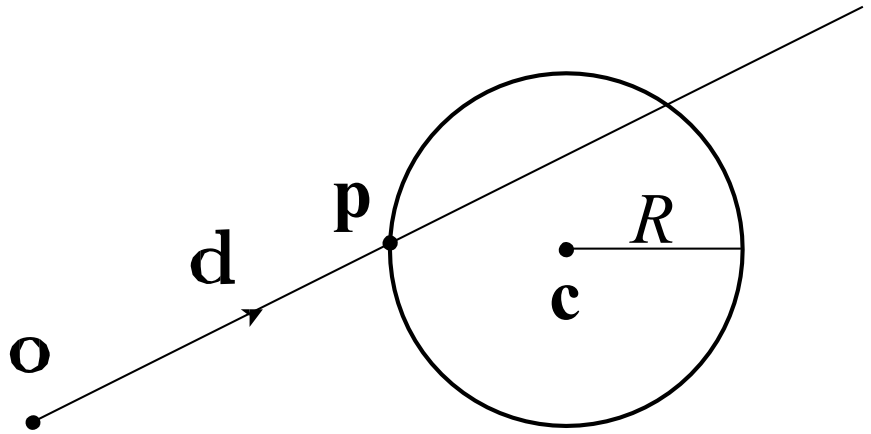
Ray Intersection With Sphere

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

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



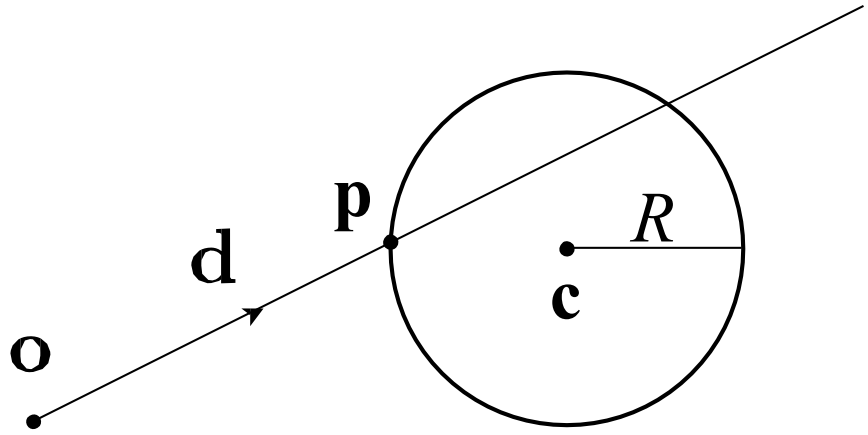
Solve for intersection:

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

Ray Intersection With Sphere

Solve for intersection:

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

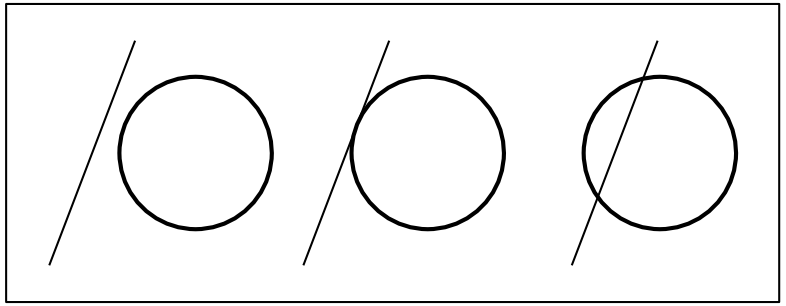


$$a\,t^2 + b\,t + c = 0, \quad \text{where}$$

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

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

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



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

Ray Intersection With Implicit Surface

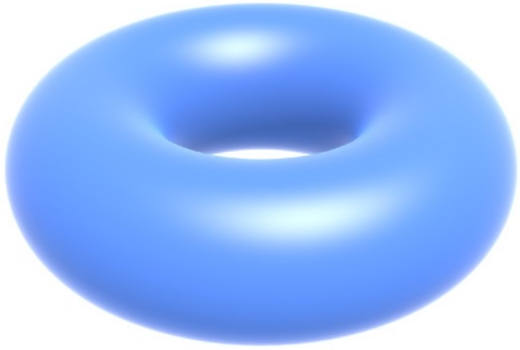
Ray: $\mathbf{r}(t) = \mathbf{o} + t\,\mathbf{d}, \quad 0 \le t < \infty$

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

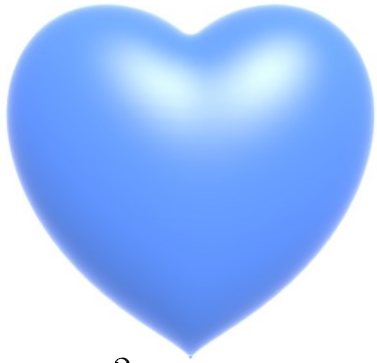
Substitute ray equation: $f(\mathbf{o} + t\,\mathbf{d}) = 0$

Solve for **real, positive** roots

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

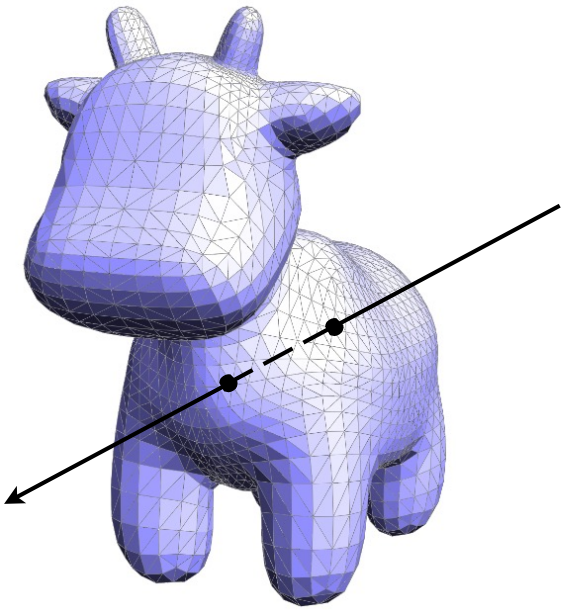


$(R - \sqrt{x^2+y^2})^2 + z^2 = r^2$



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

Ray Intersection With Triangle Mesh



Why?

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

How to compute?

Let's break this down:

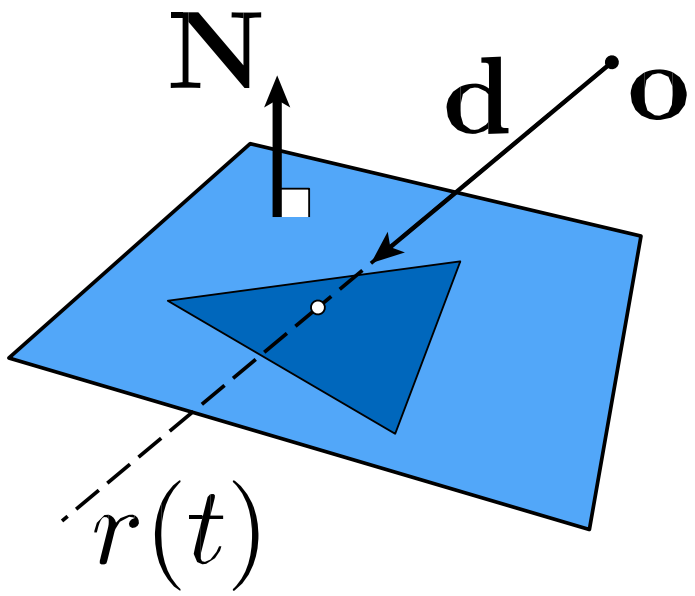
- Simple idea: just intersect ray with each triangle
- Simple, but slow (acceleration?)
- Note: can have 0, 1 intersections (ignoring multiple intersections)

Ray Intersection With Triangle

Triangle is in a plane

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

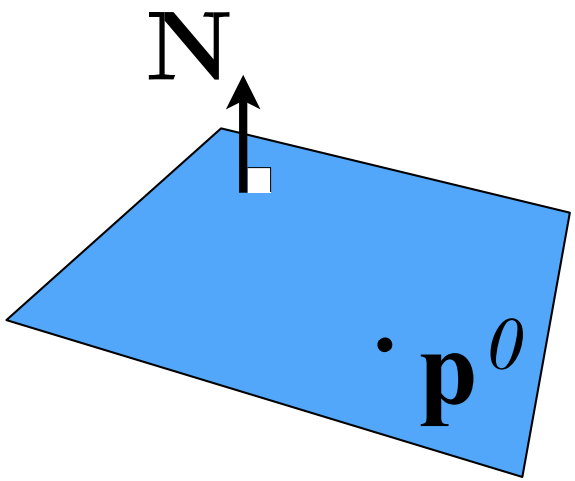
Many ways to optimize…



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):

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

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

all points on plane

one point on plane

normal vector

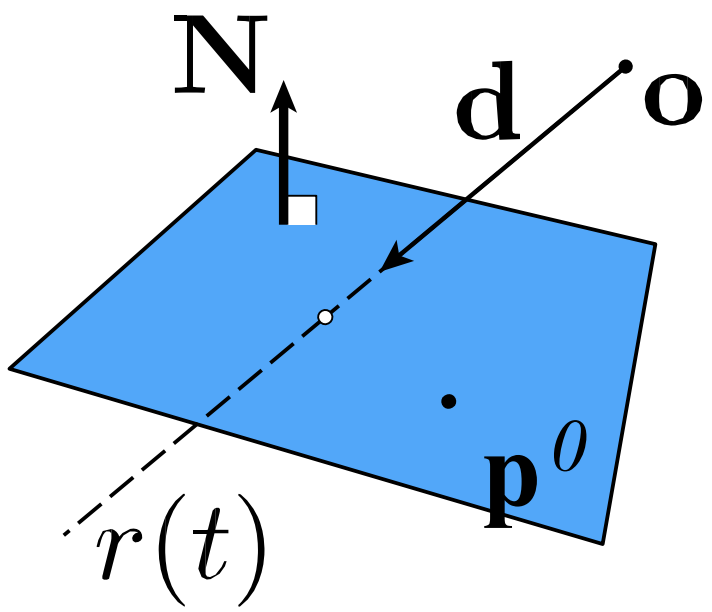
Ray Intersection With Plane

Ray equation:

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

Plane equation:

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



Solve for intersection

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

$$(\mathbf{p} - \mathbf{p}^0) \cdot \mathbf{N} = (\mathbf{o} + t\,\mathbf{d} - \mathbf{p}^0) \cdot \mathbf{N} = 0$$

$$t = \frac{(\mathbf{p}^0 - \mathbf{o}) \cdot \mathbf{N}}{\mathbf{d} \cdot \mathbf{N}}$$

Check: $0 \le t < \infty$

Möller Trumbore Algorithm

A faster approach, giving barycentric coordinate directly

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

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

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

Where:

$$\vec{\mathbf{E}}_1 = \vec{\mathbf{P}}_1 - \vec{\mathbf{P}}_0$$

$$\vec{\mathbf{E}}_2 = \vec{\mathbf{P}}_2 - \vec{\mathbf{P}}_0$$

$$\vec{\mathbf{S}} = \vec{\mathbf{O}} - \vec{\mathbf{P}}_0$$

$$\vec{\mathbf{S}}_1 = \vec{\mathbf{D}} \times \vec{\mathbf{E}}_2$$

$$\vec{\mathbf{S}}_2 = \vec{\mathbf{S}} \times \vec{\mathbf{E}}_1$$

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

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

Thank you!

(And thank Prof. Ravi Ramamoorthi (UCSD), Prof. Ren Ng (UC Berkeley), and Prof. Lingqi Yan (UCSB) for many of the slides!)