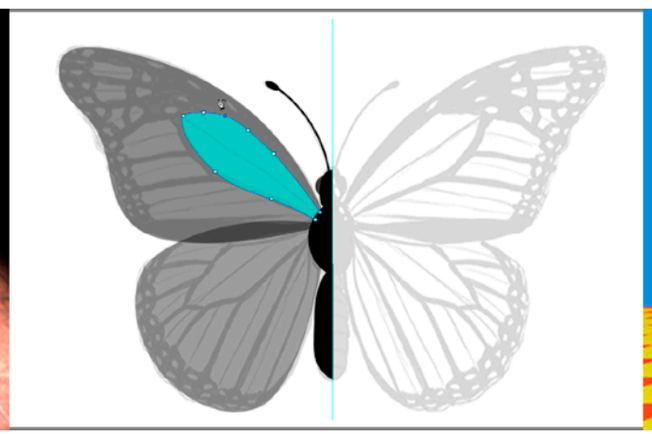
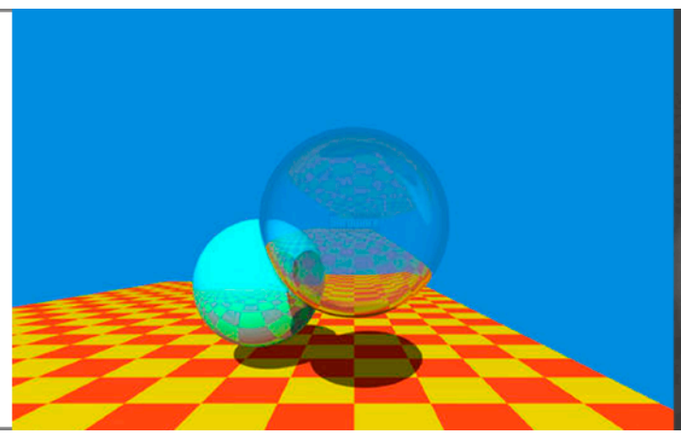
Introduction to Computer Graphics

GAMES101, Lingqi Yan, UC Santa Barbara

Lecture 13:
Ray Tracing 1
(Whitted-Style Ray Tracing)

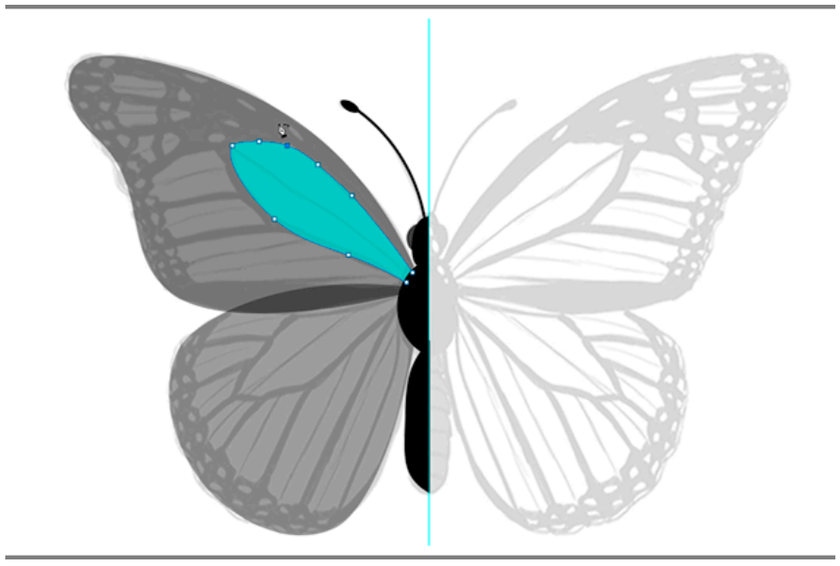

http://www.cs.ucsb.edu/~lingqi/teaching/games101.html

Announcements

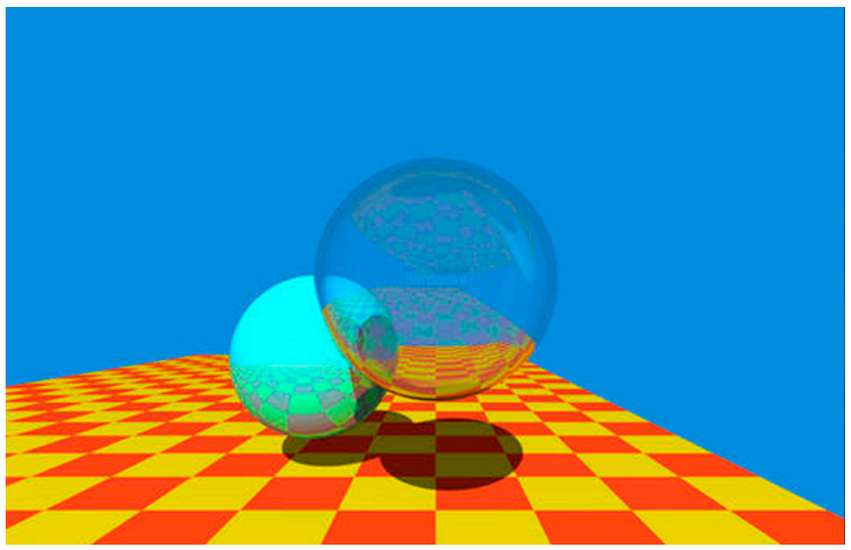
- Homework 4 — 252 submissions so far
- Homework 1 — 80 resubmissions so far
- For universities that use this course
  - Feel free to do so, and if you need scores
  - On your side: give me a name (TA or professor)
    On my side: give you access
    You determine whether to account for resubmissions

Course Roadmap

Rasterization



Geometry



Ray tracing

Animation / simulation

Why Ray Tracing?

Blinn-Phong 经验模型的缺陷：即对于全局效果的表示不足，比如1.阴影的表示 2. Gloosy反射（类似镜面反射的物体表面反射） 3. 间接光照，也就是空间中不只有一个单一光源，可能还存在着其他光源以及光在空间中的多次反射。

- 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

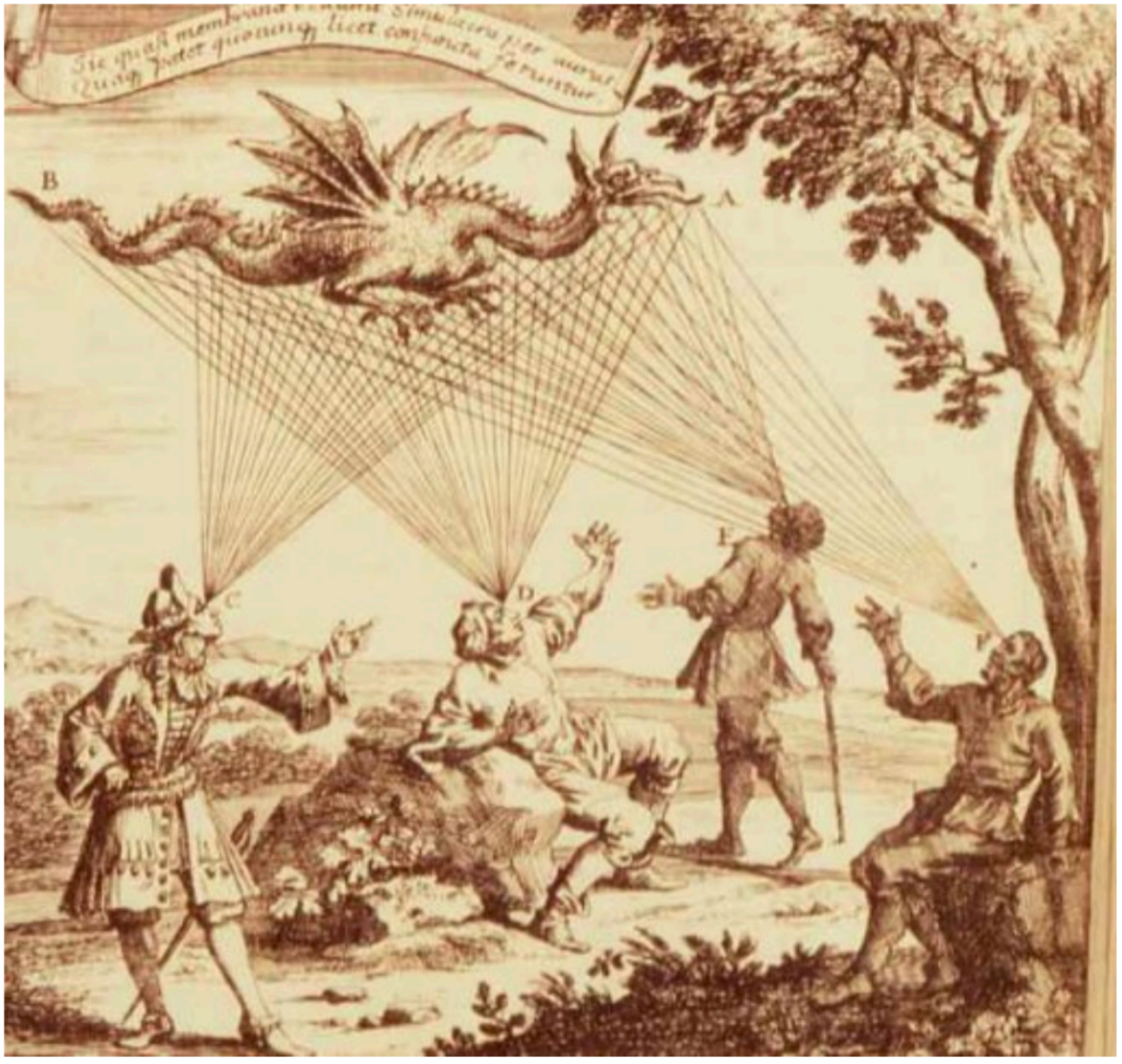
1. Light travels in straight lines (though this is wrong)
   沿直线传播
2. Light rays do not “collide” with each other if they cross (though this is still wrong) 光线互不碰撞
3. Light rays travel from the light sources to the eye (but the physics is invariant under path reversal - reciprocity).
   光线一定是从光源出发最后到达接收点（人眼），并且光具有可逆性，可以沿光路原路返回

“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*

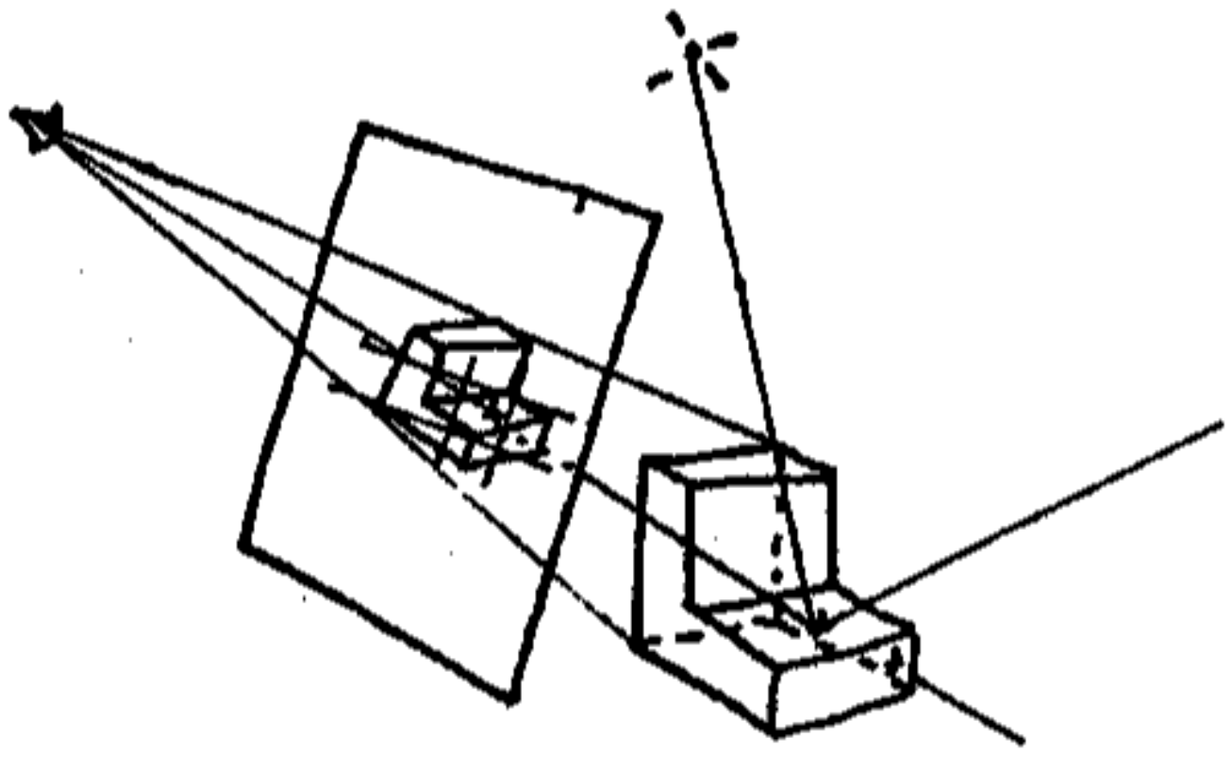
*http://www.ncbi.nlm.nih.gov/pubmed/12094435?dopt=Abstract

Slide courtesy of Prof. Alexei Efros, UC Berkeley

Ray Casting

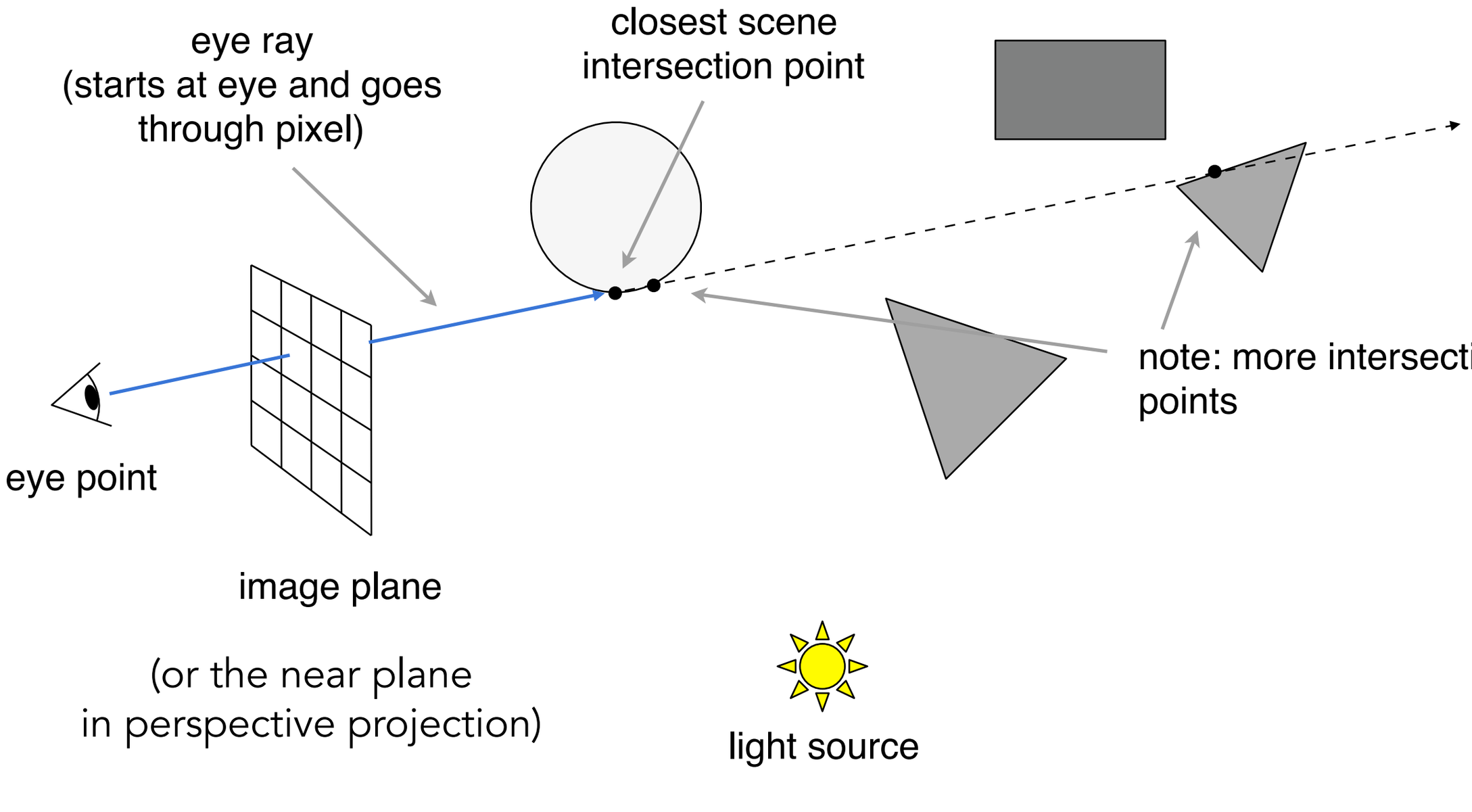
Appel 1968 - Ray casting

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

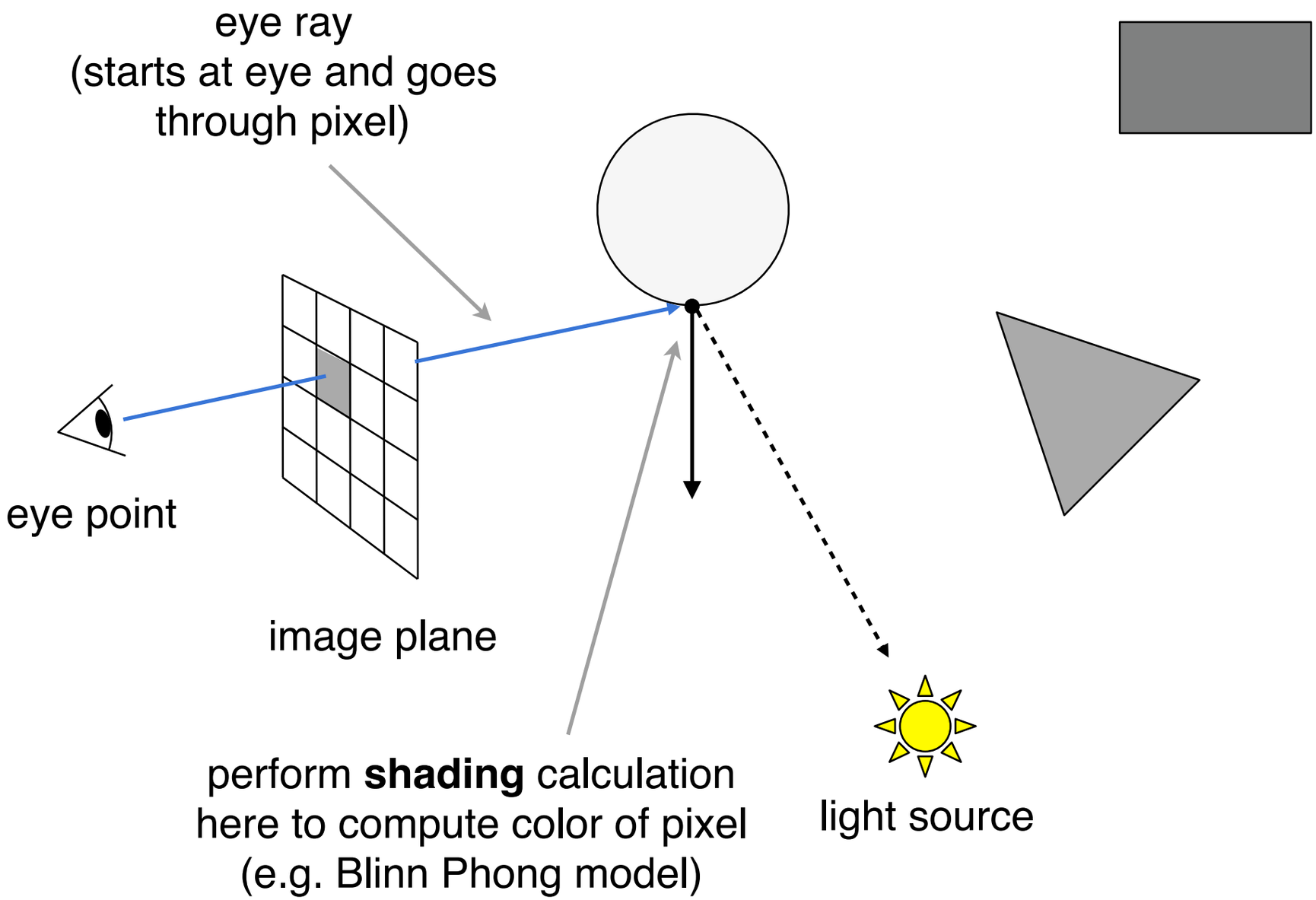
Pinhole Camera Model



Ray Casting - Shading Pixels (Local Only)

Pinhole Camera Model

从人眼（视作点光源1）发出eye ray ，即发出一条光线（经过2维像素点，也视作这个像素发出的光线），并且这条光线只记录在路径上遇见的第一个物体点（这也完美解决了z-buffer问题）然后将交点与光源连线，若此连线路径无遮挡则这个点被照亮，若有遮挡则此点在阴影中，自此就可以对交点着色并写入像素点。



Recursive (Whitted-Style) Ray Tracing

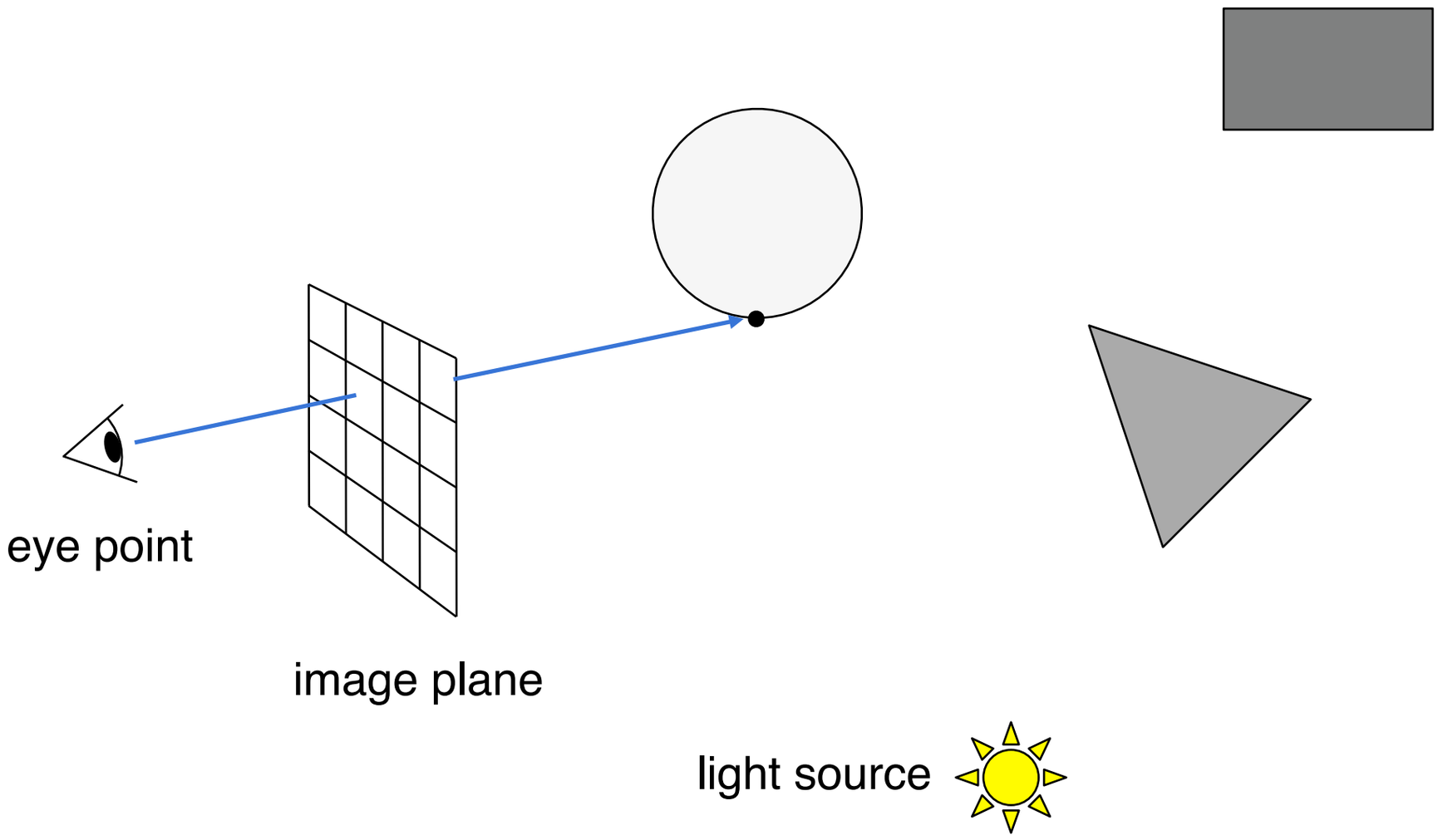
"An improved Illumination model for shaded display"
T. Whitted, CACM 1980

Time:

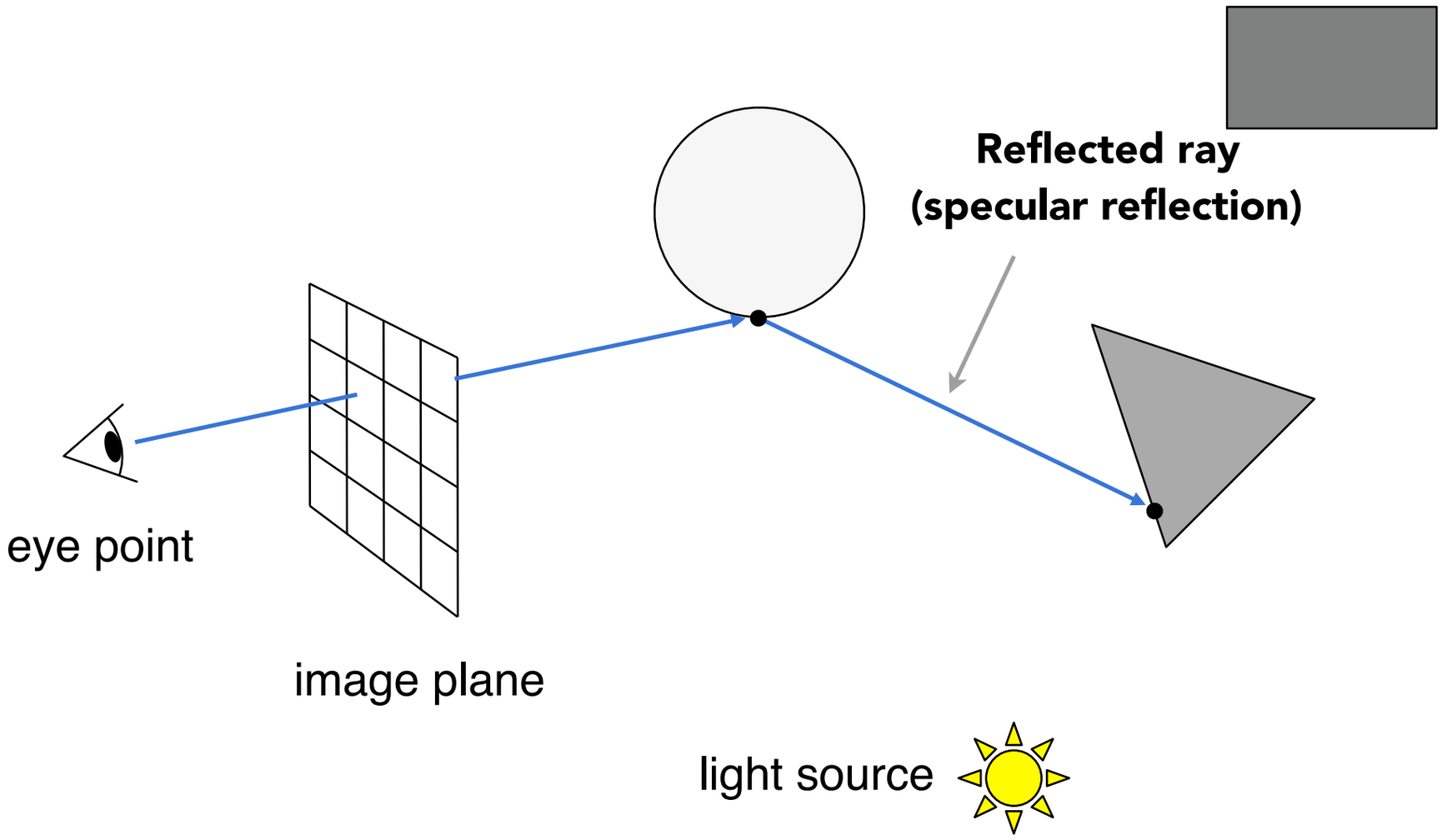
- VAX 11/780 (1979) 74m
- PC (2006) 6s
- GPU (2012) 1/30s

Spheres and Checkerboard, T. Whitted, 1979

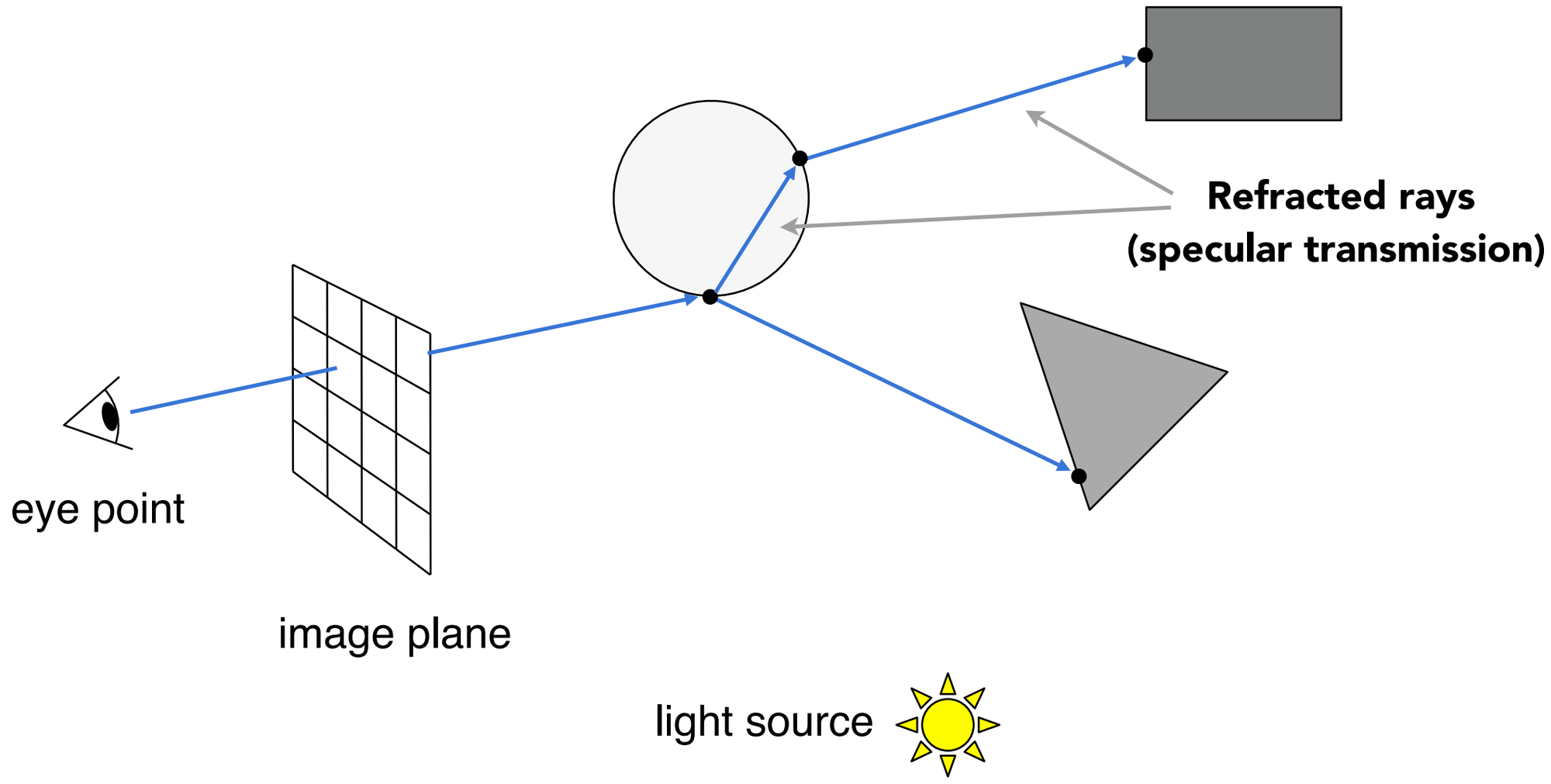
Recursive Ray Tracing



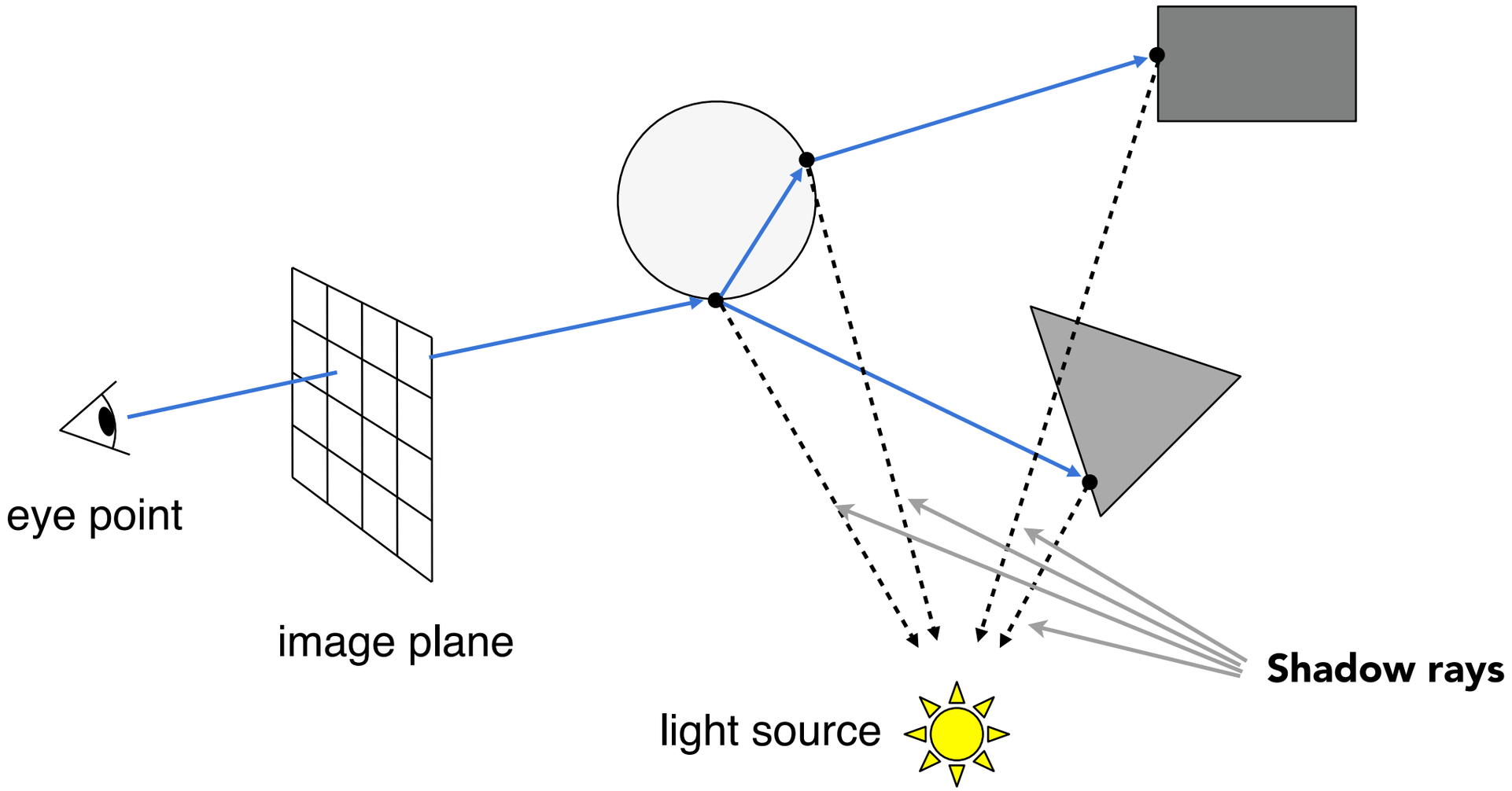
Recursive Ray Tracing



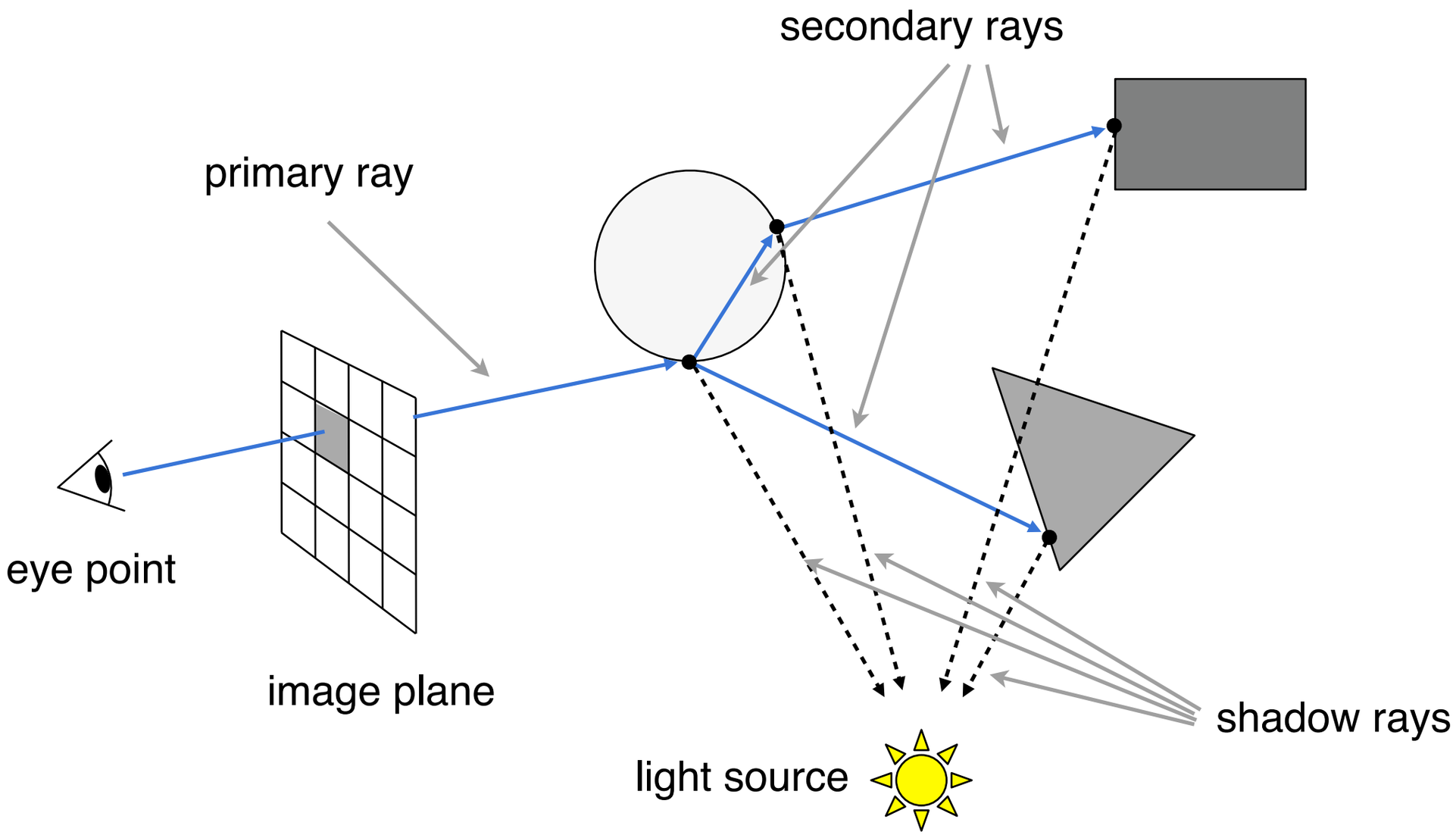
Recursive Ray Tracing



Recursive Ray Tracing



Recursive Ray Tracing



Recursive(Whitted-Style) Ray Tracing假定：在任意一个接收到光线的点可以视作光源继续发射折射光线。这些由同一根光线（同一个像素点发射出来的光线）与物体产生的交点以及折射所产生的交点都与光源做连线并着色返回给像素点。

Recursive Ray Tracing

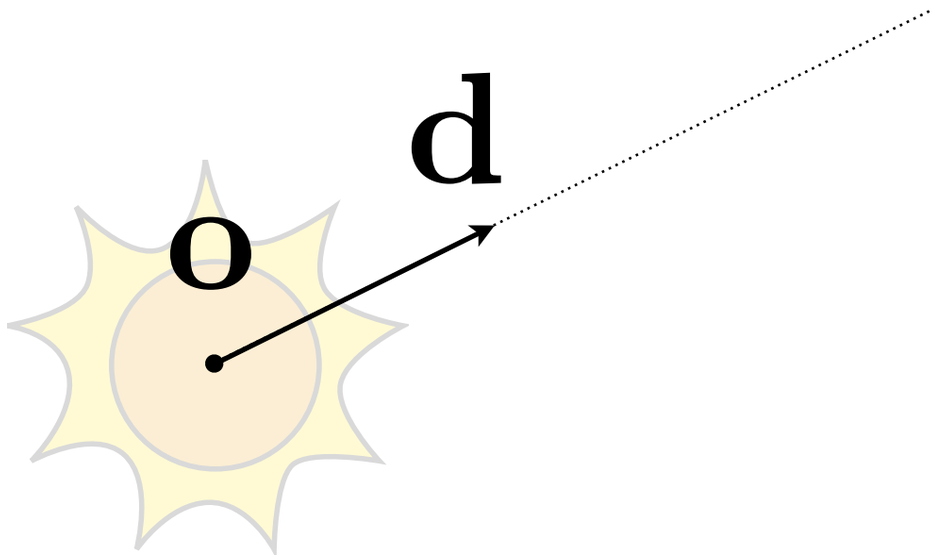
Ray-Surface Intersection

现在目标就明确了，就是求交点。求交点第一步：先在数学上定义光线

Ray Equation

Ray is defined by its origin and a direction vector

Example:



Ray equation: 射线的定义

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

point along ray　“time”　origin　(normalized) direction

Ray Intersection With Sphere

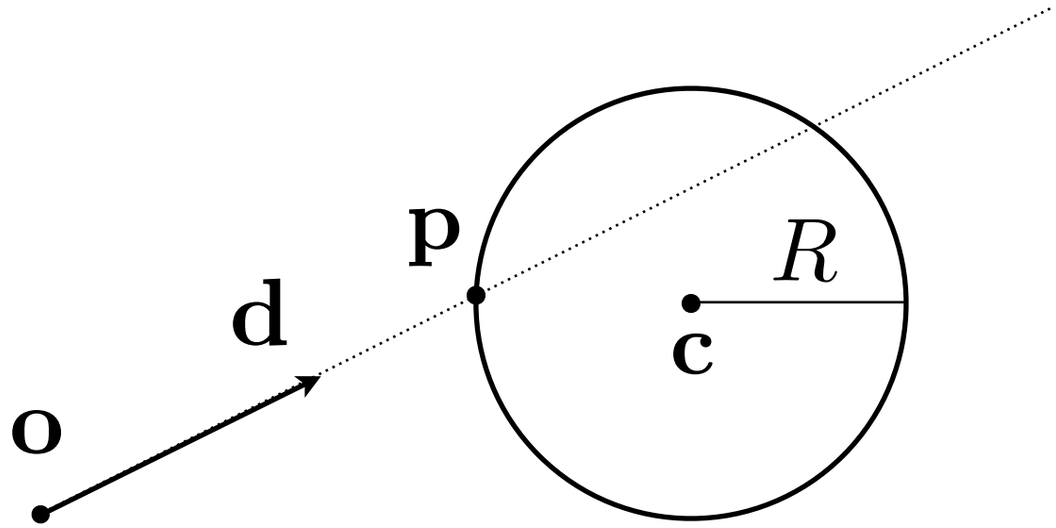
隐式表面：以球面为例

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

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



假设有一道光线打在物体c上，交点为p

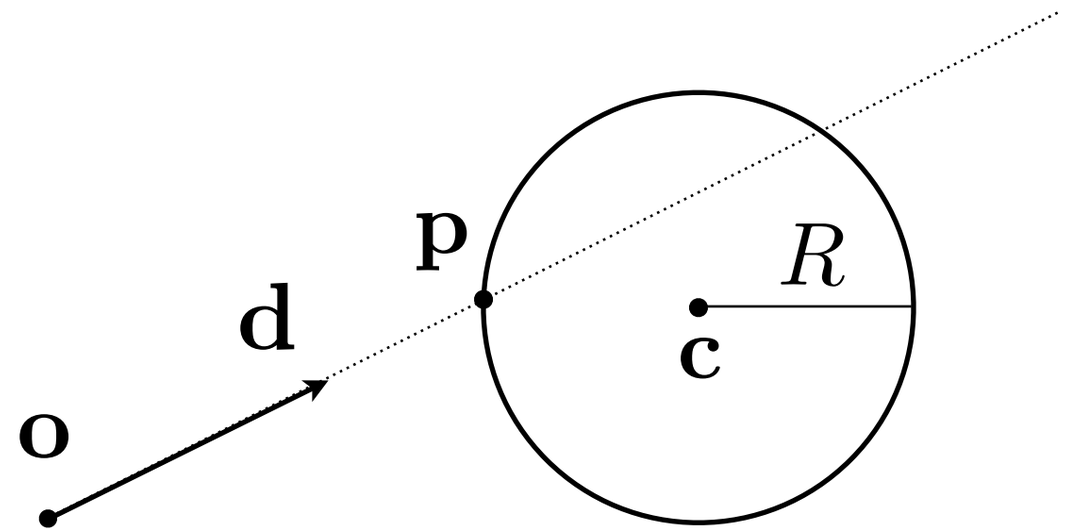
Solve for intersection:

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

Ray Intersection With Sphere

Solve for intersection:

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



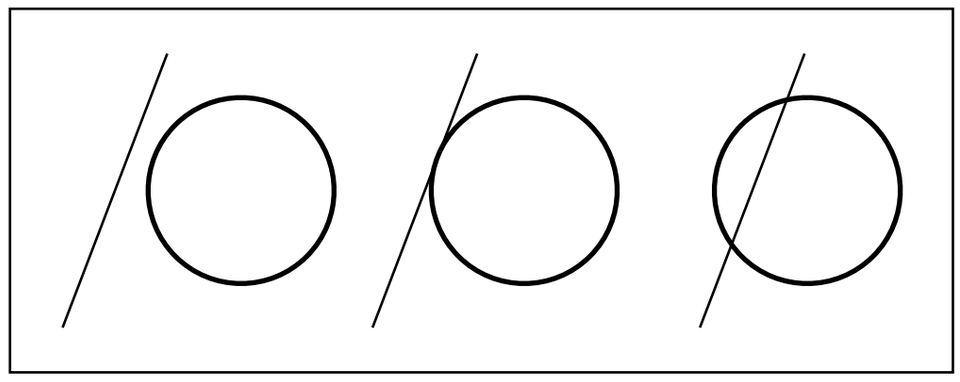
$$a\,t^2 + b\,t + c = 0, \text{ where}$$

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

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

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

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



Ray Intersection With Implicit Surface

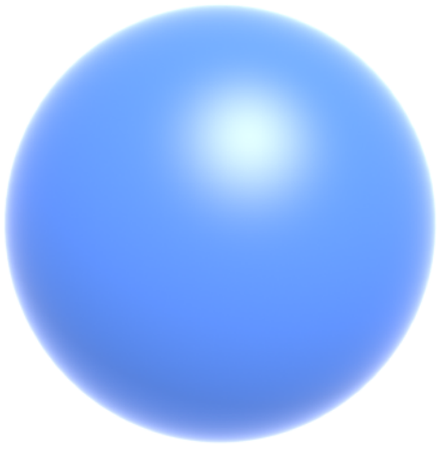
一般隐式图像方程与光线交点求解方法：对于t展开即求解二次函数，且t>0，同理对于任意隐式表面都可以求解t

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

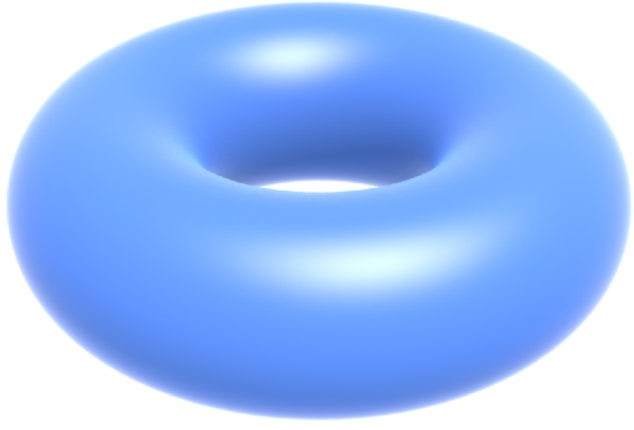
General implicit surface: $\mathbf{p} : f(\mathbf{p}) = 0$

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

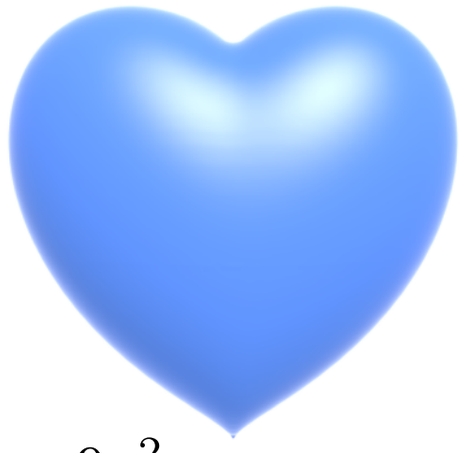
Solve for **real, positive** roots



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



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



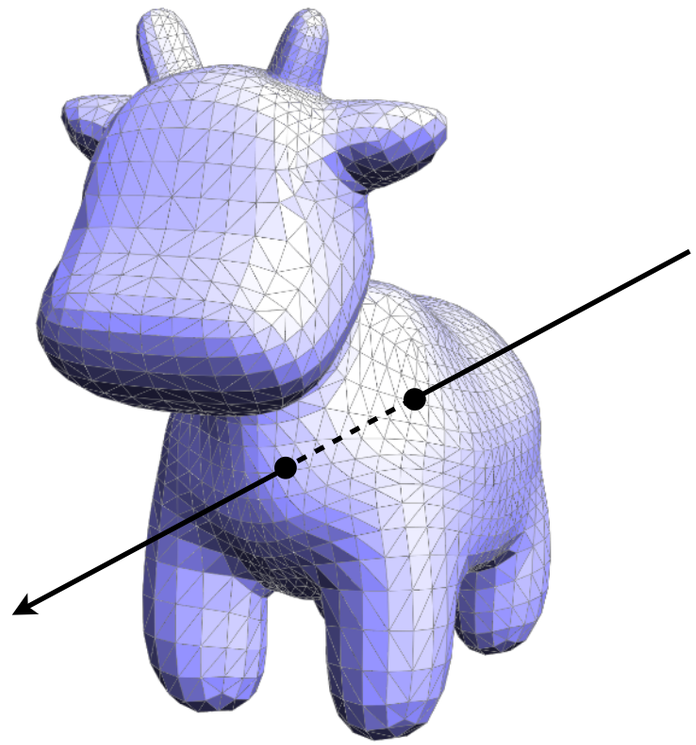
$$(x^2 + \frac{9y^2}{4} + z^2 - 1)^3 = x^2z^3 + \frac{9y^2z^3}{80}$$

Ray Intersection With Triangle Mesh

对于显式表面：一个简单的想法是一个个三角形都和光线判断是否有交点，有交点的取t最小的。先不去考虑计算量的优化下，我们要求一个三角形是否在一个面内可以先求这个光线和平面的哪个点相交，再判断这个点在不在三角形内部。

Why?

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



How to compute?

Let’s break this down:

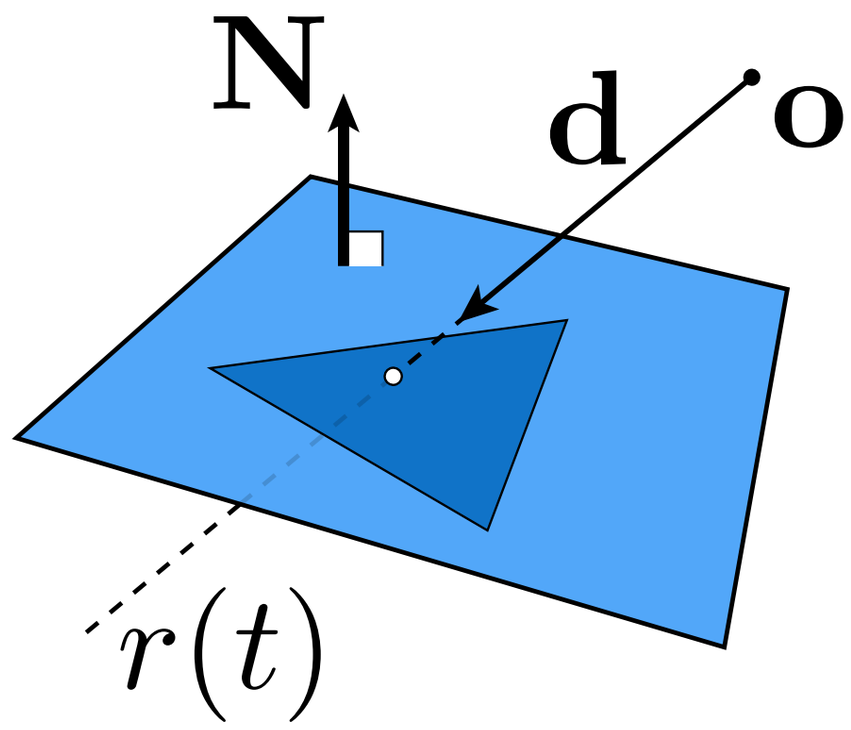
- Simple idea: just intersect ray with each triangle
- Simple, but slow (acceleration?)
- Note: can have 0, 1 intersections (ignoring multiple intersections)

Ray Intersection With Triangle

Triangle is in a plane

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

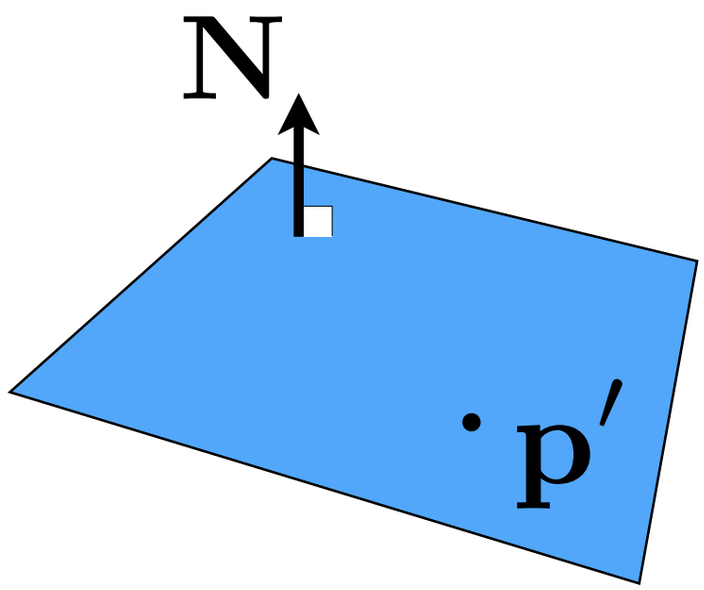
Many ways to optimize…



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):

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

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

all points on plane

one point on plane

normal vector

Ray Intersection With Plane

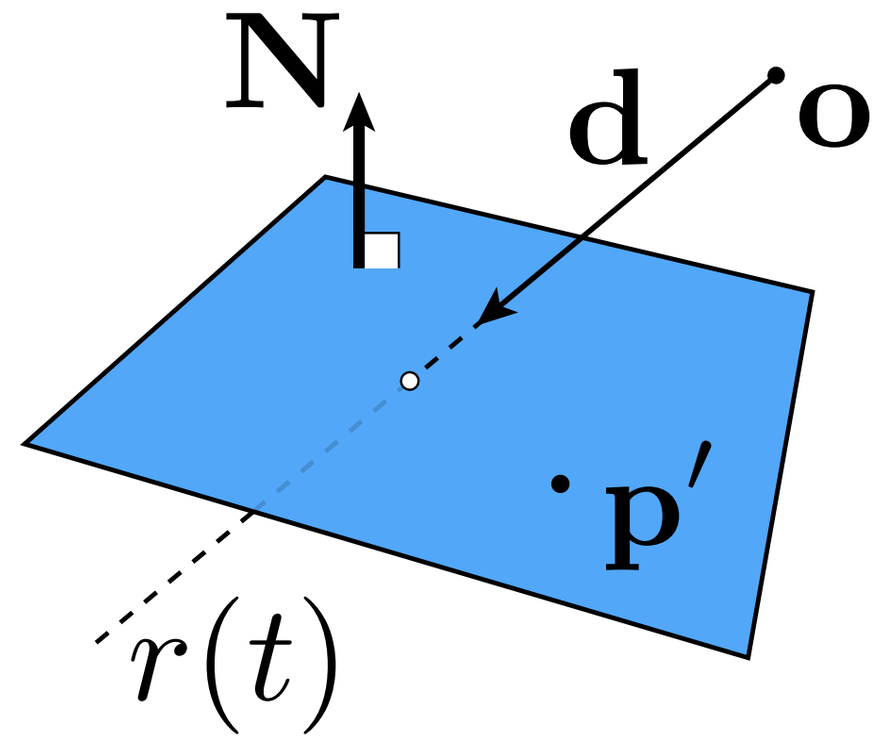
如图直接用一条法向量加一个点就可以定义一个平面（法向量定义平面朝向，平面上任意一点定义平面位置），则这个平面上任意一点都满足公司 (p-p')N=0 (与 pp'向量法向量垂直)，带入 p = o+td 可得

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

利用重心坐标表示法直接判断直线与三角形的交点是否在三角形内

A faster approach, giving barycentric coordinate directly

Derivation in the discussion section!

这里求解 t 、b1、b2、（1-b1-b2），三个系数若都在 0-1 区间则在三角形内部。

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

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

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 x # traingles (x #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

San Miguel Scene, 10.7M triangles

Ray Tracing – Performance Challenges

Plant Ecosystem, 20M triangles

Bounding Volumes

Bounding Volumes

包围盒（Bounding Box）的概念：任意复杂的物体，我们都可以用一个简单形状将其包围起来，光线若连盒子都碰不到就肯定碰不到里面的物体，这是首要优化思想

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**

对于一个 AABB 包围盒我们将其看作三组对面所围成的交集；则现在问题就变为了判断光线如何与一个包围盒相交

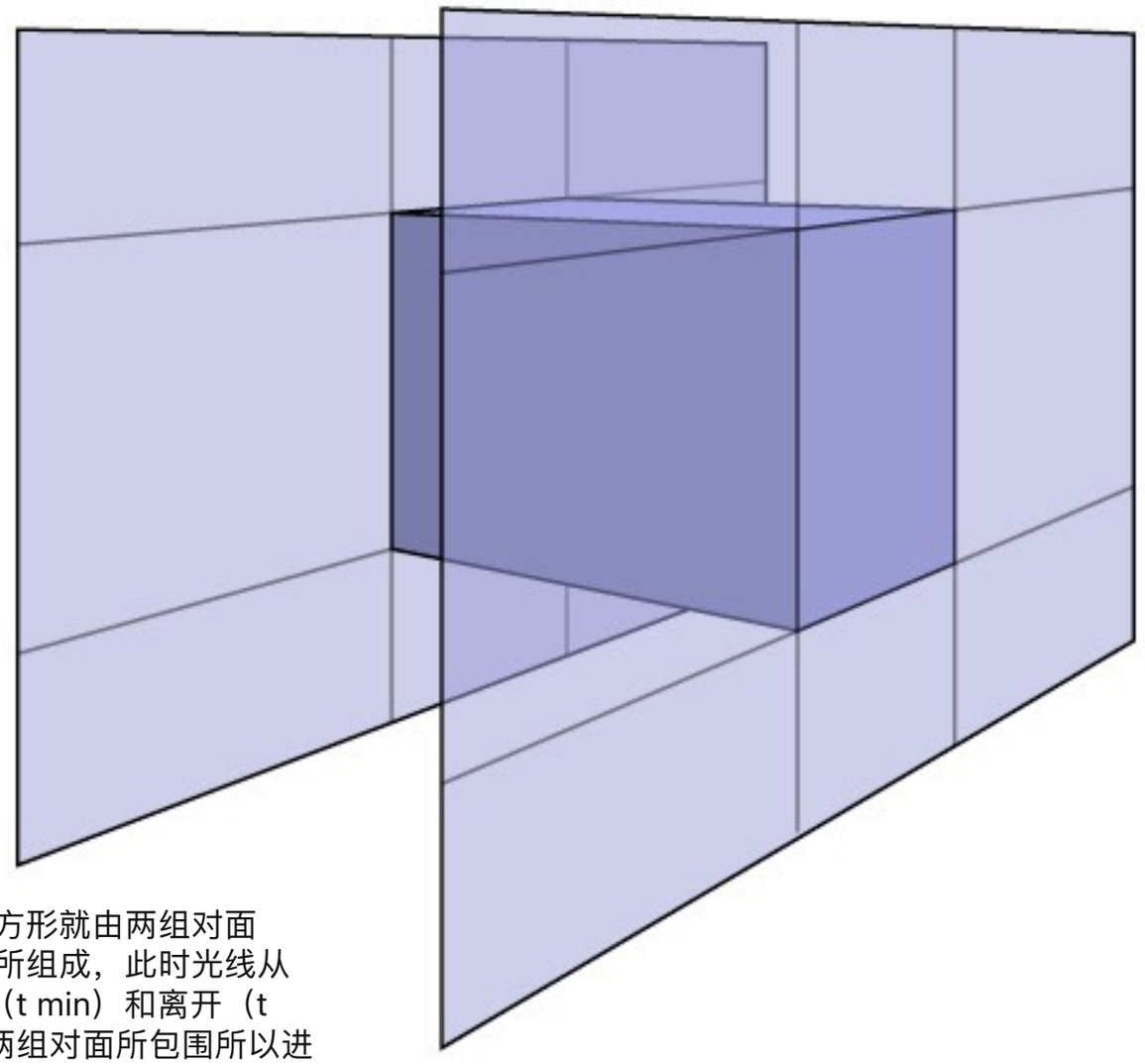
Specifically:

We often use an

Axis-Aligned Bounding Box (AABB)

(轴对齐包围盒)

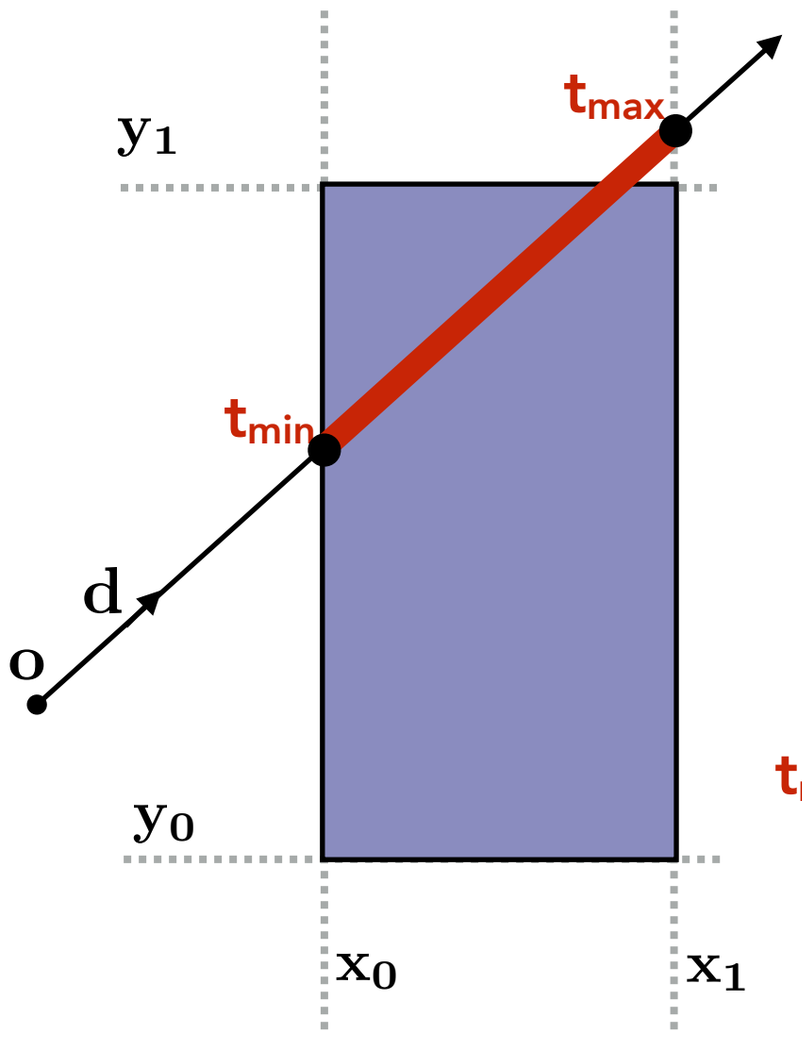
i.e. any side of the BB is along either x, y, or z axis



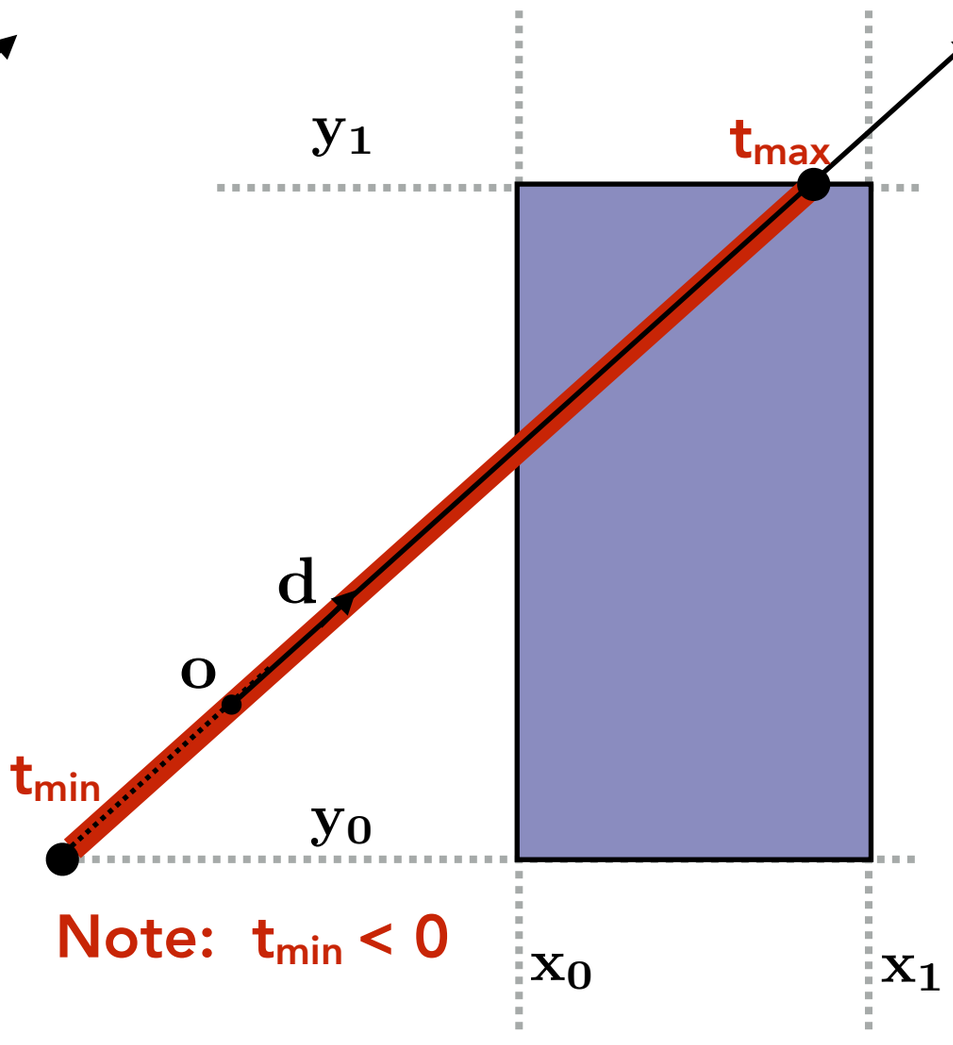
首先我们先看二维的情况，二维情况下一个长方形就由两组对面（其实这里是直线段，我更愿意称其为对线）所组成，此时光线从 o 点出发，我们可以计算出光线什么时候进入（t min）和离开（t max）纵向和横向的对面，由于 o 点被横向的两组对面所包围所以进入对面的时间（t min）是负的。从图中可看到，对于这两组 t min-t max 我们取一个交集就可以得到光线在包围盒中的所停留的时间。扩展到三维也是同理。

Ray Intersection with Axis-Aligned Box

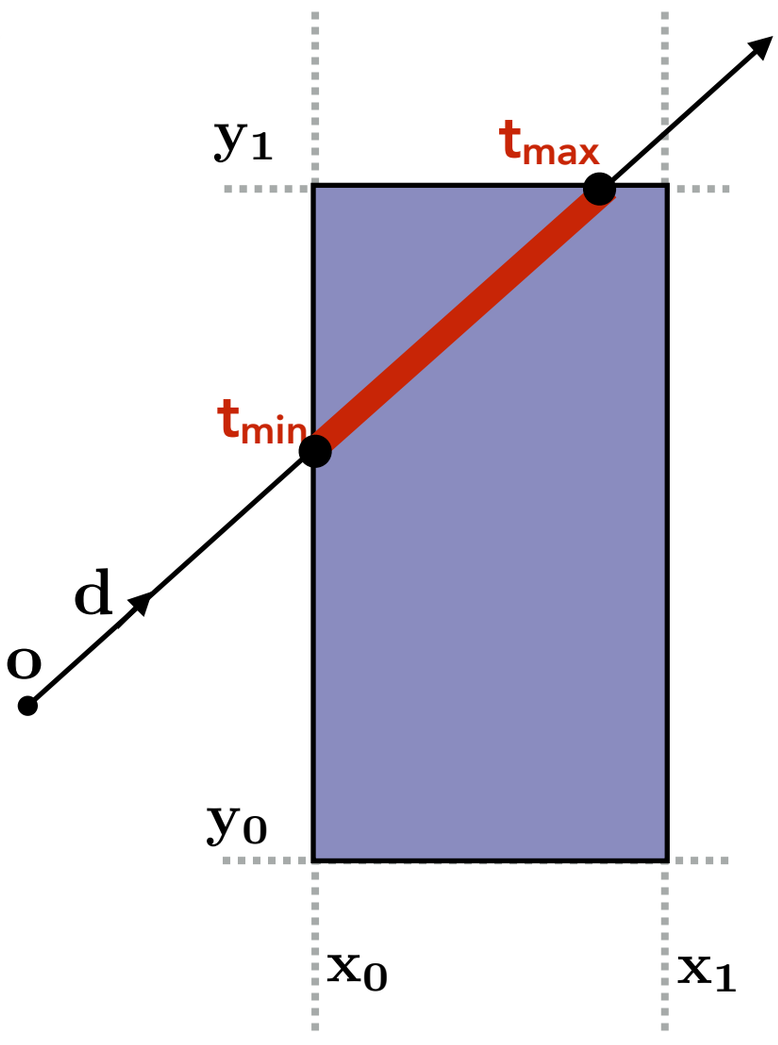
2D example; 3D is the same! Compute intersections with slabs and take intersection of t_{min}/t_{max} intervals



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 核心结论：当光线至少进入了所有对面，才进入了包围盒；以及光线至少离开了任意一个对面，就算是离开了包围盒。
  - 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, $t_{enter} = \mathbf{max}\{t_{min}\}$, $t_{exit} = \mathbf{min}\{t_{max}\}$

- 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!

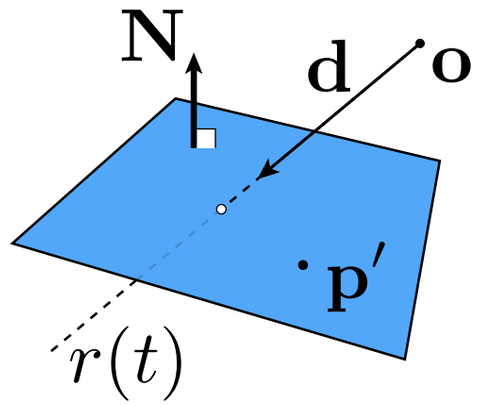
- What if $t_{exit} < 0$?
  - The box is “behind” the ray — no intersection!

- 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?

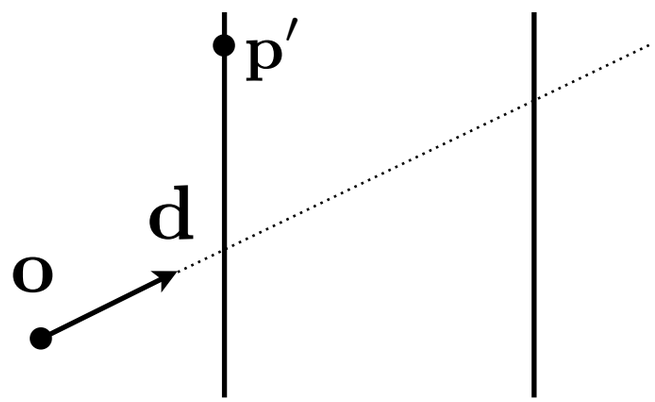
General



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

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!)