Computer Graphics

Ray Tracing

Teacher: A.prof Chengying Gao (高成英)

E-mail: mcsgcy@mail.sysu.edu.cn

School of Computer Science and Engineering

Slides from Dr. Lingqi Yan

Why Ray Tracing?

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

Soft shadows

Indirect illumination

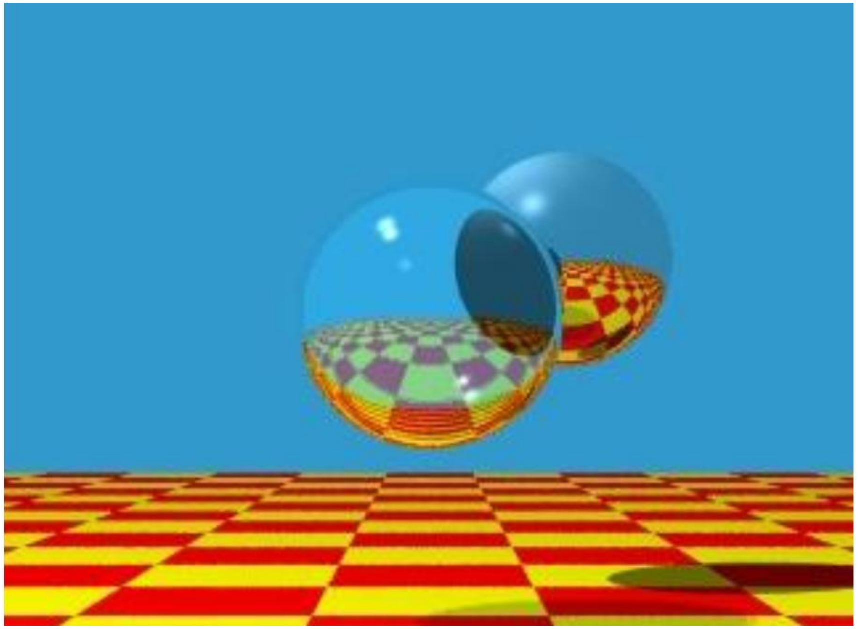
Why Ray Tracing?

- Rasterization is fast, but quality is relatively low

Buggy, from PlayerUnknown’s Battlegrounds (PC game)

Why Ray Tracing?

- Ray tracing techniques could generate impressive images including a lot of visual effects, such as hard/soft shadows, transparence(透明), translucence(半透明), reflection, refraction and so on.



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


RTX OFF
RTX ON

Why we see objects?

- Light can be interpreted as a collection of rays that begin at the light sources and bounce around the objects(reflections and refraction etc.) in the scenes.
- We see objects become rays finally come into our eyes.

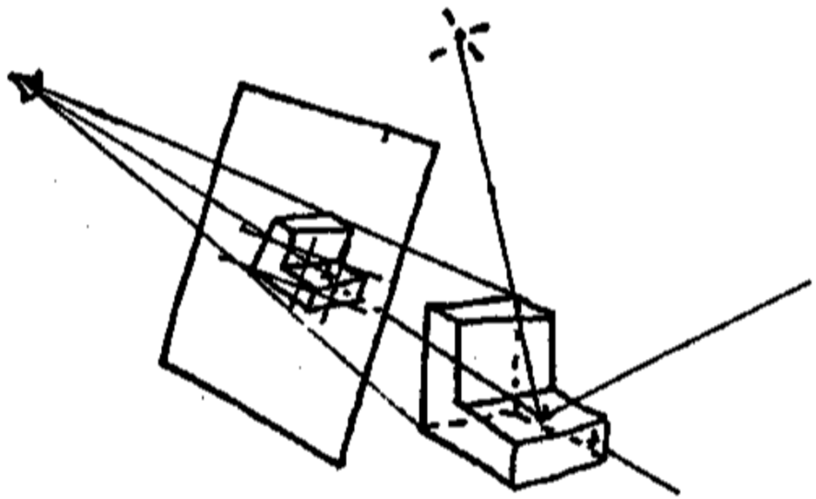
Three ideas about light rays

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

Ray casting

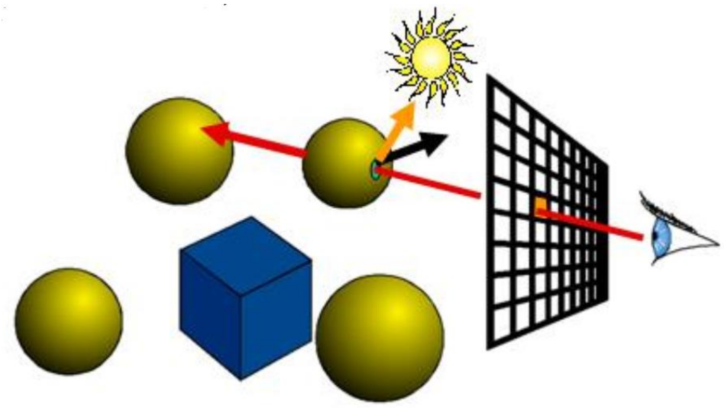
Appel 1968 - Ray casting

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

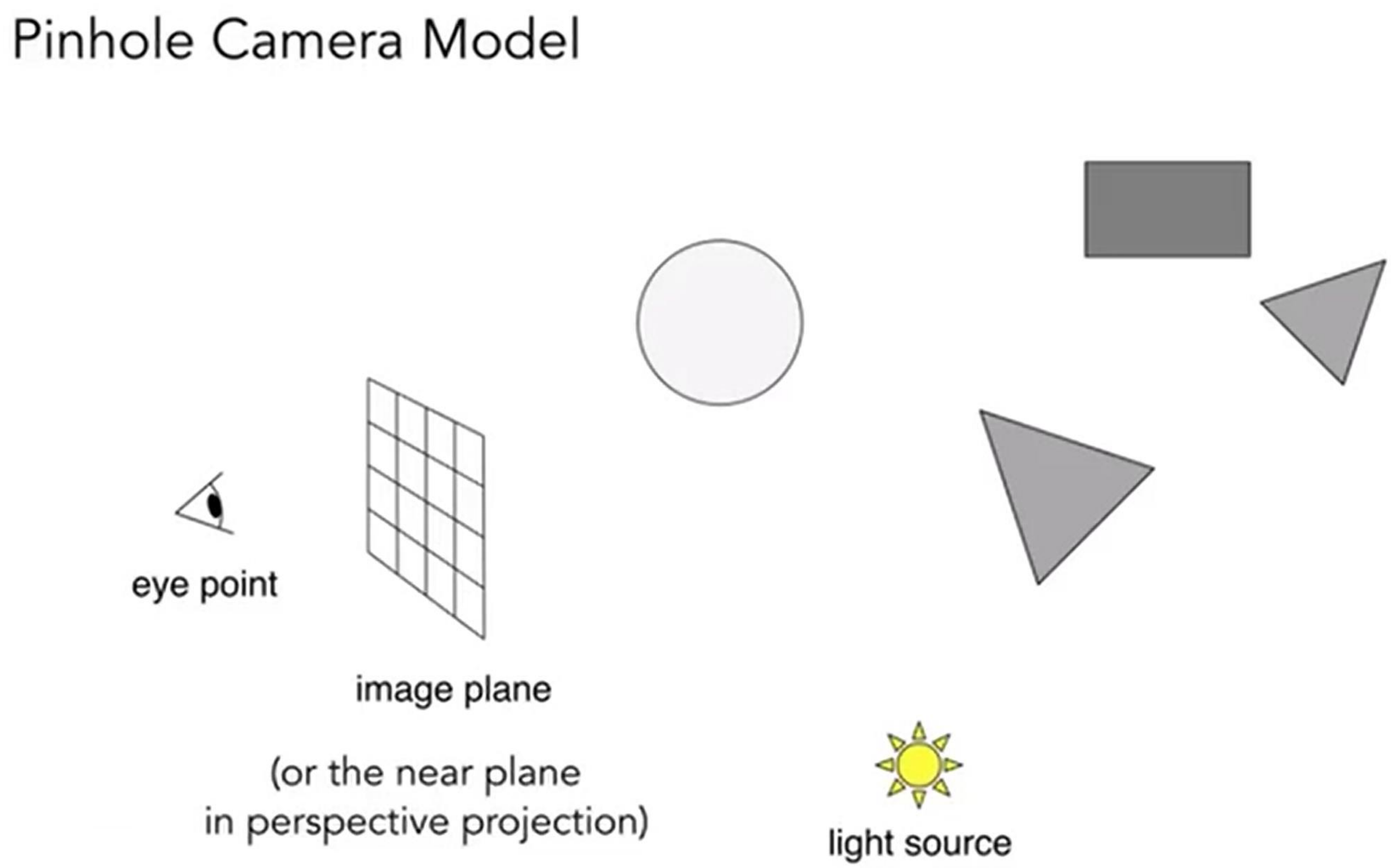


Basic idea of ray tracing

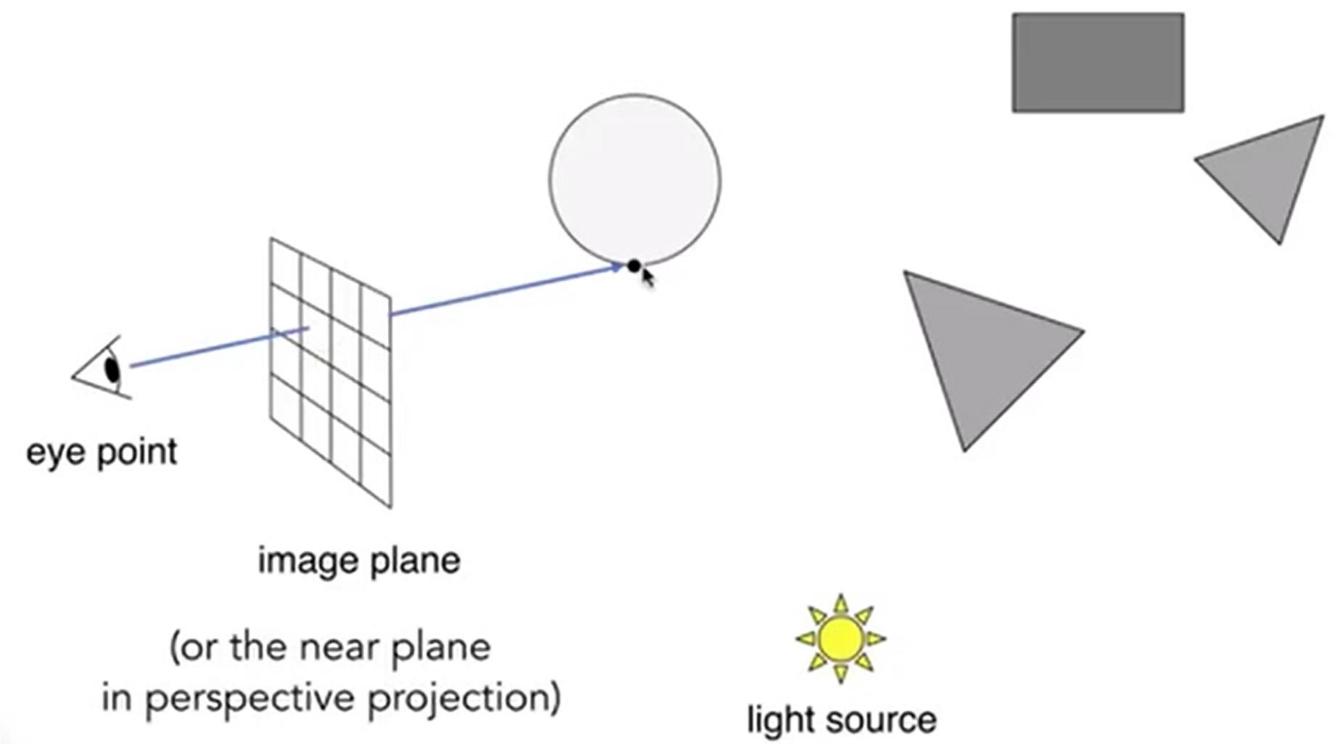
- For each pixel, what can we “see”?
- A ray casting from the eye through the center of the pixel and out into the scene, its path is traced to see which object the ray hits first.
- Check for shadows by sending a ray to the light
- Calculate the shading value of the point by the Phong model.



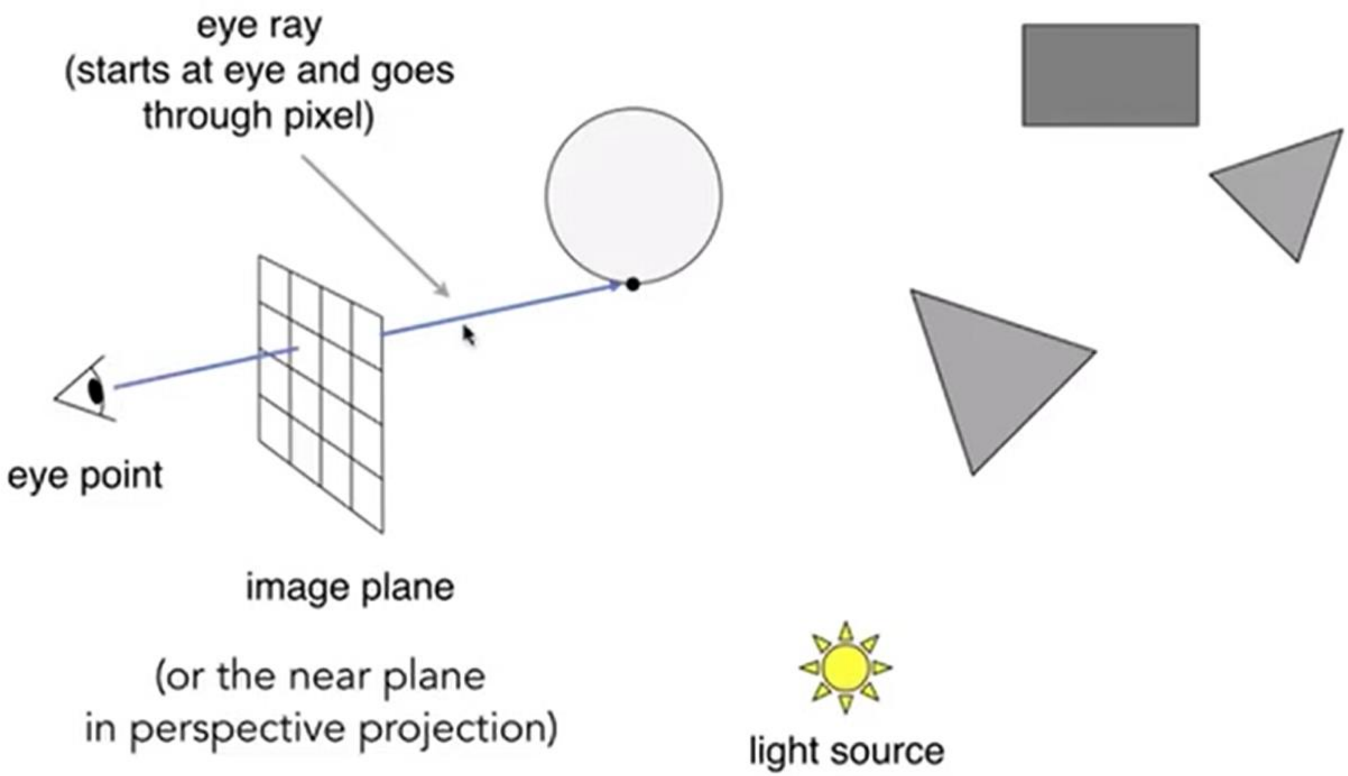
Ray Casting - Generating Eye Rays



Ray Casting - Generating Eye Rays

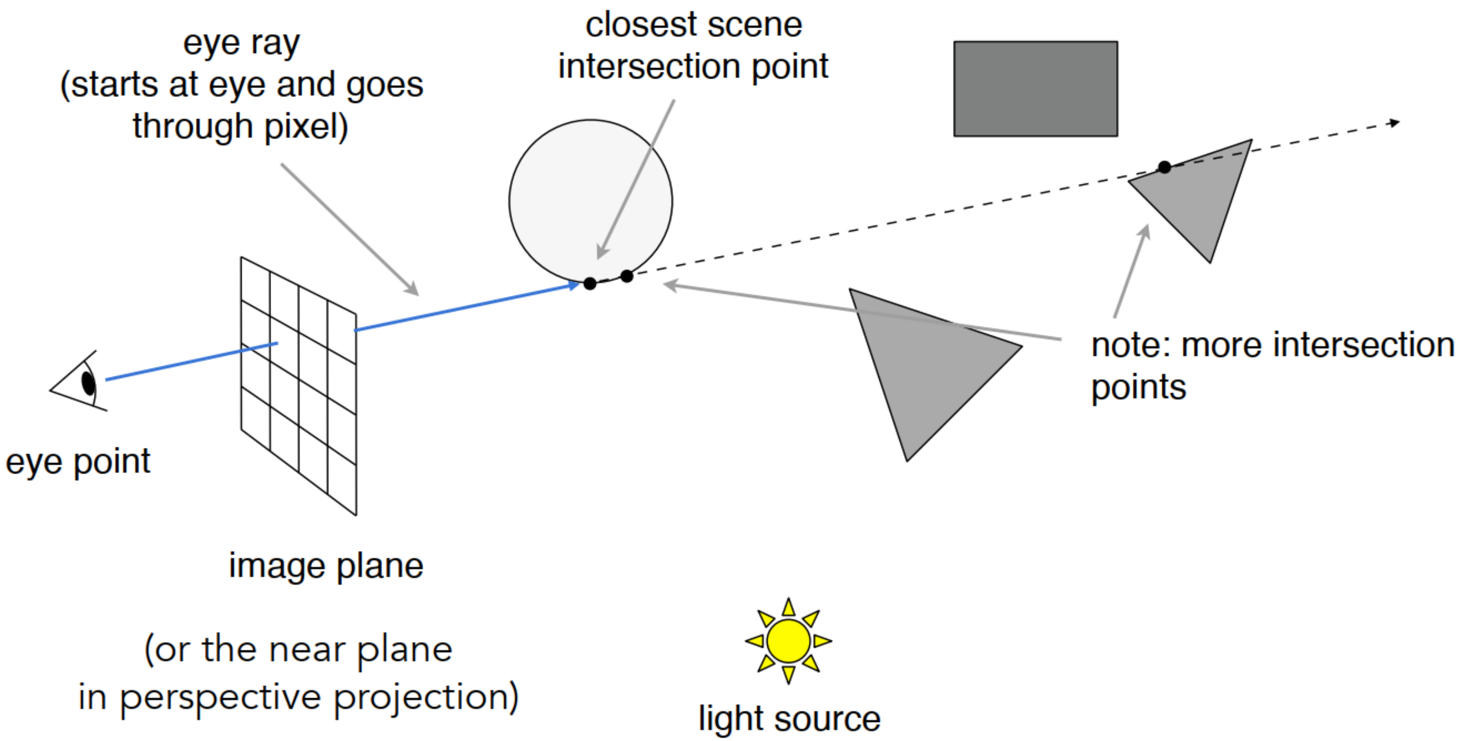


Ray Casting - Generating Eye Rays

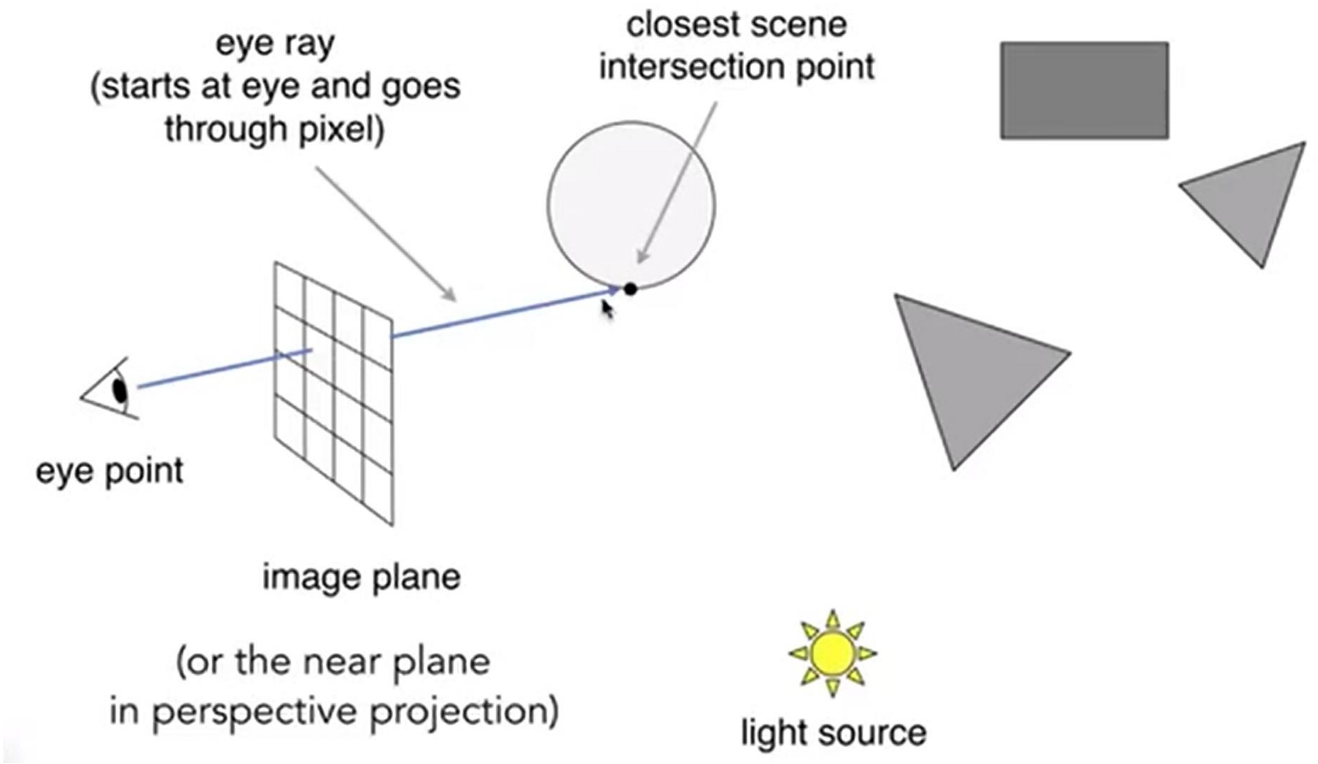


Ray Casting - Generating Eye Rays

Pinhole Camera Model

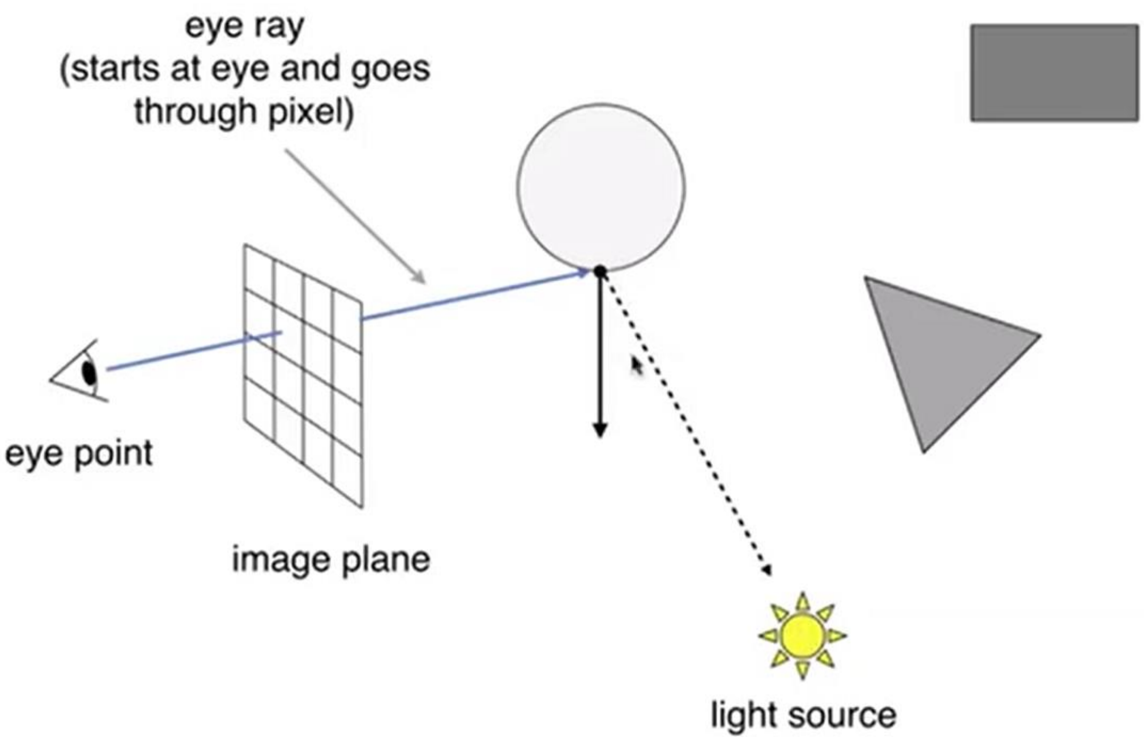


Ray Casting - Generating Eye Rays

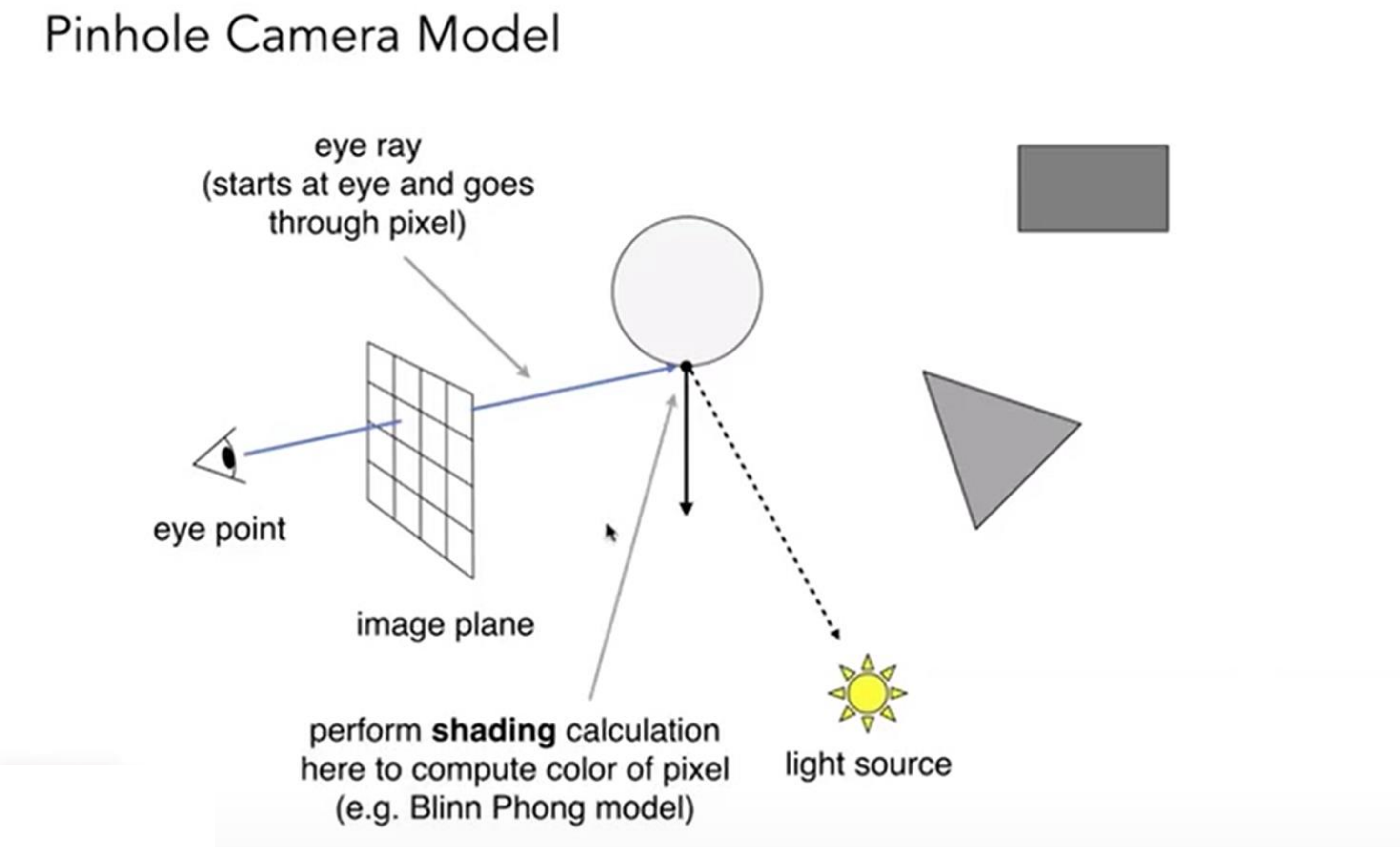


Ray Casting - Shading Pixels (Local Only)

Pinhole Camera Model

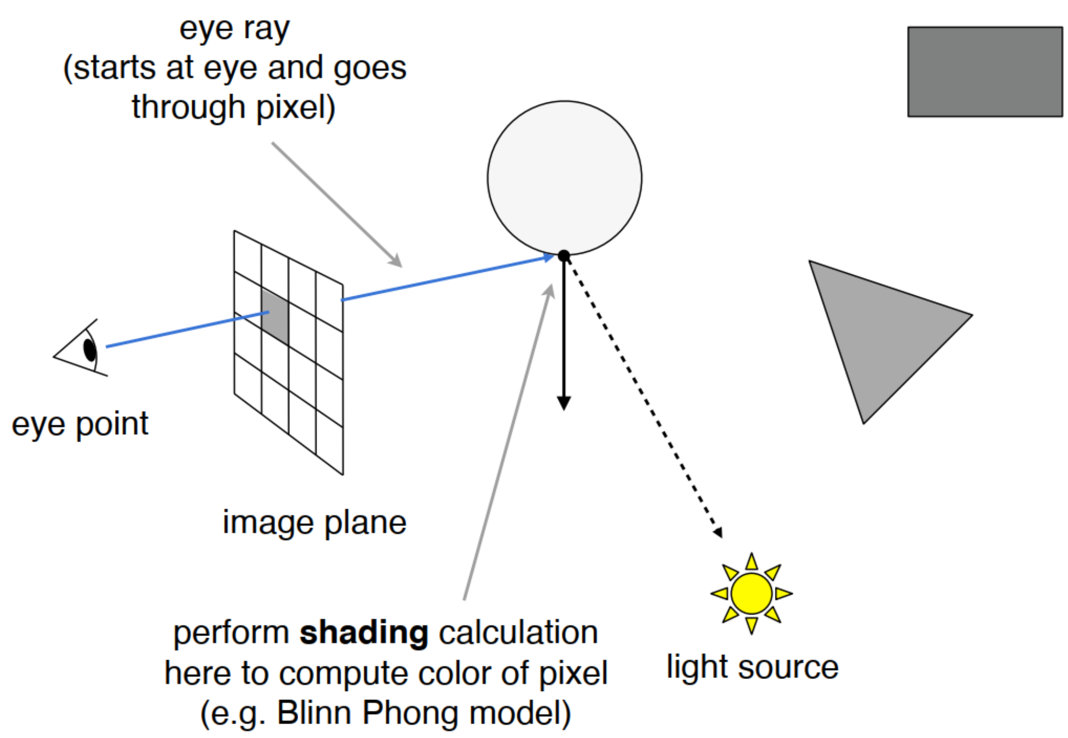


Ray Casting - Generating Eye Rays



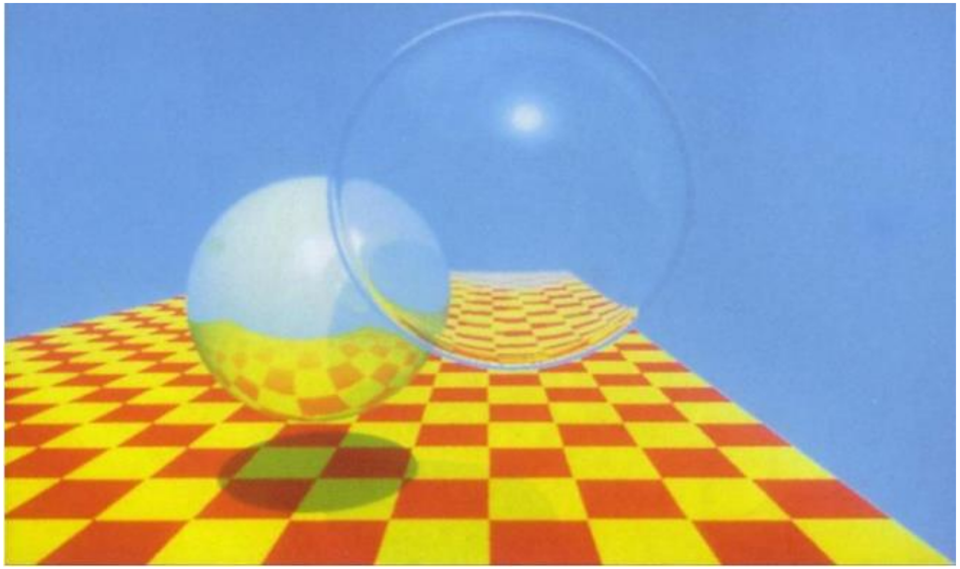
Ray Casting - Shading Pixels (Local Only)

Pinhole Camera Model



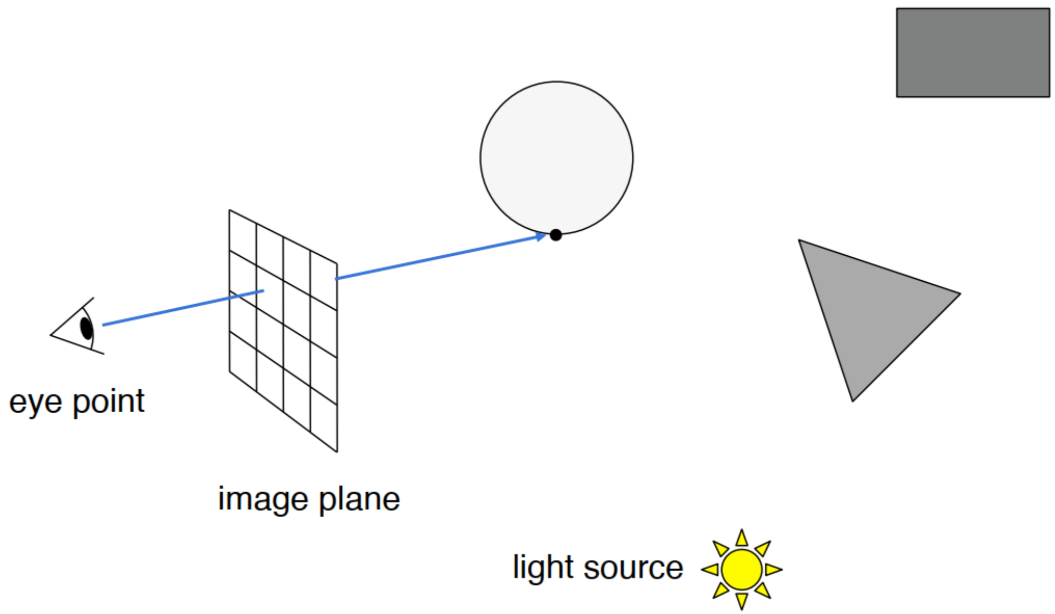
Recursive(Whitted-Style) Ray Tracing

- In 1980, Whitted proposed a ray tracing model, include light reflection and refraction effects.
  - *Turner Whitted, An improved illumination model for shaded display, Communications of the ACM, v.23 n.6, p.343-349, June 1980.*
- A Milestone of Computer Graphics.
  - VAX 11/780 (1979) 74m
  - PC (2006) 6s
  - GPU (2012) 1/30s
  - today 1/100s,1000s

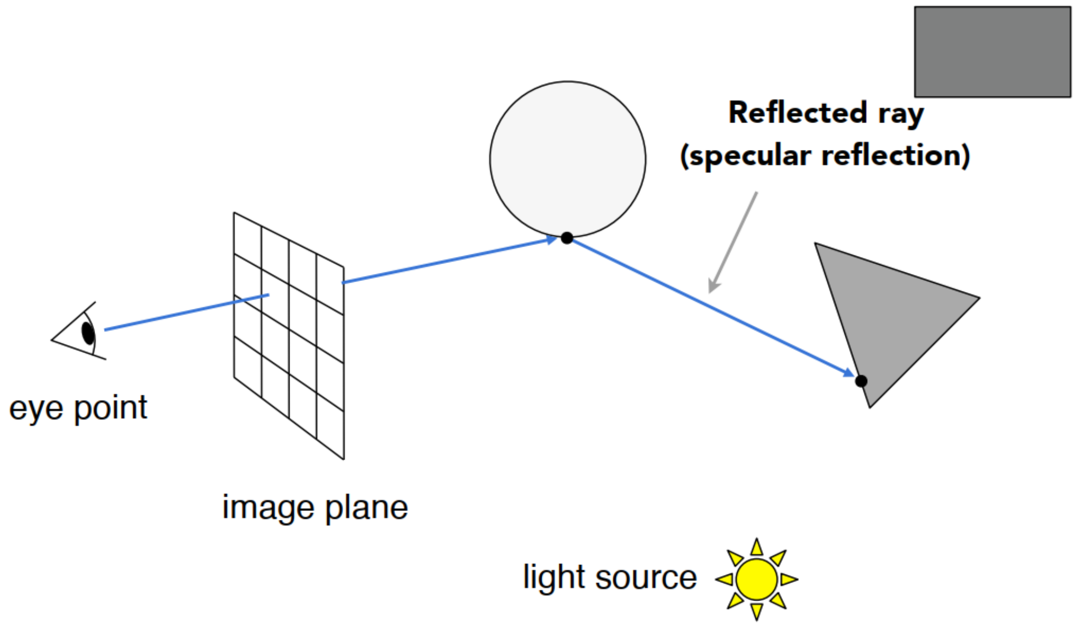


Spheres and Checkerboard, T.Whitted

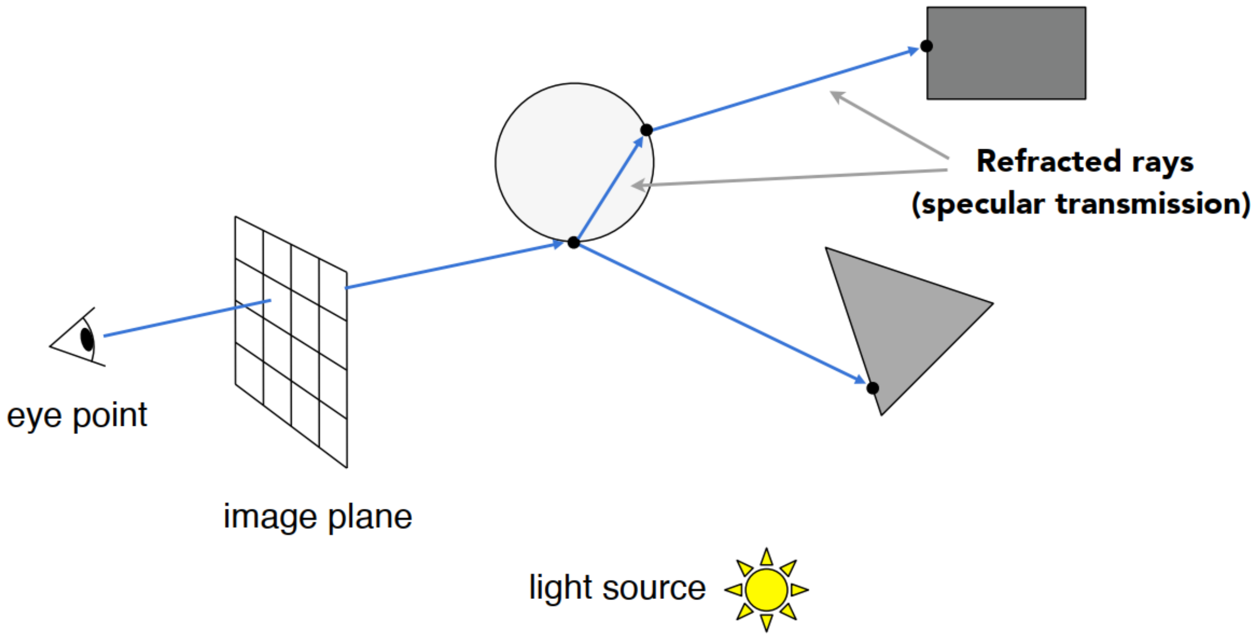
Recursive Ray Tracing



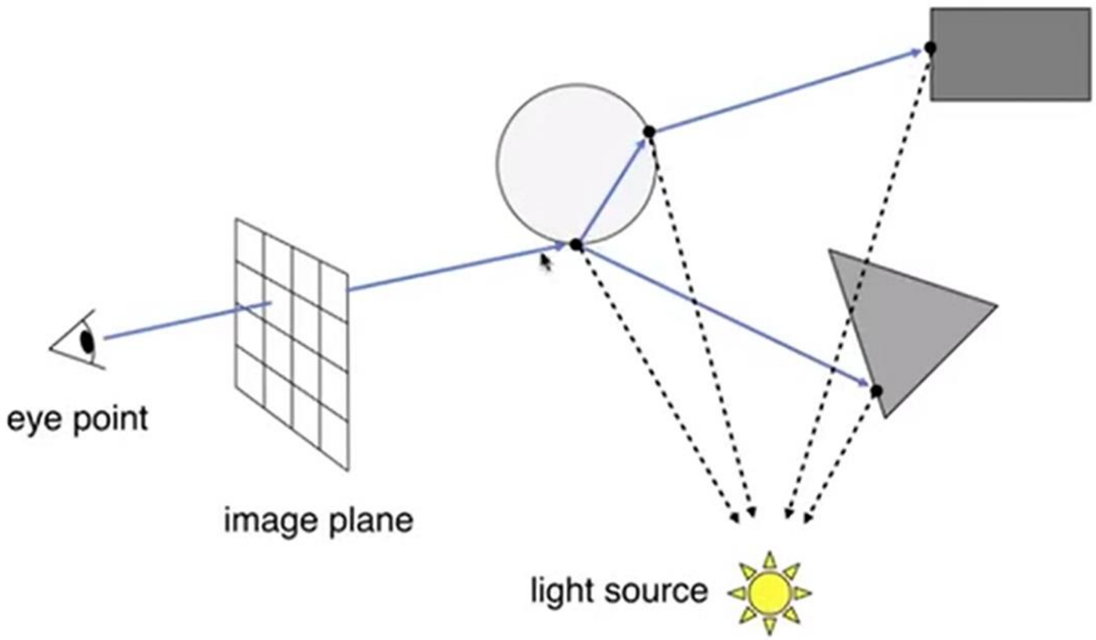
Recursive Ray Tracing



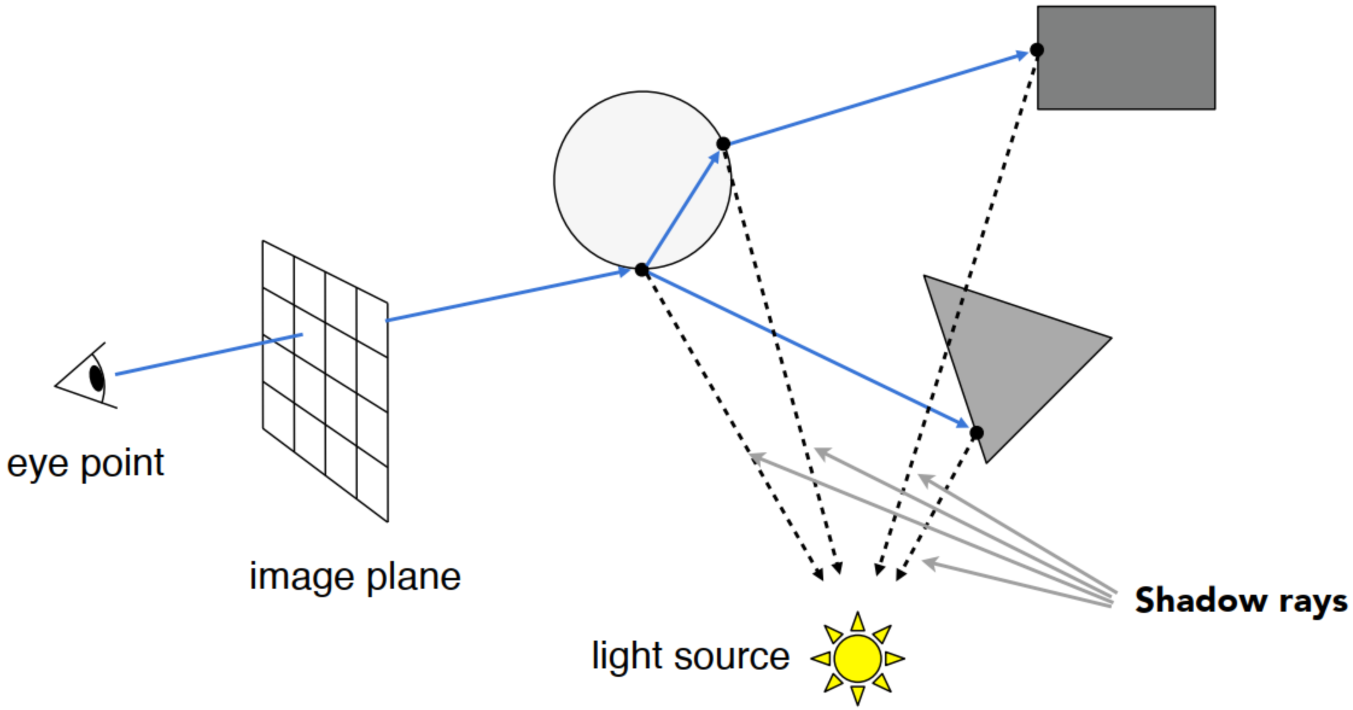
Recursive Ray Tracing



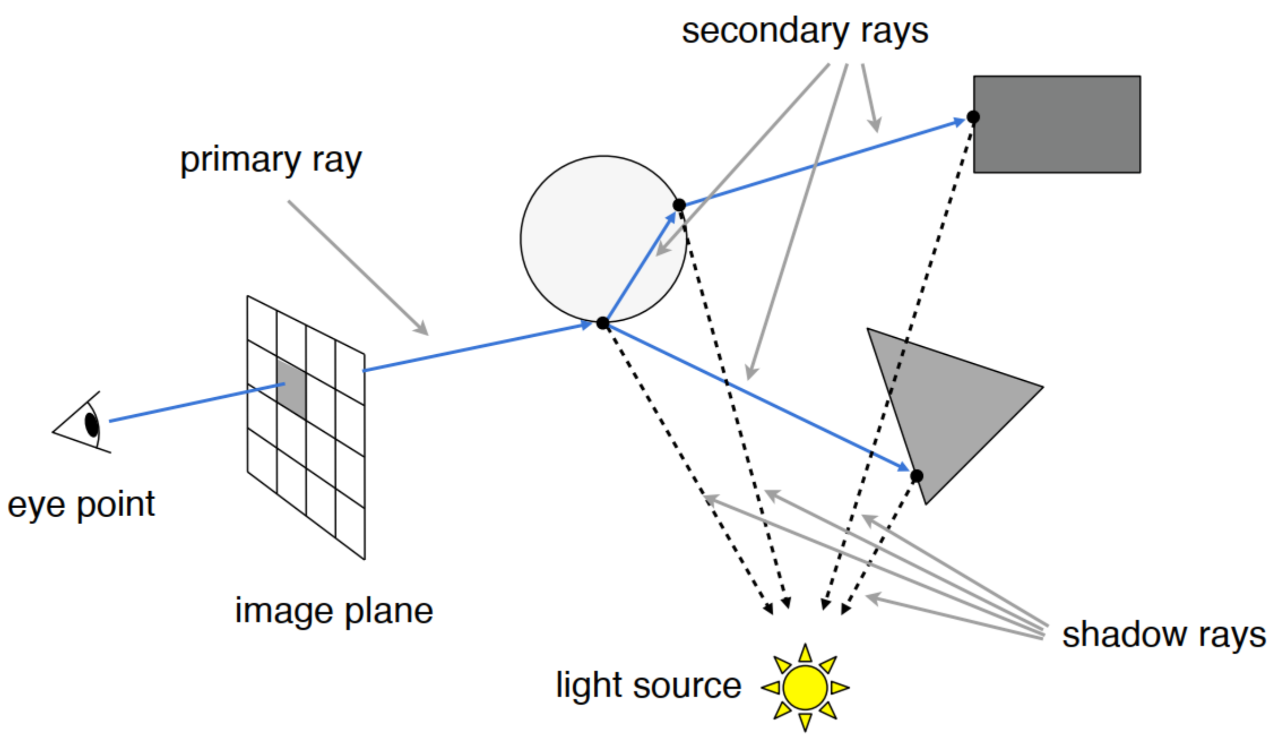
Recursive Ray Tracing



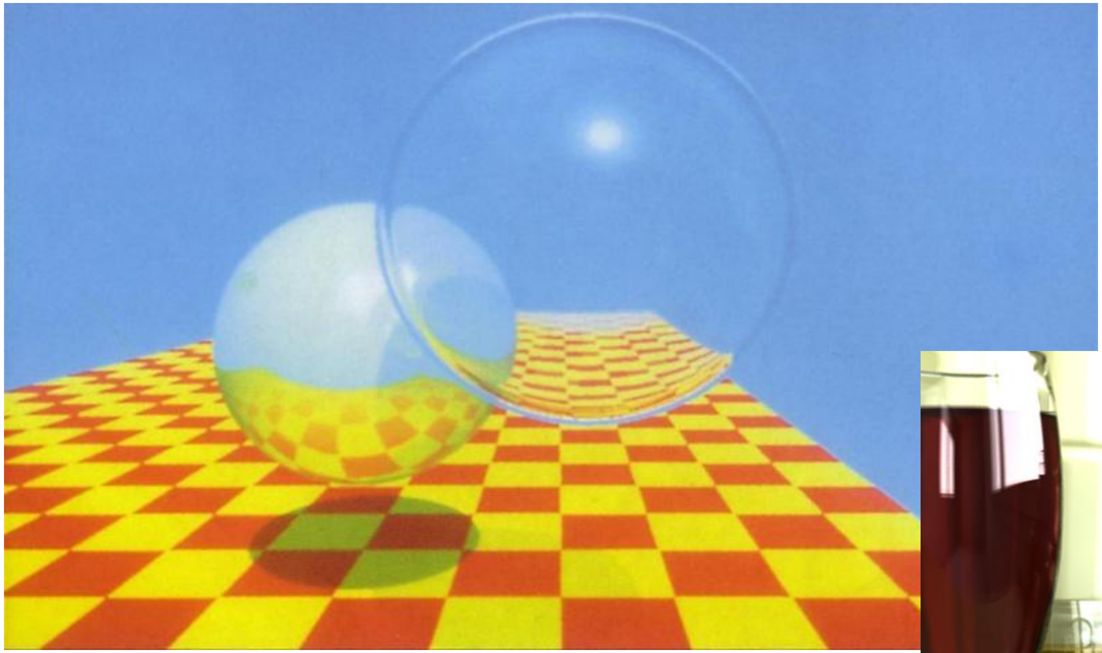
Recursive Ray Tracing



Recursive Ray Tracing



Recursive Ray Tracing

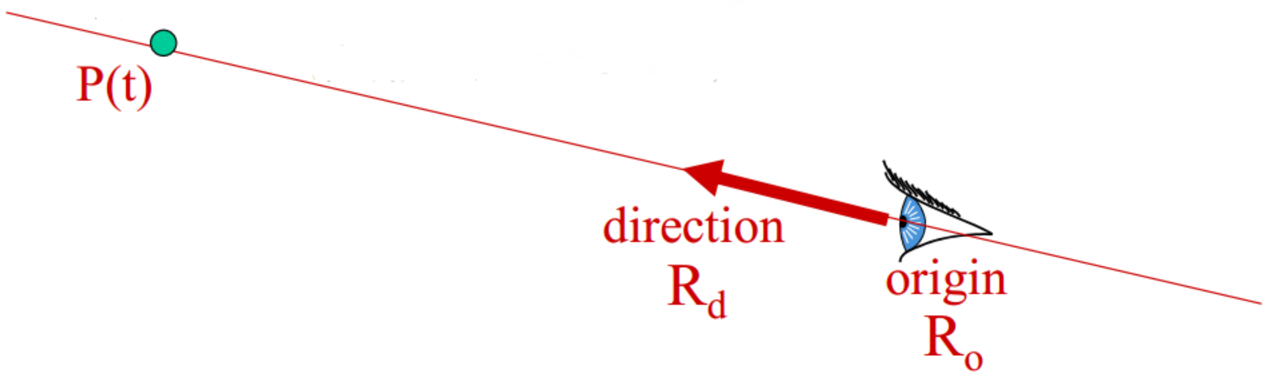


Ray-Surface intersection

- Ray Representation
- Sphere intersection （Implicit Surface）
- Triangle intersection
- Box intersection

Ray representation

- Ray is defined by its origin and a direction vector
- $P(t) = R_o + t * R_d$,
  - where $R_o=(x_o,y_o,z_o)$ is the original point of the ray, $R_d = (x_d,y_d,z_d)$ is the direction the ray is going on, usually the direction is normalized.
  - t value determines the point the ray arrives at, its value is always larger than 0



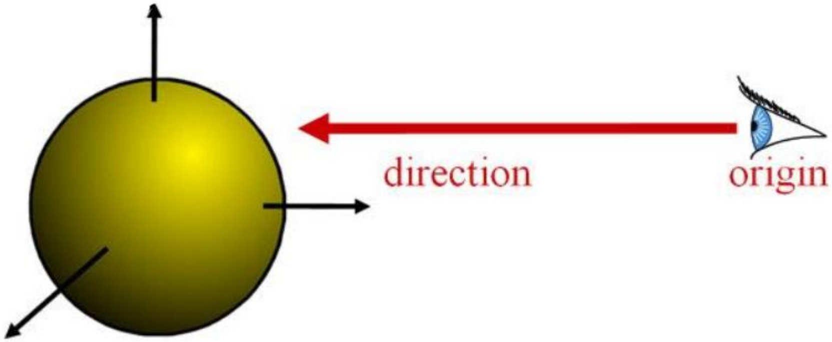
Ray Intersection With Sphere

- sphere defined as:
  - a center point P_c , and a radius r.
  - the implicit formula for the sphere is :

$$f(P) = \|P - P_c\| - r = 0$$

  - To solve for the intersection between a ray and a sphere, simply replace P in the ray equation to yield:

$$\|P(t) - P_c\| - r = 0$$



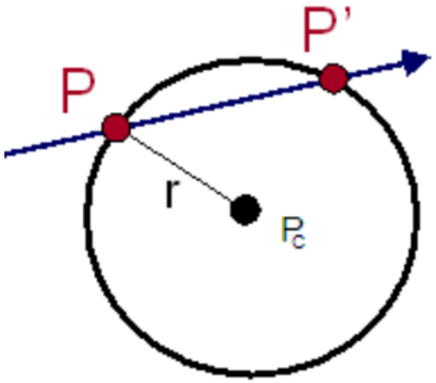
What is an intersection?

The intersection p must satisfy both ray equation and sphere equation

Sphere Intersection: Algebra method

- The equation of last page is simplified as follows

$$\|P(t)-P_c\|-r=0$$



$$\|R_0+tR_d-P_c\|=r$$

$$(R_0+tR_d-P_c)\cdot(R_0+tR_d-P_c)=r^2$$

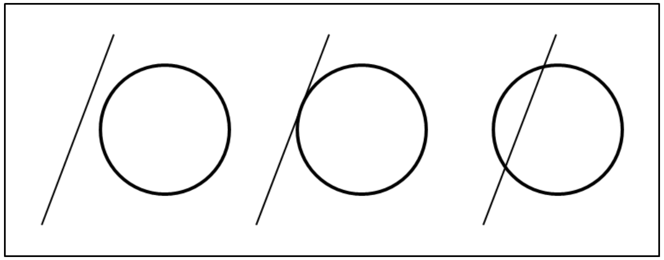
$$t^2(R_d\cdot R_d)+2t(R_d\cdot(R_0-P_c))+(R_0-P_c)\cdot(R_0-P_c)-r^2=0$$

- Since R_d is normalized:

$$t^2+2t(R_d\cdot(R_0-P_c))+(R_0-P_c)\cdot(R_0-P_c)-r^2=0$$

$$t^2+2tb+c=0$$

$$t=-b\pm\sqrt{b^2-c}$$



Ray Intersection With Implicit Surface

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

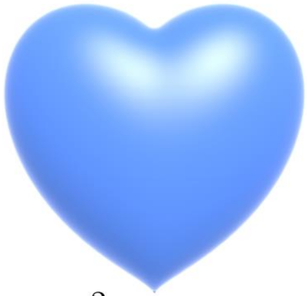
General implicit surface: $\mathbf{p} : f(\mathbf{p}) = 0$

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

Solve for **real, positive** roots

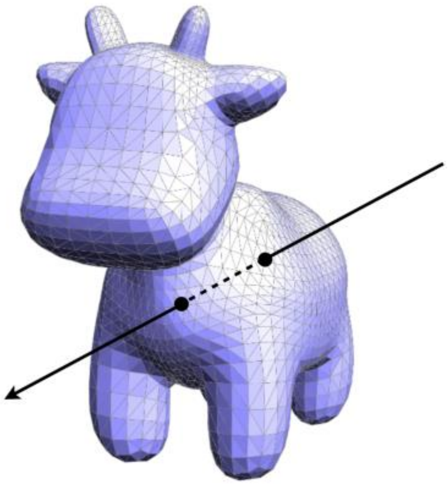
$x^2+y^2+z^2=1$

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



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

Ray Intersection With Triangle Mesh



Why?

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

How to compute?

Let's break this down:

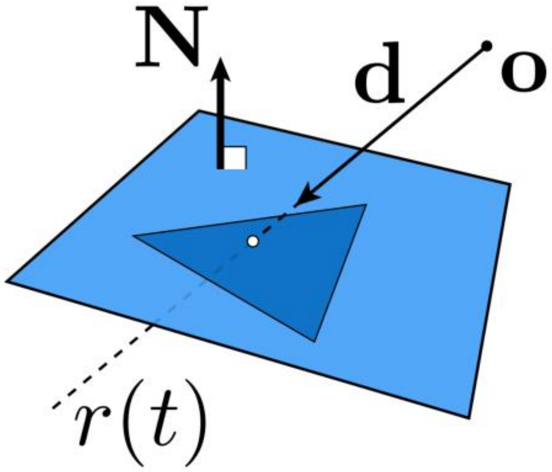
- Simple idea: just intersect ray with each triangle
- Simple, but slow (acceleration?)
- Note: can have 0, 1 intersections (ignoring multiple intersections)

Ray Intersection With Triangle Mesh

Triangle is in a plane

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

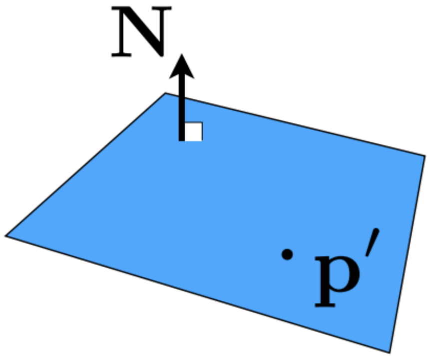
Many ways to optimize…



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):

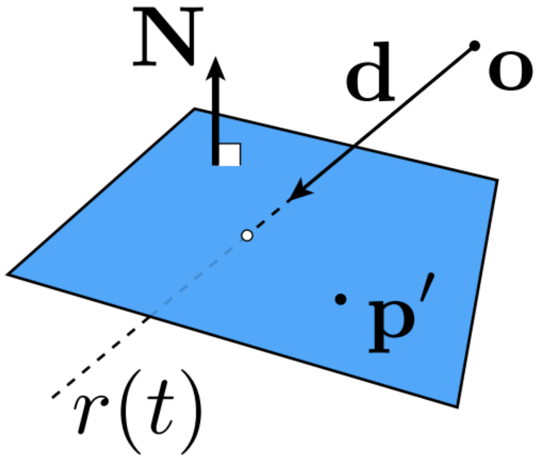
$$\mathbf{p} : (\mathbf{p} - \mathbf{p}') \cdot \mathbf{N} = 0 \qquad\qquad ax + by + cz + d = 0$$

all points on plane — one point on plane — normal vector

Ray Intersection With Plane

Ray equation:

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



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: $0 \leq t < \infty$

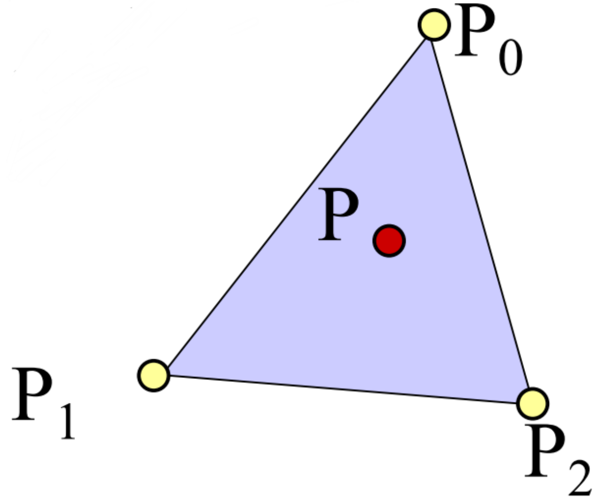
A faster approach, giving barycentric coordinate directly

Triangle Intersection

- Barycentric coordinates:
  - A point P, on a triangle $P_0P_1P_2$, is given by the explicit formula:

$$P = \alpha P_0 + \beta P_1 + \gamma P_2$$

  - where (α, β, γ) are the barycentric coordinates, which must satisfy $0 \le \alpha, \beta, \gamma \le 1$, $\alpha + \beta + \gamma = 1$



Triangle Intersection

- Since α + β + γ = 1, we can write α = 1 - β - γ, we have:

$$P = (1-\beta-\gamma)P_0 + \beta P_1 + \gamma P_2$$

- Set ray equation equal to barycentric equation:

$$R_o + tR_d = (1-\beta-\gamma)P_0 + \beta P_1 + \gamma P_2$$

- Rearrange the terms, gives:

$$\begin{pmatrix} R_d & P_0 - P_1 & P_0 - P_2 \end{pmatrix} \begin{pmatrix} t \\ \beta \\ \gamma \end{pmatrix} = P_0 - R_0$$

- The means the barycentric coordinate and distance t can be found by solving this linear system of equations

Triangle Intersection

- Denoting $E_1 = P_0 - P_1, \mathrm{E}_2 = P_0 - P_2, S = P_0 - R_0$
- the solution to the equation above is obtained by Cramer's rule:

$$\begin{pmatrix} t \\ \beta \\ \gamma \end{pmatrix} = \frac{1}{\det(d, E_1, E_2)} \begin{pmatrix} \det(S, E_1, E_2) \\ \det(d, S, E_2) \\ \det(d, E_1, S) \end{pmatrix}$$

- Then check $0 \le \beta, \gamma \le 1, \beta + \gamma \le 1$ to determine whether or not the intersect point is inside the triangle

Homework 4