SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of Information Technology & Artificial Intelligence & Data Science

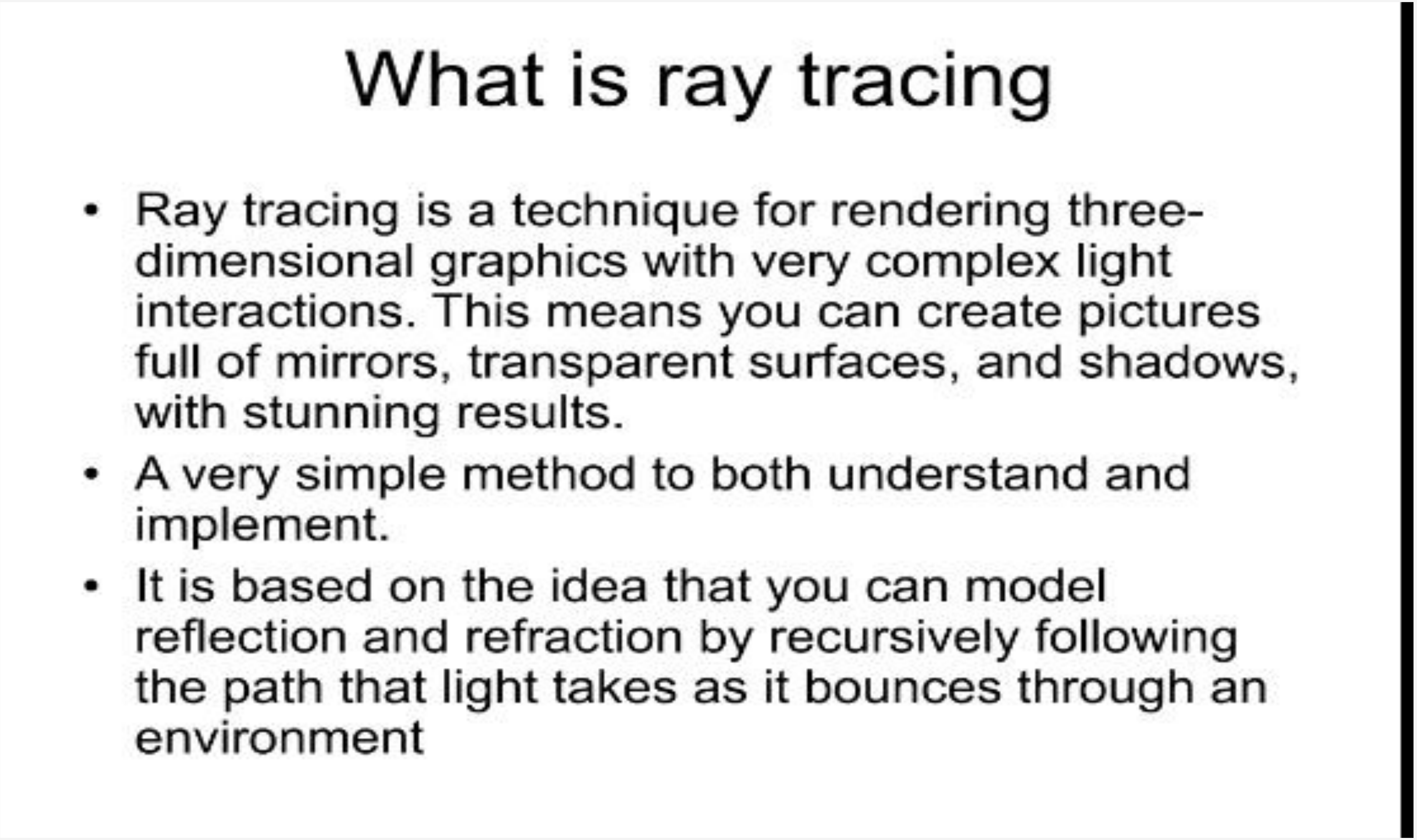
Course Name – COMPUTER GRAPHICS

III Year / V Semester

Unit 4 – SURFACE DESIGN

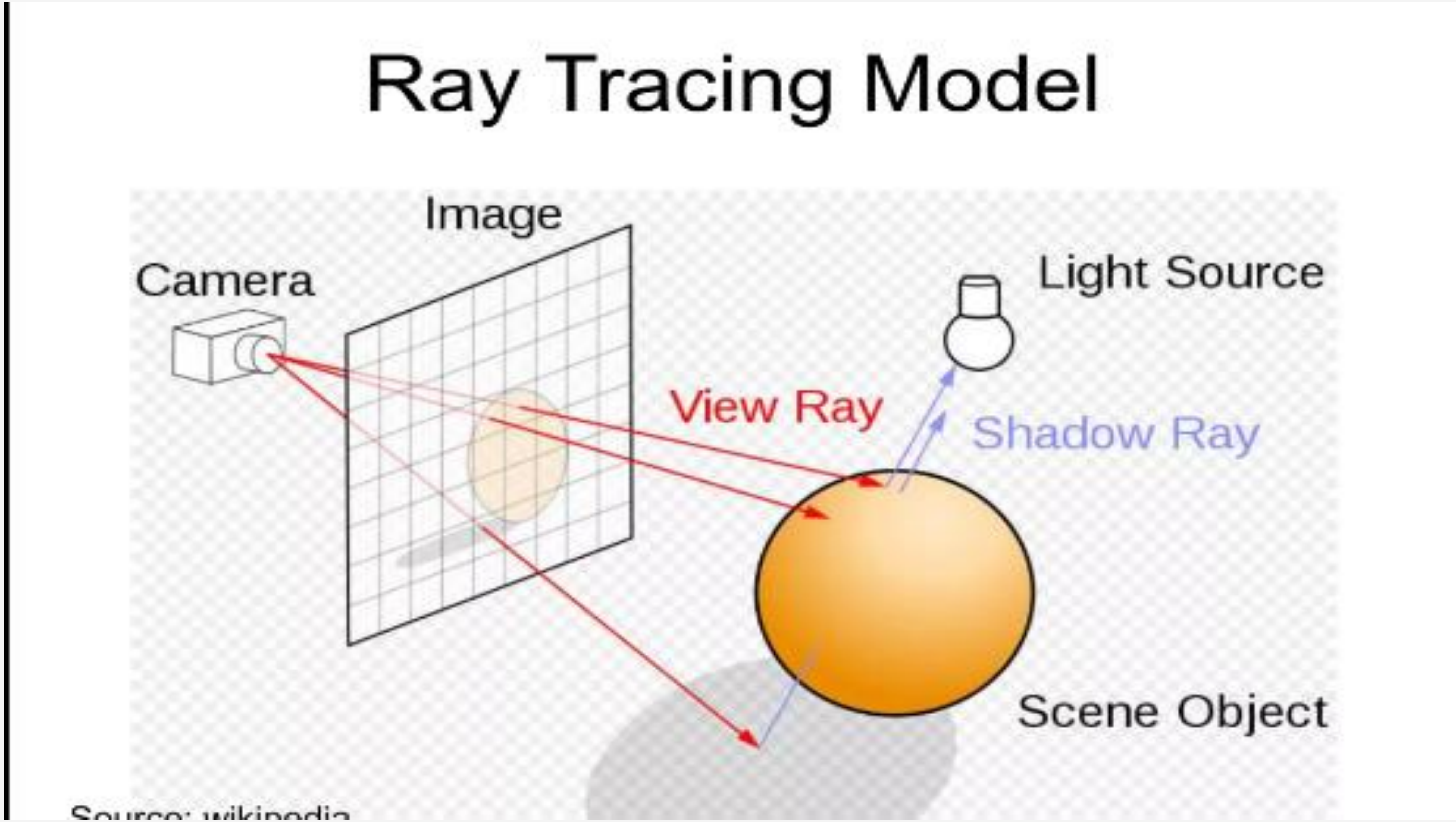
Topics-Overview of the Ray Tracing Process – Intersecting Rays with other Primitives – Adding Shadows for Greater Realism – Reflections and Transparency – Boolean Operations on Objects – Ray Casting.

OVERVIEW OF THE RAY TRACING PROCESS



What is ray tracing

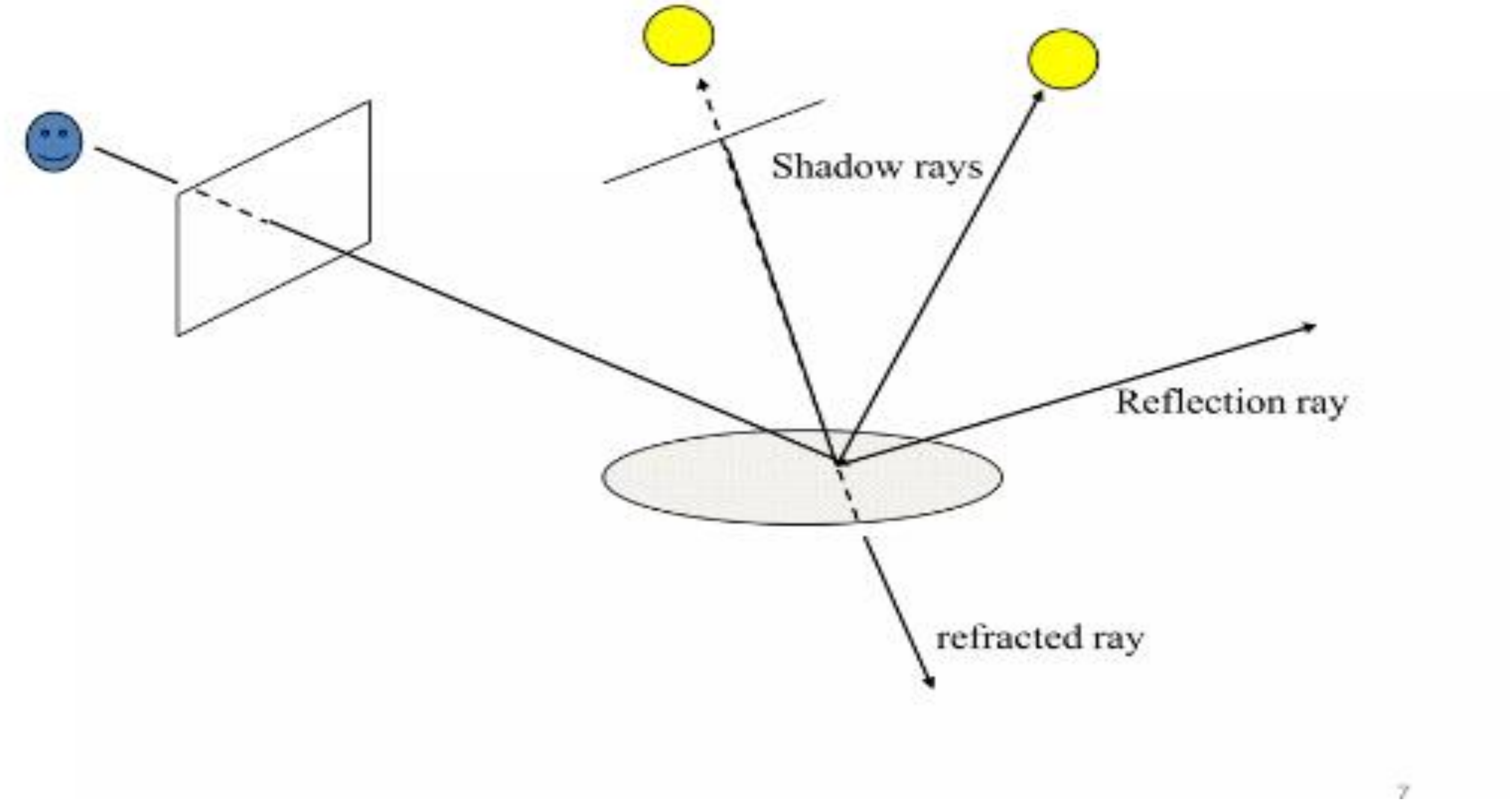
- Ray tracing is a technique for rendering three-dimensional graphics with very complex light interactions. This means you can create pictures full of mirrors, transparent surfaces, and shadows, with stunning results.
- A very simple method to both understand and implement.
- It is based on the idea that you can model reflection and refraction by recursively following the path that light takes as it bounces through an environment

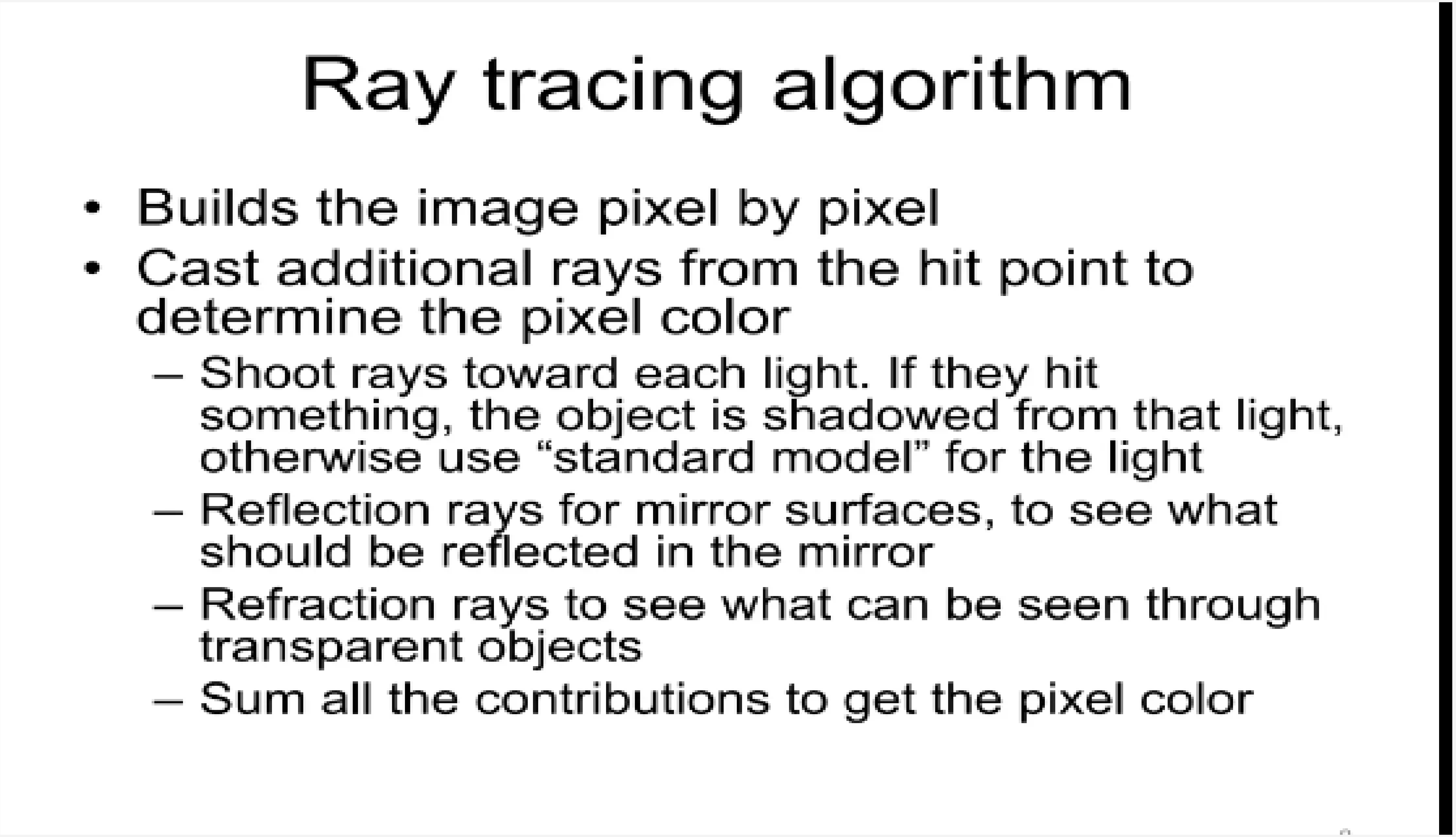


Ray Tracing Model

Source: wikipedia

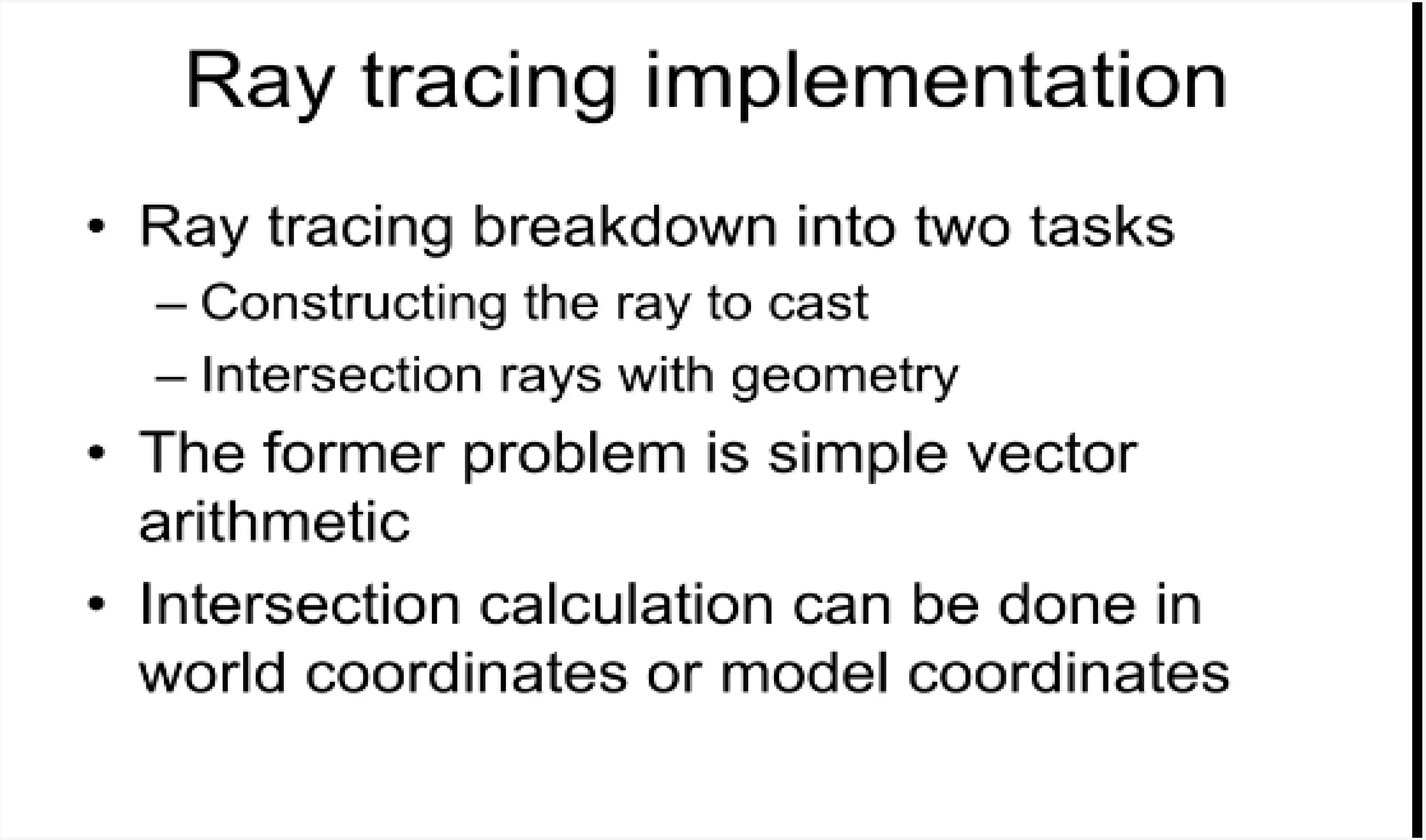
Ray tracing





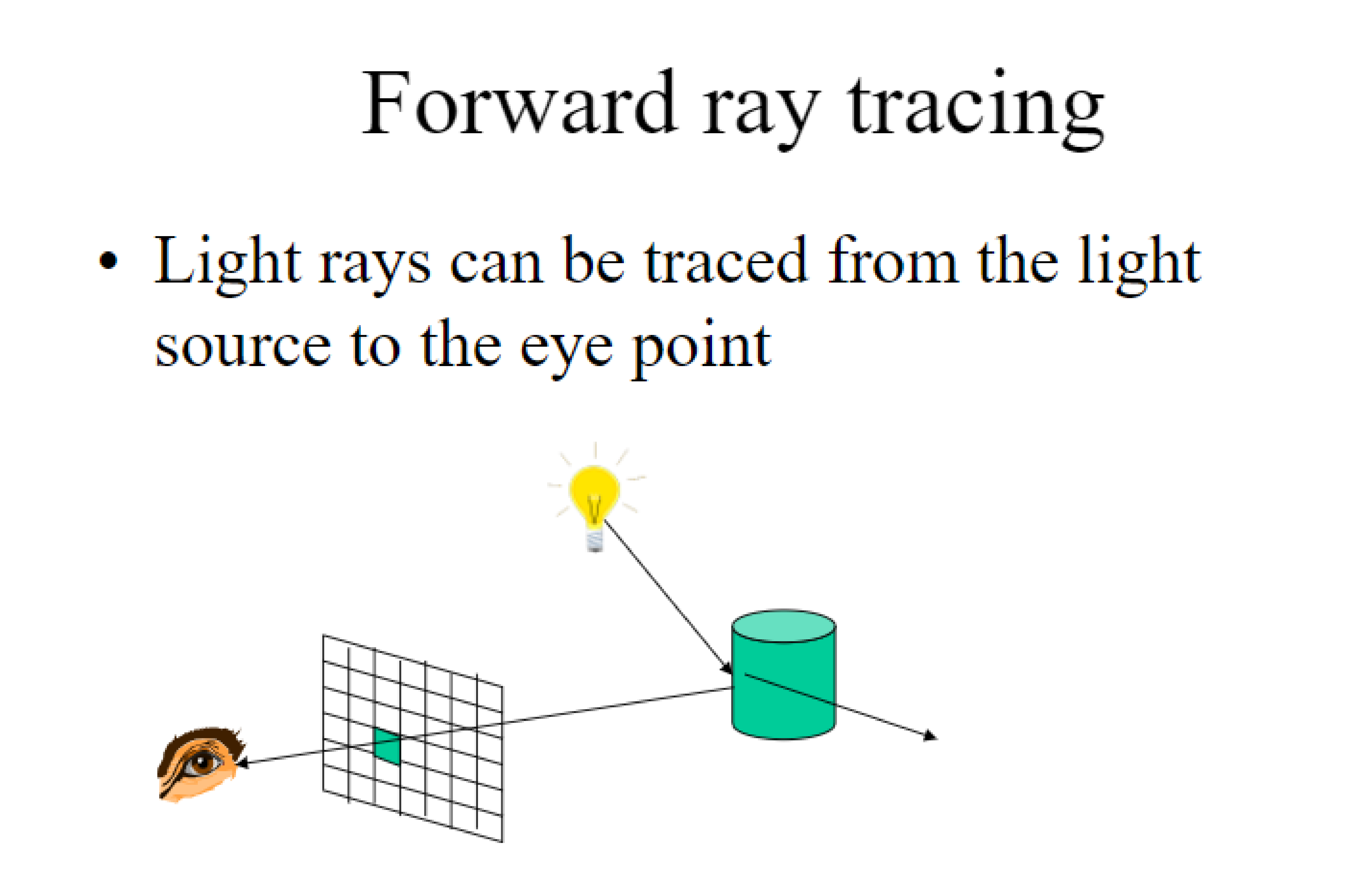
Ray tracing algorithm

- Builds the image pixel by pixel
- Cast additional rays from the hit point to determine the pixel color
  - – Shoot rays toward each light. If they hit something, the object is shadowed from that light, otherwise use “standard model” for the light
  - – Reflection rays for mirror surfaces, to see what should be reflected in the mirror
  - – Refraction rays to see what can be seen through transparent objects
  - – Sum all the contributions to get the pixel color



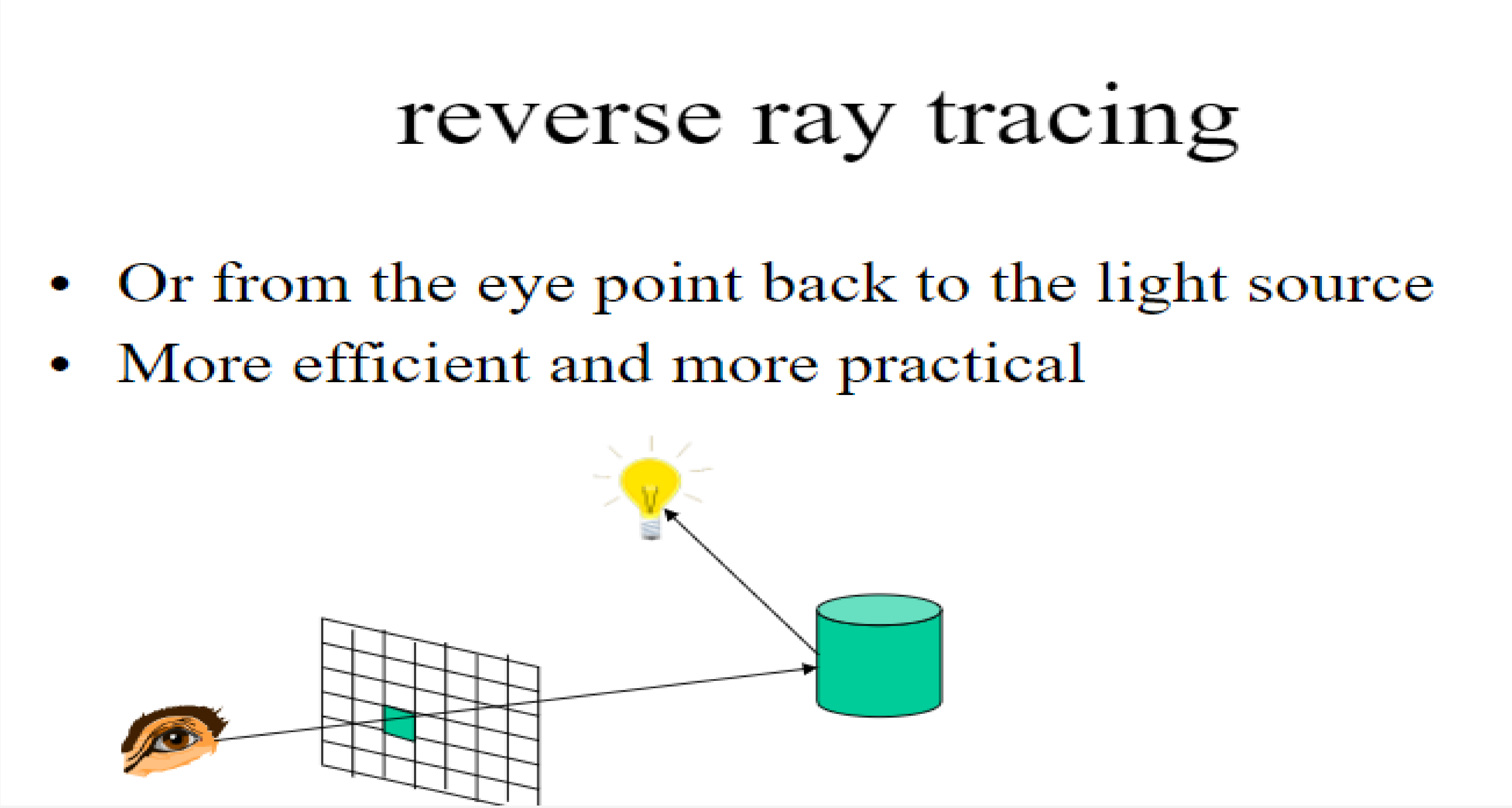
Ray tracing implementation

- Ray tracing breakdown into two tasks
  - Constructing the ray to cast
  - Intersection rays with geometry
- The former problem is simple vector arithmetic
- Intersection calculation can be done in world coordinates or model coordinates



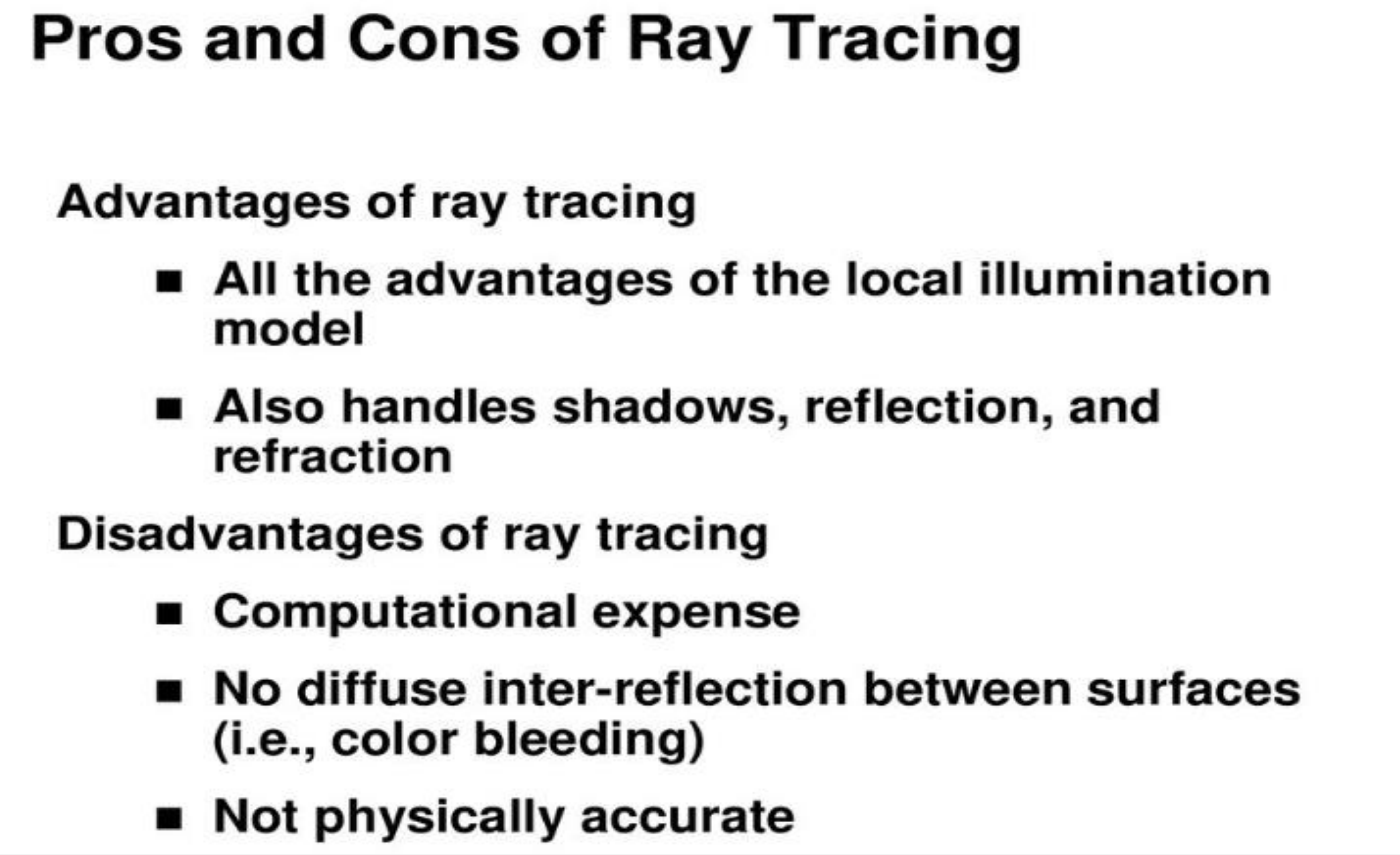
Forward ray tracing

- Light rays can be traced from the light source to the eye point



reverse ray tracing

- Or from the eye point back to the light source
- More efficient and more practical



Pros and Cons of Ray Tracing

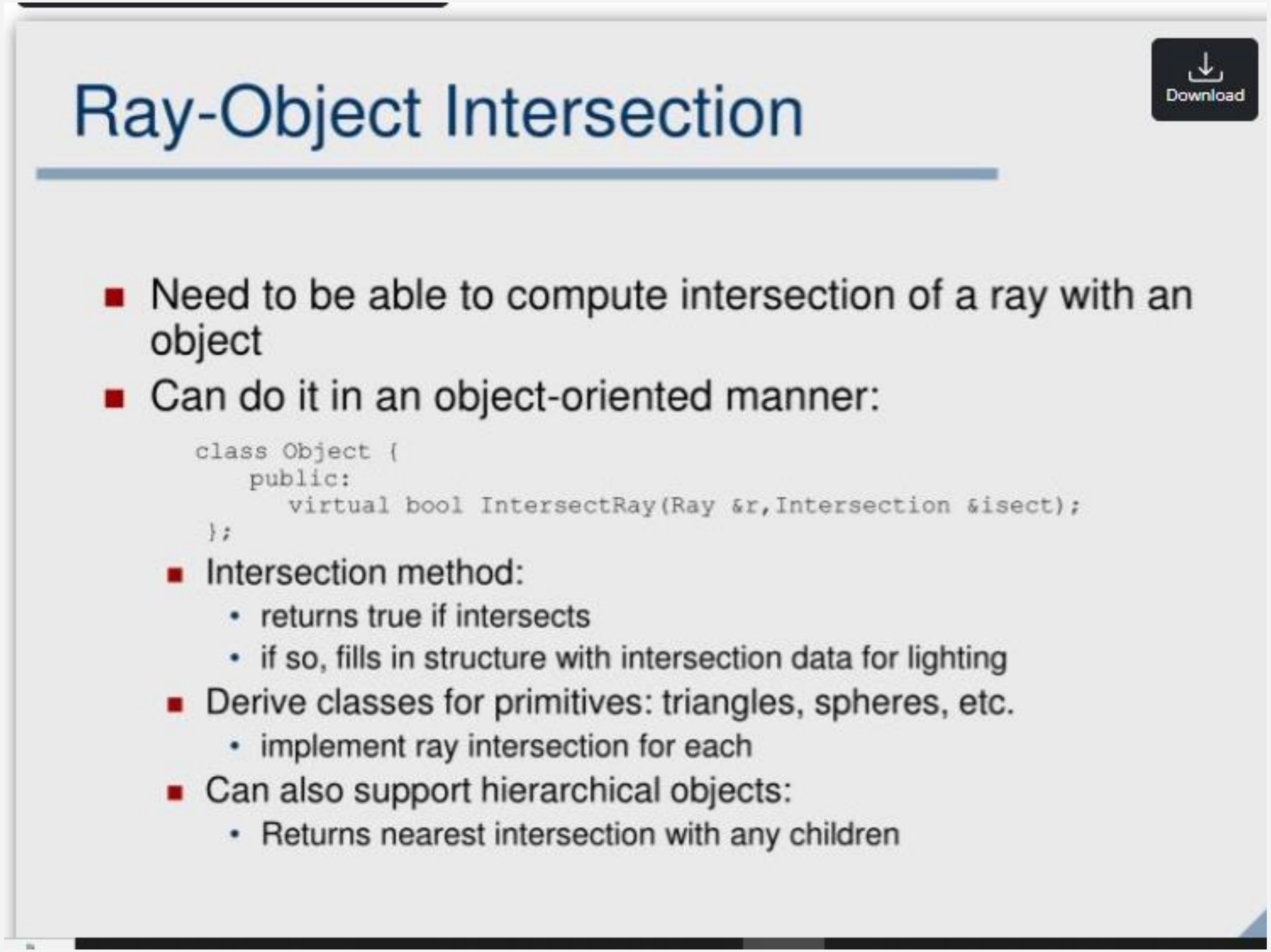
Advantages of ray tracing

- All the advantages of the local illumination model
- Also handles shadows, reflection, and refraction

Disadvantages of ray tracing

- Computational expense
- No diffuse inter-reflection between surfaces (i.e., color bleeding)
- Not physically accurate

INTERSECTING RAYS WITH OTHER PRIMITIVES

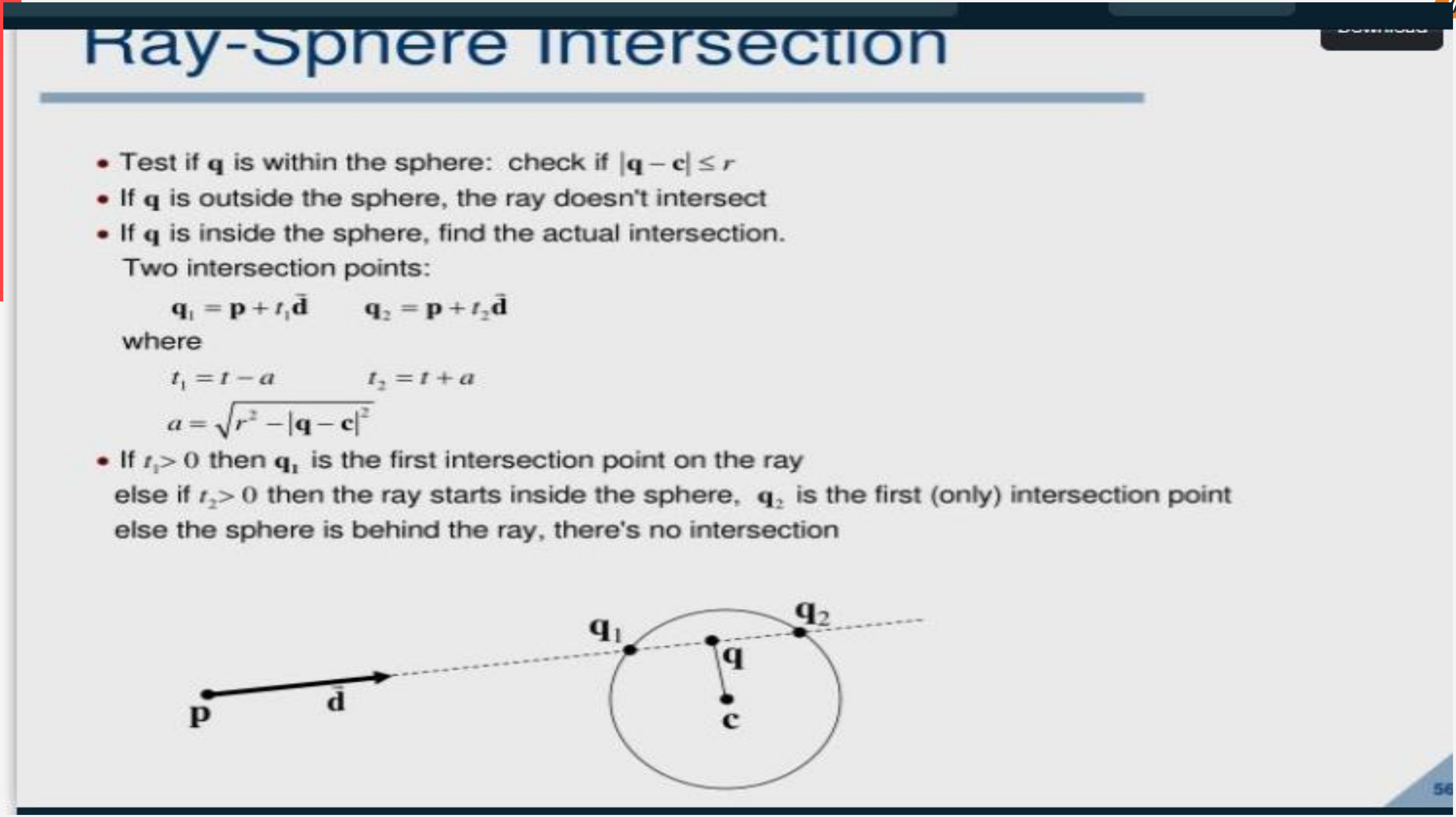


Ray-Object Intersection

- Need to be able to compute intersection of a ray with an object
- Can do it in an object-oriented manner:

```
class Object {
    public:
       virtual bool IntersectRay(Ray &r,Intersection &isect);
 };
```

  - Intersection method:
    - returns true if intersects
    - if so, fills in structure with intersection data for lighting
  - Derive classes for primitives: triangles, spheres, etc.
    - implement ray intersection for each
  - Can also support hierarchical objects:
    - Returns nearest intersection with any children



Ray-Sphere Intersection

- Test if $\mathbf{q}$ is within the sphere: check if $|\mathbf{q}-\mathbf{c}| \leq r$
- If $\mathbf{q}$ is outside the sphere, the ray doesn't intersect
- If $\mathbf{q}$ is inside the sphere, find the actual intersection.

  Two intersection points:

  $$\mathbf{q}_1 = \mathbf{p} + t_1\vec{\mathbf{d}} \qquad \mathbf{q}_2 = \mathbf{p} + t_2\vec{\mathbf{d}}$$

  where

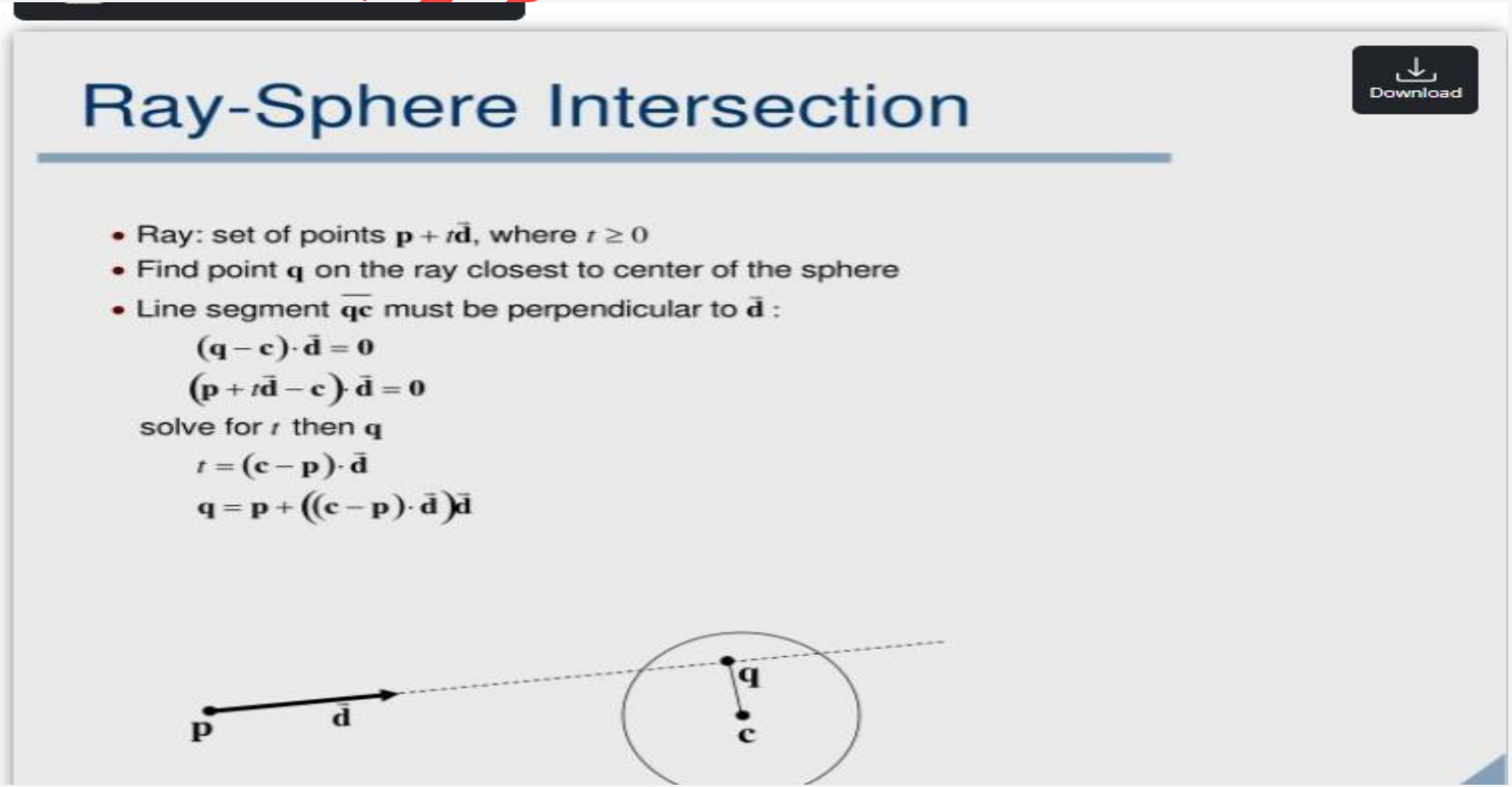
  $$t_1 = t - a \qquad t_2 = t + a$$

  $$a = \sqrt{r^2 - |\mathbf{q}-\mathbf{c}|^2}$$

- If $t_1 > 0$ then $\mathbf{q}_1$ is the first intersection point on the ray

  else if $t_2 > 0$ then the ray starts inside the sphere, $\mathbf{q}_2$ is the first (only) intersection point

  else the sphere is behind the ray, there's no intersection



Ray-Sphere Intersection

- Ray: set of points $\mathbf{p} + t\vec{\mathbf{d}}$, where $t \geq 0$
- Find point $\mathbf{q}$ on the ray closest to center of the sphere
- Line segment $\overline{\mathbf{qc}}$ must be perpendicular to $\vec{\mathbf{d}}$:

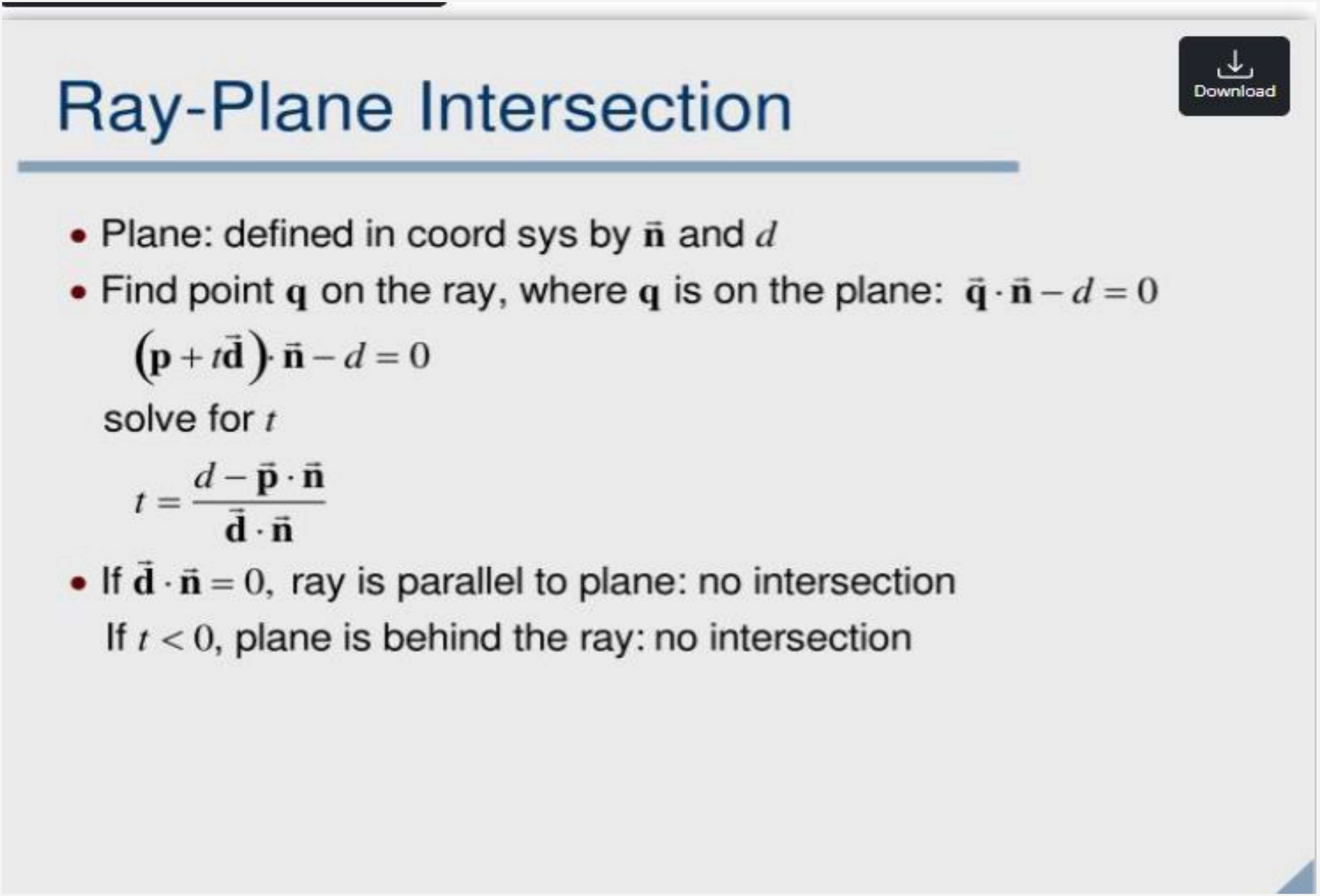
$$(\mathbf{q} - \mathbf{c}) \cdot \vec{\mathbf{d}} = 0$$

$$\left(\mathbf{p} + t\vec{\mathbf{d}} - \mathbf{c}\right) \cdot \vec{\mathbf{d}} = 0$$

solve for t then $\mathbf{q}$

$$t = (\mathbf{c} - \mathbf{p}) \cdot \vec{\mathbf{d}}$$

$$\mathbf{q} = \mathbf{p} + \left((\mathbf{c} - \mathbf{p}) \cdot \vec{\mathbf{d}}\right)\vec{\mathbf{d}}$$



Ray-Plane Intersection

- Plane: defined in coord sys by $\vec{\mathbf{n}}$ and d
- Find point $\mathbf{q}$ on the ray, where $\mathbf{q}$ is on the plane: $\vec{\mathbf{q}}\cdot\vec{\mathbf{n}} - d = 0$

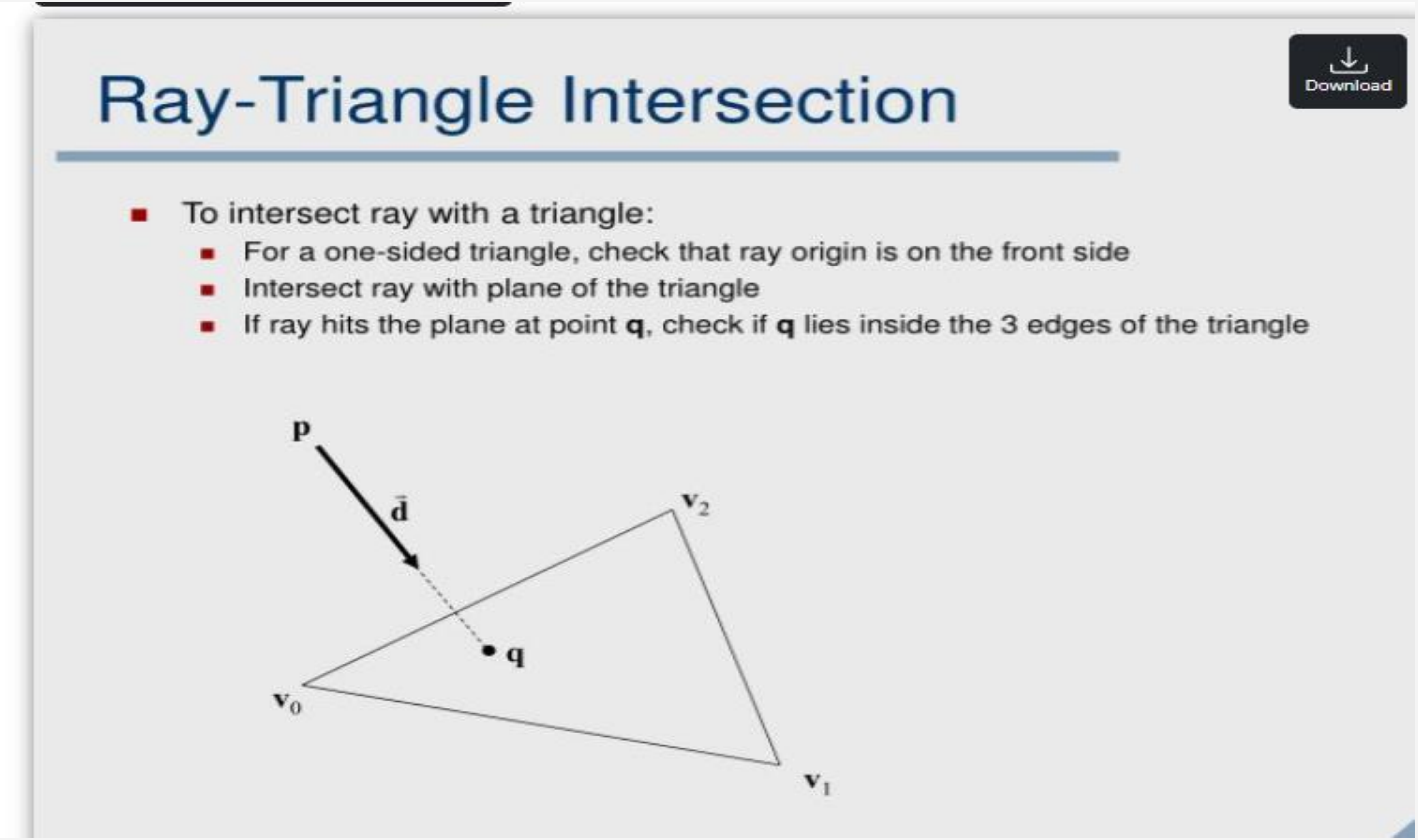
$$\left(\mathbf{p} + t\vec{\mathbf{d}}\right)\cdot\vec{\mathbf{n}} - d = 0$$

solve for t

$$t = \frac{d - \vec{\mathbf{p}}\cdot\vec{\mathbf{n}}}{\vec{\mathbf{d}}\cdot\vec{\mathbf{n}}}$$

- If $\vec{\mathbf{d}}\cdot\vec{\mathbf{n}} = 0,$ ray is parallel to plane: no intersection

  If $t < 0$, plane is behind the ray: no intersection



Ray-Triangle Intersection

- To intersect ray with a triangle:
  - For a one-sided triangle, check that ray origin is on the front side
  - Intersect ray with plane of the triangle
  - If ray hits the plane at point **q**, check if **q** lies inside the 3 edges of the triangle