

# Ray Tracing

## Ray Definition

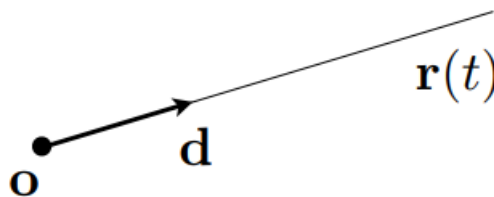
Ray is an ideal model for light transportation with the following assumptions:

- Ray is geometric:
  - no diffraction, no polarization, no interference
- Ray travels in a straight line in a vacuum
  - no atmospheric scattering
  - no gravity effects
- Discrete-wavelength approximation
  - RGB
  - quantized approximation of certain effects (dispersion, fluorescence)
- Superposition
  - no non-linear reflecting materials

## Parametric Form

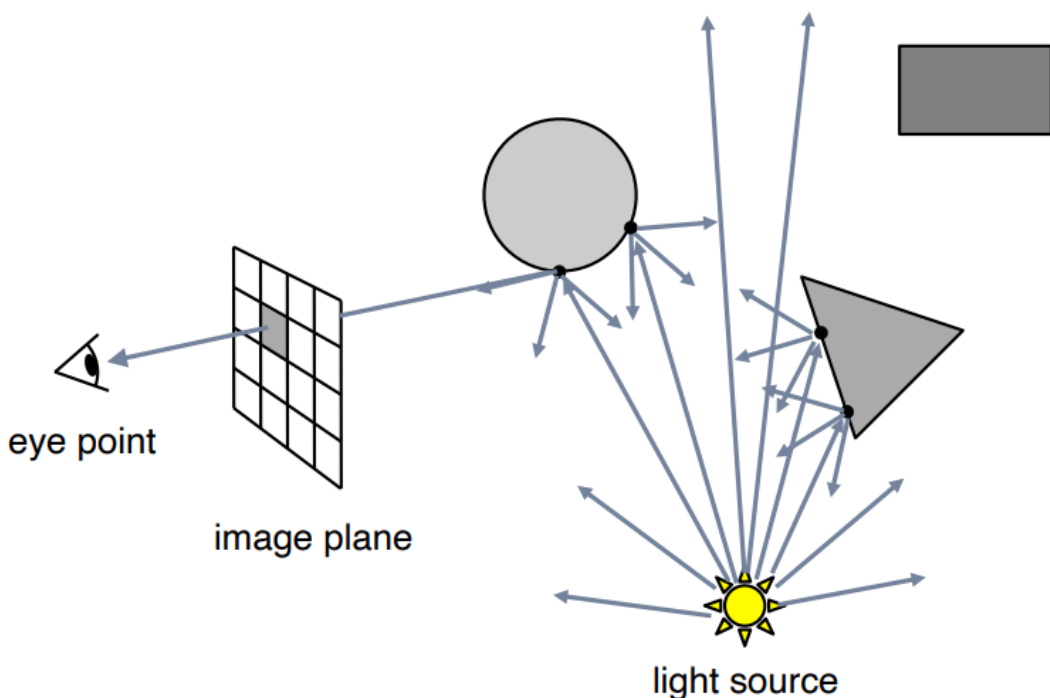
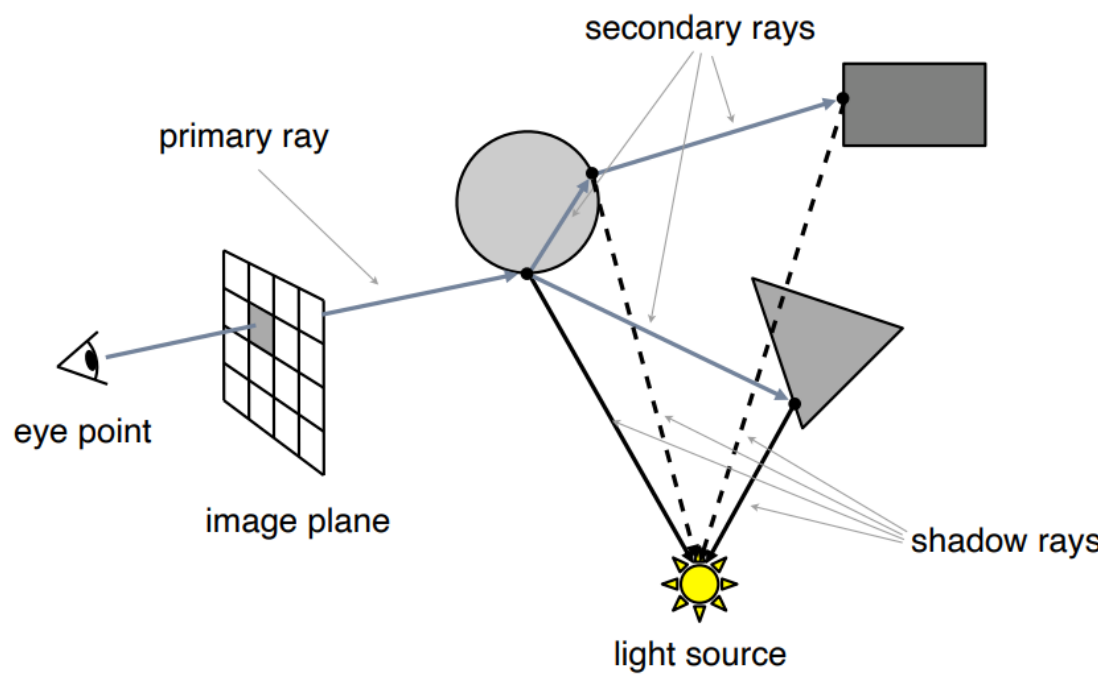
Ray can be expressed in parametric form

$$\underbrace{\mathbf{r}(t)}_{\text{ray}} = \underbrace{\mathbf{o}}_{\text{origin}} + t \underbrace{\mathbf{d}}_{\text{direction}}$$



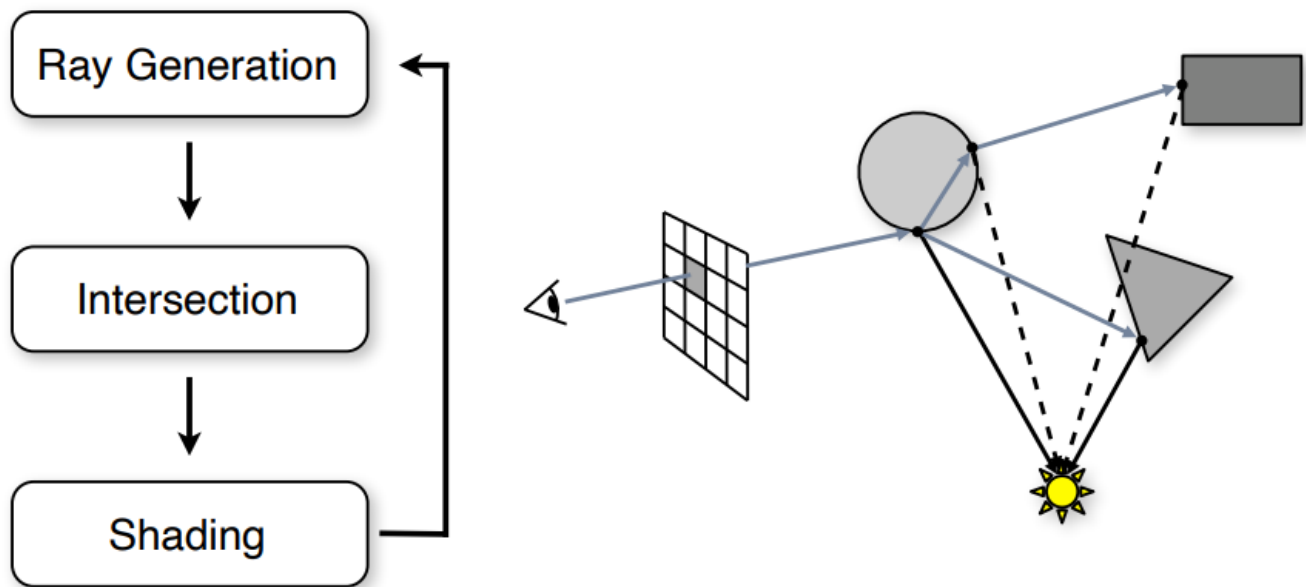
## General Theory

Type	Presentation

Type	Presentation
Light tracing	 <p>The diagram illustrates light tracing. A light source (a yellow sun) emits numerous rays in all directions. These rays hit various objects: a sphere, a triangle, and a rectangle. From each hit point, multiple secondary rays are emitted, creating a complex web of light paths. One ray from the sphere hits the image plane (a grid). Another ray from the triangle hits the image plane. A ray from the rectangle hits the image plane. Finally, a ray from the image plane hits the eye point (an eye icon). The labels 'eye point', 'image plane', and 'light source' are present.</p>
Camera tracing	 <p>The diagram illustrates camera tracing. It shows the same scene as the light tracing diagram. However, the focus is on the path of a single ray starting from the eye point, passing through the image plane, and hitting the sphere. From that point on the sphere, a 'primary ray' is shown. From the same point, 'secondary rays' are shown as dashed lines. One secondary ray hits the triangle, and another hits the rectangle. 'Shadow rays' are shown as dashed lines originating from the light source and hitting the objects. The labels 'eye point', 'image plane', 'light source', 'primary ray', 'secondary rays', and 'shadow rays' are present.</p>

We see the world because we see the light. To simulate the light transporting in the world especially those who arrive at the camera, we need an efficient way to model this. Instead of modeling all lights emitted from light sources, we only simulate those **can** arrive at our camera. **Because light path is inversive**, it absolutely legal to do that.

The general ray tracing pipeline is as follows



## Ray Generation

To project 3D objects onto a 2D image, we use [pinhole camera model](#) or other [Advanced Camera Models](#). A ray(or multiple rays) is generated through the image pixels.

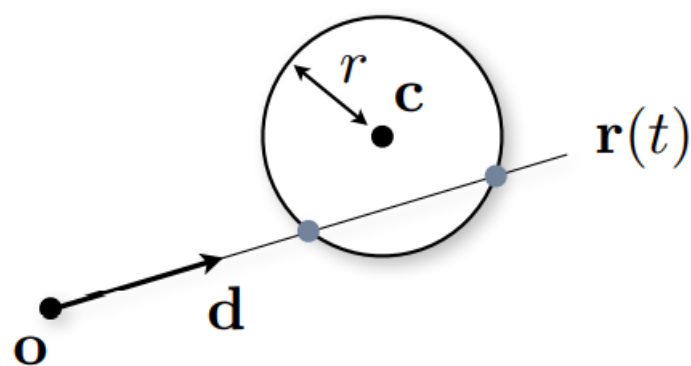
## Intersection

The next step we need to decide whether the ray intersects with other objects. We distinguish two cases: intersections with analytical shapes and intersections with meshes.

### Intersection with Analytical Shapes

Analytical shapes are defined by implicit surface. Usually we can derive the intersections from solving equations.

#### Spheres



The implicit sphere equations is

$$\|\mathbf{x} - \underbrace{\mathbf{c}}_{\text{center}}\|^2 = \underbrace{r^2}_{\text{radius}}$$

Plug in the ray definition

$$\|\mathbf{o} + t\mathbf{d} - \mathbf{c}\|^2 - r^2 = 0$$

which reduces to a quadratic equation on  $t$

$$At^2 + Bt + C = 0$$

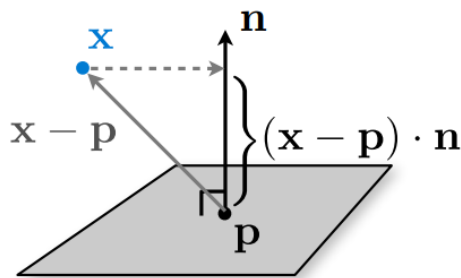
where

$$\begin{aligned} A &= \|\mathbf{d}\|^2 \\ B &= 2(\mathbf{o} - \mathbf{c}) \cdot \mathbf{d}^\top \\ C &= (\mathbf{o} - \mathbf{c})^2 - r^2 \\ \Delta &= B^2 - 4AC \end{aligned}$$

We have the following cases

1.  $\Delta < 0$ : there's no intersection
2.  $\Delta > 0$ : there exist intersections, solve for  $t$ .  $t = t_1$  and  $t = t_2$  with  $t_1 < t_2$ :
  1.  $t_1 \in [t_{min}, t_{max}]$ : intersection point at  $t_1$
  2.  $t_1 \notin [t_{min}, t_{max}]$  and  $t_2 \in [t_{min}, t_{max}]$ : intersection point at  $t_2$
  3.  $t_1 \notin [t_{min}, t_{max}]$  and  $t_2 \notin [t_{min}, t_{max}]$ : no intersection

## Planes



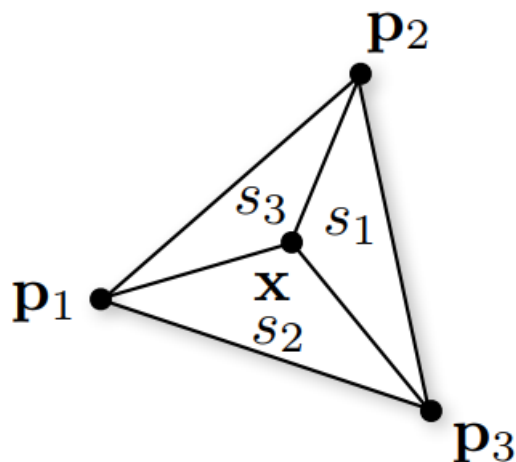
Implicit plane equation:

$$ax + by + cz + d = 0 \iff (\mathbf{x} - \underbrace{\mathbf{p}}_{\text{point on plane}}) \cdot \underbrace{\mathbf{n}}_{\text{plane normal}} = 0$$

Plug in ray definition we have

$$t = -\frac{(\mathbf{o} - \mathbf{p}) \cdot \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}}$$

## Triangles



Barycentric coordinates:

$$\mathbf{x} = s_1\mathbf{p}_1 + s_2\mathbf{p}_2 + s_3\mathbf{p}_3$$

Point  $\mathbf{x}$  is inside the triangle if

$$s_1 + s_2 + s_3 = 1, \quad 0 \leq s_i \leq 1$$

Two way to calculate intersection:

1. Plug in the ray definition

$$\mathbf{o} + t\mathbf{d} = s_1\mathbf{p}_1 + s_2\mathbf{p}_2 + s_3\mathbf{p}_3$$

2. Intersection with plane and test if inside the triangle

## More

Find guidance on intersection on [Object/Object Intersection \(realtimerendering.com\)](http://realtimerendering.com)

## Shading

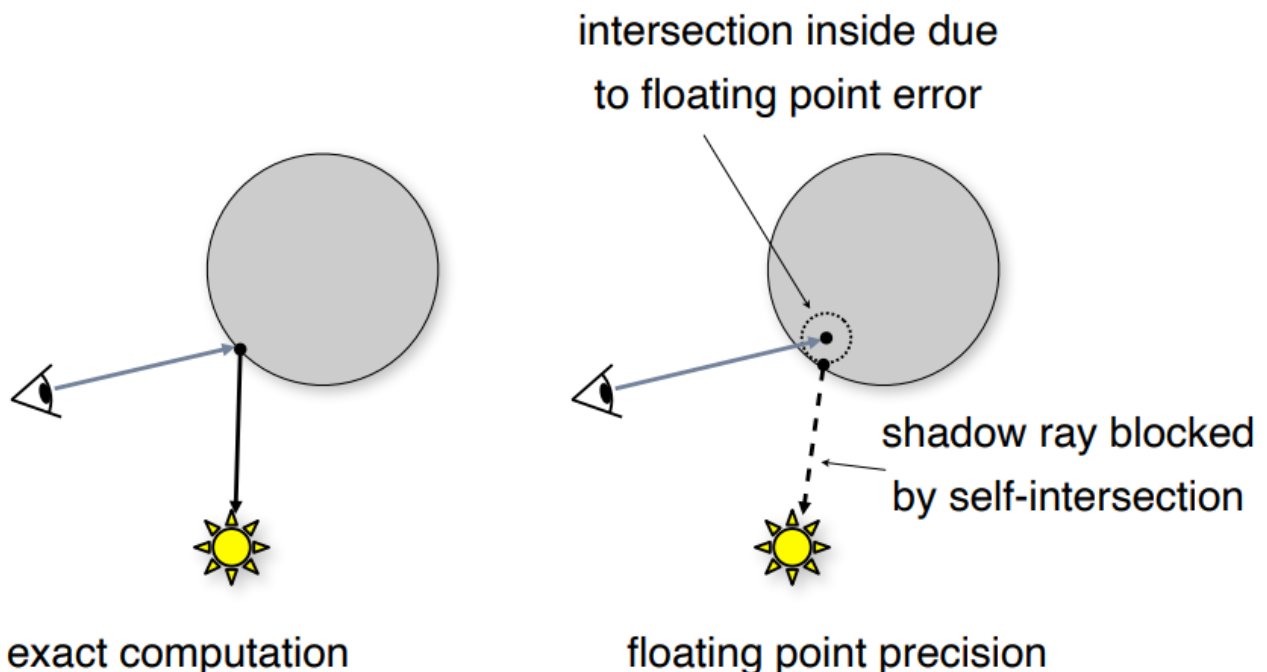
Shading is the process of determining color of primary ray (and thus corresponding pixel color) depending on hit point.

Once an intersection is found, we need to evaluate the light-material interaction, i.e. what's explained in [Appearance Modeling](#). Different shading schemes will be discussed later, for example, [Direct Illumination](#) and [Path Tracing](#). The main problem is to determine the integral

$$L_r(\mathbf{x}, \mathbf{w}_r) = \int_{H^2} f_r(\mathbf{x}, \mathbf{w}_i, \mathbf{w}_r) L_i(\mathbf{x}, \mathbf{w}_i) \cos \theta_i d\mathbf{w}_i$$

which will be illustrated in [Monte Carlo Integration](#).

## Precision Issues



To solve the precision issues in intersection, we can add an intersection tolerance  $\epsilon = 1e - 6$  to  $t_{\min}$  of the ray for example.