Table of Contents

Principles of Ray Tracing

Ray tracing (and its many derivative methods) is probably the most powerful way of rendering realistic images. The technique consists of tracing a ray from the origin of the viewing coordinates through a pixel, and extend this ray into the scene to determine what object(s) intersect with it, and ultimately compute the color of the pixel. A scene is generated when all pixels within it are ray-traced. Ray tracing provides a natural solution to the hidden surface problem, since the first intersection of the ray with an object locates the visible surface for a pixel. It also provides elegant solutions to casting shadows by way of feeler (or shadowing rays).

The same pipeline of transformation matrices is used with ray tracing. The main difference is the way by which the image is formed. If θ is the viewing angle of the camera and A is the aspect ratio of the near plane (the rendering window should have the same aspect ratio), then the width and the height of the near plane $(2W, 2H)$ (in viewing coordinates, not pixel coordinates) are given by:

$$H = N \tan\left(\frac{\pi}{180}\frac{\theta}{2}\right)$$

and

$$W = HA$$

If we define a pixel at row r and column c expressed as $(u_c, v_r)^T$, we can then find its coordinates on the near plane as

$$(u_c, v_r)^T = \left(-W + W\frac{2c}{n}, -H + H\frac{2r}{m}\right)^T$$

where n and m are the number of columns and rows of pixels in the rendering window, respectively. The 3D point corresponding to this pixel on the near plane in viewing coordinates is written as $(u_c, v_r, -N)^T$. In world coordinates, this point turns out to be:

$$\begin{bmatrix} \vec{u} & \vec{v} & \vec{n} & \vec{e} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_c \\ v_c \\ -N \\ 1 \end{bmatrix} = u_c \vec{u} + v_r \vec{v} - N \vec{n} + \vec{e}$$

where $\vec{e}$ is the origin of the viewing coordinate system expressed in world coordinates (note that technically this is a point and using the vector notation for it is a bit of an abuse).

We are now in a position to parametrize a ray from the origin of the viewing coordinate system passing through pixel $(u_c, v_c)^T$ as

$$\vec{r}(t) = \vec{e}(1-t) + (\vec{e} - N\vec{n} + u_c\vec{u} + v_r\vec{v})t$$

Simple manipulation of this expression allows us to write

$$r(t) = \vec{e} + \vec{d}\,t$$

where

$$\vec{d} = -N\vec{n} + W\left(\frac{2c}{n} - 1\right)\vec{u} + H\left(\frac{2r}{m} - 1\right)\vec{v}$$

Ray Tracing Process

The steps generally involved in rendering a scene using a ray tracing technique are the following:

1. Define objects and light sources within the scene
2. Set up the viewing coordinate system (the synthetic camera)
3. For each pixel, set up a ray $\vec{r}(t)$
4. Find the intersections of this ray with objects in the scene
5. Identify the intersection point that is closest to the viewer (it is the smallest value of t for which an intersection is found)
6. Determine color and intensity of light along the ray from the intersection point
7. If the object is opaque, consider this object only. Otherwise, continue refining the computation of light color and intensity with further ray-object intersections.

If we create an instance of a generic object and place it where we want it in the scene, then there exists a transformation matrix M that performs this operation:

$$\vec{q} = M\vec{q}\,'$$

where $\vec{q}$ is the generic object transformed as we desire it to appear in the scene. At this point, we need to find out if the ray for a given pixel intersects this object. An efficient way of performing this computation is to keep the object in its generic form $\vec{q}\,'$ and apply the inverse

transformation M^{-1} to the ray $\vec{r}(t)$ instead. The transformed ray can thus be written in the following way:

$$M^{-1}\vec{r}(t)=M^{-1}(\vec{e}+\vec{d}\,t)=M^{-1}\vec{e}+M^{-1}\vec{d}\,t$$

Conveniently, the values for t at intersections are the same in both spaces.

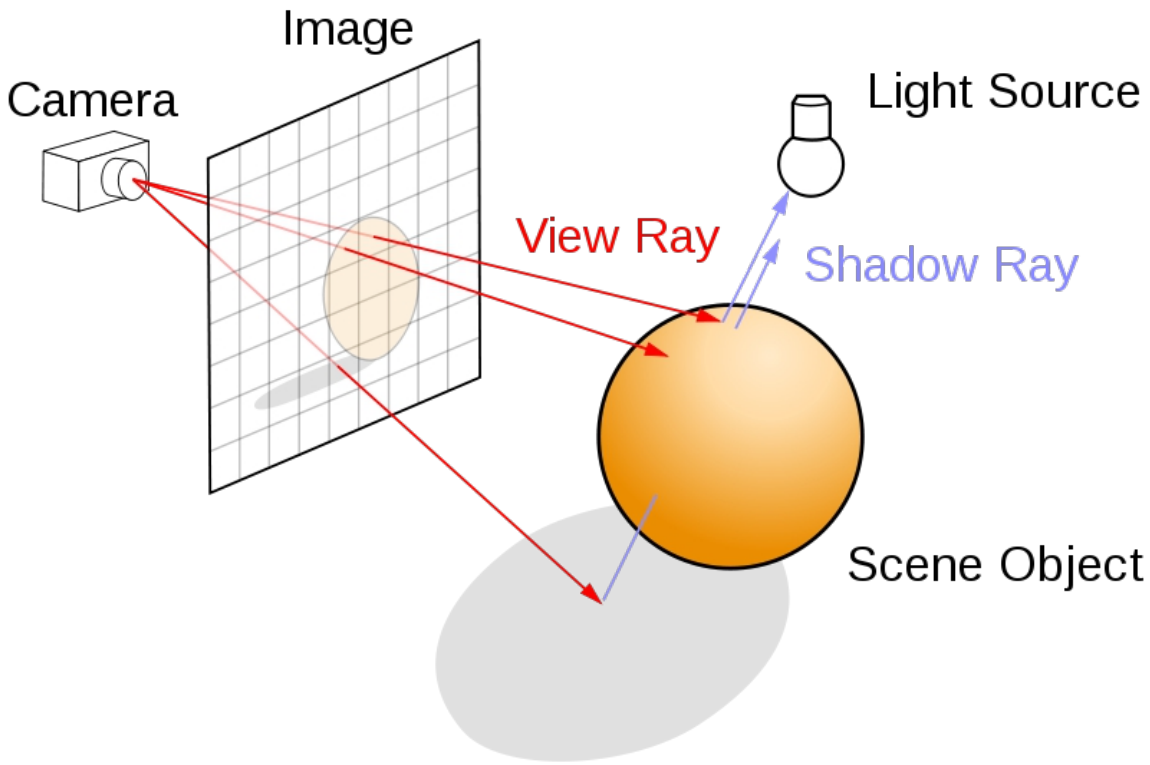


Illustration 1: The principles behind ray-tracing

Generic Objects

Scaled, rotated, and translated generic objects are very often used with ray tracing. We examine some of these objects and the techniques to compute ray intersections with them. In what follows, an assumption is made that the ray $\vec{r}(t)$ is transformed into the generic object space with matrix M^{-1}.

Generic Sphere

The implicit equation of the generic sphere (centered at the origin, with radius equal to 1) is written as $x^2+y^2+z^2-1=0$. Any point p on the sphere satisfies $\|p\|^2=1$. If $\vec{r}(t)$ is on the surface of the sphere (for a given t), then $\|\vec{r}(t)\|^2=\|\vec{e}+\vec{d}\,t\|^2=1$. Because $\|\vec{a}+\vec{b}\|^2=\|\vec{a}\|^2+2\vec{a}\cdot\vec{b}+\|\vec{b}\|^2$, we can write:

$$\|\vec{e}+\vec{d}\,t\|^2-1 \;=\; \|\vec{e}\|^2+2\vec{e}\cdot\vec{d}\,t+\|\vec{d}\|^2t^2-1 \;=\; 0$$

By posing $a=\|\vec{d}\|^2$, $b=\vec{e}\cdot\vec{d}$, and $c=\|\vec{e}\|^2-1$, we can rewrite the equation as a quadratic:

$$at^2+2bt+c=0$$

and solve for t as:

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

$b^2-ac<0$	The ray misses the sphere
$b^2-ac=0$	The ray grazes the sphere at t
$b^2-ac>0$	The ray intersects the sphere when entering it and when leaving it (2 solutions)

Generic Plane

The equation of the generic plane is given by $z=0$. The intersection of the ray with the generic plane is straightforwardly obtained with setting the ray equation with $z=0$ as a constraint, which implies $e_z+d_z t=0$, and in turn, $e_z=-d_z t$, yielding

$$t=-\frac{e_z}{d_z}$$

If $d_z=0$ then the ray is parallel to the plane and does not intersect it.

Generic Cylinder

The implicit equation for a generic cylinder of infinite extent in z is given by $x^2+y^2-1=0$, for $z\in[-\infty,\infty]$. The ray intersects the cylinder if

$$(e_x+d_x t)^2+(e_y+d_y t)^2-1=0$$

which is quadratic in t . Posing $a=d_x^2+d_y^2$, $b=e_x d_x+e_y d_y$, and $c=e_x^2+e_y^2-1$, yields

$$at^2+2bt+c=0$$

which we solve in a manner identical to the intersection of the ray and the generic sphere. Given an existing intersection, we find z as $e_z+d_z t$, and verify if $z\in[-1,1]$. To find out if the ray intersects with the generic cylinder caps at $z=-1$ and $z=1$, we intersect the ray with these two planes, and verify if the solution is within the circle (of radius equal to 1) defined by the cap on the respective planes.

Generic Cone

The implicit equation of the generic implicit cone of infinite wall extent is given by:

$$x^2+y^2-(1-z)^2=0$$

To find out if the ray intersects the cone, we need to substitute $\vec{r}(t)=\vec{e}+\vec{d}\,t$ into the implicit equation. Since $x=e_x+d_x t$ and $y=e_y+d_y t$, we form the quadratic $at^2+2bt+c=0$, where

$$a=d_x^2+d_y^2-d_z^2$$

$$b=e_x d_x+e_y d_y+d_z(1-e_z)$$

$$c=e_x^2+e_y^2-(1-e_z)^2$$

Once t is obtained, we find $z=e_z+d_z t$ and if $z\in[-1,1]$, then there is an intersection between the ray and the cone wall. We also need to verify if the ray intersects with the base of the cone which is contained within the plane $z=-1$, as before. The base of the cone is a generic circle (of radius equal to one).

Convex Polyhedra

Polyhedra are composed of facets that are planar. Given a normal $\vec{n}$ the equation of a plane is $\vec{n}\cdot P=d$, where P is a point on the plane and d is a constant. As usual, the normal vector is made to point towards the outside of the object, in this case, a convex polyhedra. We need to determine if ray $\vec{r}(t)$ intersects with one of the facets. If the ray is not parallel to the facet, then we can write:

$$\vec{n}\cdot(\vec{e}+\vec{d}\,t)=d$$

and solve for t as: $t=d-\dfrac{\vec{n}\cdot\vec{e}}{\vec{n}\cdot\vec{d}}$.

$\vec{n}\cdot\vec{d}<0$	Ray from outside to inside
$\vec{n}\cdot\vec{d}>0$	Ray from inside to outside
$\vec{n}\cdot\vec{d}=0$ and $d-\vec{n}\cdot\vec{e}>0$	Ray totally outside
$\vec{n}\cdot\vec{d}=0$ and $d-\vec{n}\cdot\vec{e}<0$	Ray totally inside

If $\vec{n}\cdot\vec{d}=0$ then $\vec{n}$ and $\vec{d}$ are perpendicular, which implies that the ray is parallel to the plane. For any point Q, $Q\cdot\vec{n}<d$ implies that the point is inside, while $Q\cdot\vec{n}>d$ implies

that it is outside. If $\vec{n}\cdot\vec{d}\neq 0$, then the extended surface and the ray must intersect. If $\vec{n}\cdot\vec{d}>0$ then the angle between the two vectors is smaller than 90 degrees, and this ray passes from the inside to the outside, and conversely if $\vec{n}\cdot\vec{d}<0$.

Computational Concerns

The amount of computation that a ray-tracer must effect is often times staggering, and prevents its use for real-time applications in general. Techniques have been developed to reduce the computational burden. Below is a description of some of the simplest ones:

1. Ray tracers use bounding boxes to simplify the computation of intersections with generic objects. Also, projection extents are used for the same purpose. These extents are rectangles enclosing all the image points of a particular object.
2. Space sub-division techniques are also used. They consist of enclosing the scene in a cube and to proceed sub-dividing this cube into smaller ones until each one contains only a pre-set number of surfaces, while eliminating the cubes which are void of surfaces to render.

Shading

For each ray that intersects an object, we must find the reflected light at the hit point. Hence we need to know the intersection point, and the surface normal at that point. Let's define the following vectors:

1. $\vec{s}$: a vector from the intersection point of the ray and the surface to the light source
2. $\vec{v}$: a vector to the center of projection of the viewing coordinates
3. $\vec{n}$: surface normal at ray-surface intersection point
4. $\vec{r}$: direction of specular (perfect) reflection

The intensity observed at the ray-surface intersection point is composed of ambient, diffuse, and specular components, which we can mathematically express as:

1. $I_a=\rho_a I_m$
2. $I_d=\rho_d I_p(\vec{s}\cdot\vec{n})$
3. $I_s=\rho_s I_p(\vec{r}\cdot\vec{v})^f$

where I_m is the ambient light intensity and I_p is the point light source intensity. ρ_a , ρ_d and ρ_s are coefficients with $\rho_a=\rho_d$, usually. Note that $\rho_a+\rho_d+\rho_s=1$.

With these elements, we can define the shading model as:

$$I=I_a+I_p(\rho_d(\vec{n}\cdot\vec{s})+\rho_s(\vec{v}\cdot\vec{r})^f)$$

Since light intensity falls with the square of the distance,

$$\frac{I_p}{(a_0 + a_1 d + a_2 d^2)}$$

can be used to model depth effects, where d is depth. In addition, in the presence of multiple light sources, only two additional components need to be calculated: I_d and I_s. These equations are defined for a single channel (monochrome). They immediately generalize to the RGB system of colors as one equation per color channel:

$$I_r = \rho_{ra} I_{rm} + I_{rp}(\rho_{rd}(\vec{n} \cdot \vec{s}) + \rho_{rs}(\vec{v} \cdot \vec{r})^f)$$

$$I_g = \rho_{ga} I_{gm} + I_{gp}(\rho_{gd}(\vec{n} \cdot \vec{s}) + \rho_{gs}(\vec{v} \cdot \vec{r})^f)$$

$$I_b = \rho_{ba} I_{bm} + I_{bp}(\rho_{bd}(\vec{n} \cdot \vec{s}) + \rho_{bs}(\vec{v} \cdot \vec{r})^f)$$

where $\vec{I}_m = (I_{rm}, I_{gm}, I_{bm})$ is the ambient light intensity per color channel, $\vec{\rho}_a = (\rho_{ra}, \rho_{ga}, \rho_{ba})$ are coefficients expressing how the object reflects ambient light, $\vec{I}_p = (I_{rp}, I_{gp}, I_{bp})$ are the intensities of the light source per color channel, $\vec{\rho}_d = (\rho_{rd}, \rho_{gd}, \rho_{bd})$ are the diffuse reflection coefficients of the object per color channel, and $\vec{\rho}_s = (\rho_{rs}, \rho_{gs}, \rho_{bs})$ are the specular reflection coefficients of the object per color channel. Note that the sum of the elements of $\vec{\rho}_a$, $\vec{\rho}_d$, and $\vec{\rho}_s$ must sum to unity for each vector. Additionally, vectors $\vec{n}$, $\vec{s}$, $\vec{v}$, $\vec{r}$ must have unit length for the shading equations to work correctly.

Surface Normals at Intersections

Given a point P on a generic sphere, the normal to the sphere at this point is simply the point itself, considered as a vector.

For planes, the generic equation is $z = 0$ and the normal is given by $\vec{n} = (0,0,1)$. In the case of a generic cylinder, if the intersection point is on the wall, the normal is computed as:

$$\vec{n} = (x, y, 0)$$

If the intersection is on the base of the cylinder, then $\vec{n} = (0,0,-1)$. Conversely, if it is on the top, the normal is then $\vec{n} = (0,0,1)$.

For generic cones, if the intersection is on the wall, the normal is

$$\vec{n} = (x, y, 1 - z)$$

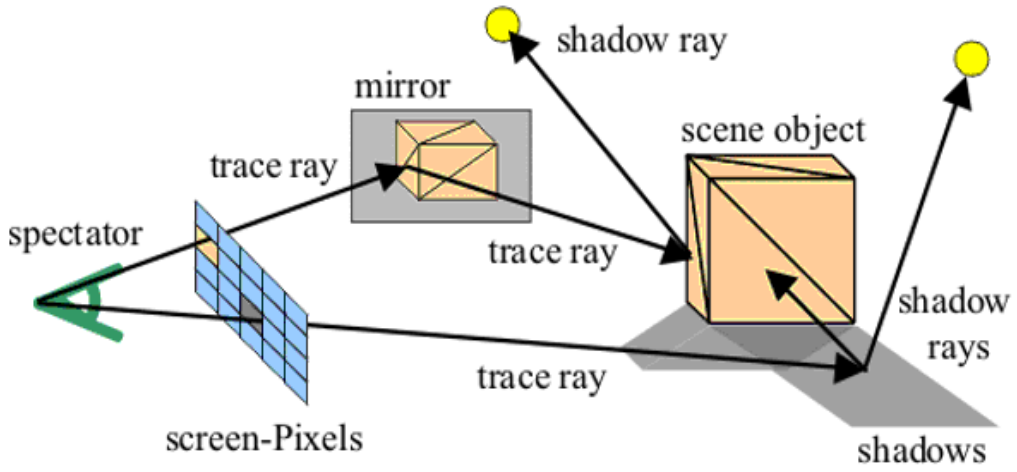
If the intersection is on the base, then $\vec{n} = (0,0,-1)$.

Shadowing

Suppose we have a light source at L and an intersection point on an object $\vec{r}(t_0)$. If other objects are found between the light source and the intersection point, then this point should not receive any incident light from the source and be lit with ambient light, or other models to that effect. Let's trace a secondary ray, from the intersection point and the light source as:

$$\vec{S}(u)=\vec{r}(t_0)+(L-\vec{r}(t_0))u$$

Illustration 2: Shadow rays



If this secondary ray intersects and opaque object with $u\in]0,1]$ then the point $\vec{r}(t_0)$ is in shadow of the light source. Assuming a single light source, the intensity received at the point should be ambient only. As we can see, ray tracing naturally solves the shadowing problem on a per rendered pixel basis.

Reflection and Refraction

Typically, light is reflected by many surfaces before it reaches the observer. Additionally, light may be refracted (or partially transmitted) by translucent objects or surfaces. The light that reaches the observer may have up to five components, if the surface is mirror-like, translucent, or both. The first three components are local, and they correspond to ambient, diffuse, and specular light at the point. However, two more non-local components can be added to these. Namely, they are the reflected and refracted components coming from other objects in the environment, leading to this general equation for the light intensity received by a pixel in the viewport: $I=I_a+I_d+I_s+I_l+I_r$. The last two components are the contributions from reflected and refracted light, respectively.

The component I_l is from reflected light that is incident at $\vec{r}(t_0)$, along direction $-\vec{l}$ where $\vec{l}=\vec{r}(t_0)-2(\vec{r}(t_0)\cdot\vec{n})\vec{n}$. The component I_r is light that is bent by another, translucent object, and reaches $\vec{r}(t_0)$. Light that is refracted changes direction as it passes through translucent material. This change in direction is dependent on the density of the media that

light crosses. For instance, if c_1 and c_2 are the respective speeds of light in two different materials, then light gets refracted according to Snell's law:

$$\frac{\sin\theta_1}{c_1} = \frac{\sin\theta_2}{c_2}$$

with the index of refraction defined as:

$$n_{21} = \frac{c_1}{c_2}$$

Interestingly, there is a critical angle beyond which the totality of incident light gets reflected (no refraction), and is given by:

$$\frac{c_1}{c_2}\sin\theta_1 = 1$$

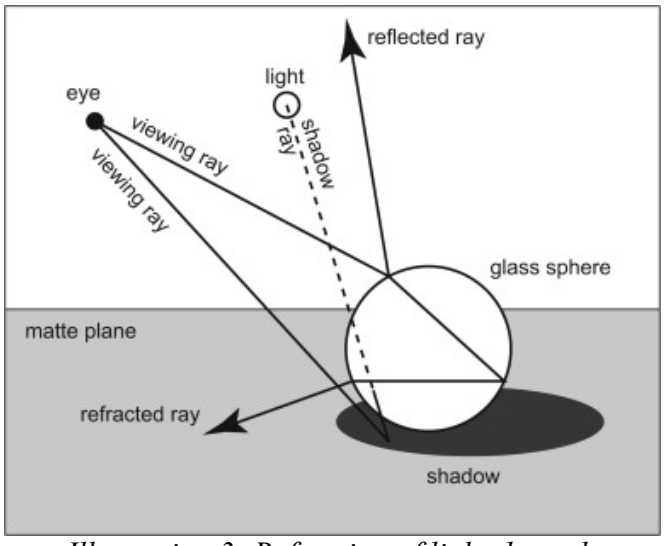


Illustration 3: Refraction of light through translucent material

Medium 1	Medium 2	Refraction Index
Air	Water	1.33
Air	Diamond	2.42
Air	Glass	1.5 to 1.9

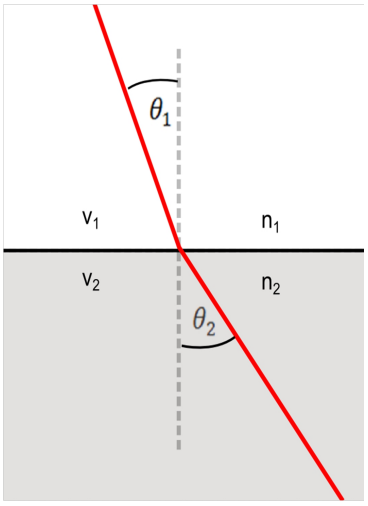


Illustration 4: Snell's law of light refraction

Lighting Equations

The lighting equations for ray tracing include three local terms and both the reflection and refraction terms:

$$I = I_a + I_d + I_s + I_l + I_r$$

At the intersection $\vec{r}(t_0)$ of the ray with an object, the local components are obtained directly. The reflection and refraction terms need to be evaluated by considering the specular reflection direction and its intersection with a scene object, which contributes to the reflected light component. If the material at $\vec{r}(t_0)$ is translucent, then there is a ray penetrating the material at that point and it is refracted according to Snell's law. This ray and its intersection with other objects contribute the refracted light component.

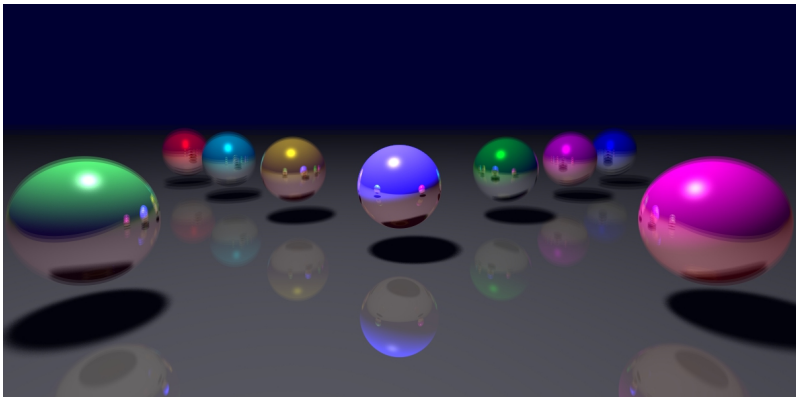


Illustration 5: An example of a ray-traced image