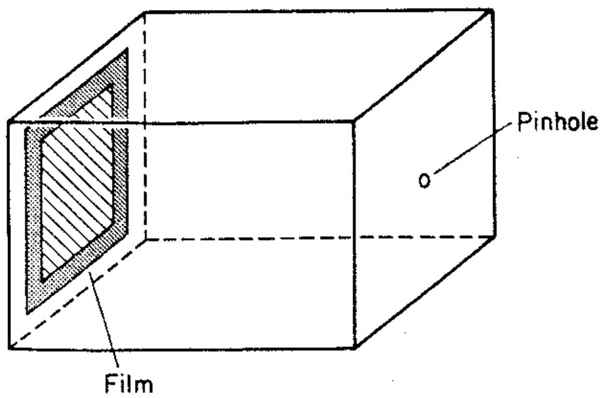
RAYTRACING

. introduction

- objective: to simulate how a camera records a physical scene onto film

- the pinhole camera model



. a flat piece of photographic film is placed at the back of a light-tight box

. a single pinhole is made in the front of the box and is covered with an piece of opaque tape

- **the pinhole camera model, cont.**

  - **a photo is made by holding the camera steady and removing the tape for a while**

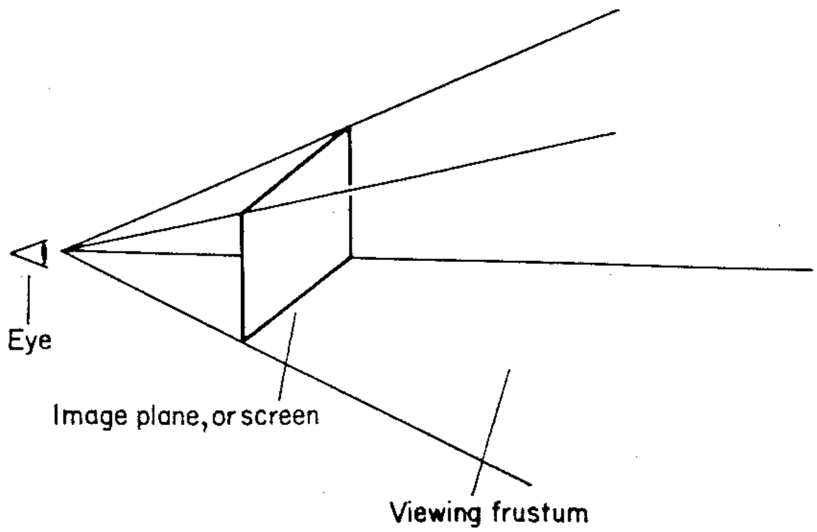
    - **light passing through the pinhole and striking the film causes a chemical change in the emulsion**

    - **each point on the film can receive light only along the line joining the piece of film and the pinhole**

    - **the larger the pinhole, the brighter and blurrier the image**

  - **it actually works!**

- **computer graphics version of the pinhole camera**

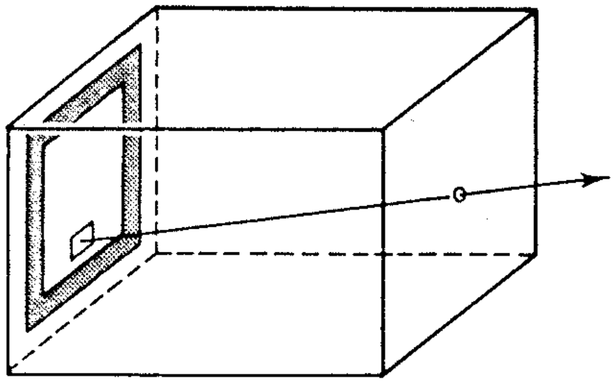


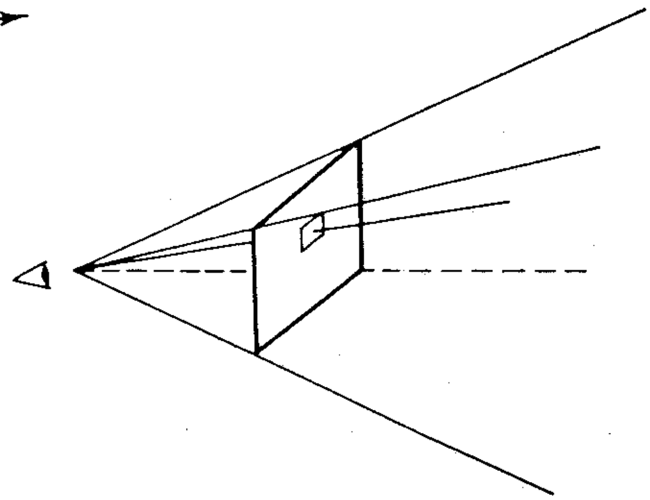
  - **the film is placed in front of the pinhole**
  - **the pinhole is renamed the "eye"**

- pixels and rays

  - each pixel may be thought of as an independent window onto the scene

  - each pixel may be associated with a region of film



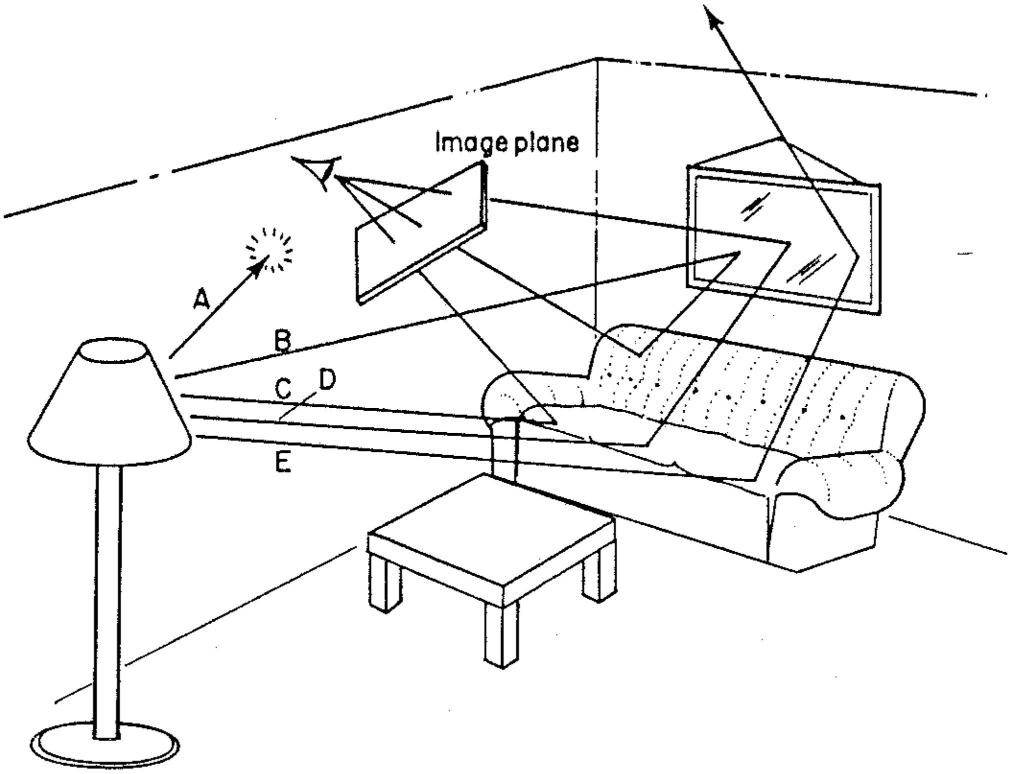


  - light rays arrive from the scene, pass through the pinhole and strike the film

  - all these must be represented by a single color

  - good first approximation: average all the colors striking the film

- **tracing rays**



- **forward ray tracing: consider a living room with a viewer, a light source, and various furnishings**
  - **photon A strikes the wall and is absorbed, so it doesn't contribute to the picture**
  - **photons B, C and D strike furnishings and reach the film**
  - **photon E strikes furnishings, but does not reach the film**
  - **photons get a bit dimmer with each bounce**

- tracing rays, cont.
  - forward ray tracing and backward ray tracing
    - to follow all photons would be an overwhelming task
    - also, too many photons play no role at all!
    - ask instead, "Which photons contribute to the image (by striking the image plane and passing into the eye)?"
      - the ray emanates from the eye, passes through the image plane and continues until it strikes one or more objects and a light source

- tracing rays, cont.
    - ray combination
        - the color of a ray is determined by the all the light that contributed to it
        - four classes of light rays
            - **pixel rays** or **eye rays** which carry light through a pixel to the eye
            - **illumination rays** or **shadow rays** which carry light from a light source directly to an object surface
            - **reflection rays** which carry light reflected by an object and
            - **transparency rays** which carry light passing through an object
        - two classes of light
            - **incoming light, incident light,** or **illumination**
            - **outgoing light** or **radiated light,** leaving as a reflected ray, a transparency ray or a pixel ray

- tracing rays, cont.

  - shadow and illumination rays

    - if a light source can be "seen" from a point on a surface, light from that source strikes the surface at that point

    - a shadow ray "feels around" for shadows; sometimes it is called a **shadow feeler**

      - if it reaches the light source without interruption, it becomes an **illumination ray** (which is thought of as travelling from the light source)

- tracing rays, cont.
  - propagated light
    - illuminating light that leaves a surface in a direction that matters
    - of the four mechanisms for light transport, two are considered now
      - specular reflection
      - specular transmission
    - for a perfectly flat, shiny surface, light can arrive in only one direction and still be reflected into the eye
  - transparency rays
    - again, incident light can be transmitted in a single direction, but it is bent or refracted as it passes from one medium to the other
  - surface physics (determining how light behaves at a surface)
    - techniques for determining reflection or transparency can be simple or complicated
      - fortunately, simple models seem to work fairly well

- recursive visibility
    - the color of radiated light is a function of
        - combined light from the light sources
        - light the object reflects
        - light the object transmits
    - the color of combined light from the light sources is determined in the same manner, suggesting a recursive algorithm

        step 1: begin with a ray that starts at the eye (the eye ray or **pixel ray**)

        step 2: determine what object or light source this ray hits (done over and over again, splitting rays whenever a partially transparent surface is struck and terminating when the ray leaves the scene)

        - a **ray tree** shows the whole process in schematic form
        - recursion stops when
            - a ray leaves the scene
            - its contribution becomes too small

- aliasing
    - among the many differences between synthesizing a computer image and exposing a piece of film is the fundamental fact that a digital computer cannot represent a continuous signal
    - spatial aliasing
        - due to the uniform nature of the pixel grid
        - no matter how many rays are used or how closely they are packed, small objects or larger, far away objects can be missed all together
            - even though they are small, they still matter
    - temporal aliasing
        - examples
            - (even excellent) "stills" of a spoked wheel, shown one at a time to produce an animated sequence, can give the impression that the wheel is spinning backwards
            - when a very small object moves across the screen, it sometimes is hit by a ray and sometimes is missed
            - a horizontal edge moving up the screen and "jerking" from scan-line to scan-line
        - solution
            - create stills that look blurry when things are moving fast (i.e. include **motion blur**)

- antialiasing solutions
    - supersampling
        - use lots of rays per pixel and average them
        - reduces rather than solves the problem
        - very expensive
    - adaptive supersampling
        - start with five rays (one for each corner and one for the center)
            - if they are all nearly the same color, a single object probably covers the pixel
            - if they are of sufficiently different colors, subdivide the pixel and process each subdivision with five rays
            - easy, not too slow, and works fairly well
            - however, little objects can slip between the five initial rays

- antialiasing solutions, cont.
    - stochastic ray tracing (or distributed ray tracing)
        - send out a fixed number of rays in random directions, still covering the pixel fairly evenly
        - elimination of a regular grid eliminates regular aliasing artifacts
        - bias the randomness to direct rays where lots of light is arriving rather than where light is sparse
        - supports motion blur, depth of field and soft edges on shadows (the penumbra region)
        - new problem, however: noise
            - error doesn't correlate to a regular grid
            - instead it spreads out
            - the human visual system is very forgiving of this form of noise
    - statistical supersampling
        - statistical analysis helps determine if "enough" rays are being sent
        - look at the colors of the rays
        - perform some statistical tests
        - "good enough" might mean that we are 90% confident the color is correct

- essential raytracing algorithms
  - the heart of a ray tracing package is the set of routines to find the intersection point of a ray and an object
  - problems to be addressed
    - for a ray spawned from the eye
      - the closest intersection point must be found
      - the surface normal at this point must be determined
    - for a ray sent towards the light
      - if there is an intersection point closer than the light, it must be determined
    - for a ray tested against a bounding volume
      - a simple hit/no hit determination may suffice
  - additional useful information
    - the intersection point's location relative to some reference frame for the surface (used in texture mapping)
  - mathematically elegant solutions sometimes make for slow algorithms
    - efficient algorithms are more important than elegant solutions
  - so that simple algebra can be used, the discussion is limited to quadric surfaces
    - planes and spheres are of particular interest

- **ray/sphere intersection and mapping**
  - **spheres commonly are used in ray tracing**
  - **testing for intersection is easy**
  - **algebraic solution**
    - **ray definitions**

$$\mathbf{R}_{\text{origin}} \equiv \mathbf{R}_0 \equiv [X_0 \ Y_0 \ Z_0]$$
$$\mathbf{R}_{\text{direction}} \equiv \mathbf{R}_d \equiv [X_d \ Y_d \ Z_d]$$
where $X_d^2 + Y_d^2 + Z_d^2 = 1$ (i.e. normalized)

which define a ray as

set of points on line $\mathbf{R}(t) = \mathbf{R}_0 + \mathbf{R}_d * t$, where $t > 0$.

- **restricting t > 0 avoids problems discussed later**

- **algebraic solution, cont.**

  - **sphere definitions**

    Sphere's center $\equiv \mathbf{S_c} \equiv [X_c\ Y_c\ Z_c]$
    Sphere's radius $\equiv S_r$
    Sphere's surface is the set of points $[X_s\ Y_s\ Z_s]$
    where $(X_s - X_c)^2 + (Y_s - Y_c)^2 + (Z_s - Z_c)^2 = S_r^2$.

  - **solving the intersection problem: substitute the ray equation into the sphere equation and solve for t**

    - **express the ray equation as a set of equations for the set of points [X Y Z] in terms of t**

$$X = X_0 + X_d * t$$
$$Y = Y_0 + Y_d * t$$
$$Z = Z_0 + Z_d * t.$$

- **algebraic solution, cont.**

  - **substitute the set of equations into the sphere equations's variables [X_s Y_s Z_s] obtaining**

$$(X_0 + X_d * t - X_c)^2 +$$
$$(Y_0 + Y_d * t - Y_c)^2 +$$
$$(Z_0 + Z_d * t - Z_c)^2 = S_r^2.$$

which simplifies to

$$A * t^2 + B * t + C = 0$$

where

$$A = X_d^2 + Y_d^2 + Z_d^2 = 1$$
$$B = 2 * (X_d * (X_0 - X_c) + Y_d * (Y_0 - Y_c) + Z_d * (Z_0 - Z_c))$$
$$C = (X_0 - X_c)^2 + (Y_0 - Y_c)^2 + (Z_0 - Z_c)^2 - S_r^2.$$

    - . **A always equals 1 since the ray direction is normalized**

  - **the solution for t is**

$$t_0 = \frac{-B - \sqrt{(B^2 - 4 * C)}}{2}$$

$$t_1 = \frac{-B + \sqrt{(B^2 - 4 * C)}}{2}.$$

    - . **when the part under the radical is negative, the ray misses the sphere**

    - . **the smaller, positive real root is the closer intersection point, if one exists**

- **algebraic solution, cont.**

  - **the actual intersection point is**

$$\mathbf{r}_{\text{intersect}} \equiv \mathbf{r}_{\mathrm{i}} = [x_{\mathrm{i}}\ y_{\mathrm{i}}\ z_{\mathrm{i}}]$$

$$= [X_0 + X_{\mathrm{d}} * t \quad Y_0 + Y_{\mathrm{d}} * t \quad Z_0 + Z_{\mathrm{d}} * t].$$

  - **the unit vector normal at the surface is**

$$\mathbf{r}_{\text{normal}} \equiv \mathbf{r}_{\mathrm{n}} = \left[\frac{(x_{\mathrm{i}} - X_{\mathrm{c}})}{S_{\mathrm{r}}} \frac{(y_{\mathrm{i}} - Y_{\mathrm{c}})}{S_{\mathrm{r}}} \frac{(z_{\mathrm{i}} - Z_{\mathrm{c}})}{S_{\mathrm{r}}} \right].$$

    - **if the ray originates inside the sphere, the unit vector normal at the surface should be negated so it points back toward the ray**

algebraic solution, cont.

example

Given a ray with an origin at [1 −2 −1] and a direction vector of [1 2 4], find the nearest intersection point with a sphere of radius $S_r = 3$ centered at [3 0 5].

First normalize the direction vector, which yields:

$$\text{direction vector magnitude} = \sqrt{(1 * 1 + 2 * 2 + 4 * 4)} = \sqrt{21}$$
$$\mathbf{R}_d = [1/\sqrt{21} \quad 2/\sqrt{21} \quad 4/\sqrt{21}]$$
$$= [0.218 \ 0.436 \ 0.873].$$

Now find A, B, and C

$$A = 1 \text{ (because the ray direction is normalized)}$$
$$B = 2 * (0.218 * (1 - 3) + 0.436 * (-2 - 0) + 0.873 * (-1 - 5))$$
$$= -13.092$$
$$C = (1 - 3)^2 + (-2 - 0)^2 + (-1 - 5)^2 - 3^2$$
$$= 35.$$

We now check if the discriminant is positive

Is $B^2 - 4 * C > 0$?
Substituting: is $-13.092^2 - 4 * 35 > 0$?
Yes, $31.400 > 0$.

This means the ray intersects the sphere. From this we can calculate t_0 from (A6):

$$t_0 = \frac{-B - \sqrt{(B^2 - 4 * C)}}{2}$$
$$= \frac{13.092 - \sqrt{(31.400)}}{2}$$
$$= 3.744$$

Since t_0 is positive, we don't have to calculate t_1. The actual intersection point is.

$$\mathbf{r}_i = [X_0 + X_d * t \ \ Y_0 + Y_d * t \ \ Z_0 + Z_d * t]$$
$$= [1 + 0.218 * 3.744 \quad -2 + 0.436 * 3.744 \quad -1 + 0.873 * 3.744]$$
$$= [1.816 \quad -0.368 \quad 2.269].$$

The unit vector normal is.

$$\mathbf{r}_n = \left[\frac{(x_i - X_c)}{S_r} \quad \frac{(y_i - Y_c)}{S_r} \quad \frac{(z_i - Z_c)}{S_r} \right]$$
$$= [\ (1.816 - 3)/3 \quad (-0.368 - 0)/3 \quad (2.269 - 5)/3]$$
$$= [\ -0.395 \quad -0.123 \quad -0.910].$$

- geometric solution
    - improving on the previous solution
        - avoid square root calculations
        - use multiplicative inverses to avoid division
        - make simple checks to avoid some calculations
            - a ray may point away from a sphere
    - another strategy
        - determine if the ray's origin is outside the sphere
        - determine the closest approach of the ray to the sphere's center
        - determine if the ray is outside and points away from the sphere
        - otherwise, determine the distance between the closest approach and the sphere's surface
            - if the value is negative, the ray misses
            - otherwise find the ray/surface distance
        - calculate the [x_i y_i z_i] intersection coordinates
        - calculate the normal at the intersection point

- more detailed explanation

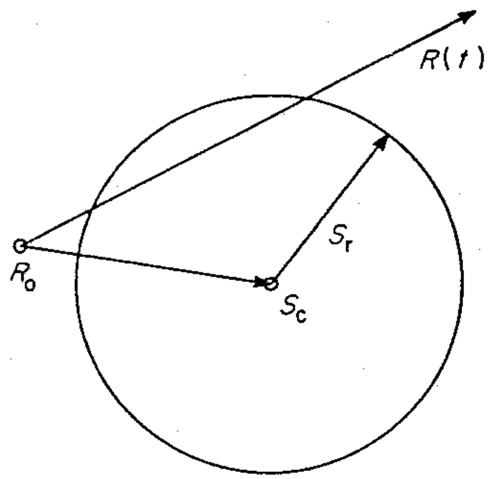
  - begin with the original ray and sphere equations

  - determine if the ray's origin is outside the sphere

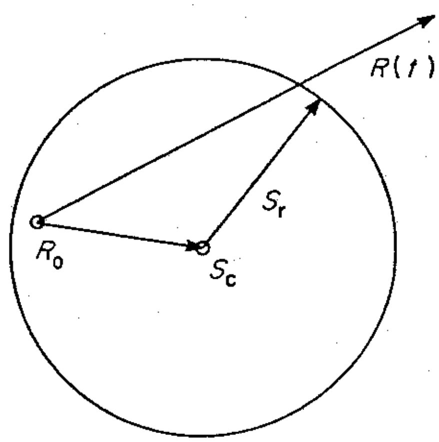
origin to center vector $\equiv \mathbf{OC} = \mathbf{S_c} - \mathbf{R_0}$
length squared of $\mathbf{OC} \equiv L_{2oc} = \mathbf{OC} \cdot \mathbf{OC}$.

- if $L^2_{oc} < S^2_r$, the ray origin is inside the sphere

- otherwise, the ray origin is outside the sphere and the ray may not hit the sphere



$(S_c - R_0) \bullet (S_c - R_0) > S_r * S_r$
so origin is outside sphere



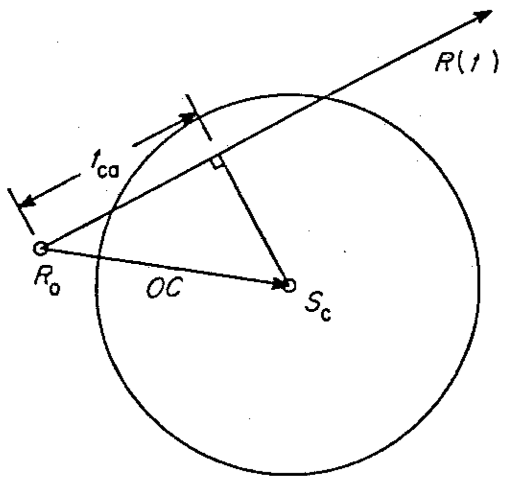
$(S_c - R_0) \bullet (S_c - R_0) < S_r * S_r$
so orgin is inside sphere

- **more detailed explanation, cont.**

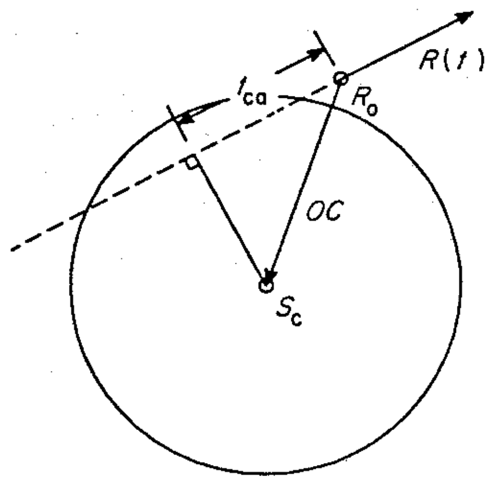
  - **calculate the distance from the ray's origin to the point on the ray closest to the sphere's center (equivalent to finding the intersection of the ray with the plane perpendicular to it which passes through the center of the sphere)**

closest approach along ray $\equiv t_{ca} = \mathbf{OC} \cdot \mathbf{R_d}$.

- . **if $t_{ca} < 0$, the center of the sphere lies behind the origin of the ray**
- . **for rays originating outside the sphere, $t_{ca} < 0$ means they cannot hit the sphere; testing is completed**



$t_{ca} > 0$, so the ray points toward the sphere



$t_{ca} < 0$, so the ray points away from the sphere

- more detailed explanation, cont.

  - determine the distance between the closest approach and the sphere's surface

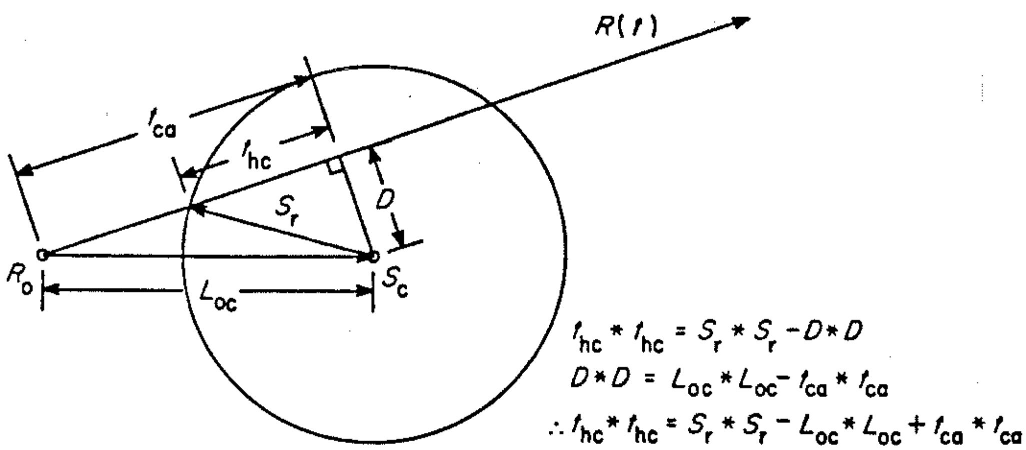
half chord distance squared $\equiv t_{hc}^2 = S_r^2 - D^2$

where D is the distance from the ray's closest approach to the sphere's center

$$D^2 = L_{oc}^2 - t_{ca}^2.$$

  - substituting,

$$t_{hc}^2 = S_r^2 - L_{oc}^2 + t_{ca}^2.$$



- **more detailed explanation, cont.**

    - **test whether the ray hits the sphere**

        - **if $t^2_{hc} < 0$, the ray misses the sphere (i.e. the ray originates outside the sphere)**

    - **calculate the intersection point's distance along the ray**

$t = t_{ca} - \sqrt{t_{hc}^2}$ for rays originating outside the sphere,
$t = t_{ca} + \sqrt{t_{hc}^2}$ for rays originating inside or on the sphere.

        - **use the smaller value for rays which originate outside the sphere**

        - **use the larger value for rays which originate inside the sphere (the smaller distance is negative - behind the ray)**

    - **calculate the intersection point and the normal as before**

example

Given a ray with an origin at [1 −2 −1] and a direction vector of [1 2 4], find the intersection point with a sphere of radius $S_r = 3$ centered at [3 0 5].

As before, first normalize the direction vector, which yields:

$$\text{direction vector magnitude} = \sqrt{(1 * 1 + 2 * 2 + 4 * 4)} = \sqrt{21}$$
$$\mathbf{R}_d = [1/\sqrt{21} \quad 2/\sqrt{21} \quad 4/\sqrt{21}]$$
$$= [0.218 \; 0.436 \; 0.873].$$

First find the ray to the center and its length squared

$$\mathbf{OC} = [3 \; 0 \; 5] - [1 \; -2 \; -1]$$
$$= [2 \; 2 \; 6]$$
$$L_{2oc} = [2 \; 2 \; 6] \cdot [2 \; 2 \; 6]$$
$$= 44.$$

Checking if $L_{2oc} \geq S_r^2$, it is found that the ray originates outside the sphere. Now calculate the closest approach along the ray to the sphere's center

$$t_{ca} = [2 \; 2 \; 6] \cdot [0.218 \; 0.436 \; 0.873]$$
$$= 6.546.$$

Checking if $t_{ca} < 0$, it is found that the center of the sphere lies in front of the origin, so calculation must continue. Calculate the half chord distance squared

$$t_{2hc} = 3 * 3 - 44 + 6.546 * 6.546$$
$$= 7.850.$$

$t_{2hc} > 0$, so the ray must hit the sphere. The intersection distance is then

$$t = 6.546 - \sqrt{7.850}$$
$$= 3.744$$

This is the same answer calculated for t_0 in the earlier algebraic example. As before, the intersection point is

$$\mathbf{r}_i = [X_0 + X_d * t \quad Y_0 + Y_d * t \quad Z_0 + Z_d * t]$$
$$= [1 + 0.218 * 3.744 \quad -2 + 0.436 * 3.744 \quad -1 + 0.873 * 3.744]$$
$$= [1.816 \quad -0.368 \quad 2.269]$$

The unit vector normal is

$$\mathbf{r}_n = \left[\frac{(x_i - X_c)}{S_r} \; \frac{(y_i - Y_c)}{S_r} \; \frac{(z_i - Z_c)}{S_r}\right]$$
$$= [(1.816 - 3)/3 \quad (-0.368 - 0)/3 \quad (2.269 - 5)/3]$$
$$= [-0.395 \quad -0.123 \quad -0.910]$$

- comparison of algebraic and geometric solutions
  - . the strength of the geometric solution is in the timely use of comparisons
    - a randomly placed ray faces away from the sphere half the time
- precision problems
  - . often the origin of a ray is a point on the sphere itself
    - theoretically, $t = 0$
    - but, imprecision can cause rays shot from the surface to hit the surface itself so that a small surface area is shadowed by itself, producing surface acne
  - . one solution
    - pass a flag telling if the origin is on the sphere
    - if the sphere is a transmitter, refraction rays must be allowed to pass through and to hit the other side
  - . simple solution
    - determine if t is within some tolerance
      - . the tolerance should be scaled to the size of the environment, perhaps based on the radius of the sphere intersected

- precision problems, cont.

  - root polishing methods

    - trace the ray and find t of the closest object

    - determine the intersection point and use this as the origin of a new ray which uses the same direction

    - intersecting the sphere with the new ray and accepting the solution for t closest to zero produces a more accurate intersection point

    - if t exceeds the acceptable tolerance, repeat the procedure

  - move the intersection point along its normal until it is outside or inside the sphere

    - when new rays are spawned, make certain the new origins are on the proper side of the surface

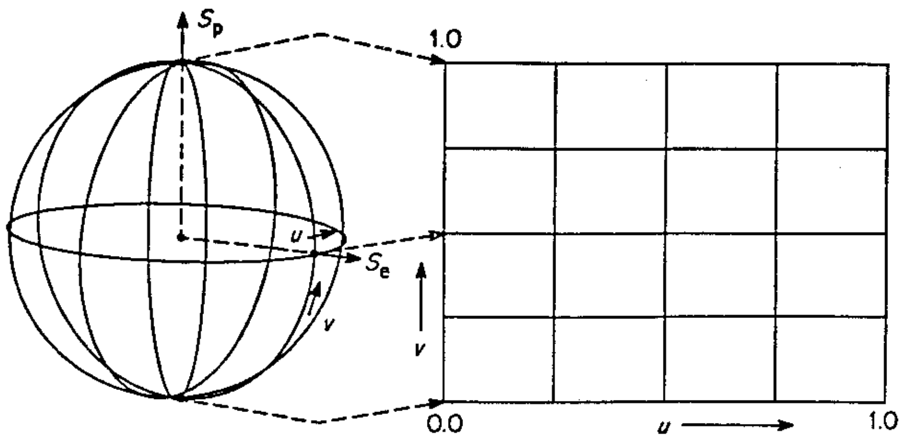
      - reflection and shadow rays always move positively

      - refraction rays move negatively

- spherical inverse mapping
  - texture or color mapping may be desired
  - problem: convert the intersection point into a longitude and latitude
  - input
    - the normal $\mathbf{S}_n$ at the point of intersection $\mathbf{R}_i$
    - the description of the sphere and its axes

      $\mathbf{S}_{pole} \equiv \mathbf{S}_p \equiv [X_p\ Y_p\ Z_p]$
      $\mathbf{S}_{equator} \equiv \mathbf{S}_e \equiv [X_e\ Y_e\ Z_e]$
      by definition, $\mathbf{S}_p \cdot \mathbf{S}_e = 0$ (i.e. are perpendicular)

      - $\mathbf{S}_p$ is a unit vector from the sphere's center to the "north pole" of the sphere
      - $\mathbf{S}_e$ is a unit vector to a point of reference on the equator
      - u varies along the equator from 0 to 1 and is defined to be 0 at the poles
      - v varies from 0 to 1 from the "south pole" to the "north pole"



- **spherical inverse mapping, cont.**

  - **calculating u and v**

    - **obtain the latitudinal parameter (the arccosine of the dot product between the intersection point's normal and the "north pole")**

$$\phi = \arccos(-\mathbf{S}_n \cdot \mathbf{S}_p)$$
$$v = \phi/\pi.$$

if v = 0 or 1, u = 0
else compute

$$\theta = \frac{\arccos((\mathbf{S}_e \cdot \mathbf{S}_n)/\sin(\phi))}{2 * \pi}$$

    - **obtain the longitudinal parameter (the cross product of the two sphere axes defining angles, compared with the direction of the normal)**

$$\begin{aligned}&\text{if } ((\mathbf{S}_p \otimes \mathbf{S}_e) \cdot \mathbf{S}_n) > 0\\ &\quad \text{then } u = \theta;\\ &\quad \text{else } u = 1 - \theta.\end{aligned}$$

      - **this test determines which side of the $\mathbf{S}_e$ vector the intersection point lies on**

- **spherical inverse mapping, cont.**

  . **example**

Begin with an intersection point normal $\mathbf{S_n}$ = [0.577 − 0.577 0.577] on a sphere whose axes are:

$$\mathbf{S_p} = [0\ 0\ 1]$$
$$\mathbf{S_e} = [1\ 0\ 0].$$

From these first find the latitudinal parameter

$$\phi = \arccos\,(-[0\ 0\ 1]\cdot[0.577\ \ -0.577\ 0.577]) = 2.186$$
$$v = 2.186/3.14159 = 0.696.$$

The longitudinal parameter calculations are

$$\theta = \frac{\arccos\,([1\ 0\ 0]\cdot[0.577\ \ -0.577\ 0.577]/\sin\,(2.186))}{2 * 3.14159}$$

$$= 0.125.$$

Now test which side of the axis $\mathbf{S_e}$ the point is on

$$([0\ 0\ 1] \otimes [1\ 0\ 0])\cdot[0.577\ \ -0.577\ 0.577]) = \ -0.577.$$

This value is less than 0, so:

$$u = 1 - 0.125 = 0.875.$$

The final answer is then $(u, v) = (0.875,\ 0.696)$.