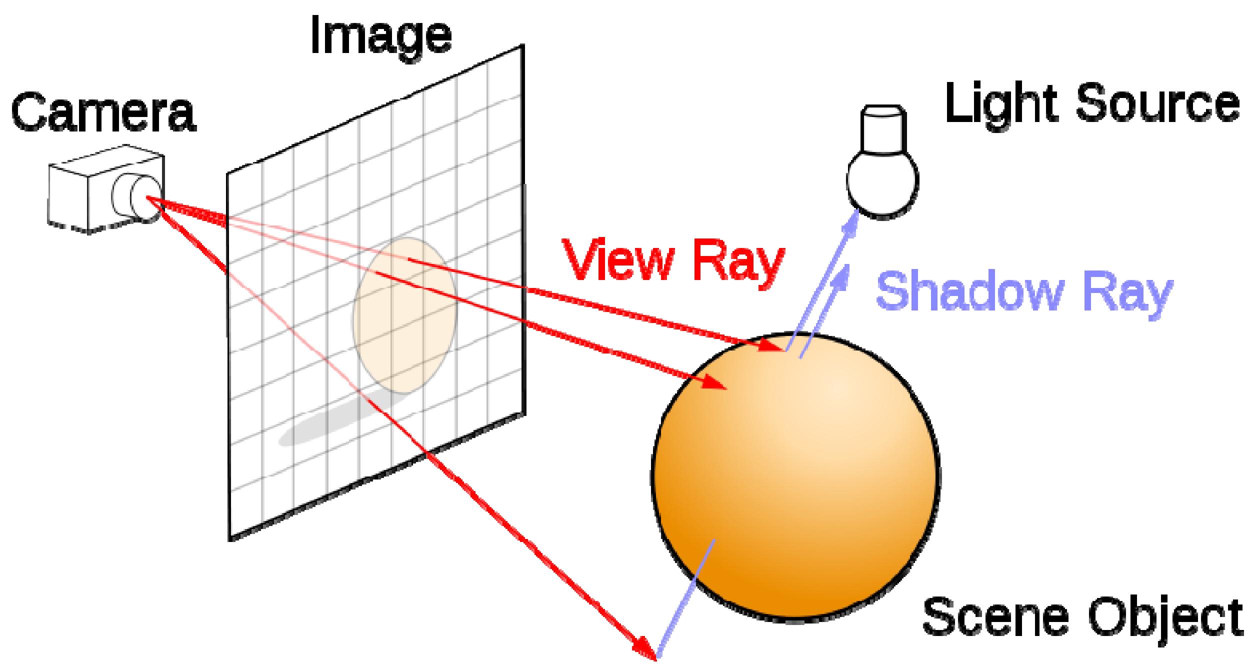
Computer Graphics

Ray Tracing

What is Ray Tracing?

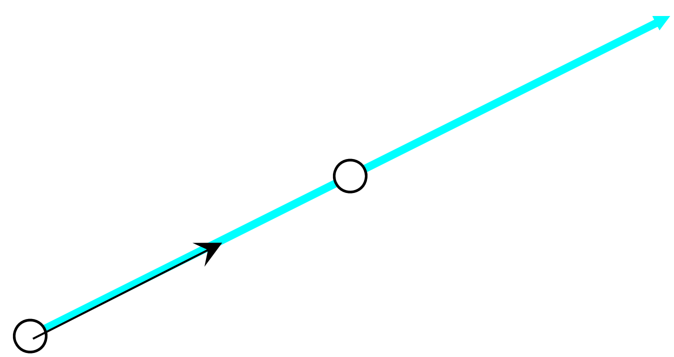


(Adapted from Wikipedia)

- Rays through each pixel in an image plane are traced back to the light source(s)

Rays

- A Ray is a line which may be calculated either by two pints or by a point and a vector
  - Vector – The ray direction
  - Point – The starting point of the ray
  - This describes an infinite line starting at the point and going in the ray direction
- Ray t values
  - A distance along the ray



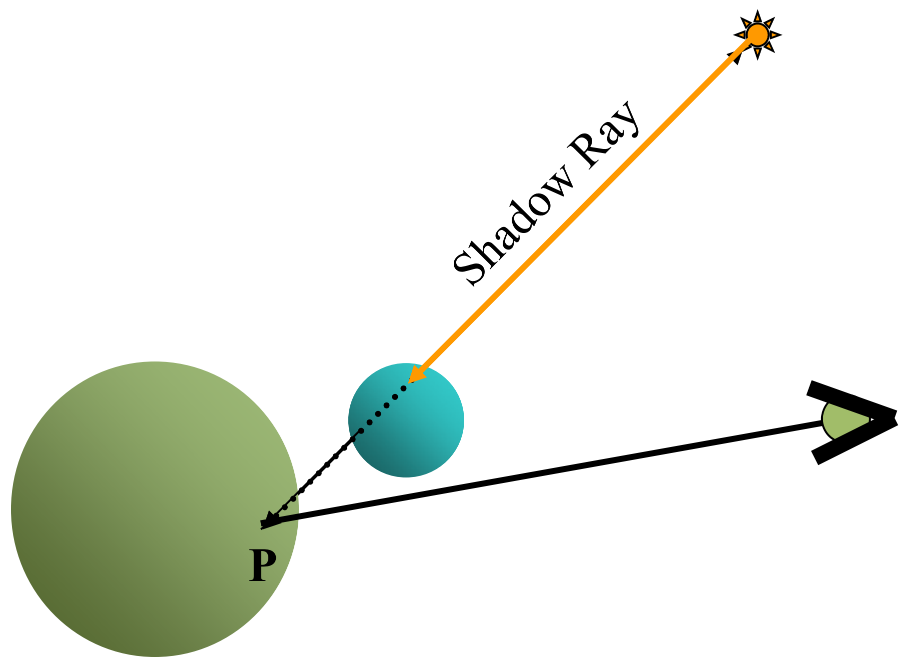
Ray Tracing Concept

- Shoot a line from the center of projection through the center of a pixel and off into space…
  - What does it hit first?
  - If it hits an object, compute the color for that point on the object
  - That's the pixel color

Simple Ray Tracing Algorithm

```
for (each scanline in image) {
    for (each pixel in the scanline) {
        calculate a ray from center of projection passing through pixel;
        for (each object in scene) {
            find the intersection of the ray with object
            if (object is intersected) and (is closest object)
                record intersection point
        }
        calculate pixel's color for the closest intersection point
    }
}
```

Finding Shadows



The Shadow Ray

- A ray from the intersection point to the light source is known as the **shadow ray**
- If any object intersects the shadow ray between the intersection point and the light source then the surface is in shadow with respect to that source

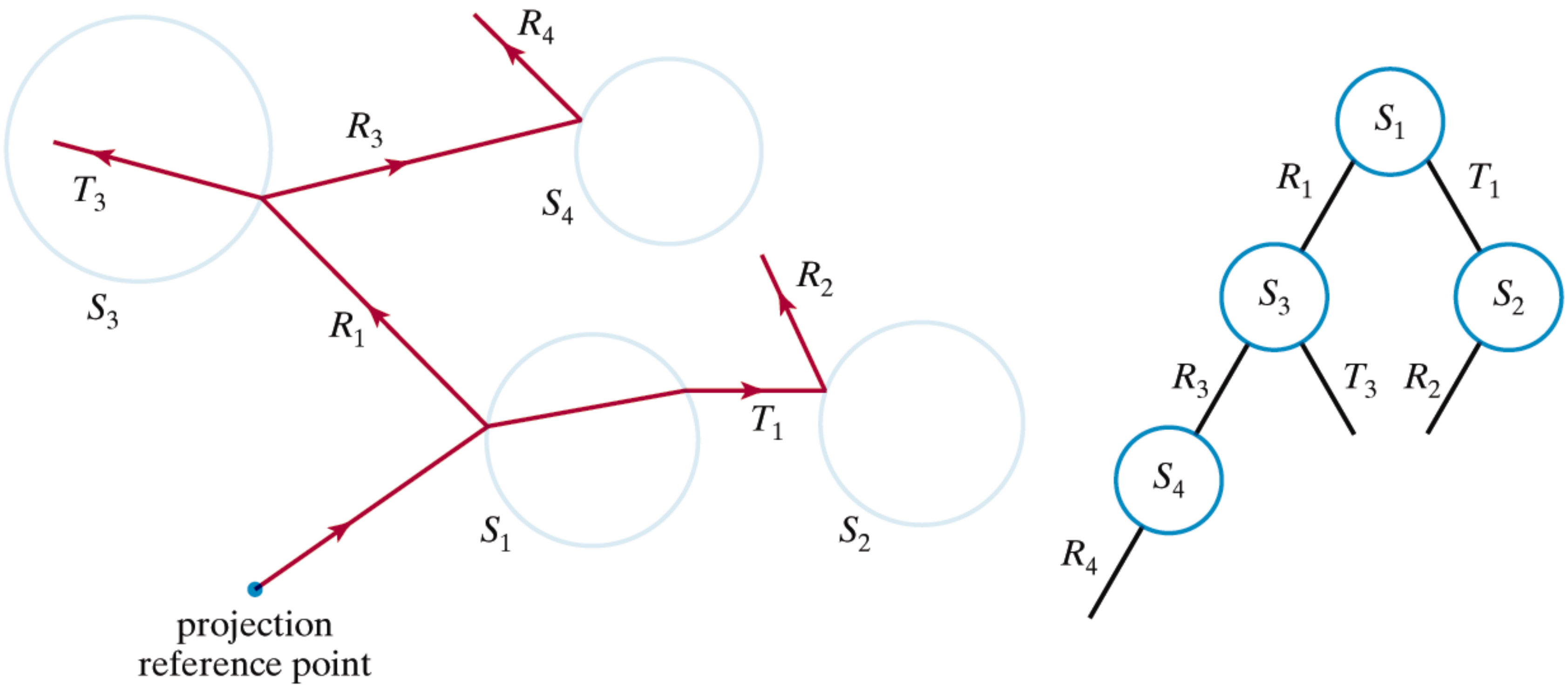
Recursive Ray Tracing Steps

- Follow a single ray from eye to each pixel position
- Find the intersection of the ray with the closest object. The nearest surface to the pixel is the visible surface for that pixel.
- Reflect the ray off the surface. For transparent surfaces also send a ray through the surface in the refraction direction. These rays are called secondary rays.
- Repeat the process for the secondary rays until a ray intersects no object or maximum allowable number of reflections is reached.

Ray-Tracing Tree

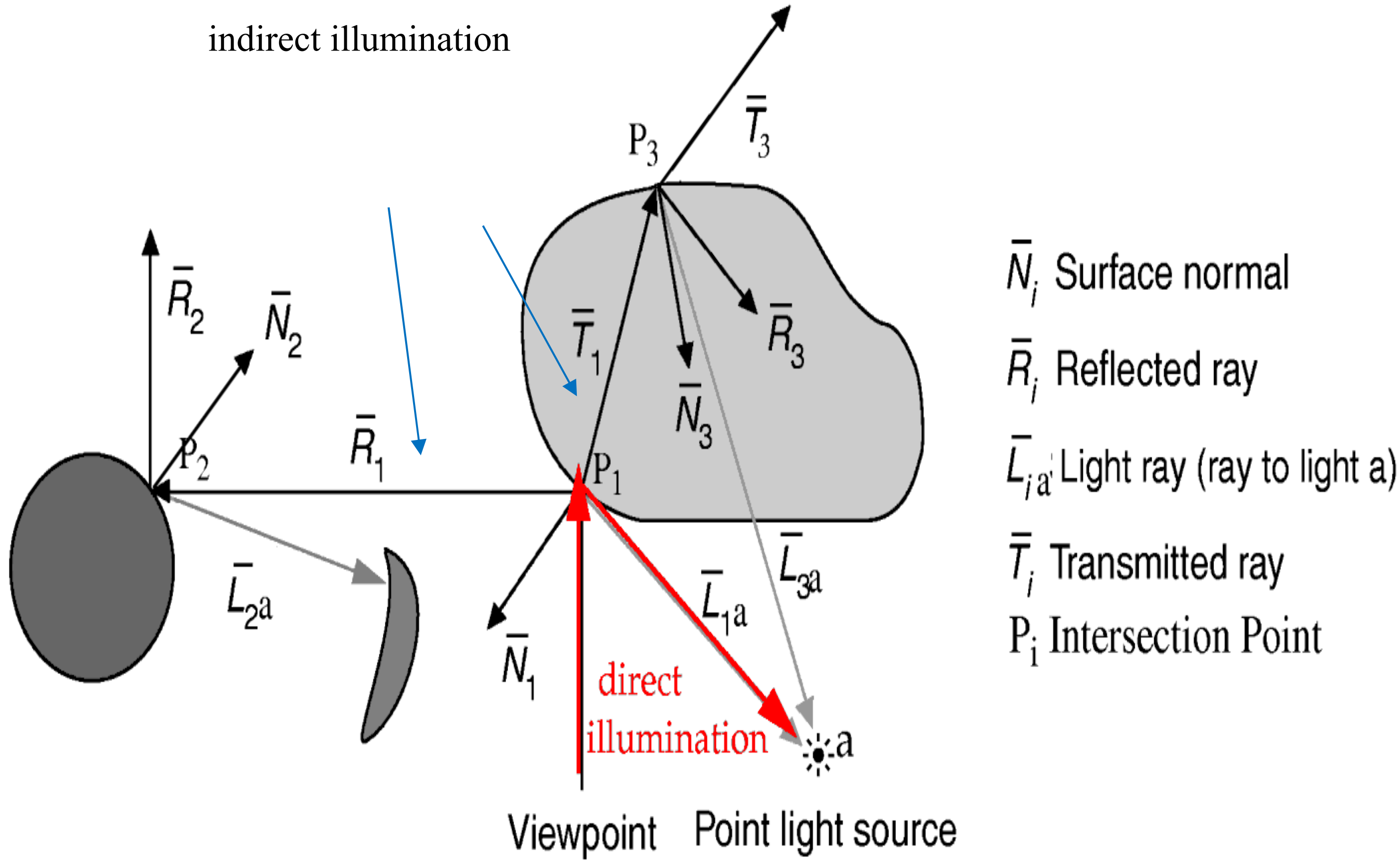
- As the rays reflect around the scene each intersected surface is added to a binary **ray-tracing tree**
- The tree's nodes store the intensity at that surface
- The tree is used to keep track of all contributions to a given pixel
- After the ray-tracing tree has been completed for a pixel the intensity contributions are accumulated
- We start at the terminal nodes (bottom) of the tree
- The surface intensity at each node is attenuated by the distance from the parent surface and added to the intensity of the parent surface
- The sum of the attenuated intensities at the root node is assigned to the pixel

Ray-Tracing Tree



Images taken from Hearn & Baker, "Computer Graphics with OpenGL" (2004)

Recursive Ray Tracing



Recursive Ray Tracing

- Your new lighting equation (Phong lighting + specular reflect + transmission):

$$I_\lambda = \underbrace{L_{a\lambda} k_{a\lambda}}_{ambient} + \sum_{lights} S_i f_{att_i} L_{p\lambda} [\underbrace{k_{d\lambda} (\vec{n} \bullet \vec{l})}_{diffuse} + \underbrace{k_{s\lambda} (\vec{r} \bullet \vec{v})^n}_{specular}] + \underbrace{\underbrace{k_{s\lambda} I_{r\lambda}}_{reflected} + \underbrace{k_{t\lambda} I_{t\lambda}}_{transmitted}}_{recursive\ rays}$$

  - I is the total color at a given point (lighting + specular reflection + transmission, λ subscript for each r,g,b)
  - L is the light intensity; L_P is the intensity of a point light source
  - k is the attenuation coefficient for the object material (ambient, diffuse, specular, etc.)
  - S_i is 0 if light is blocked at the intersection point 1 otherwise.
  - f_{att} is the attenuation function for distance
  - $\vec{n}$ is the normal vector at the object surface
  - $\vec{l}$ is the vector to the light
  - $\vec{r}$ is the reflected light vector
  - $\vec{v}$ is the vector from the eye point (view vector)
  - n is the specular exponent
  - note: intensity from recursive rays calculated with the same lighting equation at the intersection point
  - light sources contribute specular and diffuse lighting
- Note: single rays of light do not attenuate with distance; purpose of f_{att} is to simulate diminishing intensity per unit area as function of distance for point lights (typically an inverse quadratic polynomial)

Terminating Ray-Tracing

- Terminate a ray-tracing path when any one of the following conditions is satisfied:
  - The ray intersects no surfaces
  - The ray intersects a light source that is not a reflecting surface
  - A maximum allowable number of reflections have taken place

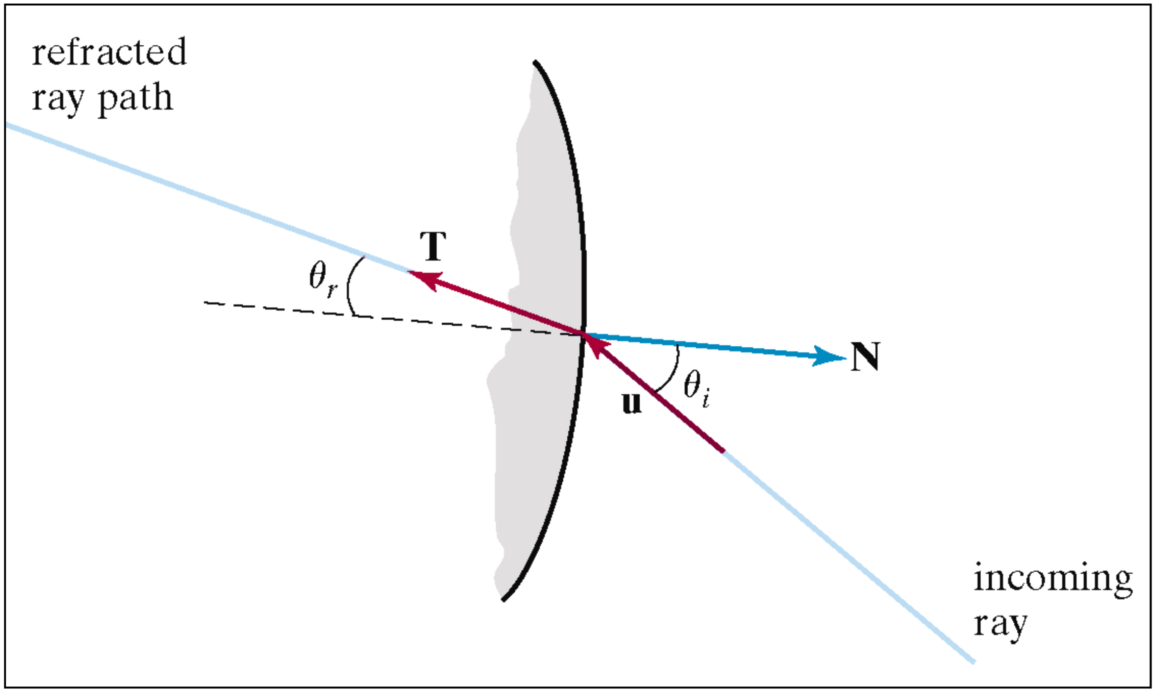
Transparent Surfaces

- We model the way light bends at interfaces with Snell's Law

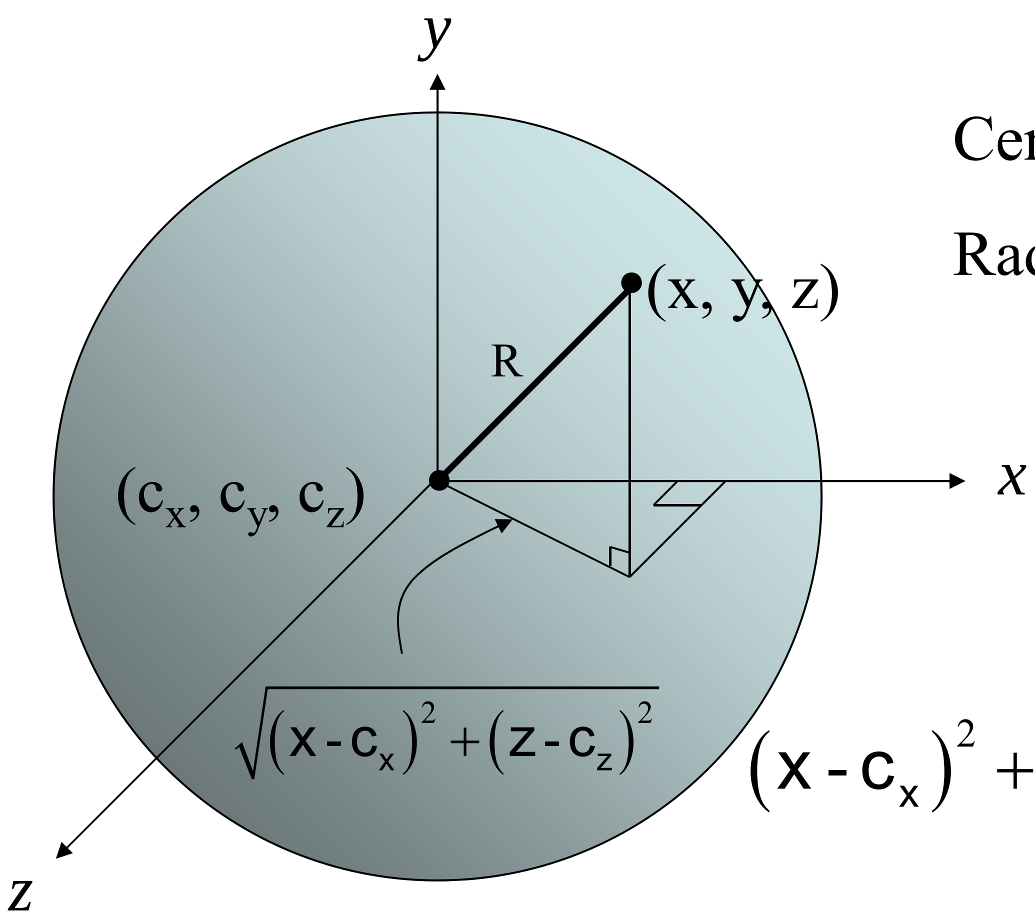
$$\sin\theta_r = \frac{\sin\theta_i \eta_{i\lambda}}{\eta_{r\lambda}}$$

$\eta_{i\lambda}$ = index of refraction of medium 1

$\eta_{r\lambda}$ = index of refraction of medium 2



Intersection of a Ray with Sphere



Center C = (c_x, c_y, c_z)

Radius R

$$(x-c_x)^2+(y-c_y)^2+(z-c_z)^2=R^2$$

Intersection of a Ray with Sphere

$P(t) = P_0 + t(P_1 - P_0)$ $\quad 0 <= t <= 1$

is really three equations

$x(t) = x_0 + t(x_1 - x_0)$

$y(t) = y_0 + t(y_1 - y_0)$

$z(t) = z_0 + t(z_1 - z_0)$ $\quad 0 <= t <= 1$

Substitute x(t), y(t), and z(t) for x, y, z, respectively in

$$(x - c_x)^2 + (y - c_y)^2 + (z - c_z)^2 = R^2$$

$$((x_0 + t(x_1 - x_0)) - c_x)^2 + ((y_0 + t(y_1 - y_0)_1) - c_y)^2 + ((z_0 + t(z_1 - z_0)) - c_z)^2 = R^2$$

Intersection of a Ray with Sphere

$$\left(\left(x_0 + t(x_1 - x_0)\right) - c_x\right)^2 + \left(\left(y_0 + t(y_1 - y_0)_1\right) - c_y\right)^2 + \left(\left(z_0 + t(z_1 - z_0)\right) - c_z\right)^2 = R^2$$

The above equation is a quadratic equation in variable t.

x_0, y_0, z_0, x_1, y_1, z_1, c_x, c_y, c_z, and R are all constants.

Solve for t using the quadratic formula.

Intersection of a Ray with Sphere

$$\left(\left(x_0 + t(x_1 - x_0)\right) - c_x\right)^2 + \left(\left(y_0 + t(y_1 - y_0)_1\right) - c_y\right)^2 + \left(\left(z_0 + t(z_1 - z_0)\right) - c_z\right)^2 = R^2$$

Set $d_x = x_1 - x_0$

$d_y = y_1 - y_0$

$d_z = z_1 - z_0$

Now find the the coefficients:

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

Intersection of a Ray with Sphere

$$\left(\left(x_0 + t\left(x_1 - x_0\right)\right) - c_x\right)^2 + \left(\left(y_0 + t\left(y_1 - y_0\right)\right) - c_y\right)^2 + \left(\left(z_0 + t\left(z_1 - z_0\right)\right) - c_z\right)^2 = R^2$$

$$\left(\left(x_0 + td_x\right) - c_x\right)^2 + \left(\left(y_0 + td_y\right) - c_y\right)^2 + \left(\left(\left(z_0 + td_z\right) - c_z\right)^2 = R^2$$

$$\begin{aligned}
&\left(x_0 + td_x\right)^2 - 2c_x\left(x_0 + td_x\right) + c_x{}^2 + \\
&\left(y_0 + td_y\right)^2 - 2c_y\left(y_0 + td_y\right) + c_y{}^2 + \\
&\left(z_0 + td_z\right)^2 - 2c_z\left(z_0 + td_z\right) + c_z{}^2 - R^2 = 0
\end{aligned}$$

$$\begin{aligned}
&x_0{}^2 + 2x_0td_x + t^2d_x{}^2 - 2c_xx_0 - 2c_xtd_x + c_x{}^2 + \\
&y_0{}^2 + 2y_0td_y + t^2d_y{}^2 - 2c_yy_0 - 2c_ytd_y + c_y{}^2 + \\
&z_0{}^2 + 2z_0td_z + t^2d_z{}^2 - 2c_zz_0 - 2c_ztd_z + c_z{}^2 - R^2 = 0
\end{aligned}$$

Intersection of a Ray with Sphere

$$x_0^2 + 2x_0td_x + t^2d_x^2 - 2c_xx_0 - 2c_xtd_x + c_x^2 +$$
$$y_0^2 + 2y_0td_y + t^2d_y^2 - 2c_yy_0 - 2c_ytd_y + c_y^2 +$$
$$z_0^2 + 2z_0td_z + t^2d_z^2 - 2c_zz_0 - 2c_ztd_z + c_z^2 - R^2 = 0$$

$$\boxed{A = d_x^2 + d_y^2 + d_z^2}$$

$$\boxed{B = 2d_x(x_0 - c_x) + 2d_y(y_0 - c_y) + 2d_z(z_0 - c_z)}$$

$$\boxed{\begin{aligned} C = {} & c_x^2 + c_y^2 + c_z^2 + x_0^2 + y_0^2 + z_0^2 + \\ & -2(c_xx_0 + c_yy_0 + c_zz_0) - R^2 \end{aligned}}$$

Intersection of a Ray with Sphere

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

$$\text{discriminant} = D(A,B,C) = B^2 - 4AC$$

$$D(A,B,C)\begin{cases} < 0 & \text{no intersection} \\ = 0 & \text{ray is tangent to the sphere} \\ > 0 & \text{ray intersects sphere in two points} \end{cases}$$

Intersection of a Ray with Sphere

The intersection nearest P_0 is given by:

$$t = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

To find the coordinates of the intersection point:

$$x = x_0 + td_x$$

$$y = y_0 + td_y$$

$$z = z_0 + td_z$$

Finding Normal

$$\mathbf{N} = \frac{(x-cx,\ y-cy,\ z-cz)}{|(x-cx,\ y-cy,\ z-cz)|}$$