**Shade Ray Diffuse**

To complete the shade ray diffuse function I needed to calculate the diffuse component of each R,G, and B component of the color. I used the formula

I also made sure to normalize the vector, l, before using it in the calculation.

**Shade Ray Local**

Shade ray local had the same implementation as shade ray diffuse with the addition of the specular component. The formula I used was

Where v is the direction the viewer which I calculated by negating the eye vector.

**Sphere Intersection**

To complete the ray sphere intersection function I calculated a,b, and c components of the quadratic formula to simplify the problem. a is equal to the dot product of the ray direction with itself, b is equal to the 2 times the ray direction times the hitpoint minus the sphere center, and c is equal to the hitpoint minus the sphere center squared, minus the radius squared. After this I calculated the discriminant of the quadratic formula which allows us to break the problem down into three cases. The first case is when the discriminant is less than 0 which means there are no intersection points. The second case is when the discriminant is equal to 0 meaning there is only one intersection point. This can be calculated as -b/2a. The third cases is when there are two intersection points.

This final case is further divided into three cases. The first case is when both intersections points are less than 0 meaning they are behind the ray origin and disregarded. The second case is when one point is in front of the ray origin and one is behind. In this case t would be whichever hitpoint is greater than 0. The last case is when both intersections are in front of the ray origin. In this case, t would be equal to the smallest value (or the closest value).

After t is calculated, we need to calculate the other properties of the intersection to parametrize them. The hitpoint is calculated using the o+td formula. The normal vector is calculated by subtracting the sphere position from the hitpoint and normalizing it. The surface material is equal to the sphere material.

I ran into some trouble with the ray sphere intersection because I was receiving a Thread 1: EXC\_BAD\_ACCESS (code=1, address=0x0) error. I was not able to find the cause of this error and as a result was not able to test out my sphere intersection function or use any spheres in the test.scene file. However, I am confident that the implementation is close to correct!

**Shade Ray Recursive**

To complete the shade ray recursive function, I needed to add the contributions from the reflection vector. This was done using the formula r = l - 2(n . l) n to compute the direction of the reflection ray. When I started the implementation I accidentally used the formula from the phong illumination but was able to fix the problem without starting over by negating the result. This calculation was saved as the new ray direction and I recursively called the shade ray recursive function at the current level + 1.