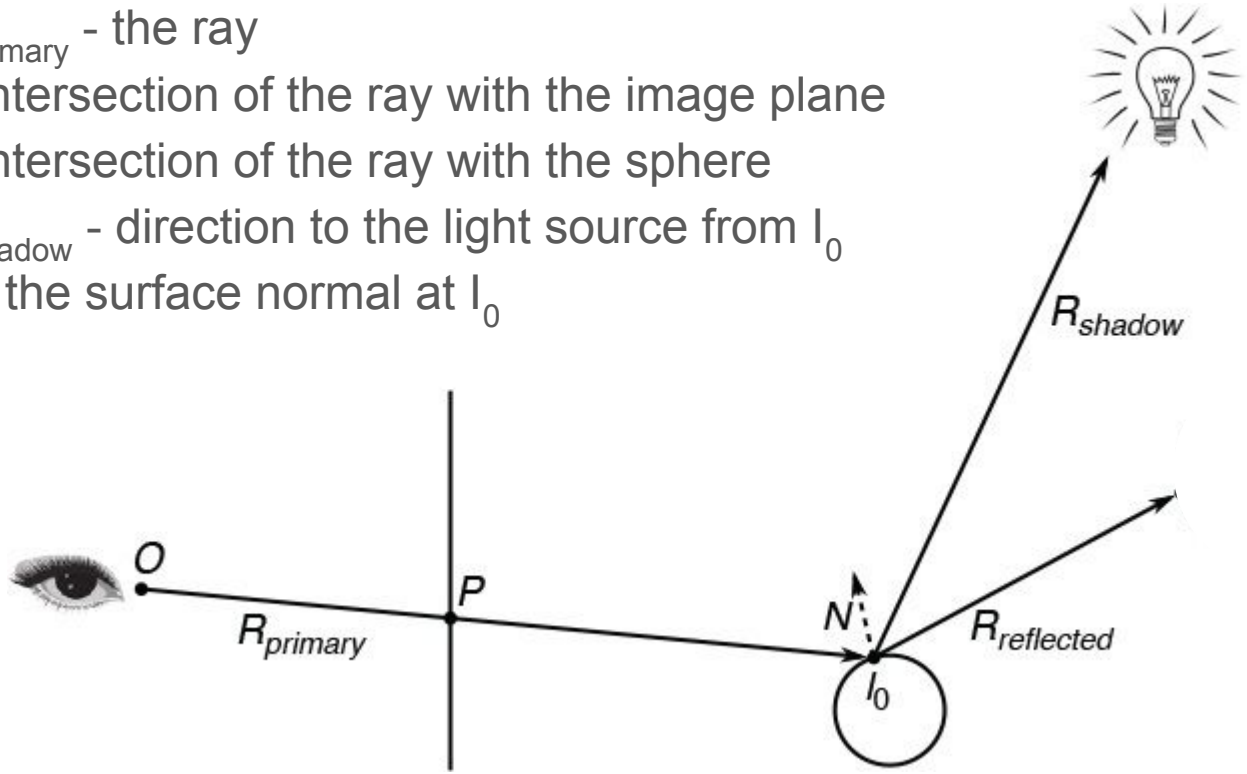
Ray Tracing

Vadim Surov

Introduction

- We discuss the construction of a simple non-recursive ray tracer for Assignment 3.
- The ray tracer is capable of rendering images using mathematical models of objects. In our case, it's a sphere.
- Knowledge of geometry is enough to produce a perfectly shaped image from the model on the screen using a fragment shader.

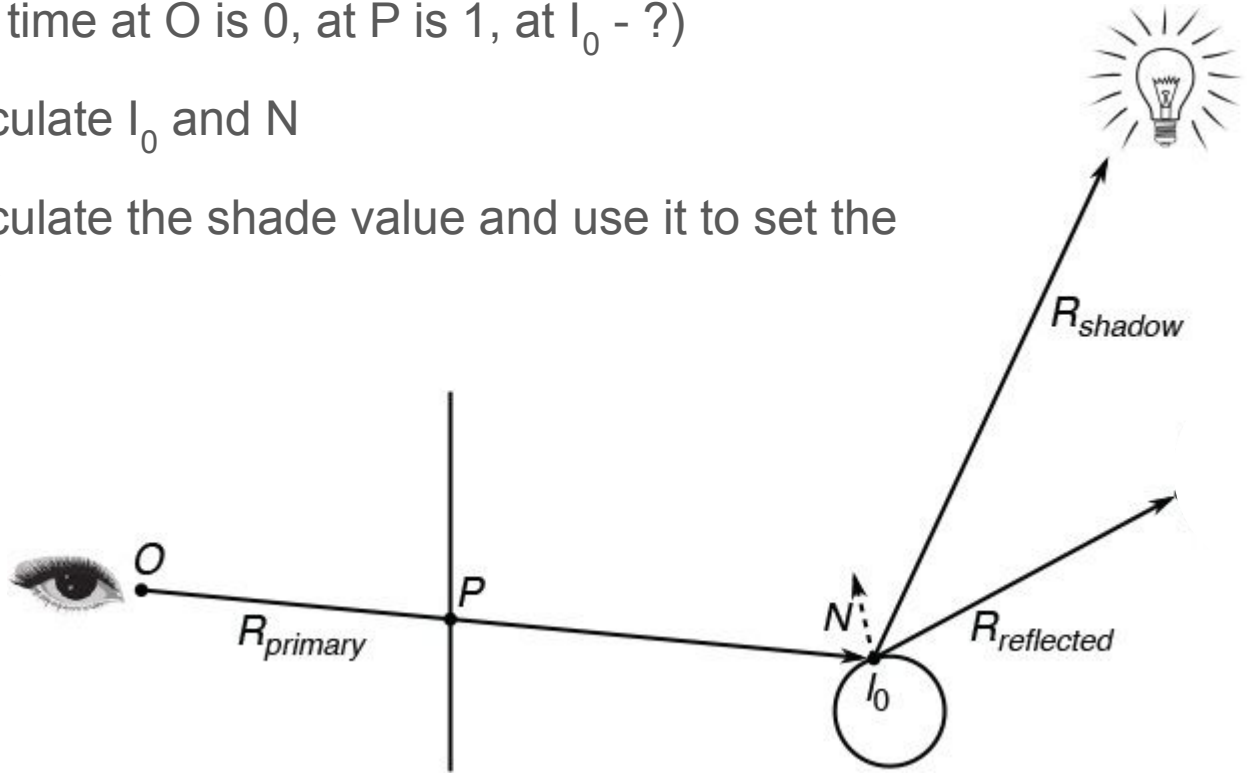
- Point O - the eye position that forms the origin of a ray
- Vector $R_{primary}$ - the ray
- Point P - intersection of the ray with the image plane
- Point I_0 - intersection of the ray with the sphere
- Vector R_{shadow} - direction to the light source from I_0
- Vector N - the surface normal at I_0



Step 1: Calculate t - time of intersection of the ray with the sphere (Ex: time at O is 0, at P is 1, at I_0 - ?)

Step 2: Calculate I_0 and N

Step 3: Calculate the shade value and use it to set the traced color



Step 1: Calculate t - time of intersection of the ray with the sphere

Step 1: Calculate t (1 / 4)

- Given a ray R with origin O and direction as vector D, then at time t, a point on that ray is O + tD.
- Also, given a sphere at center C with radius r. Any point on its surface is at distance r from C.
- The squared distance between C and any point on the sphere's surface is r^2. This is convenient because the dot product of a vector with itself is its squared length.
- Thus, we can say that for a point P at O + tD the following is correct:

$$(P - C) \cdot (P - C) = r^2$$

Step 1: Calculate t (2 / 4)

- Substituting for P, we have

$$(O + t\vec{D} - C) \cdot (O + t\vec{D} - C) = r^2$$

- Expanding this gives us a quadratic equation in t:

$$(\vec{D} \cdot \vec{D})t^2 + 2(O - C) \cdot \vec{D}t + (O - C) \cdot (O - C) - r^2 = 0$$

- Writing this in the more familiar form of $At^2 + Bt + C = 0$, we have

$$A = \vec{D} \cdot \vec{D}$$
$$B = 2(O - C) \cdot \vec{D}$$
$$C = (O - C) \cdot (O - C) - r^2$$

Step 1: Calculate t (3 / 4)

- As a simple quadratic equation, we can solve for t, knowing that there are either zero, one, or two solutions.

$$t = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

- Given that we know that our direction vector is normalized, then its length is 1, and, therefore, A is 1 as well. This simplifies things a little, and we can simply say that our solution for t is

$$t = \frac{-B \pm \sqrt{B^2 - 4C}}{2}$$

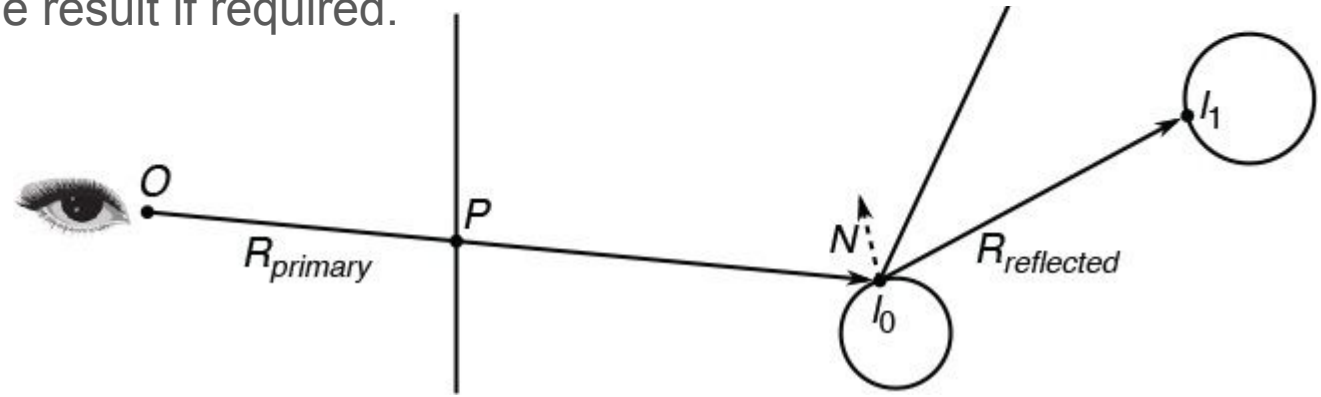
Step 1: Calculate t (4 / 4)

- If $4C > B^2$, then the term under the square root is negative and there is no solution for t, which means that there is no intersection between the ray and the sphere.
- If $B^2 == 4C$, then there is only one solution, meaning that the ray just grazes the sphere. If that solution is positive, then this occurs in front of the viewer and we have found our intersection point. If the single solution for t is negative, then the intersection point is behind the viewer.
- Finally, if there are two solutions to the equation, we take the smallest non-negative solution for t as our intersection point.

Step 2: Calculate the intersection point I_0 and normal vector N

Step 2: Calculate I_0 and N

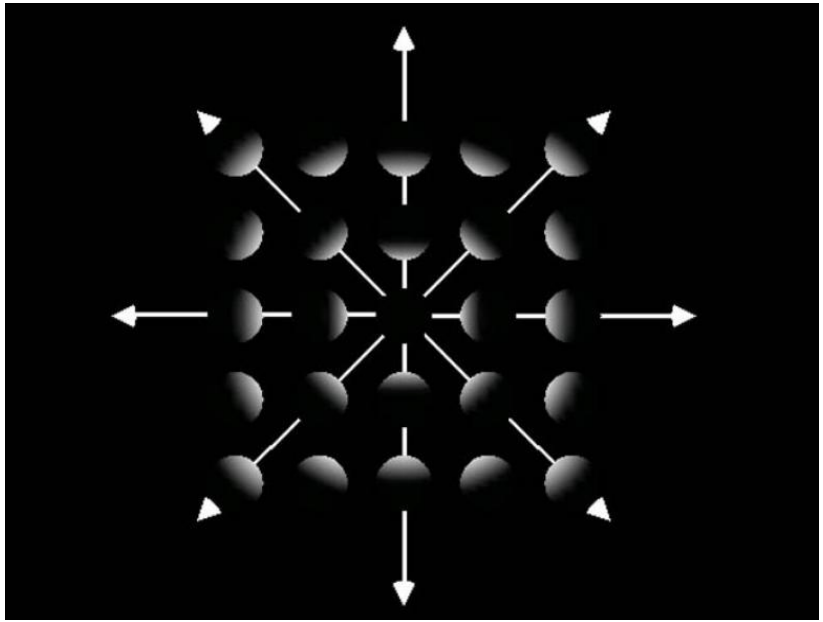
- So, let's say there is one or there are two solutions to the equation, we take the smallest non-negative solution for t as our intersection point.
- To find I_0 we simply plug t back into O + tD and retrieve the coordinates of the intersection point in 3D space.
- To find N we calculate I_0- C, where C is the center of the sphere. Normalize then the result if required.



Step 3: Calculate the shade value and use it to set the traced color

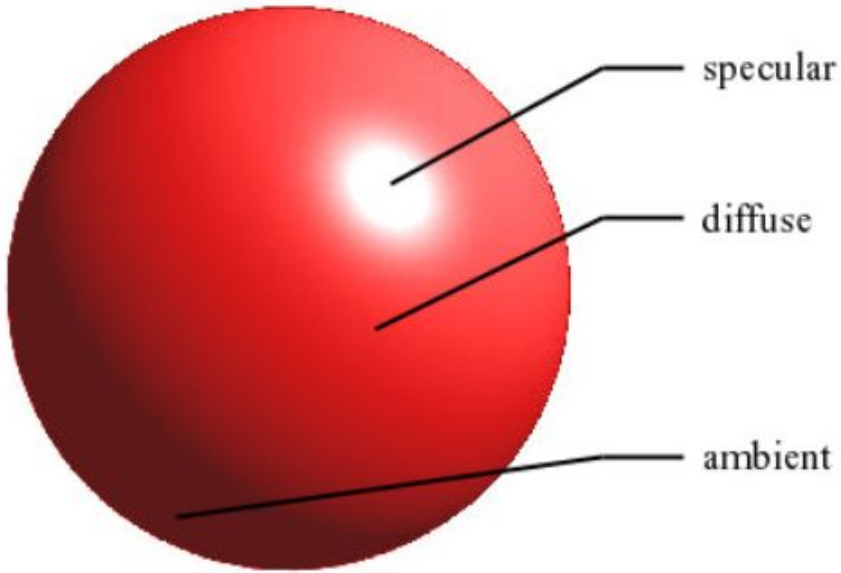
Step 3: Calculate the shade (1 / 4)

- Need to set the light source type and parameters.
- Simplest light: point light type with given position and intensity parameters. Radial intensity attenuation is ignored.
- We also ignore the material properties for sake of simplicity.



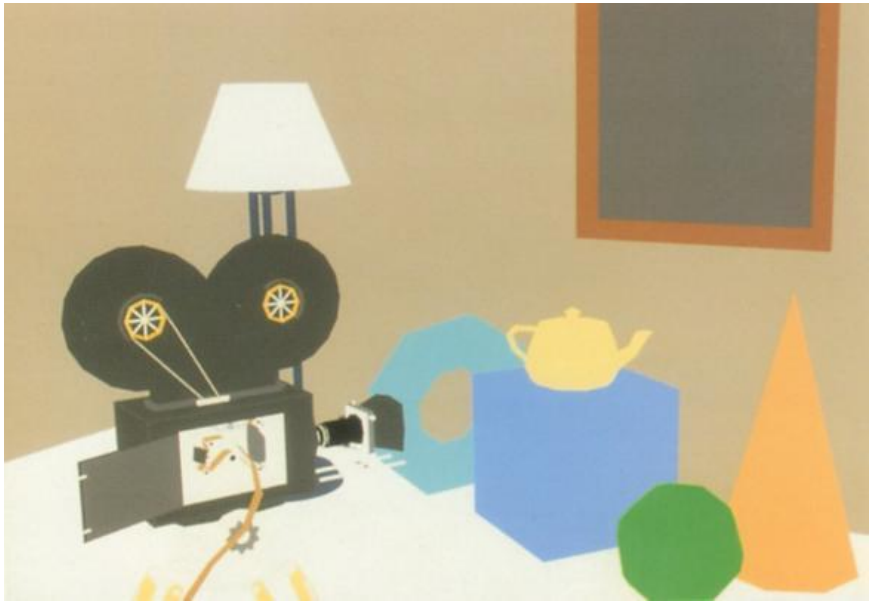
Step 3: Calculate the shade (2 / 4)

- Need to set the Illumination Model.
- In Assignment 3 we are using simple model that consider only the light position, intensity and 2 types of light contribution to compute the final illumination of an object: **ambient** and **diffuse.**



Step 3: Calculate the shade (3 / 4)

- **Ambient Light Contribution** is a very simple approximation of the global illumination**.**
- Ambient light has the same intensity in all locations and directions.
- We set the intensity as vec3(0.1f, 0.1f, 0.1f) as minimum for all points on the surface.



Rendered using Pexar's PhotoRealistic RenderMan ™ software

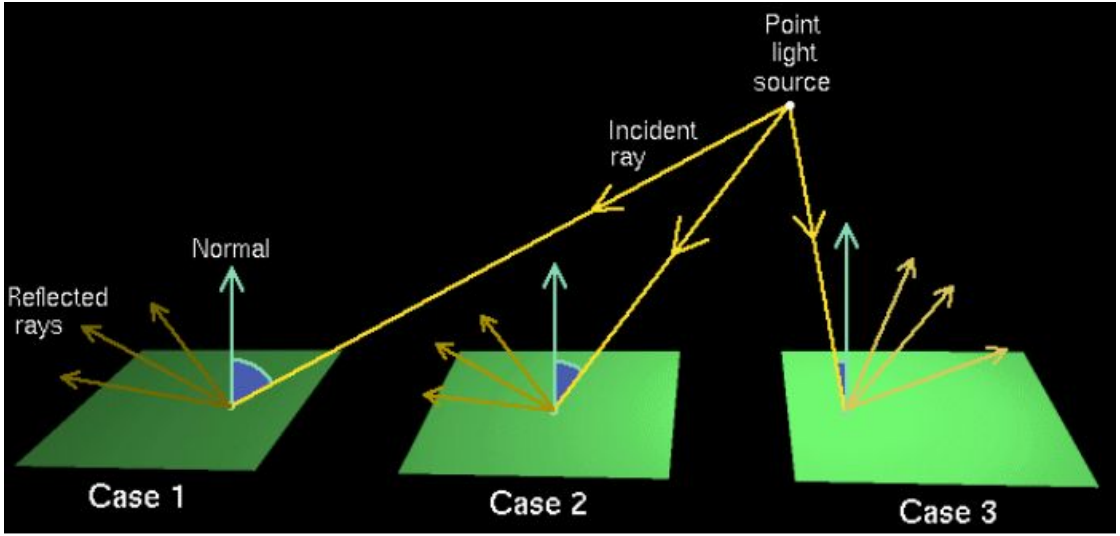
Step 3: Calculate the shade (3 / 4)

- **Diffuse Light Contribution** represents direct light that hits a surface and reflects equally in all direction.
- This contribution is independent of viewer position. Only depends on relative position of light source to surface.

Rendered using Pexar's PhotoRealistic RenderMan ™ software

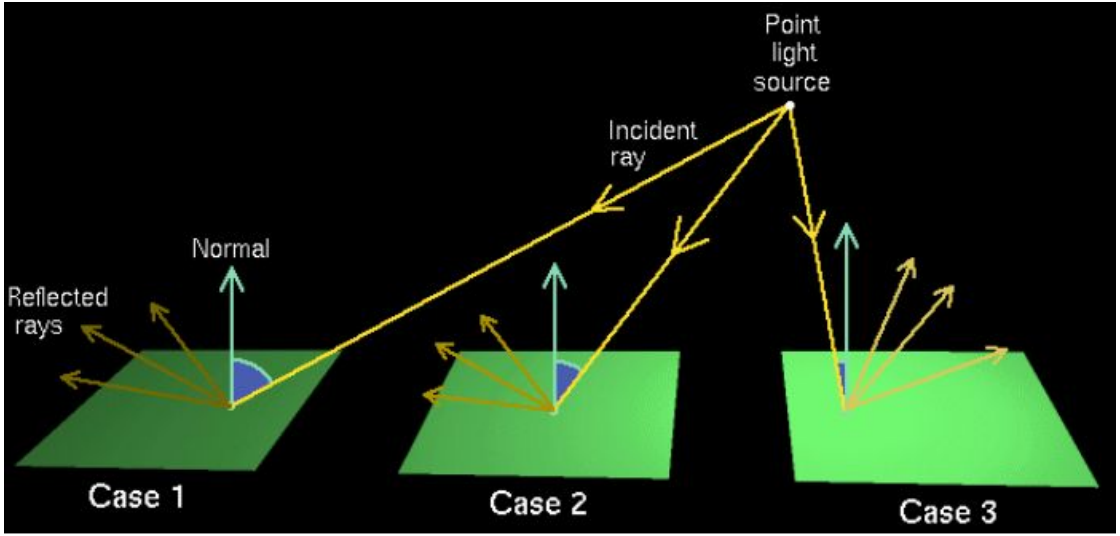
Lambert's Cosine Law (1 / 3)

- For diffuse contribution we need to decide how much light the object point receive from the light source



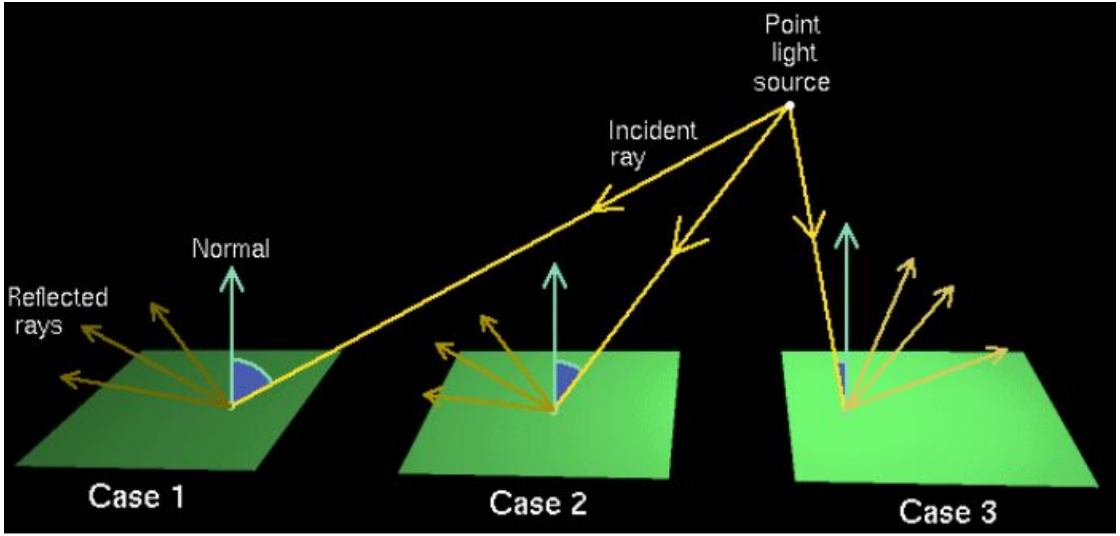
Lambert's Cosine Law (2 / 3)

- Lambert's law states that the reflected energy from a small surface area in a particular direction is proportional to cosine of the angle between that direction and the surface normal

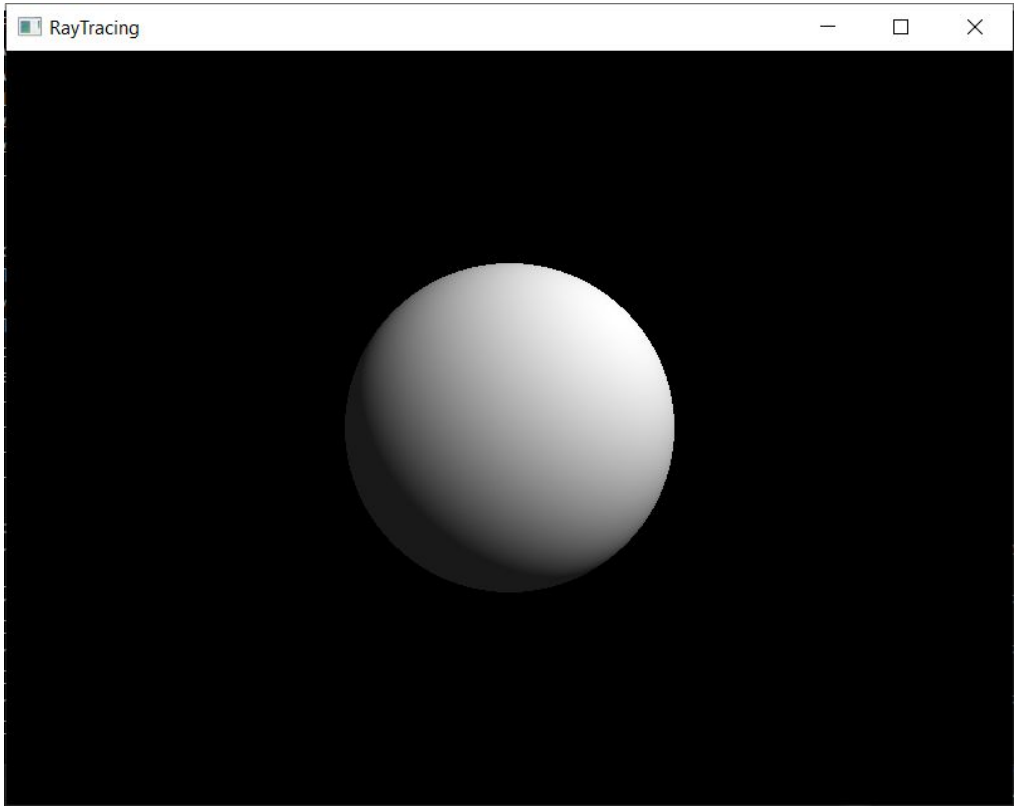


Lambert's Cosine Law (3 / 3)

- Finally, to find the cosine of the angle, we use dot product.
- Geometrically, the dot product of A and B equals the length of A times the length of B times the cosine of the angle between them: $A \cdot B = |A||B| \cos(\theta)$



Assignment 3. The result



Reference

- Chapter 13. OpenGL SuperBible : Comprehensive Tutorial And Reference, 7th Edition by Graham Sellers, Richard S Wright Jr., and Nicholas Haemel ISBN-10: 0672337479 ISBN-13: 978-0672337475