Computer Graphics Project

Deepak Yadav (14075020)
Vinayaka Batwara (14074016)
Shivam Garg (14074017)

Part III CSE
15th Nov 2016

What is Ray Tracing?

Ray tracing is a technique for rendering three-dimensional graphics with very complex light interactions. This means we can create pictures full of mirrors, transparent surfaces, and shadows, with stunning results. It is based on the idea that we can model reflection and refraction by recursively following the path that light takes as it bounces through an environment.

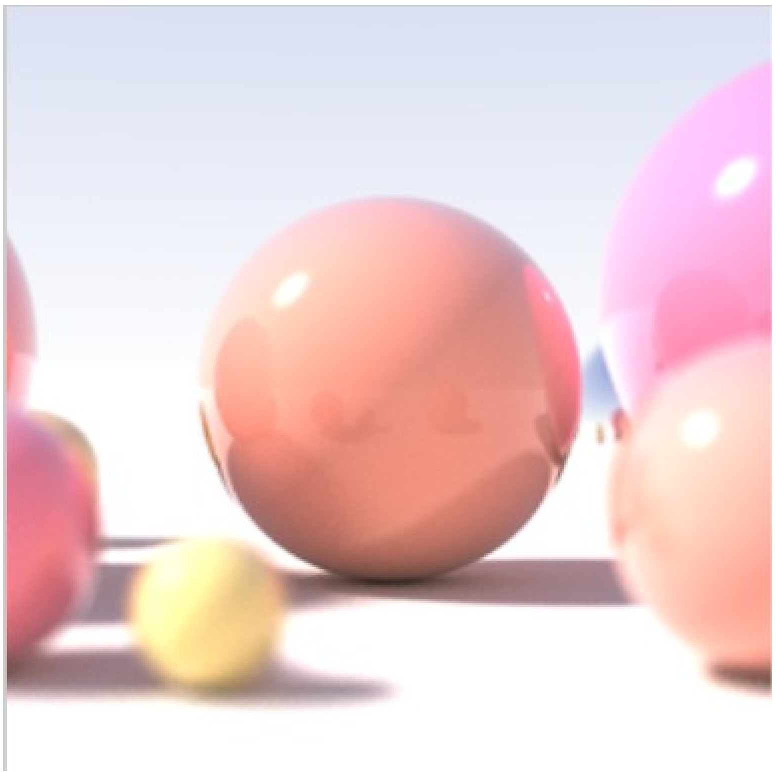

Fig: Photo-realistic ray tracing

Our Implementation

We have implemented a simplified version based on the basic philosophy of ray tracing. Our implementation traces a ray as it collides (reflection) with objects present in the scene (in our case 3 circles of different radius).
This program uses a function ray() that draws a ray given the starting point and the angle that the ray makes with the horizontal. The function ray() uses a generalized form of the DDA Line Generation Algorithm.

Given the initial point and angle of ray, we start tracing; checking at each point if a collision occurs. If there is a collision we calculate the change in the angle of the ray, and then continue this whole routine again.

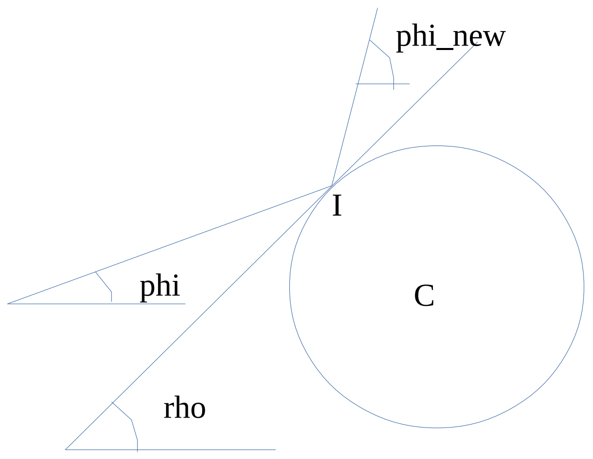

Fig: Calculation of phi_new

The calculations done are shown below:

phi_new = 2*rho-phi;

rho = arctan(-(intersection_x-center_x)/(intersection_y-center_y));

phi_new: angle of ray after collision
phi: angle of ray before collision
rho: angle of the tangent to the circle at the point of collision

phi, phi_new: Vary from 0 to 360 degrees
rho: Varies from 0 to 180 degrees

rho is calculated by the use of the vector connecting the center of the circle with the point of intersection/collision of the ray with the circle boundary. Then using the property that tangent is normal to the radius, we come up with the value of rho.
This calculation is explicit and works only for circles. Future works mention a more generalized method.
The whole implentation is in C++ using the OpenGl and GLUT libraries. Following are the screenshots of our program running:

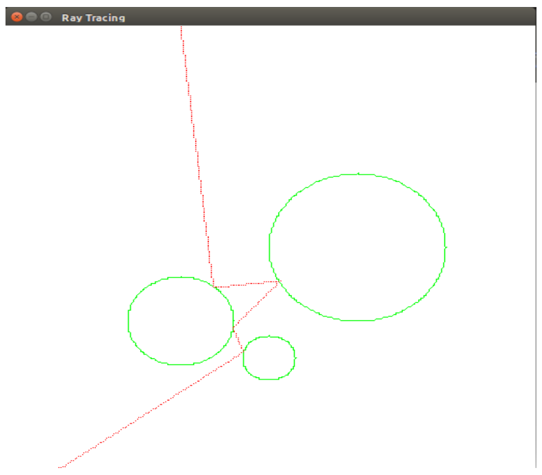

Fig: Program Screenshot for phi = 276

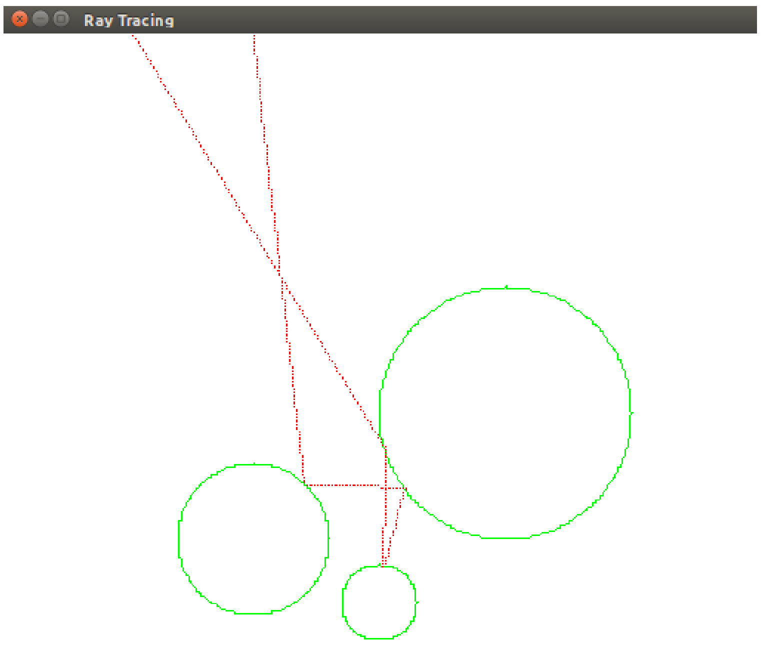

Fig: Program Screenshot for phi = 276.51

Conclusion:

Our program models the basic idea of ray tracing in an easy to understand manner. It also highlights and solves the basic problems involved in a photo-realistic ray tracing model. Ray Tracing is extremely important for creating shadows, textures and reflections in a complex 3D environment. It is a crucial part of rendering process.

Future Work

We will extend the idea so that it works for more complex geomerical surfaces. In this project we have explicity calculated the angle of reflected ray using the properties of circles, tangents and normals. This will not work for more complex surfaces.

We can generalize the calculation of rho for any surface using the property that the slope of tangent is the derivative of the curve at the point of collision. The derivative can be calculated using the approximate formula:

$$df/dx = (f(x+h)-f(x))/h, \text{ h is a small value}$$

This will generalize the program to any given surface. Other modification is the inclusion of refraction to the framework. This will also lead to "tree of rays" which can be easily handled using recursion. Final touch can be given by extending whole code into 3D.

References:

https://en.wikipedia.org/wiki/Ray_tracing_(graphics)
https://www.cs.unc.edu/~rademach/xroads-RT/RTarticle.html