Project Report

160050014 Chinmay Awale

Problem

To make a raytracer for hobbyists, which takes definations as collection of objects and creates an image from the definations in **reasonable time**.

Design

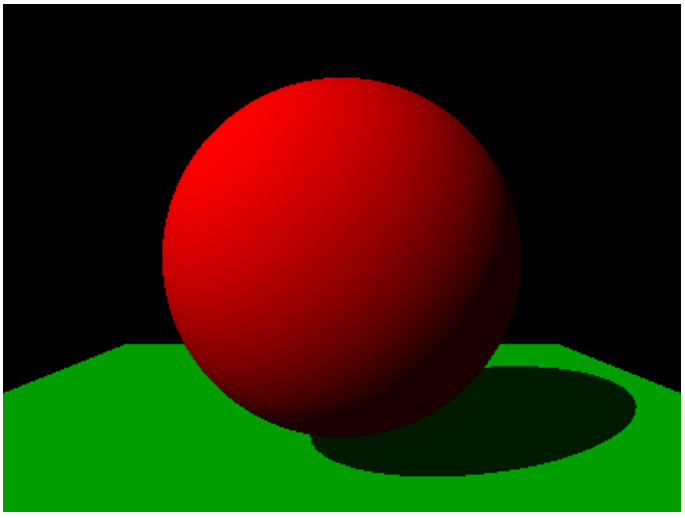
The program is split into five files:

(1) **render.rkt :** the *rendering loop and camera* resides in this file.
(2) **shader.rkt :** the functions for *shading* a pixel, *scene and light*.
(3) **primitives.rkt :** *shapes, color and miscellaneous structues*.
(4) **vectorlib.rkt & matrixlib.rkt :** *lists as vectors and matrices.*
(5) **racket/draw library** was used, **Examples.rkt** has scenes shown below.

Algorithm

- trace a ray from the camera point to a screen point and check its intersection with other objects.
- Get the closest point of intersection , If no intersection is found use background color
- check the visiblity of light source from that point in case it hits.
- Calculate the color intensity using the normal at the point and sample rays from source.

Sample Output



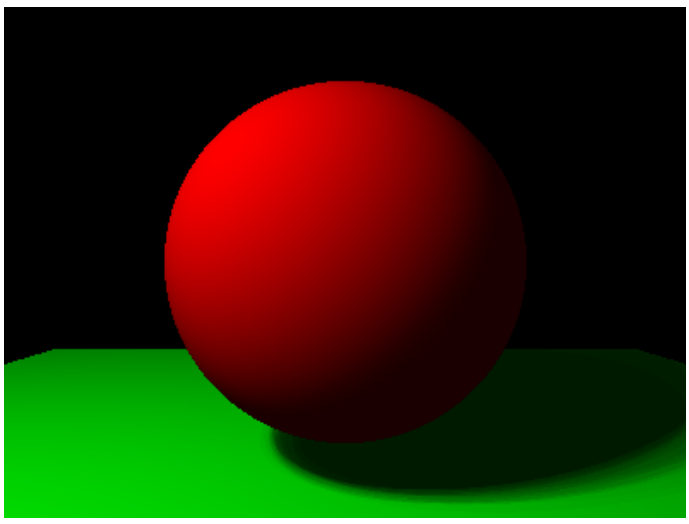
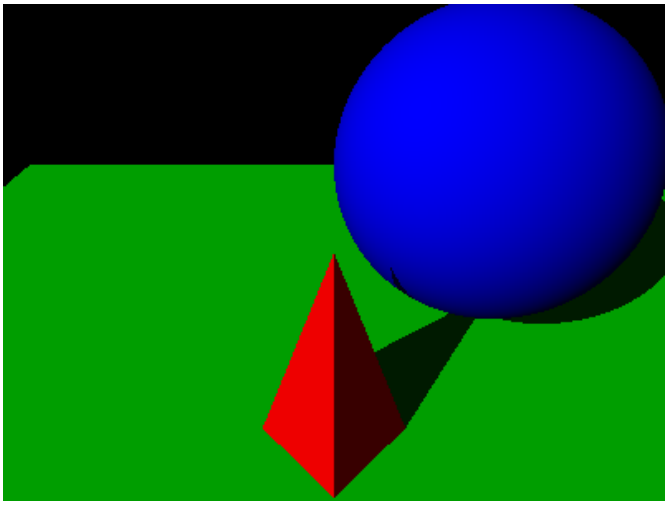
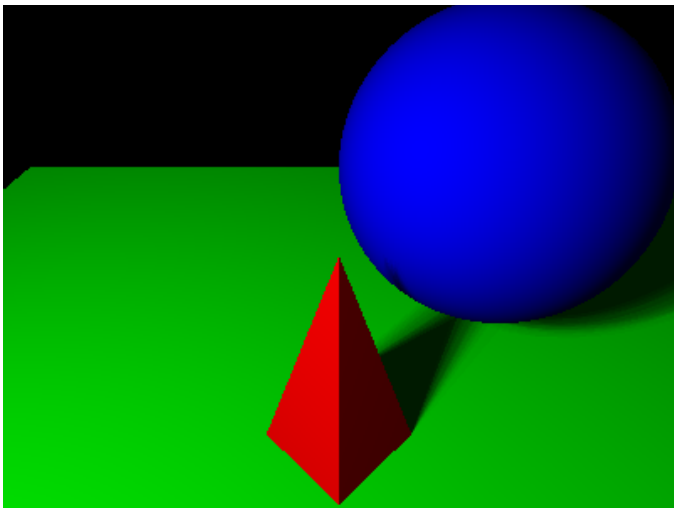


Image I, II : image on the left is rendered in the *global lamp mode* (render time ~ 3sec) (source at infinity) while one at the right side (render time ~ 30 sec) used *distributed source* with 30 samples. *Notice the dramatic difference in shadows*





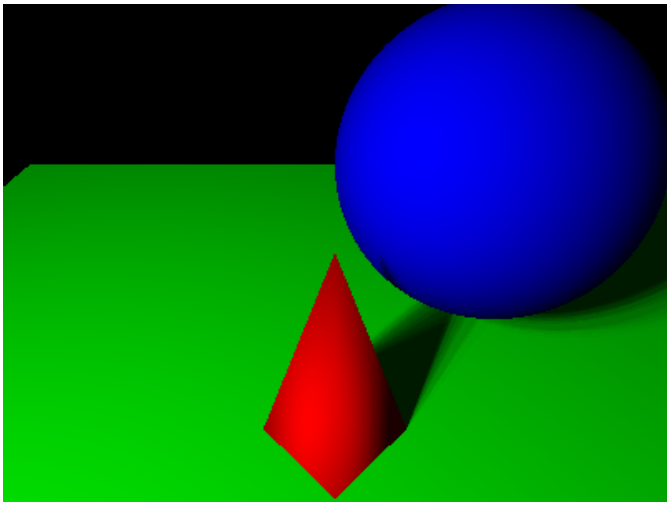


Image III: (*global lamp mode)*
render time ~ 6 sec

Image IV : (*distributed source)*
render time ~ 1 min 25 sec

Image V : (*distributed source and gourand shading)*
render time ~ 1 min 25 sec

Image VI : (*distributed source)*
render time ~ 1 min 52 sec

Limitations

- lack of hardware support causes machine to render slowly
- *recursive raytracing and spectacular reflections* were not implemented as *they drastically increased time*.
- *Smooth (Gourand) shading* was implemented but normals are user specified hence prone to falure

Points of interest

- to detect the amount of light recieved sampling of the light object was done and rays were traced to the concerned object, this was a workaround to what modern day renderers call *ambient occlusion*
- the triangle ray intersection calculation was improved by *Moller-Trumbore algorithm* which simply avoids recalculation.
- The sphere used to model the mesh is is inherited by light object, thus making it possible to make the light visible (*it is made invisible by default*)

*Note : all images are of resolution 400 x 300