CMPE460 Ray Intersection Assignment 1

Sevde SARIKAYA
2017400081

Problem: We are asked to implement a simple ray-tracer that renders a scene using ambient illumination, checking only for shadow and multiplying colors by 0.1 if the intersection point is in shadow. Assume the eye point is at the origin (0,0,0); and the center of the screen is at (0,0,100).The screen extends from (-50,-50,100) to (50,50,100) and the resolution is to be 1000x1000 pixels. Assume there is only one light source and it is at (500,500,500).

How my ray tracer works

1-) Define:

- **eye location**
- **light location**
- **scene location**
- **pixel step**

2-) Iterate through each pixel starting from the left lower corner by sending rays to each pixel from the eye.

3-) If the eye-ray intersects with an object at a point, send a ray from the light source in the direction of that point.

- **If the light ray intersects with the object in the same point, object color will be seen as original.**
- **If it doesn't, object color will be multiplied by 0.1 to create a shadow.**

<u>**How intersection calculated:**</u>

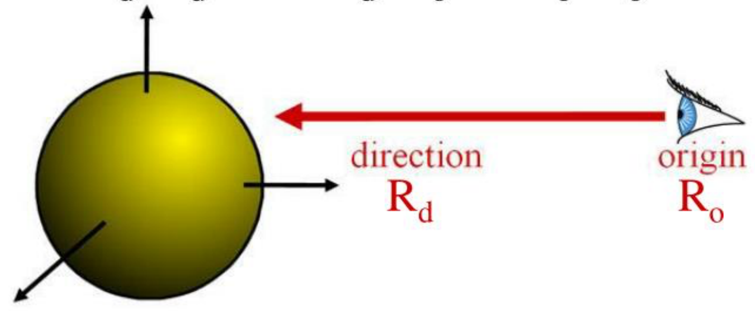
1-) Iterate through each sphere object and calculate the root.

The root is the solution of the scale of the direction in the formula below:

$$P(t) = R_o + t * R_d \qquad H(P) = P \cdot P - r^2 = 0$$

$$(R_o + tR_d) \cdot (R_o + tR_d) - r^2 = 0$$

$$R_d \cdot R_d t^2 + 2R_d \cdot R_o t + R_o \cdot R_o - r^2 = 0$$



2-) If the root is found, check if it is smaller than the minimum root so far.
3-) At the end, return the minimum root and object index.

How to run the program:

```
compile: g++ main.cpp -o main.exe
run: ./main.exe > main.ppm
```

The input is read from the input.txt file in the same folder.

File format:

Number of sphere
Color of the first sphere
Position of the first sphere
Radius of the first sphere
…

Example:

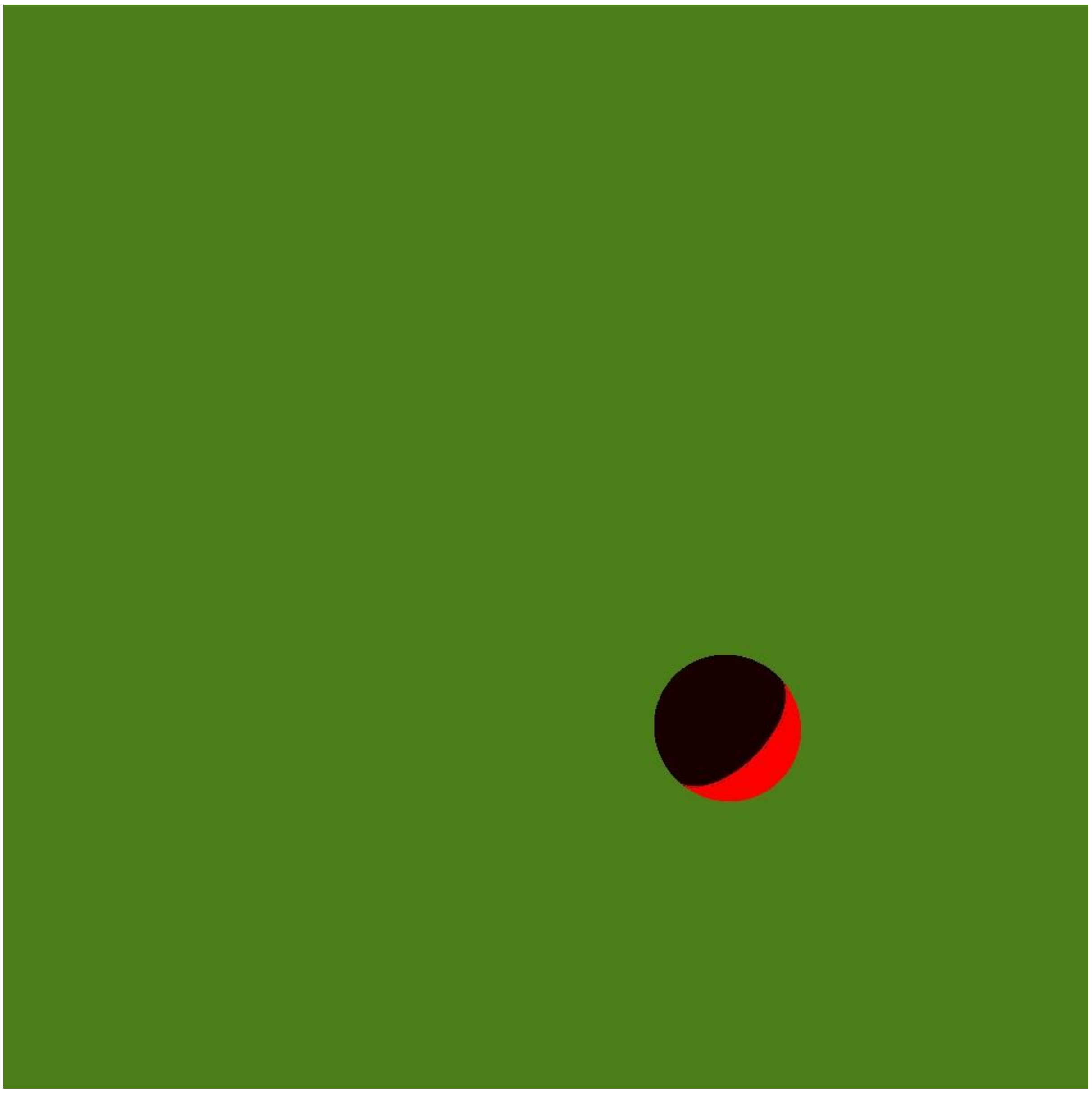
```
4
255 0 0
50.0 50.0 300.0
20
0 255 0
100.0 100.0 600.0
60
0 0 255
-100.0 150.0 150.0
15
0 255 255
-50.0 -50.0 300.0
30.0
```

RESULTS

1- Input:

```
1
255 0 0
50.0 50.0 300.0
20
```

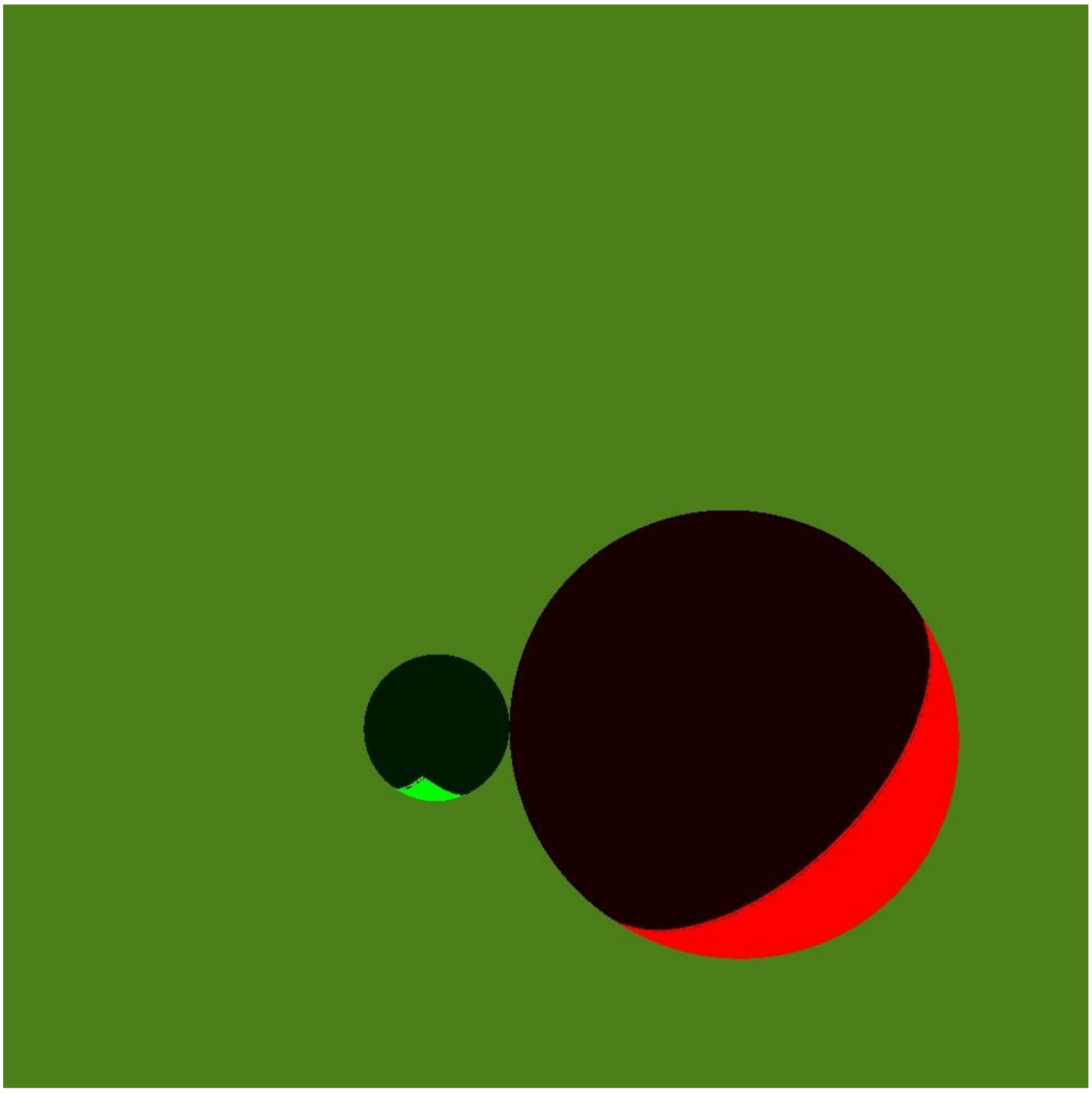
Output:



2- Input:

```
2
255 0 0
50.0 50.0 300.0
60
0 255 0
-30.0 50.0 300.0
20
```

Output:



3- Input:

```
4
255 0 0
50.0 50.0 300.0
20
0 255 0
100.0 100.0 600.0
60
0 0 255
-100.0 150.0 210.0
15
0 255 255
-50.0 -50.0 300.0
30.0
```

Output: