Cmpe460 Computer Graphics Homework-1 Report

Halil Karabacak

February 2023

1 Problem Definition

Main aim of this project is the implementation of the fundamentals of Ray Tracing algorithm on spherical objects. All the outputs will be presented are prepared with the following information :

i Center of the screen (0, 0, 100)
ii The eye is at the origin (0, 0, 0)
iii The screen extends from (-50, -50, 100) to (50, 50, 100)
iv Resolution of the screen is 1000 x 1000 pixels

2 Dependencies

Since no programming language is specified in the description, I used C++.

i **STB** : STB is used for exporting the resulting frame of the Ray Tracing in the PNG format.
ii **CMake** : CMake is used to generate makefile, and then generating the executable. In most UNIX kernel based operating systems, CMake is available. On Windows, it can be used with Visual Studio tools which enable you to create the solution file and generate executable for Windows as well.

3 Algorithm

Ray Tracing on spherical objects is pretty easy and straightforward. Here's the step-by-step explanation of the algorithm I implemented.

i Iterate through each pixel of the frame defined. Send a ray from eye to the each pixel.
ii Than having this ray's starting point, direction ; start iterating over the spheres to check if there is an intersection with a sphere. We also need to find the closest one to the eye because objects at the backward will be blocked by the ones in front of them. This process will give us the color of the pixel closest to the eye. If there is no intersection, simply either 0 or 1 will be assigned to the pixel in question.

BUT, how an intersection is detected ?

i **Ray Equation** : A ray is consists of an rigin point O (the eye) and a direction vector $\vec{D}$ (the direction from the eye to the pixel). A point P on the ray can be expressed as :

$$P = O + t \cdot \vec{D}$$

where t is a scalar parameter.

ii **Sphere Equation** : A sphere can be represented by its center C and radius r. The equation of a sphere centered at C is :

$$\|P - C\|^2 = r^2$$

This equation represents all points P that are on the surface of the sphere.

iii **Intersection Calculation** :

— Put the ray equation into the sphere equation to find the intersection point :

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

— Expand and solve for t using vector algebra :

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

— Calculate the discriminant Δ of this quadratic equation :
  — If $\Delta < 0$, there are no real solutions, and thus no intersection.
  — If $\Delta = 0$, the ray is tangent to the sphere.
  — If $\Delta > 0$, there are two intersections ; choose the one with the smallest positive t as the closest intersection point.

iv **Intersection Point** : If an intersection occurs, the intersection point $P_{\text{intersect}}$ can be found by substituting the computed t into the ray equation :

$$P_{\text{intersect}} = O + t_{\text{closest}} \cdot \vec{D}$$

After applying this algorithm for each pixel, we can fill the screen with intersections.

4 Implementation Details

Since we are iterating over 1000x1000 pixels, and this process is highly parallelizable ; I first added CPU parallelism with the simple **OMP**. Later decided to add a simple **Cuda** implementation as well, but not packaged it as the user may not have Cuda drivers, or maybe even a Nvidia GPU. But still added this implementation to the GitHub. It might be private at the time I send this file, but will be public after 1-2 days of the due of the submission.

My program reads input from **input.txt**, not directly from the command line.

5 How to Run the Code

If you are using **MacOS** or any **Linux**-based operating systems, follow these steps to get the executable :
— Create a directory named "build".
— Navigate to the "build" directory.
— Run the command : `cmake ..`
— Execute : `make`

These steps will generate an executable object.

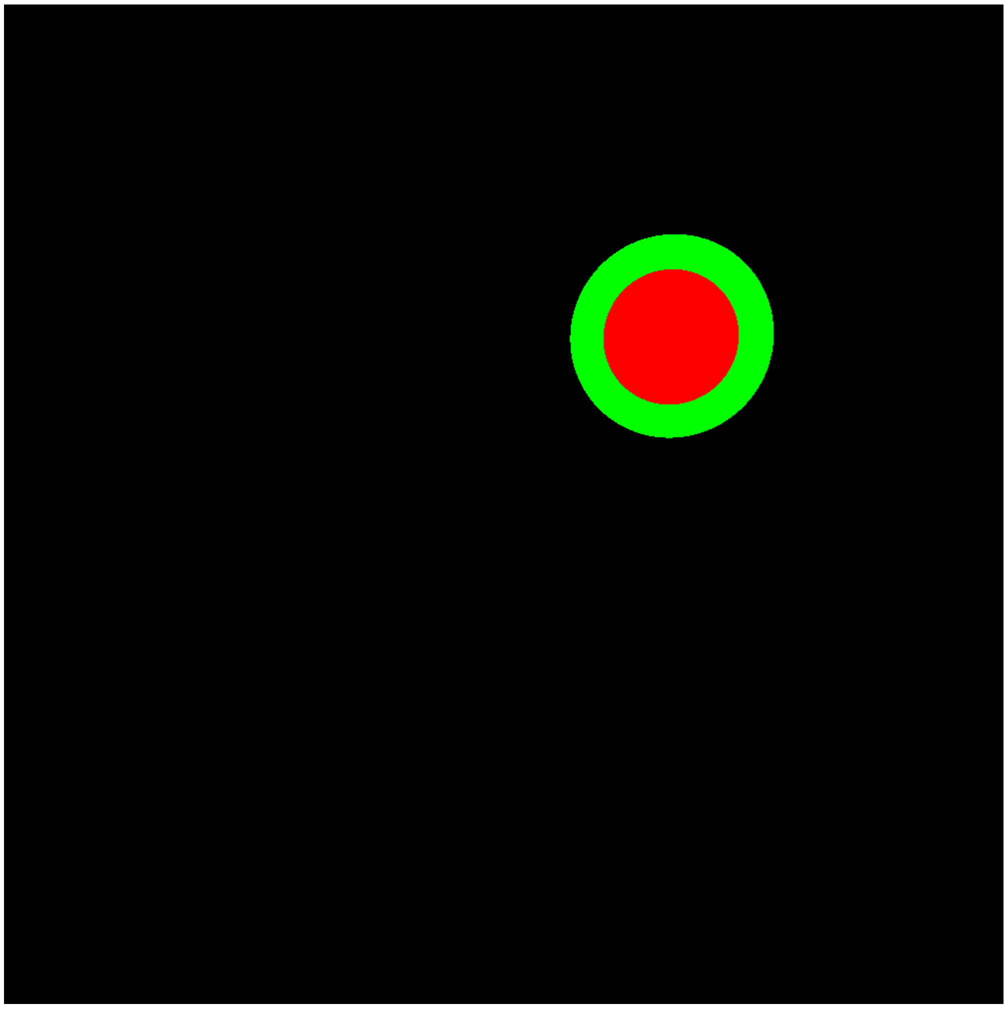
However, if you are using Windows, you'll need to install Visual Studio tools to utilize CMakeLists. Follow these instructions :
— Create a directory named "build".
— Navigate to the "build" directory.
— Run the command : `cmake ..`
— Open the solution file (.sln) and build it using Visual Studio.

I will submit an executable for both Windows and Linux based operating systems under executable folders.

6 Results

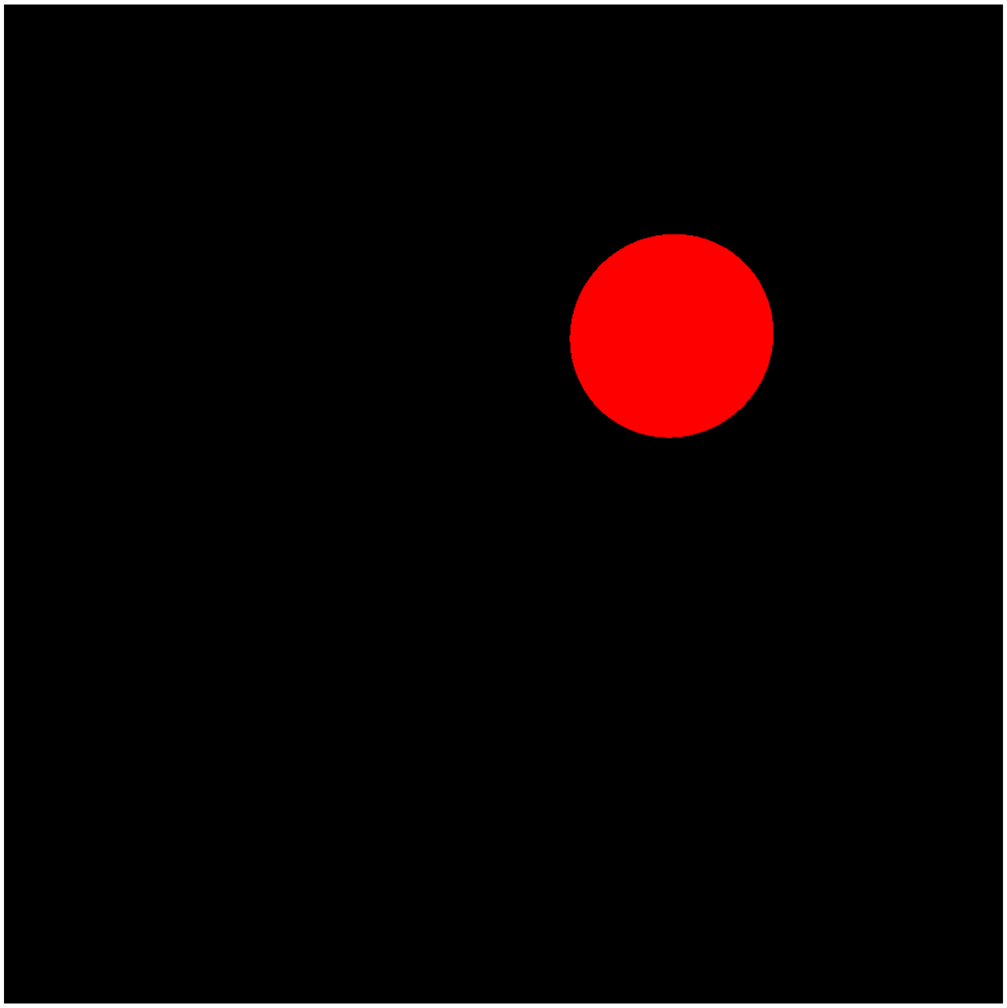
Some outputs of the program with different inputs are added.

i Input 1
2
255 0 0
50.0 50.0 300.0
20
0 255 0
100.0 100.0 600.0
60



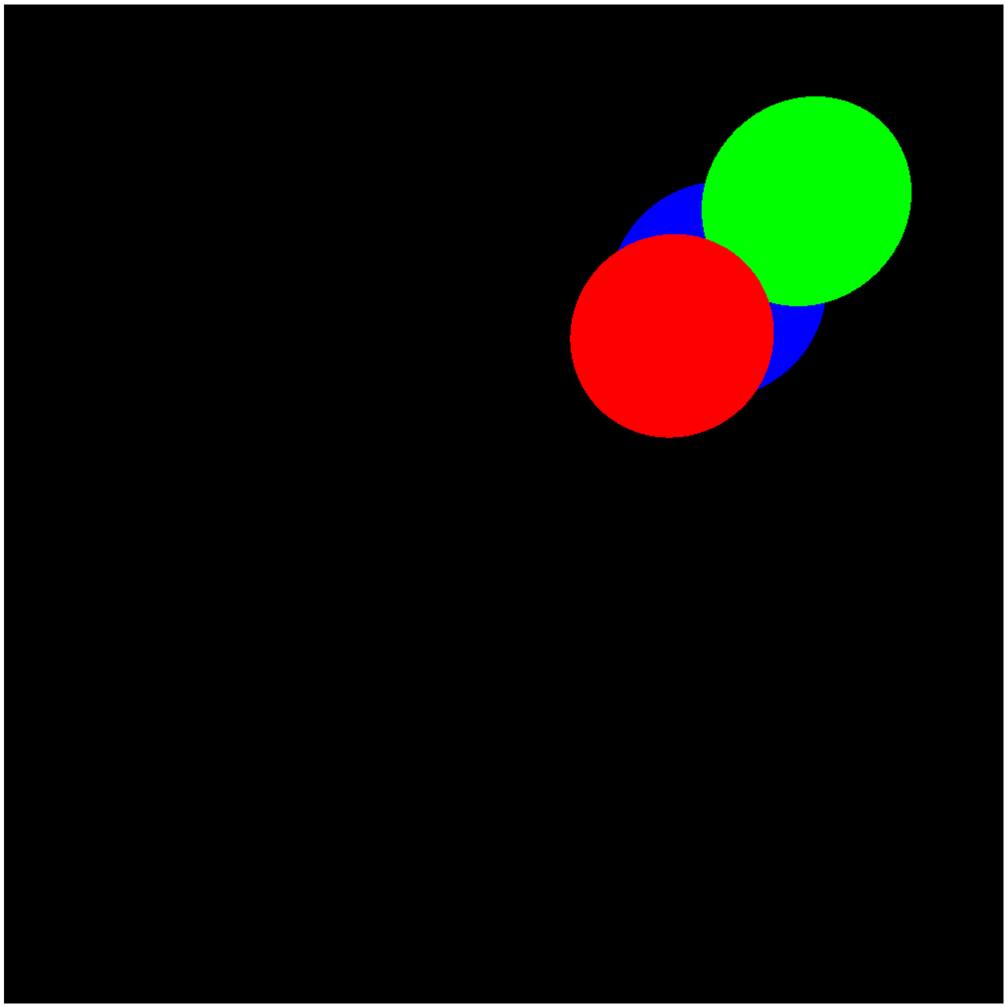
ii Input 2

```
2
255 0 0
50.0 50.0 300.0
30
0 255 0
100.0 100.0 600.0
60
```



iii Input 3

```
3
255 0 0
50.0 50.0 300.0
30
0 255 0
180.0 180.0 600.0
60
0 0 255
180.0 180.0 850.0
90
```



iv Input 4

```
4
255 0 0
50.0 50.0 300.0
30
0 255 0
180.0 180.0 600.0
60
0 0 255
180.0 180.0 850.0
90
0 255 255
90.0 90.0 425.0
40
```