Cmpe460 Computer Graphics Homework-2 Report

Halil Karabacak

February 2023

1 Problem Definition

Main aim of this project is the implementation of the of Recursive 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
- v A light source at (500, 500, 500) with color (1,1,1).
- vi Ambient light with color (0.3, 0.3, 0.3).
- vii Background color (0.0, 0.0, 0.1).
- viii Max recursive ray tracing depth of 5.

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 Scene Intersection (`sceneIntersect`)

The `sceneIntersect` function determines if a ray intersects any object in the scene. It iterates through all objects, calculating intersections, and returns the closest hit point, its normal, and the material properties.

1. Iterate through each sphere in the scene.
2. For each sphere, check if the ray intersects it.
3. If an intersection occurs, update the closest intersection point and its normal.
4. Check for intersection with a plane representing the ground.
5. Return the closest intersection point, normal, and material.

4 Lighting Calculation (`calculateLighting`)

Calculates the lighting at a point on a surface using the Phong reflection model, considering ambient, diffuse, and specular components.

1. Calculate the direction to the light source.
2. Perform shadow check by casting a ray towards the light source.
3. Calculate the diffuse lighting component based on the light's angle of incidence.
4. Return the combined color influenced by ambient and diffuse lighting.

5 Ray Casting (`castRay`)

Casts a ray from the camera into the scene to determine the color of a pixel based on intersections with objects and lighting.

1. Check for intersection with any object in the scene.
2. If no intersection is found, return the background color.
3. If an intersection is found, calculate the color at the intersection point considering lighting and material properties.
4. Calculate reflection if necessary and combine it with the object's color.
5. Return the calculated color for the pixel.

6 Rendering (`render`)

Renders the scene by casting rays through each pixel of the image plane, determining the color of each pixel.

1. Initialize a framebuffer to store pixel colors.
2. For each pixel in the image :
   (a) Calculate the direction of the ray from the camera through the pixel.
   (b) Cast the ray into the scene to determine the pixel's color.
   (c) Store the color in the framebuffer.
3. Write the contents of the framebuffer to an image file.

7 How to Run the Code

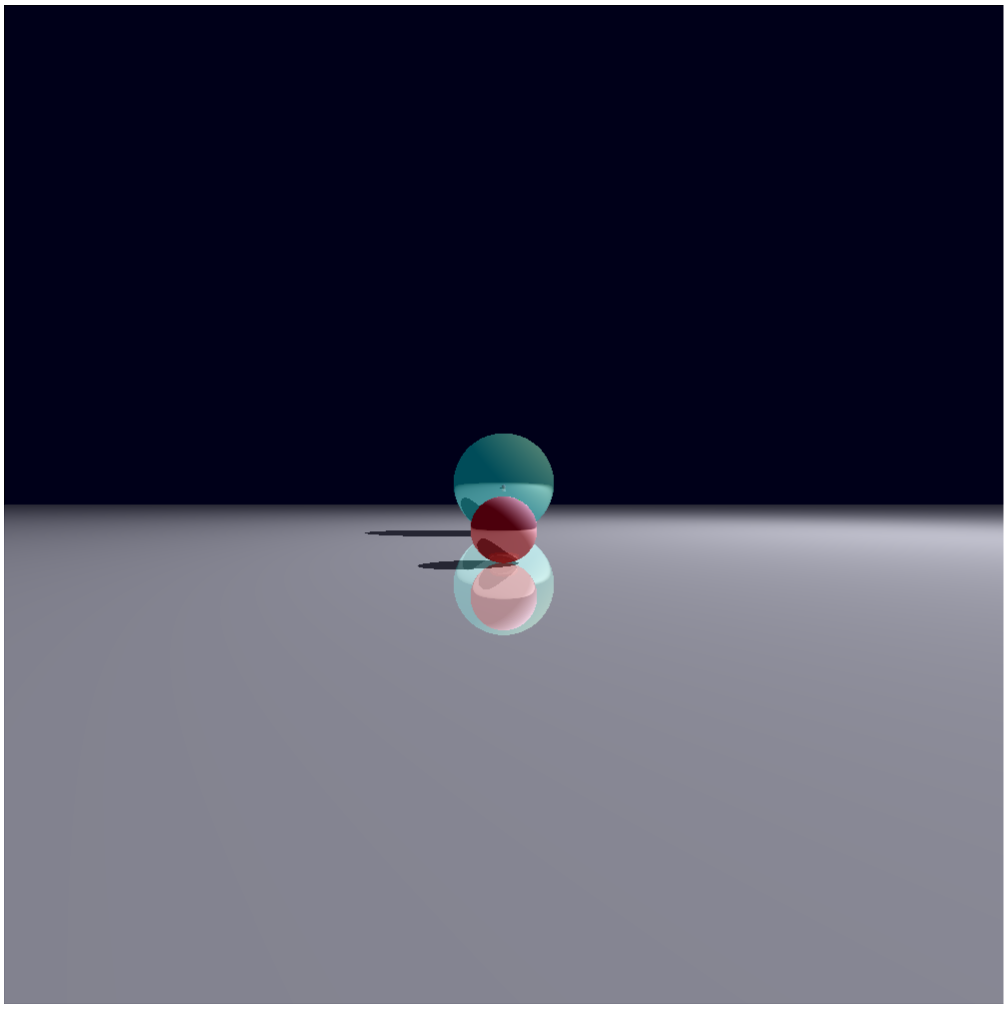
If you are using **MacOS** or any **Linux**-based operating systems, follow these steps to get the executable :
— Create a directory named "build".
— Navigate to the "build" directory.
— Run the command : `cmake ..`
— Execute : `make`
I will submit an executable for Linux based operating systems and program can be used with input.txt file.

8 Results

Some outputs of the program with different inputs are added.

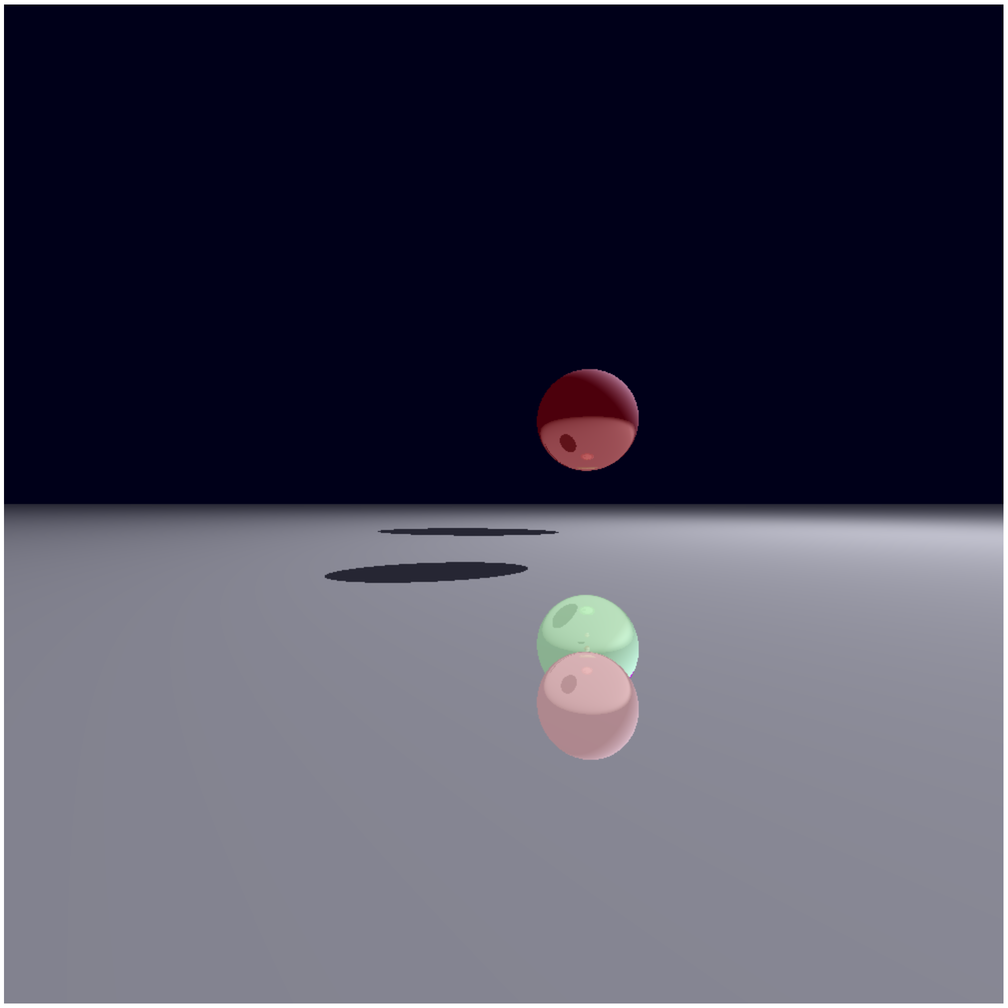
i Input 1

```
2
255 0 0
0.0 -15.0 300.0
20
0 255 0
0.0 25.0 600.0
60
```



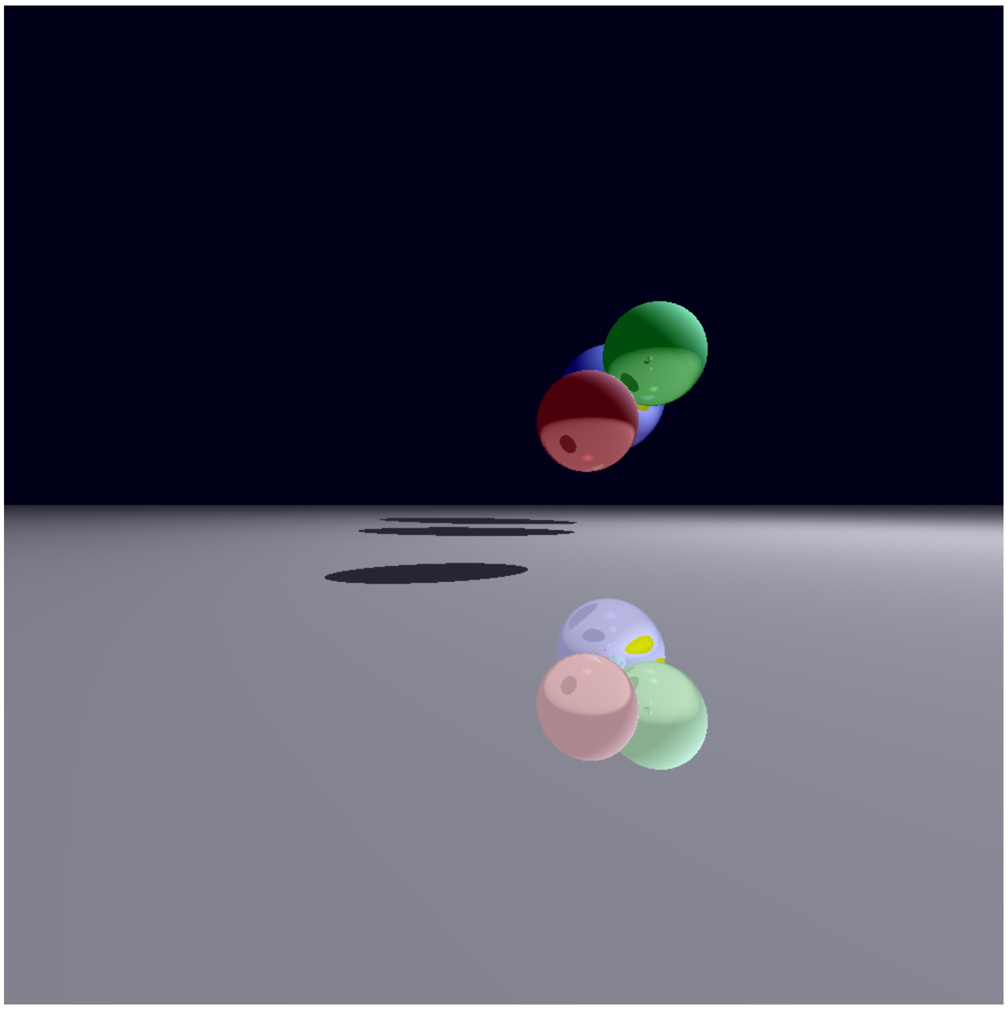
ii Input 2

```
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
```