Alperen Bağ 2018400279

CMPE 460 Project 1 Report

Compiling & Running

Code can be compiled with the following commands. I have used **OpenCV** library just for saving the output array as JPG image, therefore it is necessary for compiling.

- ***cd RayTracingRecursive/src***
- ***cmake .***
- ***make***

Run the executable with;

- ***./RayTracingRecursive***

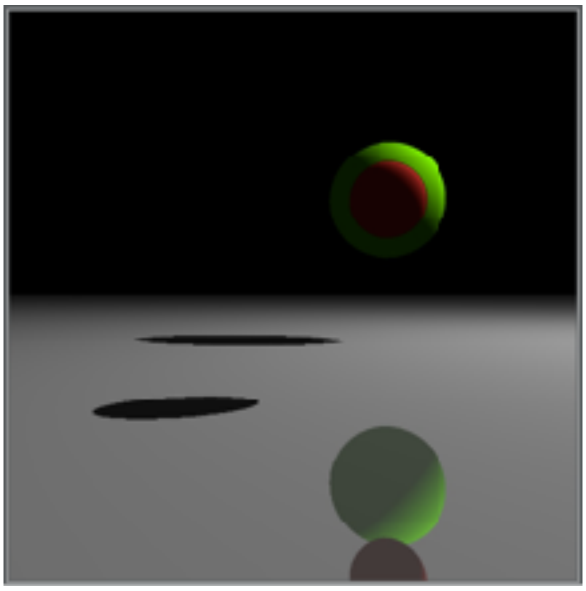
Output image will be created in the same folder with executable.

Sample Inputs and Outputs (Max Depth = 5)

- *Number of spheres*
- ***2***

- *Coordinate of sphere (input x y z coordinates with leaving one character space between them)*
- ***50 50 300***
- *Radius of the sphere*
- ***20***
- *Color of sphere (input RGB values with leaving one character space between values)*
- ***255 0 0***

- *Coordinate of sphere (input x y z coordinates with leaving one character space between them)*
- ***100 100 600***
- *Radius of the sphere*
- ***60***
- *Color of sphere (input RGB values with leaving one character space between values)*
- ***0 255 0***

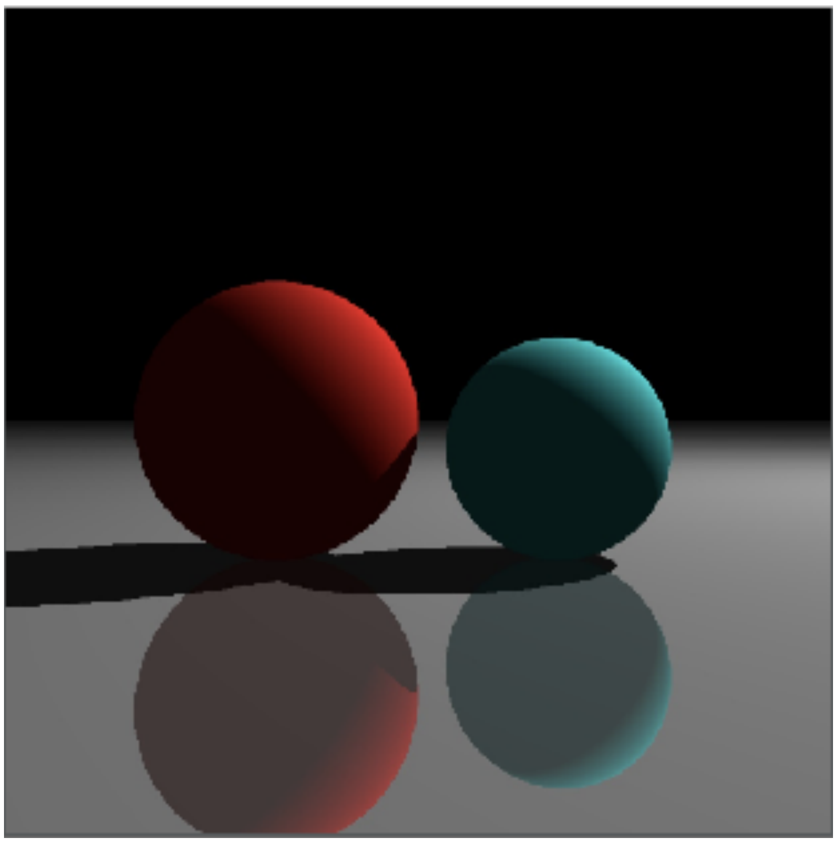


Alperen Bağ 2018400279

- *Number of spheres*
- ***2***

- *Coordinate of sphere (input x y z coordinates with leaving one character space between them)*
- ***-50 0 300***
- *Radius of the sphere*
- ***50***
- *Color of sphere (input RGB values with leaving one character space between values)*
- ***255 0 0***

- *Coordinate of sphere (input x y z coordinates with leaving one character space between them)*
- ***50 -10 300***
- *Radius of the sphere*
- ***40***
- *Color of sphere (input RGB values with leaving one character space between values)*
- ***0 255 255***

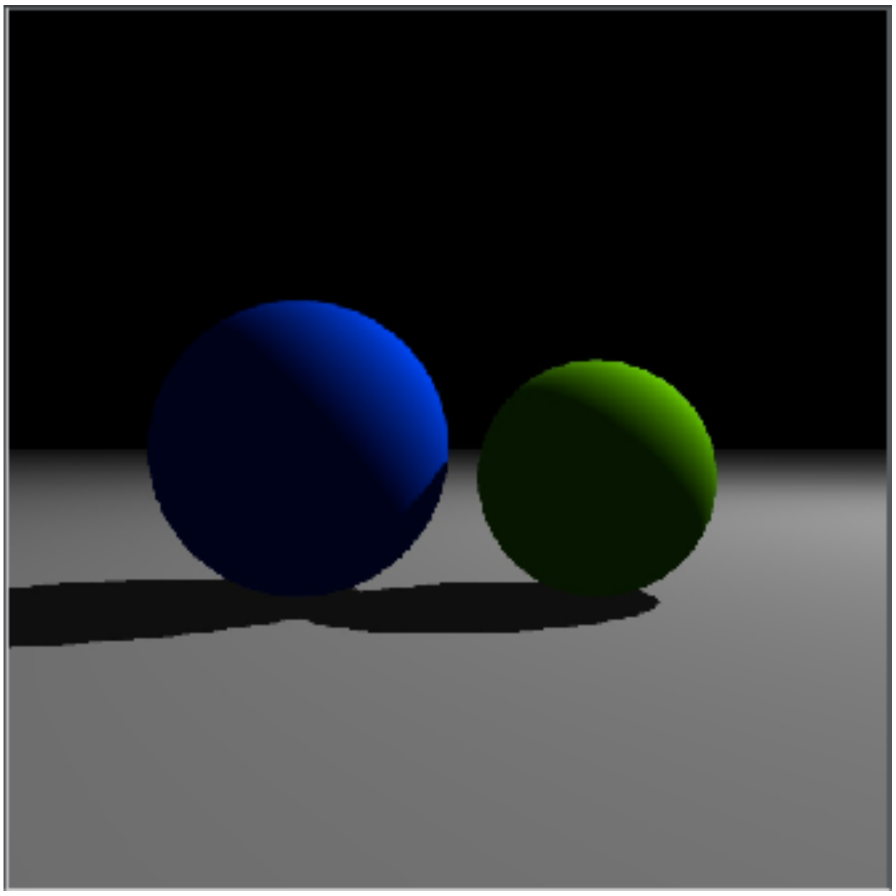


Sample Inputs and Outputs (Max Depth = 1)

- *Number of spheres*
- ***2***

- *Coordinate of sphere (input x y z coordinates with leaving one character space between them)*
- ***50 50 300***
- *Radius of the sphere*
- ***20***
- *Color of sphere (input RGB values with leaving one character space between values)*
- ***0 0 255***

- *Coordinate of sphere (input x y z coordinates with leaving one character space between them)*
- ***100 100 600***
- *Radius of the sphere*
- ***60***
- *Color of sphere (input RGB values with leaving one character space between values)*
- ***0 255 0***



Details

For the intersection detection, I used the same techniques in the first project. Additionally, I implemented Lambertian shading model and reflection model.

Lambertian Shading (Diffuse Color)

$$L = k_d \, I \ \max \ (0, \mathbf{n} \cdot \mathbf{l})$$

n is the normal unit vector of the hit point.
l is the vector from the hit point to the light source.
k$_d$ is the surface color.
I is the light source intensity.

Reflection Model (Recursive Rays)

I applied recursive ray tracing for the reflections. When a ray hits the object, another ray is generated from the hit point with a small shift to prevent the new ray from hitting the same hit point. In each generation, a depth counter is incremented. When the counter reaches max depth, generation is stopped. I used 5 as max depth value in my program.

In order to find the reflected vector, the following formula is used.

$$\mathbf{r} = \mathbf{d} - 2\,(\mathbf{d} \cdot \mathbf{n})\,\mathbf{n}.$$

d is the primary ray.
n is the unit normal of the hit point.

* Formulas were taken from our textbook.

Final Color Calculation

Color of each pixel is calculated by averaging surface colors of each point that is hit by the primary ray and the rays that are generated recursively.