Alperen Bağ 2018400279

CMPE 460 Project 1 Report

Compiling & Running

Executable is located in “exe” folder, however the code in “src” folder can be compiled again with the following commands.

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

Run the executable with;

- ***./RayTracing***

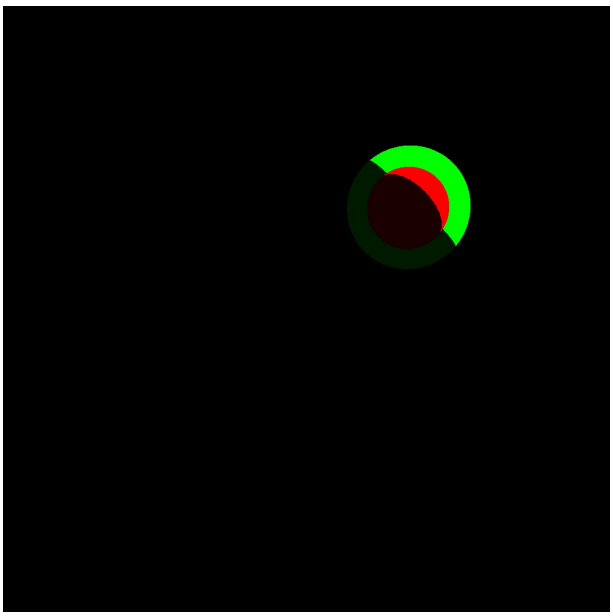
Output image will be created in the same folder with executable.

Sample Inputs and Outputs

- *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***



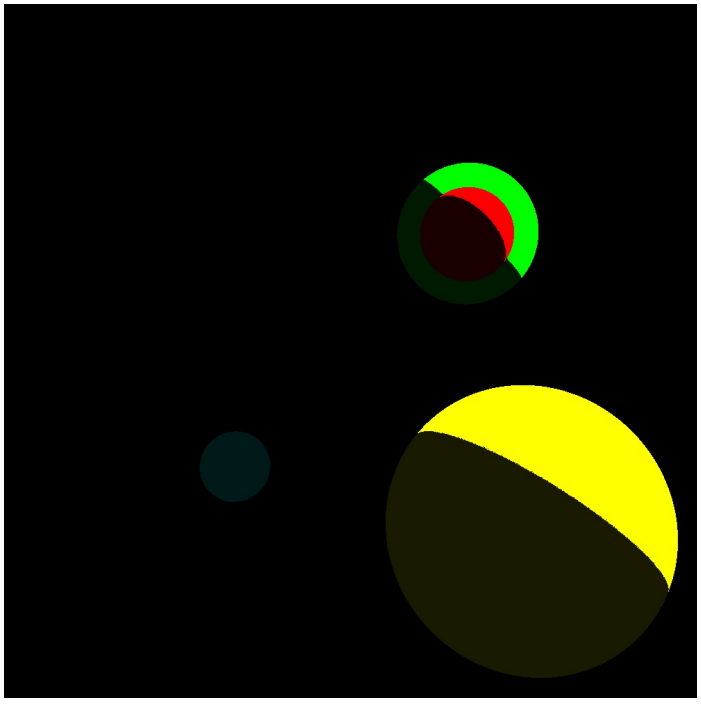
- *Number of spheres*
- ***4***

- *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***

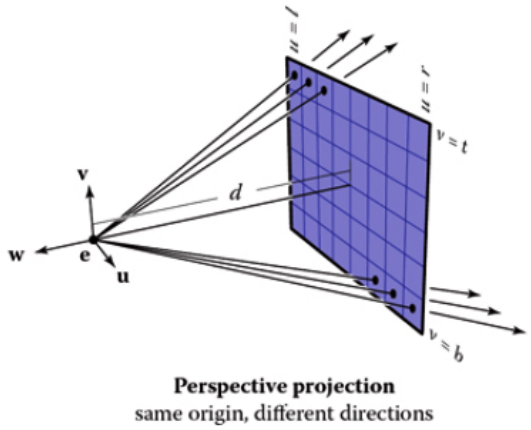
- *Coordinate of sphere (input x y z coordinates with leaving one character space between them)*
- ***-100 -100 600***
- *Radius of the sphere*
- ***30***
- *Color of sphere (input RGB values with leaving one character space between values)*
- ***0 255 255***

- *Coordinate of sphere (input x y z coordinates with leaving one character space between them)*
- ***75 -75 300***
- *Radius of the sphere*
- ***60***
- *Color of sphere (input RGB values with leaving one character space between values)*
- ***255 255 0***



Details

For the generation of the image, I used perspective projection and ray tracing method. All rays have the same origin indicated with **e.** The coordinate system I have set is very similar to one below, the only difference is that **w** vector is in the same direction with **d**.



Perspective projection
same origin, different directions

d = 100
l = -50
r = 50
b = -50
t = 50

This figure was taken from our textbook (Fundamentals of Computer Graphics, Steve Marschner, Peter Shirley, 4th ed.)

Rays are generated with the following formula.

ray = **e** + *u***u** + *v***v**

$$u = l + (r - l) * (i + 0.5)/1000$$
$$v = b + (t - b) * (j + 0.5)/1000$$

- *i* and *j are the index of the pixel through which the generated ray goes. Due to the camera-based coordinate system, Bottom-Left pixel is (i=0, j=0) and Top-Right pixel is (i=999, j=999).*

Then I found the interaction points of each sphere with the generated ray and took the closest intersection point to camera. In order to decide whether the hit point (**p**) is under the shadow or under the light, I generated another ray from the light source located at (500, 500, 500) to the hit point (**p**). If the new hit point (**h**) of the ray sent to the scene is the same as **p**, then we can say that **p** is under the light, otherwise it is under the shadow.

Because of the precision loss due to float operations, **h** and **p** might not be exactly same. Therefore I assume **p** is under the light if $|p.x - h.x| < 0.1$ && $|p.y - h.y| < 0.1$ && $|p.z - h.z| < 0.1$.

Points under the light has the RGB value given by the user, points under the shadow has the RGB value given as input, but divided by 10.