CS 405 - Computer Graphics

Ray Tracer

Can Ince - 14241

The time complexity of ray tracing is highly depends on the width and the height of the desired image, number of collusions between the ray and the scene objects and number of reflections. Within the two embedded for loop, collusion and getcolorAt functions were called recursively until there is no intersection. Most of the critical steps were commented in the render.cpp file.

```
Cans-MacBook-Pro:terminal Can$ a.out -o
Starting..
Config file success
Config file: 640 480 1.10 0.1 0.9 3 1.5 -3 -7 12 12 0.5 0.5 1 0.6 0.2 0.3 0.9 1 0
.5 1
Width is:640
Height is: 480
Scene objects has been inserted. Rendering...
Output file has been created in 14041.8 seconds
```

I created a config file for the position of camera, lights, colors and objects in the scene. Configurations can be manipulated easily depending on the desired output therefore it can be used in the future projects. An example state of the config file can be seen in the picture above. It can render most of the configurated parameters in less than 10 minutes.

The time complexity differs within the given input therefore I will assume only the worst and best hypothetical cases. If the ray hits the every object in the scene then the complexity becomes width*height*scene object size^2*light sources*interactions. Hypothetically if I assume that the maximum value that the mentioned parameters can get is N, then I can say that the complexity of my ray tracing sums up to O(N^7).

Pre-processing is the process that is done before rendering the scene such as adding lights, positioning

cameras, designate the RGB values. Computing the image is the part where the rendering is done. Sending the rays from camera and computing the color values at each pixels, computing the reflections and shadows are all done in computing/rendering.

The most critical parameters of the ray tracer are

- Width and height of the desired output
- Anti-aliasing depth
- RGB scheme
- Number of the scene objects(position parameters of the camera, evidently)

Note that the images has created using bitmap instead of TIFF since libtiff was quite buggy and hard to resolve.
Images were computed using anti-aliasing depth of 4 therefore 16 per pixel.