Programming Assignment #3
Raytracer

CSE 458 Computer Graphics, Fall 2010

1 Introduction

This assignment will require you to build a raytracer; a program that takes a high-level description of a scene (where the objects, lights, and camera are) and produces the image seen through that camera. You will do this without using OpenGL to render the image: you will use mathematical operations to calculate the RGB value that should be at each pixel in the desired image.

According to wikipedia: In computer graphics, ray tracing is a technique for generating an image by tracing the path of light through pixels in an image plane and simulating the effects of its encounters with virtual objects. The technique is capable of producing a very high degree of visual realism, usually higher than that of typical scanline rendering methods, but at a greater computational cost. This makes ray tracing best suited for applications where the image can be rendered slowly ahead of time, such as in still images and film and television special effects, and more poorly suited for real-time applications like computer games where speed is critical. Ray tracing is capable of simulating a wide variety of optical effects, such as reflection and refraction, scattering, and chromatic aberration.

2 Requirements

Your raytracer must read a description of a file of the .ray format (see your starter code). An example .ray file is shown in Fig. 1. This file format includes descriptions of camera, background color, ambient light, diffuse/specular lights, and spheres & triangles with material properties and texture images. Your raytracer must handle all of these accurately to receive full credit. With your starter code comes a set of sample scenes useful to test that your raytracer can handle all requirements. You will need to make two new .ray files of your own, and you will need to submit renderings or animations of your new scenes (as well as renderings of the sample scenes) along with your code and readme.

Figure 1: A Sample Ray File

```
# author: Kyle

Background {
        # Color of the background
        color 0.15 0.15 0.25 # blue-gray
        # low intensity white light
        ambientLight 1 1 1
}

Camera {
        eye 0 0 0
        lookAt 10 0 0
        up 0 0 1
        fovy 45
}

Lights {
        Light {
                position 10 0 5
                color 1 1 1
        }
}

Materials {
        Material {
                textureFilename NULL
                diffuseColor 0.5 0 0
                specularColor 0.3 0.3 0.3
                reflectiveColor 0.3 0.3 0.3
                shininess 50
                transparentColor 0 0 0
                indexOfRefraction 0.6
        }
}

Group {
        Sphere {
                materialIndex 0
                center 10 0 0
                radius 2
        }
}
```

Features are assigned points as follows:
10 pts- Sphere intersection.
10 pts- Triangle intersection.
10 pts- Triangle texture mapping.
10 pts- Sphere texture mapping.
10 pts- Diffuse and specular shading.
10 pts- Shadows.
10 pts- Recursive reflection/refraction.
10 pts- Code structure/Comments.
10 pts- Animation (or still) creativity.
10 pts- Two (or more) original scenes.

3 Instruction

For help, see the lecture notes on Angel for the project slides: CMPSC458-Raytracer.ppt

4 Extra Credit

As usual, be creative, describe everything you do in the readme file, and make things look nice to get more points. Some ideas:

10 pts- Good antialiasing.
10 pts- Animation with motion blur.
15 pts- Depth of field.
15 pts- Blinn blobs.
15 pts- Multiple object groups with bounding spheres.
15 pts- BSP Tree for speed up.
7.5 pts- Up to two additional object types (10 each).
15 pts- Procedural texture mapping.
?? pts- Your idea/ talk to me to see how many pts.