CSC 473
Programming Assignment 1 part 1

Due Sunday April 14th, at 11:59pm

Overview

In this assignment you will implement the basic functions of a ray tracer.
In addition to teaching the basics of ray tracing, this assignment will be used as the base for following assignments. Since you will be adding and reusing your code, it is advised that you write your code in a clean, structured object-oriented fashion.
Building on your work in lab, you will be writing a ray caster. You do not need to compute lighting, just color the pixels the pigment of the closest intersections with a sphere or plane. Your code will need to

- Parsing the scene description file, specially, simple.pov which is linked on the website.
- Compute ray-sphere and ray-plane intersections
- Color pixels based the pigment of the closest intersection

Ray Object Intersections

After parsing the scene file and creating the necessary data structures, your program should begin casting rays. The camera object should, with knowledge of the output image size and a given pixel, be able to cast the necessary rays and return the appropriate rgb color value for the pixel. In the assignment you will only cast one ray per pixel (this will change in later assignments however).
The rays should be represented by a class object. To cast a ray, simply traverse the scene object list (also represented by C++ objects) testing for intersection with each object. Each derived geometry object class should have its own intersection routine that takes a ray, performs an intersection test and returns the closest intersection (if one exists).
The closest intersection will be colored based on the pigment of the hit object.

What you should hand in

Your program should have the following syntax:

```
  raytrace [options]
```

where the options are:

```
imageWidth
imageHeight
 -I inputFilename.pov
```

So raytrace 640 480 -Isample.pov will render a 640x480 image file, sample.ppm, consisting of the scene defined in sample.pov.
Image files should be named inputFilename.ppm (or .tga, etc.)
Sample input files and images are linked on the class webpage. The pictures are generated by Povray and will not look identical to your output (due to differences in the shading model, etc.). You will be required to turn in rendered versions of these files

(ppm format, 640x480) along with the amount of time required to render (in the README file). The README file should also contain a description of which parts of the ray tracer that you believe are working, partially working, and not implemented. Make sure that your code works with files that contain spheres and planes.

Debugging

This program is large and you should attempt to approach things in a piece wise manner. You should attempt to get one feature working before attempting the next feature. Working in this code-test manner will help you in that when a problem arises, you will only have one problem to fix. Additionally, if you have a problem, try to look at the output (or internal data values) to determine a solution rather than simply tweaking the equations and code.

You may wish to start with a file that is simply a sphere and a camera. Code the camera class and sphere class (derived from a generic geometric object class). Now, write the sphere intersection code coloring all of the intersections a constant color. Test this and make sure it works and then proceed to implement the other intersection tests.