Homework 4

Due: April 26, 2010

NOTE: Your program *must* conform to everything presented in this document in order to receive any credit. You may, however, generate arbitrarily verbose debugging output of various kinds, including additional files. You may also wish to recognize additional command-line arguments for your own purpose. Just don't do anything that would prevent your program from being run by a shell script without human intervention. If you use additional files or command line arguments, you must modify `Makefile` so that your program compiles correctly. Remember that you will not receive credit if your program refuses to compile.

1 Description

In this assignment, you are to implement a simple ray tracer. Your program should be called `prog4` and should accept the following command line arguments:

```
prog4  nx  ny  inputfile  outputfile
```

where,

`nx`, `ny`	:	width and height of the output image
`inputfile`	:	command file to use
`outputfile`	:	the output ppm file

The command file (i.e., `inputfile`) will have various parameters for your ray tracer to use. An example command file looks like:

```
lookfrom   0   0   200
lookat   0   0   -1
vfov   90
vup   0   1   0
light   0   0   200
background   0   0   0
spheres spheres.sph
```

The format of the spheres file (i.e., `spheres.sph`) is:

$$c_x \;\; c_y \;\; c_z \quad \text{radius} \quad \text{r g b} \quad [\text{texture map file} \mid \text{none}]$$

$$\vdots$$

That is, a sphere is represented by the center coordinates, radius, RGB color, and the texture map (or none) it uses. Thus, an example spheres file might look like:

```
sphere   0   -300   -1200   200   0.8   0.8   0.8   texture1.ppm
sphere   -80   -150   -1200   200   0.7   0.7   0.7   none
sphere   70   -100   -1200   200   0.9   0.9   0.9   texture2.ppm
```

For each pixel, you need to shoot a ray according to the viewing parameters and find the ray color to render the pixel. The ray color should be computed according to the texture map (or from the color of the sphere if no texture map is used, i.e., `none`) using Lambertian illumination model.

2 Extra feature for bonus credits

Add the functionality for triangles. To do so, the `inputfile` should contain

```
triangles triangles.sph
```

The format of the triangles file (i.e., `triangles.sph`) is:

$$x_0 \; y_0 \; z_0 \;\; u_0 \; v_0 \quad x_1 \; y_1 \; z_1 \;\; u_1 \; v_1 \quad x_2 \; y_2 \; z_2 \;\; u_2 \; v_2 \quad \text{texture3.ppm}$$

$$\vdots$$

where, (x, y, z) are the vertex coordinate of a triangle and the corresponding (u, v) is the texture coordinate.

3 What to Submit

Please put your name and email address in each item (except texture maps) below:

1. All of your source code files needed to compile your program
2. `inputfile`, spheres file, and triangles file
3. All of your texture maps needed to run your program
4. `Readme` containing
   - Exact commands to compile and execute your program on separate lines so that a script can copy and paste to test your code automatically.
   - Anything that you want me to know before grading your code, such as new features, extra credits, user interfaces, etc.
   - Plain ASCII text file (No doc files)

Note: Get the ray tracer without texture mapping first.

- Set up the viewing parameters as simple as possible. For example,
  - Put your eye on positive z-axis.
  - Put your virtual film at $z = 0$.
  - Put the light at the same location as your eye.
- Put just one sphere centered at the negative z-axis.

Once it works generating the image as expected, test it with more spheres at various locations and the light at various locations. Your eye should stay on the positive z-axis for simplicity (we are not implementing the arbitraty view point). Add texture mapping only when it works with your complex scene.

Note: Do not try the extra feature unless all your programs work as expected.

Very important note: If your program fails to compile and/or fails to run, you will not receive any credit for this homework.