CPSC 314, Project 3: Raytracer

Out: Fri 8 Feb 2013
Due: Fri 1 Mar 2013 5pm
Value: 8% of final grade
Points: 100

In this assignment you will implement a simple raytracer that supports spheres and planes. The raytracer should cast primary rays into the scene, which spawn secondary reflection and shadow rays.

Up to four extra credit points are available for extending your program to support non-photorealistic shading or refraction, or for designing your own scene.

Template

The template code has three main subdirectories, `src`, `include`, and `demo`. The `demo` directory contains scene descriptions in the simple `.ray` format, describing the following kinds of objects: Resolution, Camera, Material, Sphere, Plane, and PointLight. The comments in those files describe the format. The directory also contains reference images created by the solution code.

You will be making additions to three of the template code files in the `src` directory: `FileParser.cpp`, `Primitives.cpp`, and `Raytracer.cpp`. You do not need to make any changes to the fourth source file, `main.cpp`.

The `README.txt` contains instructions for compiling and running your raytracer. The `raytracer` binary takes two optional arguments: the name of the scene description, and the name of the output PPM image file. The defaults are `demo/testscene.ray` and `demo/testscene.ppm`. The output of the program is two image files, a color image and a black-and-white depth map image that you might find useful for debugging. The name of the depth map image file is `filename_depth.ppm`, where `filename.ppm` is the specified output image file.

Requirements

- **14 pts** Complete the parser in `FileParser::parse`. The template code includes PointLight parsing as an example. You need to implement parsing for for Material, Sphere, Plane, Camera, and Resolution. After you have implemented these function, uncomment the `cout` lines that report on parsing results (allowing you, and the grader, to check that the parsing has occurred correctly).

- **14 pts** Implement `Sphere::intersect`

- **14 pts** Implement `Plane::intersect`

- **16 pts** Implement the missing part of `Raytracer::shade` that does a lighting calculation to find the color at a point. You should calculate the ambient, diffuse, specular, and emission terms.

- **14 pts** Implement the shadow ray calculation in `Raytracer::shade`.

- **14 pts** Implement the secondary ray recursion for reflection in `Raytracer::traceRay`

- **14 pts** Modify the orthographic camera code in `Raytracer::raytraceScene` so that it does perspective projection when generating the rays into the scene.

Extra Credit [4 pts]

- **(1 pt)**: Implement nonphotorealistic shading: cool-to-warm colors, silhouette edges, and creases where objects intersect each other.

- **(1 pt)**: Design an interesting scene.

- **(2 pts)**: Add support for refraction.

The comments in the template code above each section where are you required to add code contain the details of the specification. They also contain many hints. The recommended order of implementation is exactly the order we list the items above.

Downloads

- Linux version, with Eclipse: Download the template file and unpack it with the command

  `gunzip < p3_template_linux.tar.gz | tar xvf -`

  Open up Eclipse. You do not need to create an empty new project yourself. Under the File menu, pick the Import item. In the Select dialog that pops up, under the top General category you should select 'Existing Projects into Workspace'. Use 'Select root directory' to indicate the directory where you unpacked the template code. Then hit 'Finish'.

- Linux version, with command-line makefiles: Download the template file and unpack it with the command `gunzip < p3_template_linux.tar.gz | tar xvf -`. Type `cd src` to enter that directory, and then `make` to compile. Edit the files with the editor of your choice.

- Mac version: Same as either option above, except use `p3_template_mac.tar.gz` for the template file.

- Windows version, with Visual Studio: Download the template file `p3_template_win.zip`. Double-click to unarchive it, and then double-click on the `p3.soln` file to launch VS.

Handin/Grading/Documentation

The grading, required documentation, and handin will be the same as with the previous projects, with the following exceptions. First, use the command 'handin cs314 p3'. Do submit the images made by your program for the two example scenes provided. If you design extra-credit scenes, also submit the .ray file for them.

As before, the main requirement for style is to avoid producing near-duplicate code.

We provide the templates for multiple platforms as a convenience for those who want to do most of the work remotely, but as before you are reponsible for making sure that the code you hand in compiles and runs on the Linux machines in the lab.