CSCD18F Computer Graphics, Fall 2008

Assignment 3 (10% of course grade)

Part A (programming): Due before class (9:59am) on Friday, November 14th, 2008.
Part B (written) **will be out on November 12th**: Due 11:59pm on Wednsday, November 19th, 2008.

Part A: Basic Ray Tracer [75 marks in total]

Your Programming Task

Your task is to implement a basic ray-tracer and render a simple scene using ray casting and local shading (note no global lighting or shading yet). The starter code sets up a scene comprising of an ellipsoid and a plane, being illuminated by a point light source. Your job is to render the scene by implementing code fragments required for object intersections and Phong shading.

You will have to implement the following code fragments:

(a) [10 marks] Ray casting.

(b) [20 marks] Intersection code for ray-sphere intersection (it should also work for affinely deformed spheres).

(c) [15 marks] Intersection code for ray-square intersection (it should also work for affinely deformed squares).

(d) [15 marks] Phong shading for a point light source (including ambient, diffuse, and specular components)

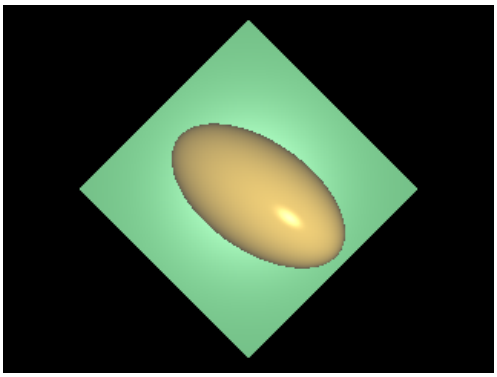
To demonstrate the working of your program, you should generate three different types of renderings:

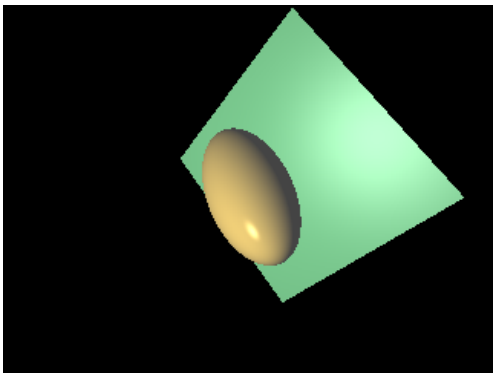
1. a *scene signature* where each pixel shows a unique color identifier for the first object hit (or background). This gives you an impression of the relative positions of the camera and the objects.
2. a rendered scene with only the diffuse and ambient components of the Phong model.
3. a rendered scene with all three terms of the Phong model.

In addition to specifying your scene, the starter code has been written to generate two distinct views (images) every time you run it. Thus, for each type of rendering above you will generate two views, comprising a total of six images. In your electronic submission, please call these image files `sig1`, `sig2`, `diffuse1`, `diffuse2`, `phong1` and `phong2`. Below are two images of what the Phong rendered scene should look like.

Finally, ray-tracing is compute-intensive, so debug your code on small images, e.g., 200x200, and debug ray intersections using the scene signature.

Also, when writing your code, keep in mind that you will need to extend it in various ways in Assignment 4, hence make sure your implementation is modular and re-usable. To encourage this [5 marks] will be devoted to programming style and comments. Make sure you read the helper code carefully and only implement the functions that are necessary, i.e. try to avoid creating unnecessary structures or classes.





Helper Code

To get started, we have created a simple demo for you. The demo sets up the scene and basic class structures required. Note that the helper code is written in C++ and does not use OpenGL. It will also provide you with a template Makefile for compilation and linking of your programs.

To unpack and compile the helper code on `mathlab`, download the file `a3_starter.zip` and use the following commands

```
unzip a3_starter.zip
cd a3/raytracer
make
raytracer
```

Turning in your Solution to Part B

All your code should remain in the directory a3/raytracer. In addition to your code, you must complete the files *CHECKLIST* and *REPORT* contained in that directory. *Failure to complete these files will result in zero marks on your assignment*.

[10 marks] The *REPORT* file should be a well-structured written (or diagramatic) explanation of your design, and what the marker will see; as well as what the relevant files are, as the file extensions will depend on the image file format used, which should be one of the well-known BMP, TIFF, PNG, or PPM formats. The description should be a clear and concise guide to the concepts, not a simple documentation of the code. In addition to correctness, you will also be marked on the clarity and quality of your writing. We expect a well-written report explaining your design and code structure.

Note that this file should *not* be thought of as a substitute for putting detailed comments in your code. Your code should be well commented if you want to receive full (or even partial) credit for it.

To pack and submit your solution, execute the following commands from the directory containing your code (i.e., `a3/raytracer`):

```
cd ../../
tar cvfz a3_solution.tgz a3
submit -N A3a cscd18f08 a3_solution.tgz
```

Compatibility

All of your assignments **must** run on `mathlab` (Linux). You are welcome, however, to develop on other platforms (and then port to `mathlab` for the final submission). This sample code is designed to work on

Linux, but should be portable to other Unix platforms (including Windows with Visual C++). If you are planning to develop for windows with Visual C++, we have included some starter code to help you out.

If you develop on a different platform, be sure you know how to compile and test your code on `mathlab`, well before the deadline. Your assignment must run on the `mathlab` Linux configuration. Several marks will be deducted if your code does not compile and/or run without modification. If the marker cannot easily figure out how to compile and execute the code, it will most likely receive zero marks. If you choose to develop the assignment in Windows, test porting your code to Linux long before the deadline, perhaps even before you have finished the assignment. Often bugs that are "hidden" when compiling on one platform, make their presence known by crashing the application on a different platform. We will not be sympathetic to porting problems that you have at the last minute.