CSC418 Winter 2008

Assignment 4
Ray-Tracer

Due April 7th at 11:59pm.

<u>PART 1</u>

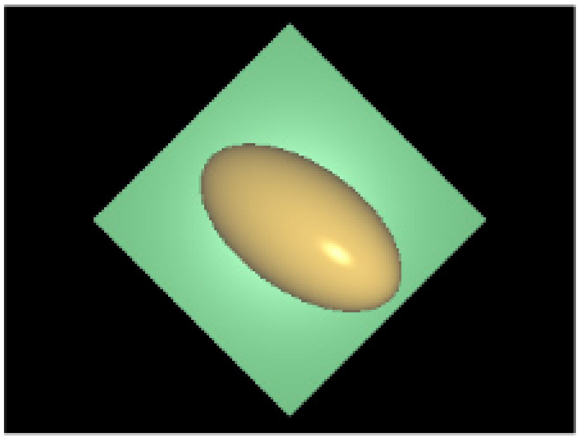
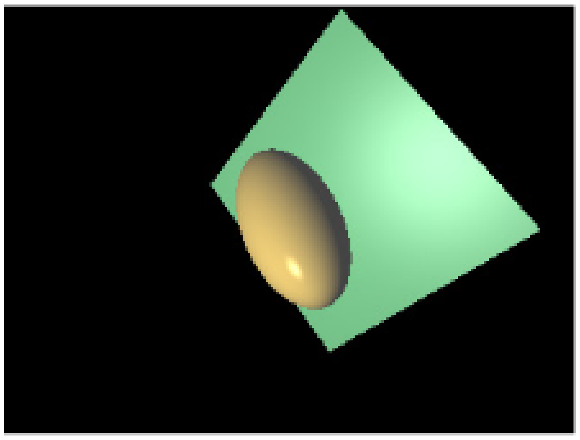
In this part of the assignment, you are required to complete the implementation of a basic ray-tracer. We will provide with starter code that sets up a scene comprising of an ellipsoid and a plane and illuminated by a point light source. Your job is to render the scene by implementing code fragments for ray generation, object intersection and Phong shading.

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

1. a scene signature, where each pixel shows a unique colour identifier for the first object hit (or background). This gives you an impression of the relative positions of the camera and objects.
2. a rendered scene with only 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 (image) every time you run it. Thus, for each type of rendering above you will generate 2 views, comprising a total of six images. In your electronic submission, please call these image files `sig1`, `sig2`, `diffuse1`, `diffuse2`, `phong1`, and `phong2`. You should also include a `readme.txt` with you submission to explain what the marker will see.

Below are two images of what the Phong rendered scene should look like.

For this part submit:

1. Modified source code in an archive file. Label it `part1_src`.
2. 6 image files described above.
3. Readme file described above.

PART 2

Extend your ray-tracer to implement the following features:

1. More primitives: cylinder, polygon, torus
2. Shadows
3. Reflection and refraction for mirror and transparent surfaces

Implement at least two of the following extensions:

4. Anti-aliasing (multi-sampling)
5. Soft-shadows and area lights
6. Acceleration structure
7. Texture mapping
8. Bump mapping

Feel free to modify any part of the starter code for this. You have total flexibility in that respect for this part of the assignment. The program will load and parse a scene description file. This file describes the elements in the scene and their properties. It is up to you to design the format of the scene file.

In order for the marker to grade you fairly and easily, make sure you submit, for a particular feature, an image which demonstrates that feature exclusively. For example, if you are showing your implementation of texture mapping, avoid having transparencies in your image. Likewise, if you are demonstrating reflection and transparency, do not use textures. For each of the above features, submit two files: a scene description file which generates the image and the generated image itself. Use filenames which clearly state what the feature you are highlighting in the image is. Of course, you are also encouraged to submit images that demonstrate the full power of your ray-tracer. For this part, also submit a text or PDF file describing the structure of your ray tracer and which features you have implemented. Include in this report any information you believe the grader should have in order to grade your submission fairly.

For this part submit:

1. Modified source code in an archive. Label it `part2_src`.
2. Pair of files for each ray tracer feature: scene description and image.
3. A few images showing the full power of your ray-tracer, with all features turned on.

4. The report file.

There should be no OpenGL in your ray-tracer.

NOTE: Ray-tracing is compute intensive, so debug your code with small images (say, 200x200) and debug ray intersections using the scene signature. Allow ample time to render your scenes before the submission deadline!

In this assignment we are looking both for technical and artistic aspects. Position your objects in a configuration that generates interesting effects. Choose colors that make the image look good. Take a look at work by students from previous terms http://www.cdf.toronto.edu/~karan/monkey/.

SUBMISSION INSTRUCTIONS

Your assignment must be submitted on the CDF system. Your code should again compile by simply calling `make`. For Part 2, name your program `rA4` and have it take a single command line argument: the name of the scene file to render.

Use the appropriate command, depending on your course registration, to submit your files:

```
submit -c csc418h -a a4 <filenames>
submit -c csc2504 -a a4 <filenames>
```