Computer Graphics

Theme 6 – Milestone 1

Due Monday, December 19, 2011 at 11:59:59pm

Ray Tracer

1. Introduction

In this assignment, you will write, test, and evaluate a simple ray tracing renderer named raytra. Even though this will be a bare-bones renderer, it will still be capable of creating some very nice images. Your ray tracer will support a number of different features that will be exciting to implement and from which you will be able to see exhilarating and realistic images. You will use the OpenEXR format which you are already familiar with to write out the images produces by your code.

2. Academic Honesty Policy

You are permitted and encouraged to discuss your work with other students. You may work out equations in writing on paper or a whiteboard. You are encouraged to use the Wiki bulletin board to converse with other students, the TA, and the instructor.
HOWEVER, you may NOT share source code or hardcopies of source code. Refrain from activities or the sharing materials that could cause your source code to APPEAR TO BE similar to another student's enrolled in this or previous years. We will be monitoring source code for individuality. Cheating will be dealt with severely. Cheaters will be punished. Source code should be yours and yours only. Do not cheat.

3. Grading and Lateness

This particular milestone will be worth the equivalent of three previous milestones. This will be your last project, and you are being given a considerable amount of time, so plan ahead.

Since this is the last milestone, no late submissions will be accepted. Please start early, as this milestone is worth more.

Plan ahead. If you believe that you have a just cause for submitting a late assignment in a non-medical-emergency circumstance, please obtain written permission from the instructor or TA at least one week prior to the assignment deadline. Plan ahead.

4. Starter Code

You can download from the resources section either of the two parsers that will be provided. They are very basic and are supposed to serve as starting points to get your ray tracer working as fast as possible.
Writing

5. Due Dates and Program Submission

This assignment is due on Monday, December 19, 2011, at 11:59:59pm eastern time. Please submit your solution to the TA as a zip file using the online submission system available at http://www.cs.columbia.edu/~au2158/ . The zip file should be named "<your_uni>_t06m01.zip" (For example: *au2158_t06m01.zip, IMPORTANT: make sure you submit a zip file*). Include a README file in your submission listing all the files included, and extra features you wish to explain. You should also include a Makefile to compile your code and make sure your code compiles in the CLIC machines. Your Makefile should compile your code by typing:

```
make
```

into an executable called main_t06m01.
If you have any questions about the submission process, please speak with the TA during office hours BEFORE the milestone is due. Further, please allow 24 hours for responses from the TA via email or the course Wiki.

6. Required Features

The required features will be layed out in different phases. This is meant so that development is incremental and so that your code is easier to test and debug. This will also let you plan the dates when you want to finish each individual phase, so that you will implement everything by the due date.

Your ray-tracer will operate by initializing itself and then reading from a command file that is passed as the only argument:

```
main_t06m01 mycommandfile
```

The command file contains a series of very simple commands from which you will construct a scene, set up the camera, position the lights, set options, and finally render the scene.

Command File Language

We will provide test files, but you can (and should) make your own as well.

The command file consists of a series of lines which describe geometry, camera, lights, or materials. Only this last one is context-sensitive in that the material definition applies to all geometry that is specified after the material definition.

All commands are unique from a single letter - the only variant of this is that the "lights" command ("l") has a first argument which says which kind of light to make (note also that lights don't take a material definition - the light color and intensity is encoded in 3 floats).

You may extend the command language as you wish, in order to accomplish any of the "extras" you implement, but you must stay backwards-compatible. This means that you must be able to read and render from a source file in the format below, so that we may test your renderer's basic capabilities.

All points, scalars, or vectors are given as floats, with distances in mm. RGB values are encoded as floats with range [0 1] for material colors, and light color and intensity are both encoded as an RGB triple, with minimum 0 and unbounded maximum (although it's reasonable to choose 1 has a nominal value).

Here are the possible commands:

Comment:

```
/ Any line starting with / should be ignored
```

Geometry:

```
/ sphere at position x y z with radius r:
s x y z r
/ triangle with counterclockwise point order:
t x1 y1 z1 x2 y2 z2 x3 y3 z3
/ plane with normal n and scalar value d:
p nx ny nz d
/ name of obj file to load
obj <filename>
```

Camera:

```
/ camera at position [x y z] looking in direction [vx vy vz], with
focal length d,
/ an image plane sized iw by ih (width, height) and number of pixels
pw ph.
c x y z vx vy vz d iw ih pw ph
```

Lights: (note second parameter to denote which kind of light)

```
/ a point light at position x y z, color & intensity coded as r g b
l p x y z r g b
/ a directional light with direction vx vy vz and color & intensity
coded as r g b
l d vx vy vz r g b
/ the ambient light (there will be, at most, only one of these):
l a r g b
```

Materials:

```
/ set the material for all geometry created after this deifinition to
be this one,
/ defined by diffuse components [dr dg db] and specular components
/ [sr sg sb], ideal specular components [ir ig ib], and with
"roughness"
```

```
/ or phong exponent "r"
m dr dg db sr sg sb r ir ig ib
```

Options:

```
/ for your own additions and coding, you may add option flags on a
single line in this way
o myopt1 myopt2 etc.
```

(for example, you might want to be able to switch shadows on/off by including a "shadows" option. Leaving the "shadows" option out will tell your renderer not to do the shadow computation, etc.)

Phase 1 – The Basics (30 points)

After this phase you will have been able to generate simple images. To get through this phase you're ray tracer has to support the following features:

- Ray tracing with only primary rays, no shading, no shadows, only object color:
- Command file read (note: we provide a parser for you!)
- Camera / film setup
- Primary ray generation
- Ray-object intersection w/ spheres - you don't need to do planes or triangles yet
- Output image setup - you must output an EXR image (use your t04m01 code!)

Phase 2 – Getting comfortable (30 points)

This phase will really makes things look much better, since adding shading and lighting will make things look 3D. By the end of this phase your ray tracer supports:

- Recursive ray tracing
- Phong/Blinn-Phong shading (as in your book and previous theme)
- Basic light types - point, directional, and ambient
- Hard shadows
- Ideal specular reflection
- Geometry types: spheres, triangles, and planes

Phase 3 – Making it Awesome (40 points)

Now comes the funnest part of this assignment, where you will add really cool features.

- Antialiasing (you should use jittered super sampling for this part)
- Read/render obj files, w/ normal interpolation on triangles during shading (you have code for this! A solution for that milestone is also available in the wiki)
- Ray-object acceleration: BVH structure

These are listed in order of complexity, so you should from the top. Notice that for files with a lot of triangles, like most obj files, you will need to implement the BVH structure, otherwise it will take forever to compute the rendered image.

Phase 4 (Optional) – You are Hardcore

Other phenomena that you might wish to explore after you've finished the required features:

- Refraction on spheres
- Multithreading
- Arbitrary transforms on any object (for example pushing and popping a la OpenGL)
- Depth of Field
- Motion blur
- Area lights & soft shadows

7. Bonus (15 points) Creative Submission

As part of this assignment you will have the option of submitting a creative command file. This submission will count towards the final grade of your assignment. A committee will judge your image on the following criteria.

1. How well the scene shows off this milestone's 'ingredients' (a la *Iron Chef)*
2. Aesthetic considerations. The more beautiful, the better.
3. Originality.

Top examples will be posted to the course wiki.