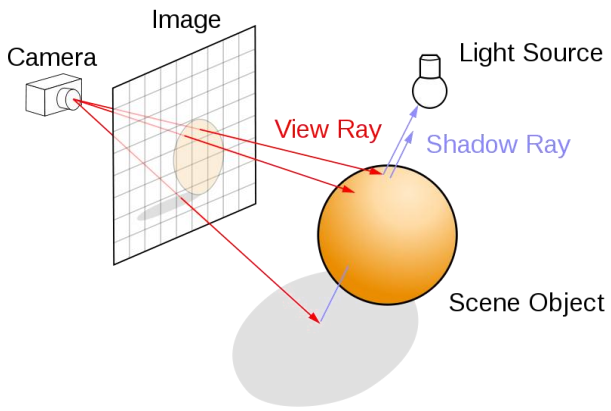
Assignment 3: Ray Tracing
CSL307 – Computer Graphics

Due Date: 31st October 2011

In this assignment you will implement a Recursive Ray Tracer.



Step 1: Ray Generation

Uniformly send out rays from the camera location. Since the camera does not have to move, you can assume that its location is (0,0,0). You should use backwards ray tracing where rays are sent from the camera, one ray per pixel. The final images should be 640x480, but for debugging you should use smaller resolutions with faster rendering times. You can use the field of view of 60 degrees.

Step 2: Ray-Object Intersection

Write the code for ray-object intersections in the scene. The mathematical solutions for the intersections are provided in the lecture notes and also available in the text book.

Step 3: Illumination

Use Phong Illumination model to determine the color of a point:

$$I = (k_a * I_a) + I_i * (k_d * (L \cdot N) + k_s * (R \cdot V)^n)$$

At each intersection point, you need to first determine if a point is in shadow, separately for each light source. You do this by launching a shadow ray to each of the lights. If the point is in shadow, its color with respect to that light should be (0,0,0), that is, black. If the point is not in shadow, use the above model to find its color with respect to that light. The final color of the point is the sum of the contributions from all lights, plus the global ambient color.

In order to compute I, you must determine the normal N at the intersection point. For triangles, you should interpolate the x,y,z coordinates of the normals given at each vertex, and then normalize the length. You should interpolate not just the normals, but also diffuse, specular and shininess coefficients. For spheres, the normal is simple to calculate based on the center of the sphere and the point location.

For materials with a non-zero specular component, you need to call your ray tracer recursively, upto a maximum depth of *at least* 3. You can choose any suitable color as your background color.

• Assignment

Your program should take a command line argument that specifies the filename containing the scene description. It should either only plot the output to the screen or both to the screen and a JPEG file (when name of the output file is provided as a second argument). See the sample code for how to write JPEG files.

• Scene Description Format

The first line is the number of objects in the file. There are three types of objects supported: lights, triangles, and spheres. Color values range from 0 to 1. The format is as follows:

Number of Objects (1 integer)
Ambient Light (3 floats)
(Then you can have lights, spheres or triangles)

- sphere
    - position (3 floats)
    - radius (1 float)
    - diffuse color (3 floats)
    - specular color (3 floats)
    - shininess (1 float)
- triangle
    - then the following, repeated three times (once for every vertex)
    - position (3 floats)
    - normal (3 floats)
    - diffuse color (3 floats)
    - specular color (3 floats)
    - shininess (1 float)
- light
    - position (3 floats)
    - color (3 floats)

Started code is provided that will load scene data from a file. Sample scene description files are also provided. You can also create your own scenes. Sample code is provided only as guide. You can modify it or re-write it.

• Submit

1. Complete source code as a tar-gzipped archive. Include only source code and no executables.
2. It should include a makefile for compiling your code. It should compile without any errors and produce the executable for testing. ***Negative marks for any problems/errors!!***
3. The code must be reasonably commented and written in an easily understandable manner. ***Negative marks for illegible code!!***
4. Include a README file to convey any details.
5. Submit/Upload it to moodle.

- **Grading: 100**

1. Details coming up!!
2. **Can discuss ideas, but STRICTLY no copying/sharing or source code.**