Problem Set 4
CMSC 427
Distributed April 12, 2007
Due: Thursday, April 26 2006

Programming

For this assignment you will write a simple ray tracer. It will be written in C++ without using OpenGL. For full credit, it must support the following features:

- A camera with arbitrary position, direction, and orientation
- Arbitrary image resolution
- Triangle and sphere primitives
- Ambient, directional, and point lights
- Arbitrary materials defined by a color, Phong exponent, and specular fraction
- Diffuse and specular shading, and mirror reflections
- Cast shadows

Note that the BMPImage library sometimes has problems with weird image resolutions, so you should stick with either powers of 2 (e.g. 64x64, 128x128, ...) or standard resolutions (e.g. 640x480, 1024x768, ...).

The program must read from the command line the name of an input file describing the scene to be ray traced, the name of the output bmp file, and the image resolution. We have provided skeleton code that parses the command line and the scene input file. We have also provided sample input files, and the output files that the ray tracer should produce when using these as input.

The input file format consists of a series of commands, each followed by a list of parameters. Blank lines and lines beginning with # are ignored. The supported commands are:

Format	***Example***	***Description***

`camera`
`eye x,y,z`
`look at point x,y,z`
`up vector x,y,z`
`y field of view degrees`
	`camera`
	`0.0 0.0 0.0`
	`0.0 0.0 -100.0`
	`0.0 1.0 0.0`
	`30.0`
		Specifies the position, orientation, and direction of the camera.
		The image is defined to be 1 unit away from the camera in the direction the camera is facing.
		"eye" is the camera's position
		"look at point" is a point in the scene the camera is pointing at
		"up vector" gives the camera's orientation
		"y field of view degrees" is the angle from the top to the bottom of the image in the camera's up and down directions

`ambientLight`
`intensity r, g, b`
	`ambientLight`
	`0.1 0.1 0.1`
		Specifies an amount of light that is striking all objects in the scene. Each scene has at most one ambient light.

`directionLight`
`intensity r, g, b`
`direction x, y, z`
	`directionLight`
	`0.3 0.0 0.0`
	`1.0 1.0 1.0`
		Specifies a light coming into the scene from infinitely far away from the given direction ("direction" is the direction TO the light from any point in the scene). Direction lights can be shadowed.

`pointLight`
`intensity r,g,b`
`position x,y,z`
	`pointLight`
	`0.0 300.0 0.0`
	`30.0 10.0 -50.0`
		Specifies a point light source located at the given position. Since the light intensity attenuates with distance, you may want to make the initial intensity much larger than 1. Point lights can be shadowed.

`material`
`color r,g,b`
`Phong cosine exponent`
`specular fraction`
	`material`
	`1.0 1.0 1.0`
	`50.0`
	`0.0`
		Specifies the material properties for all triangles and spheres declared after this point in the file (until the next material command).
		"specular fraction" is the fraction of the final color that comes from the Phong specular highlight and the mirror reflection. The remaining fraction of the final color comes from the diffuse Lambertian shading.

`triangle`
`vertex1 x,y,z`
`vertex2 x,y,z`
`vertex3 x,y,z`
	`triangle`
	`5.0 0.0 -50.0`
	`15.0 0.0 -50.0`
	`15.0 10.0 -50.0`
		Specifies a triangle primitive with the given 3 vertices.

`sphere`
`center x,y,z`
`radius`
	`sphere`
	`10.0 5.0 -47.5`
	`1.0`
		Specifies a sphere primitive with the given center and radius.

Points will be assigned as follows:

- **40 Points-** Render spheres and triangles with ray tracing under ambient light. For this you will need to shoot rays from the eye position through the pixels in the image and into the scene, and test for intersection against the primitives. The pixel color will be determined by finding the closest intersection, and multiplying that primitive's material color by the ambient light.

- **30 Points-** Add point and directional light sources with specular highlights and diffuse shading. Now, once you've found the closest ray intersection, you will need to compute a diffuse and a specular contribution to the final color from each light. The diffuse light contribution involves the angle between the surface normal and the light, while the specular light contribution involves the angle between the viewer and the light reflection, as well as the Phong exponent. The material's "specular fraction" tells what fraction of the final color should come from the specular light. The remaining fraction should come from the diffuse light, with the ambient light being added on top of these. Remember that point light intensity decreases proportional to the square of the distance from the light.

- **20 Points-** Add cast shadows. To implement this, when computing the diffuse and specular contributions from each light, you will need to first cast a ray in the direction of the light from the intersection point on the surface. If the ray intersects another object that is closer than the light, the point is shadowed from the light, and the light should not affect it.

- **10 Points-** Add mirror reflections. At each intersection point, recursively cast a ray from the point in the direction to the viewer reflected across the surface normal. This ray should not be cast if the recursion depth is greater than 5, or the surface material's "specular fraction" is 0. The light returned by this ray cast should be added to the specular light for this point.

Extra credit:

- Add anti-aliasing with super-sampling. Instead of casting one ray through each pixel in the image, cast several through different points in each pixel, and average the resulting colors.

- Add refractions. Allow the index of refraction of materials to be specified in the scene file (your program should still be able to read scene files in the original format). You can assume the index of refraction of air is 1. When a ray intersects a surface, in addition to recursively casting a reflected ray, recursively cast a refracted ray in the direction given by Snell's law. The light returned by this ray should be added to the diffuse, specular, and ambient light.

Notes:

One thing you need to be careful of when writing this program is that, due to roundoff error, the intersection distance reported by your intersect routines will be slightly incorrect. This can cause a problem if you shoot another ray from the intersection point, since it might end up intersecting the same surface it's supposed to be coming from. To prevent this, you should define some small constant epsilon (say, 0.01), and assume that any intersections at a distance along the ray less than this are not true intersections and ignore them.

The images produced by your program do not have to be pixel-for-pixel accurate to the example image, although if there's significant deviation you may be doing something wrong. It's also not impossible that there are bugs in the demo program, so let us know if your program is producing different output than the demo that you really think is correct.

Pencil and Paper exercises (5 points each)

1. Suppose we have a viewer at (0,0,0) looking at a surface located at (1,2,10) and there is a point source of light located at (10,6,5). If the surface reflects light with Phong reflectance, what surface normal would it need to have to produce the brightest possible specularity to the viewer?
2. Suppose the ground is the y=0 plane. There is an infinite wall whose base is on this plane, along the line described by the equations $y=0$, and x=0. The wall has a height of 10 feet, so that its top is described by the line y=10, x=0. There is a directional source of light, with the direction (cos π/6, sin π/6, 0).
    a. What is the length of the shadow cast by the wall? We want the distance from the edge of the shadow to the wall (that is, don't say the shadow has infinite length).
    b. Suppose the light source is an infinitely distant fluorescent tube. The top of the tube is in the direction (cos (π/6 + π/20), sin (π/6 + π/20), 0). The bottom of the tube is in the direction (cos (π/6 - π/20), sin (π/6 - π/20), 0). There is an equal amount of light coming from every direction between these two. This produces soft shadows when the tube is partially visible from a point. Suppose a point on the ground that is not in shadow has an intensity of 1. Give the intensity of an arbitrary point on the ground (x, 0, z).
3. Suppose there is a white sphere centered at the point (0,0,B) with unit radius, illuminated by a directional light source from the direction (1,0,0). B is some arbitrary, very big number. We are making B big only so that you can assume that all vectors from the viewer to any point on the sphere are in the same direction. We view the sphere from the position (0,0,0).
    a. Suppose the sphere is Lambertian. If the point on the sphere (1,0,B) appears in an image with intensity 1, what would be the intensity of the point (cos(π/20), 0, B – sin(π/20))?

b. Suppose the sphere reflects light according to the Phong model. If the point on the sphere (cos(π/4),0,B-sin(π/4)) appears in an image with intensity 1, and a point on the sphere at (cos(5π/16),0,B-sin(5π/16)) appears with intensity .4531, what will be the intensity of a point at (cos(9π/32),0,B-sin(9π/32))? If we were using Gouraud shading, and interpolated the value of the intensity at (cos(9π/32),0,B-sin(9π/32) from the intensities at (cos(5π/16),0,B-sin(5π/16)) and (cos(π/4),0,B-sin(π/4)) how much error would be caused by this interpolation?

c. **Challenge Problem:** Suppose we use Gouraud shading for the sphere, and the material has a Phong reflectance of $\cos^{20}(\theta)$. We have vertices around the equator of the sphere (in the y = 0 plane) every 2 degrees. What is the maximum amount of error that can occur? What is the maximum amount of error that can occur with Lambertian reflectance? If you find it hard to solve this problem exactly, you might want to make some reasonable approximation in the solution.

4. Instead of a point source of light, let's imagine a continuous light. For example, consider a line segment of light stretching from (-1,0,0) to (1,0,0). This is a tube that emits a constant amount of light all along its length. The strength of this light still falls off with distance, *d*, at a rate of $1/d^2$ as it does for any physically real light.

   a. Consider a surface located at the position (x,0,0), for x>1. Determine the amount of light that reaches this point from our tube of light.

   b. Explain why this lighting cannot be produced by a single point source of light whose strength drops at a rate of $1/d^2$. Show that it can be produced by a non-physically real point source of light whose strength attenuates at a rate of: $\frac{1}{a_0 + a_1 d + a_2 d^2}$. How would you choose a_0, a_1, a_2, to model this light?