Ray Tracing File Format Specification

How Your Program Should Work

Your raytracer should take as input one command-line argument, which will be the filename of a scene description file. You will probably find it useful to print status or debugging information to the screen. An estimation of "percent completed" will be very helpful as ray tracers tend to be quite slow. Your ray tracer will need to either output a bmp file to the specified directory, or display the image in a window (or better yet, do both).

No need to handle malformed input scenes, but exiting gracefully is always appreciated.

Only the terms highlighted in red will be used in Part 1 of the assignment (3A). Triangles will be incorporated in Part 2 (3B).

The input file format

This is meant to be a very easy to parse, human editable scene format. The input file is an ASCII text file ending in *.txt*. Each line of the file should be processed separately and in order; commands may not span multiple lines.

-Each line will begin with a command that specifies how to interpret the rest of the line. That command will be separated from its parameters with a ":"

-The rest of the line will be a space-separated list of parameters. The length of this list will depend on the command at the beginning of the line. A given command will always have the same number of parameters that follow it.

-A parameter will always be a number (expect for the case of the output file name). Some commands use floating point numbers, while others use integers. For example, you can safely assume that the width and height will always be integers.

-Lines beginning with "#" are comments and should be ignored.

-Here is a sample file:

```
#Renders as a black sphere lit by an ambient light
#on a white background.
material: 1 1 1 1 1 1 0 0 0 5 0 0 0 1
sphere: 0 0 2 1
ambient_light: .1 .1 .1
background: 1 1 1
```

-The tables below enumerate the different possible commands and their associated arguments.

Camera Parameters

camera_pos	px py pz	(px, py, pz) is the camera POSITION. **The default is 0 0 0**
camera_fwd	dx dy dz	(dx, dy, dz) is the viewing DIRECTION **The default is 0 0 1**
camera_up_	ux uy uz	(ux, uy, uz) is the UP vector. **The default is 0 1 0**
camera_fov_ha	ha	“ha” is one-half of the “height” angle of the viewing frustum. The half-angle of the “width” can be computed from the aspect ratio of the output image (see below). **The default is 45**
film_resolution	width height	(width,height) is the desired resolution of the output image. **The default is (640,480)**
output_image		"filename" is a string giving the name of the file to write the rendering to. **The default is "raytraced.bmp"**

Scene Geometry

max_vextices	n	A promise to the raytracer that no more than "n" vertices will appear in the rest of the file. This lets you allocate an array in which to store the vertices. **There is no default. It is an error to specify a vertex before specifying max_vertices.**
max_normals	n	A similar promise about the number of normals. **There is no default. It is an error to specify a normal before specifying max_normals.**
vertex	x y z	Adds the vertex (x,y,z) to the "pool" of available vertices. Vertices will subsequently be referred to by NUMBER, in the order they were received. The first vertex is number ZERO.
normal	z y z	Just like the "vertex" command, this adds the normal vector (x,y,z) to the normal pool. Again, these will be referred to by number.

triangle	v1 v2 v3	Adds a triangle to the scene, using three NUMBERED vertices. This triangle is assumed to be flat, so its normal vector should be COMPUTED from the vertices. You should not assume that the vertices arrive in clockwise or counterclockwise order.
normal_triangle	v1 v2 v3 n1 n2 n3	Adds a triangle to the scene using three NUMBERED vertices and three NUMBERED normals. Each normal is associated with the corresponding vertex. These normals are then interpolated during rendering using the barycentric coordinates of an intersection.
sphere	x y z r	Adds a sphere with radius "r" and centered at (x,y,z) to the scene.
background	r g b	Sets the background color to (r,g,b). This color should be returned when a ray doesn't hit anything. The default is (0,0,0)

Material Parameters

material	ar ag ab dr dg db sr sg sb ns tr tg tb ior	Specifies the "current" material properties. This material will be applied to all subsequent primitives until a new materials is specified (similar to an OpenGL "state"). (ar,ag,ab) is the ambient color of the object (dr,dg,db) is the diffuse color of the object (sr,sg,sb) is the specular color of the object ns is the phong cosine power for specular highlights (ns=0 means no highlight) (tr,tg,tb) is the transmissive color of the object, ior is the index of refraction. *[HW2 may ignore these terms]* (sr,sg,sb) should be used to determine not only the color of a highlight, but also the contribution of a reflected ray. For each channel (say red), ar, dr, sr, and tr act as Ka, Kd, Ks, and Kt (from the slide equations). **The default is 0 0 0 1 1 1 0 0 0 5 0 0 0 1, a matte white surface.**

Lighting Parameters

directional_light	r g b x y z	Adds a directional light to the scene. This light has intensity (r,g,b), and shines in a direction (x,y,z). These lights are like the sun; very, very far away.
point_light	r g b x y z	Adds a point light to the scene. This light has intensity (r,g,b), and is located at the point (x,y,z). These lights radiate equally in all directions, but their energy falls off as $1/r^2$
spot_light	r g b px py pz dx dy dz angle1 angle2	Adds a spot light to the scene. The light is located at (px,py,pz), and shines in the direction (dx,dy,dz). The two angles "angle1" and "angle2" define how the light falls off. For points at angles LESS THAN "angle1", the light should behave like a point light. For points GREATER THAN "angle2", the light contributes nothing. BETWEEN "angle1" and "angle2", the light should fall off smoothly. Although linear falloff is fine for this assignment, you might experiment with other falloff functions to give a more natural appearance.
ambient_light	r g b	Defines a scene-wide global ambient light to be applied to all primitives. **The default is 0 0 0 (no ambient light).**

Miscellaneous

max_depth	n	Certain scenes can cause your raytracer to get into an infinite loop (e.g. a hall of mirrors). This parameter sets a maximum recursion depth for reflected and refracted rays. **The default is 5.**