BIL 421/BIL 539:Spring 2020

Assignment 4: A Simple Ray Tracer

Assigned Tuesday, March 10. The program must be submitted to sksueksioglu@gmail.com by Monday, March 30 (any time up to midnight). Subject of the email should say “BIL 421 Assignment 4”. Name your Project directory as “YournameAssg4”. Before submitting, **zip** your Project directory, and email a single zipped file as attachment.Do NOT include any .exe files in your submission. Remember that your code should be fully documented. Check the course syllabus for late policy.

In this assignment, you will implement a basic ray tracer. Your program will take as input a description of the scene objects, the lights and the viewing parameters. It will produce a .ppm file containing the ray-traced image of the scene.

You will implement this in C or C++, but no OpenGL.

Your ray-tracer will support the following elements.

- sphere objects
- solid colored objects
- ambient, diffuse and specular shading
- shadows
- (optional for extra credit) reflection.

Input Format: All coordinates are given as (x,y,z) values with respect to the world frame. All colors are given as RGB values (3 floating point numbers in range [0,1]).

Input File: Your program should either prompt or accept as command line argument the name of the input file.

Output File: The first line of the input file will contain the name of the output file.The second line will contain 2 integers denoting the width and the height of the final image. For example

```
myimage.ppm
400 300
```

Camera and viewing parameters: The next 4 lines contain the coordinates of the camera, the point camera is looking at (at point), the up vector, and the y-field of view in degrees. You may assume that the viewing window is 1 unit from the camera and centered along the viewing direction (near = 1). For example

```
0.0 0.0 0.0  // camera
1.0 0.0 0.0  // at
0.0 1.0 0.0  // up
60           // fovy
```

Light Sources: The next line contains the number of light sources followed by as many lines (one per light source). Each successive line contains 9 floating point numbers: the (x,y,z) coordinates of the light source, the RGB components of its intensity, and the (a,b,c) components of the attenuation formula $(a + bd + cd^2)$. The first light source in the list will always be the ambient light, its position and attenuation factors will be ignored by your program.

Pigments: The next line contains the number of pigments $numP$ followed by a sequence of as many lines each containing the description of one pigment. Pigments are numbered from 0 to $numP - 1$. Each pigment can be thought as a function that takes a surface point (x,y,z) and maps it to an RGB value. In this project you will need to support solid colors. The line for a solid pigment will be as follows: The word **solid** followed by three floating numbers corresponding to the RGB components of the color. Each point on a solid surface has the same color.

Surface Finishes: The next line contains the number of surface finishes, $numF$, followed by a sequence of as many lines each containing the description of one set of surface finish parameters. The surface finishes are numbered from 0 to $numF - 1$. Each surface finish consists of the following five scalar values: the ambient coefficient k_a, diffuse coefficient k_d, specular coefficient k_s, the shininess α, and reflectivity coefficient k_r.

Objects: The next line contains the number of objects, followed by as many lines (one per object). Each line starts with two integers. The first integer corresponds to the pigment number (integer between 0 and $numP - 1$) and the second integer corresponds to the surface finish number (between 0 and $numF - 1$). This is followed by the type of the object and its parameters. Your raytracer is only required to support spheres, so the type and parameters will be specified as:

- the word **sphere** followed by the center point coordinates (x,y,z) and the radius r.

Output File Format: You will produce your final image in PPM P6 format. The first line contains the string "P6". The second line contains the width and the height of the image. The third line contains the integer 255. After this, you will output RGB values per pixel, three bytes per pixel (in binary). You will output these row by row, from top to bottom and from left to right. For each floating point component of the RGB vector, clamp the values to [0,1], scale by 255 and cast to **unsigned char** and output to the PPM file. You can view the PPM file using GIMP.

Other: The slides on raytracing explain all details you need to implement this ray tracer. Study them thoroughly. START early!

You may assume that there are at most 20 light sources, 25 pigments, 25 surface finishes, 100 objects in the scene.

Use (0.5, 0.5, 0.5) as your background color.

Set your recursion depth to 4 if you implement reflections.

Debug Mode: It will difficult to find bugs in your program. For that reason, in your program provide a debug mode by which you can trace only one ray given the pixel row and column number. Print a detailed trace of this ray (objects intersected, point of closest intersection, normal vector, light visibility, reflection ray, etc.)