Assignment on Ray Tracing

1 Overview

In this offline you have to create a ray tracer. This is an individual assignment. You should follow the steps as stated below:

1. Your very first step is to parse the given SDL (Scene Description Language) file. If you do not perform this step, or parse files of a different format, your submission will not be evaluated (you'll get 0).

2. You have to create a world in OpenGL according to the info obtained from the parsed file. Make sure you have your camera integrated in your code.

3. Finally you have to perform ray tracing and generate an image. Keep in mind that you must have correspondence between OpenGL window and the generated image. To clarify, let's say you fire up your OpenGL window and generate an image - **image1.bmp**. If you move your camera and generate a new image - **image2.bmp**, it should be different from **image1.bmp**. Bottomline - generate image of the world as your are currently viewing it. Play around with the provided exe file to clarify this point. Necessary controls for the exe:

    - up arrow - forward movement (zoom in)
    - down arrow - backward movement (zoom out)
    - left arrow - walk left
    - right arrow - walk right
    - PgUp - fly up
    - PgDn - fly down
    - 1 and 2 - yaw
    - 3 and 4 - pitch

- 5 and 6 - roll
- left click - generate image (status message is printed on cmd)

Image generation is not a trivial task, it takes a lot of work. Use the cross platform C++ bitmap library created by Arash Partow to make your life easier:
`http://partow.net/programming/bitmap/index.html`

2 Characteristics of The World

In our world we will have an infinite checkerboard on the xz plane ($y = 0$). We will also have spheres, cylinders (y-axis aligned), and triangles in our world. All of these will be reflective to some extent. In addition, our triangle will be refractive. We will have 0 or more light sources. While generating image, you have to take into account rendering of objects, reflection, refraction, and shading (ambient, diffuse, and specular component).

3 SDL File Format

The SDL file is quite self-explanatory; nevertheless, a brief description is stated below (match with the supplied **Spec.txt** file):

recDepth depth of recursion, i.e. upto what depth you will be performing reflection / refraction. If it is 1, you will only generate ray from the eye; you do not have to generate incident rays from the hit points.

pixels width and height of the image to be generated. So basically the total size will be $pixels \times pixels$.

objStart ... objEnd marks the start and end of an object's description. The object type is immediately followed after **objStart**.

ambCoeff ambient coefficient of the material of the object.

difCoeff diffuse coefficient of the material of the object.

refCoeff reflection / refraction coefficient of the material of the object.

specCoeff specular coefficient of the material of the object. Notice the sum of the four coefficients is 1.

specExp shininess i.e. the term which comes as an exponent upon Phong coefficient.

refractiveIndex refractive index of the material of the object.

colorOne, colorTwo the two colors of the checkerboard. The colors are given as rgb values (0.0 to 1.0).

center center of sphere in 3d coordinates.

radius radius of sphere / cylinder.

color color of the object.

xCenter, zCenter x and z coordinate of center of cylinder.

yMin, yMax lower and upper boundary of y axis aligned cylinder.

a, b, c three vertices of a triangle.

light position of light source. Assume color of light to be white.

4 Marks Distribution

Total marks allotted for this offline is 100, it has been distributed as follows:

- Window image correspondence - 15
- Checkerboard (rendering 5, reflection 5) - 10
- Sphere (rendering 5, reflection 5) - 10
- Cylinder (rendering 10, reflection 5) - 15
- Triangle (rendering 10, reflection 5, refraction 20) - 35
- Lighting (ambient, diffuse, specular) - 15

Note that 60% of the marks for reflection / refraction will depend on successfully incorporating recDepth.

5 Coding Tips

These are some guidelines that I found quite helpful:

- Try to use features of Object Oriented Programming. To be a bit more specific; you can have a superclass Shape from which Sphere, Cylinder, Checkerboard, and Triangle will be derived. This will save you from a lot of repitition of code and make your code more well structured.

- Keep in mind that the Lambert's and Phong's coefficient can never be less than zero. So if you ever come up with a negative value, just put zero in its place.

- The bitmap library of Arash Partow uses values from 0 to 255 to represent color range. Recall that we are using 0.0 to 1.0 in our SDL file. Whenever you are about to color a pixel, make sure you perform necessary conversions.

- There's a very well known problem in ray tracing, known as the Epsilon problem. Due to precision limitation of our computer, it might happen that the hit point is actually inside our object rather than on the surface of our object. In this case if we generate reflected rays or shadow rays (ray from hit point to light source), we will find the ray is being obstructed. So we will see arbitrary dark dots on our object. To get rid of this, we need to move a bit forward in the direction of the reflected ray or the shadow ray and then perform rest of the steps.

6 Submission Procedure

You will have to follow a two-step submission procedure:

1. You have to submit a zipped file by December 11, 11:55 pm to moodle. The zipped file should contain your full code and the SDL file. The name of the zipped file should be your student ID.

2. You then have to show us your offline in person. The exact time and place will be informed to you later on. You will download your submitted code from moodle, compile it, and then show us your output.