Middle East Technical University | Department of Computer Engineering

CENG 477

Introduction to Computer Graphics

Fall '2014-2015

Programming Assignment 1

Due date: 24 October 2014, Friday, 23:55

1 Objectives

In this assignment, you are going to implement a basic ray tracing that applies primitive light characteristic in C or C++ Language. Although basic, you will see that image quality will be great.

Keywords: *C, C++, Ray Tracing, Computer Graphics*

2 Specifications

1. You should name your executable as "raytracer".
2. Your executable will take a scene file as argument. (E.g. "Scene.txt") The details of the file will be elaborated in the next chapter. You should be able to run your executable as, for example, "./raytracer scene.txt"
3. The scene files may contain multiple camera configurations. You should render as many images as number of cameras. The output filenames should be formatted as ¡fileName_camId.ppm¿. (E.g. for a scene file with 2 cameras: "scene_1.ppm" "scene_2.ppm") PPM is a simple uncompressed image format that consists of RGB values of pixels. Its details can be found at: http://netpbm.sourceforge.net/doc/ppm.html
4. You will have at most 30 minutes to render scenes for each camera configuration on inek machines.
5. You will implement two types of light sources. Point and ambient. Point light sources may be multiple whereas ambient light source is singleton. Computed Light values may be greater than 255, however when outputting the pixels, you need to clamp the values between 0-255.
6. The point light sources start with equal intensities in all directions, but decreases according to distance between surface and the source. This decline in intensity is called light attenuation. The attenuation formula at point P for all channels R,G,B is given as

$$I_{Pr} = \frac{I_r}{4\pi d^2}; I_{Pg} = \frac{I_g}{4\pi d^2}; I_{Pb} = \frac{I_b}{4\pi d^2}$$

where $I_{channel}$ is initial light intensity and $I_{Pchannel}$ is light intensity at point P.
Source: http://zebu.uoregon.edu/ soper/Light/luminosity.html:

3 The Scene File

1. A sample scene file has the following format:

```
#Any line starting with a "#" symbol is a comment, therefore it can be skipped.
# Each line starts with a capital letter which annotates the item types:
# (G, C, M, L, S, T)

G imgWidth imgHeight maxRayDepth backGroundColor ambientLight

C index camPos gaze up left right bottom top distance nearClip farClip

M index ambient diffuse specular specExp reflection refrIndex Attenuation

L lightPos intensity

S centerPos radius materialIndex

T vertex1Pos vertex2Pos vertex3Pos materialIndex
```

2. For readability of the scene file, components of multi-valued variables such as Vectors or RGB values are written in parantheses.
   A 3-D vector `pos = (posX posY posZ)` is a tuple of floats denoting X, Y, Z coordinates.
   Similarly, a RGB value `rgb=(rgbR rgbG rgbB)` is a tuple denoting red, green, blue values. Types of tuple values can a float between 0 and 1 in case of material properties, or a float larger than 0 in the case of light intensity.

3. There should be only one entry starting with "G". Rest can be any number. The order of the entries is not relevant.

4. `G imgWidth imgHeight maxRayDepth backGroundColor ambientLight`
   `#e.g:`
   `G 100 100 2 (0 0 0) (50 50 50)`

   "G" denotes the entry of "G"eneral properties of the ray tracer. `imgWidth, imgHeight` denotes the integer image dimensions, width and height respectively, in pixels. `maxRayDepth` is an integer used for defining the maximum number of bounces of rays on mirror-like surfaces. `backGroundColor` and `ambientLight` are the Intensity values of the background color for each color channel and ambient light respectively.

5. `C index camPos gaze up left right bottom top distance nearClip farClip`
   `#e.g:`
   `C 0 (0 0 700) (0 0 -1) (0 1 0) -10 10 -10 10 10 0.1 20000`

   "C" denotes a "C"amera entry. `index` is the camera index that will be appended to the end of the image file. `camPos` is a vector denoting the position of the camera. `gaze` is the direction camera is looking at, `up` specifies the up direction for the camera. Floats `left, right, bottom` and `top` define the image plane in the camera coordinate system. `distance` is a float denoting distance between camera and the image plane. Lastly, `nearClip, farClip` are the floats for minimum and maximum distances of the frustrum. You do not need to render objects outside these boundaries.

6. `M index ambient diffuse specular specExp reflection refrIndex attenuation`
   `#e.g:`
   `M 0 (0.9 0.1 0.1) (0.9 0.1 0.1) (0.9 0.9 0.9) 100 (0.9 0.9 0.9) 1.2 (0.5 0.5 0.5)`

   "M" denotes a "M"aterial entry. `index` is the material index. `ambient, diffuse, specular, reflection` are all relevant material properties for all color channels between 0-1. `specExp` denotes the specular exponent while calculating the Phong shading.
   For the refraction calculation, `refrIndex` is used to calculate the bending of the light when entering the material as stated in the Snell's law according to the formula:
   $nsin\phi = n_t sin\gamma$ where:
   ϕ and γ are the angles between the ray and the normal before and after entering respectively, and n and n_t are the refrective indices of the materials, passing from n to n_t.
   `attenuation` is the degree of attenuation for each color band. It is used for the intensity loss as the ray travels through the material.

7. ```
   L lightPos intensity
   #e.g:
   L (0 300 1000) (3000 3000 3000)
   ```

   "L" denotes that the line defines a Point "L"ight source. `lightPos` is the position vector of the light, `intensity` is the intensity of the light for each color channel.

8. You will use two types of objects, spheres and triangles.

   ```
   S centerPos radius materialIndex
   T vertex1Pos vertex2Pos vertex3Pos materialIndex
   #e.g:
   S (500 500 500) 1000 0
   T (100 100 100) (200 200 200) (500 700 600) 1
   ```

   "S" and "T" specifies if the entry is a "S"phere or a "T"riangle.

   In the case of a sphere, `centerPos` is a vector defining the center of the sphere. `radius` is a float defining the radius of the sphere and the `materialIndex` is the integer defining the material of the sphere.

   In the case of a triangle, `vertex1Pos, vertex2Pos, vertex3Pos` defines the coordinates of the three vertex positions of the triangle. `materialIndex` is the material of the triangle.

4 Hints & Tips

1. Graphics programs are very well suited to object-oriented programming, therefore you are advised to use such approach. Some ideas of classes that you may want to use can be Camera, Vector (Both as RGB component and Vector), Ray, Triangle, Sphere, Material...
2. All objects will be in vacuum, (i.e. refraction index = 1)
3. Tests will use g++. Make sure your codes run on ineks before submitting.
4. You may use -O2 option while compiling your code for optimization.
5. Try to pre-compute anything that would be used many times. (E.g. squared-length of vectors, normals, etc) This will greatly speed-up your code.
   ```

6. Avoid computations whenever you can. For instance, if a material's attenuation coefficients are 1 for each channel, the object is opaque. Then you do not have to do refraction computations.

7. Assume no reflections inside the object.

5 Regulations

1. **Programming Language:** C/C++

2. **Late Submission** You can submit your codes up to 3 days late. Each late day will incur a penalty of 10 points. After 3 days, you will get 0.

3. **Cheating: We have zero tolerance policy for cheating**. People involved in cheating will be punished according to the university regulations and will get 0. You can discuss algorithmic choices, but sharing code between each other or using third party code is strictly forbidden.

4. **Newsgroup:** You must follow the newsgroup (news.ceng.metu.edu.tr) for discussions and possible updates on a daily basis.

5. **Submission:** Submission will be done via COW. Create a tar.gz file named `hwX.tar.gz` that contains all your source code files and a makefile. The executable should be named "raytracer" and should be able to be run using command `./raytracer sceneName.txt`

6. **Evaluation:** Your codes will be evaluated based on several input files including, but not limited to the test cases given as example. Rendering all scenes correctly within time limit will get you 100 points. The fastest tracers will be awarded 20 points of bonus.