Middle East Technical University | Department of Computer Engineering

CENG 477

Introduction to Computer Graphics

Fall '2020-2021

Assignment 1 — Ray Tracing

(v.1.0)

Due date: November 15, 2020, 23:59

1 Objectives

Ray tracing is a fundamental rendering algorithm. It is commonly used for animations and architectural simulations, in which the quality of the created images is more important than the time it takes to create them. In this assignment, you are going to implement a basic ray tracer that simulates the propagation of light in the real world.

Keywords: ray tracing, light propagation, geometric optics, ray-object intersections, surface shading

2 Specifications

a You should name your executable as “raytracer”.

b Your executable should take an XML scene file as argument (e.g. “scene.xml”). A parser will be given to you so that you do not have to worry about parsing the file yourself. The format of the file will be explained in the next section. You should be able to run your executable via the command “./raytracer scene.xml”.

c The scene file may contain multiple camera configurations. So you should render as many images as the number of cameras. The output filenames for each camera are also specified in the XML file.

d You will save the resulting images in PPM format. A PPM writer will be given to you so you don't have to worry about writing it yourself. Interested readers can find the details of the PPM format at: http://netpbm.sourceforge.net/doc/ppm.html.

e For any input file, your program will have 30 minutes at most to produce output image(s). Upon exceeding this limit, your program will be terminated and will be assumed to have produced no image.

f You will implement two types of light sources: point and ambient. There will be an ambient light and zero or more point light sources. The intensity values of these lights will be given as decimal number (R, G, B) triplets which are not restricted to the [0, 255] range (however, they cannot be negative since negative light does not make sense). As a result of shading computations if you find a pixel color value greater than 255, you must clamp that value to 255 and round it to the nearest integer before writing it in the PPM file.

g Point lights will be defined by their intensity (power per unit solid angle). The intensity due to such a light source falls off as inversely proportional to the squared distance from the light source. To simulate this effect, you must attenuate the intensity of point lights as:

$$I_p(d) = \frac{I}{d^2},$$

where I is the original light intensity (a triplet of RGB values given in the XML file) and $I_p(d)$ is the intensity at distance d from the light source.

h You should use Blinn-Phong shading model for the specular shading computations.

3 The Scene File

The scene file will be formatted as an XML file (see Section 6). In this file, there may be a number of different materials, vertices, triangles, spheres, lights, and cameras. Each of these are defined by a unique integer id. The ids for each type of element will start from one and increase sequentially.

a **BackgroundColor:** Specifies the R, G, B integer values of the background. If a ray through from a pixel does not hit any object, the color value of the pixel will be this.

b **ShadowRayEpsilon:** When a ray hits an object, you are going to send a ray from the intersection point to each point light source. Due to floating-point precision errors, sometimes the ray hits the same object even if it should not. Therefore, you must use ShadowRayEpsilon, which is a floating-point number, to move the intersection point a bit further so that it does not intersect the same object again.

c **MaxRecursionDepth:** Specifies how many bounces a ray makes from mirror-like objects. Applicable only when a material has non-zero MirrorReflectance value. Primary rays are assumed to start with zero bounce count.

d **Camera:** Position parameters define the coordinates of the camera. The parameters of Gaze define the direction that the camera is looking at. On the other hand, the parameters of Up define the up vector of the camera. NearPlane attribute defines the coordinates of the image plane with Left, Right, Bottom and Top parameters. NearDistance defines the distance of the image plane to the camera. ImageResolution defines the resolution of the image. Lastly, ImageName defines the name of the output. All numbers are floating-point numbers except image resolution values, which are integers. You must assume that the Gaze vector of the camera is always perpendicular to the image plane.

e **AmbientLight:** Ambient light can be defined just as intensity which is a decimal number triplet. This is the amount of light received by each object even when the object is in shadow.

f **PointLight:** A point light can be defined by a position and an intensity, which both are decimal number triplets. It is a decimal number.

g **Material:** A material can be defined with ambient, diffuse, specular, and mirror reflectance properties for each color channel. Values are decimal numbers which are between 0.0 and 1.0. Phong exponent defines the specularity exponent in Blinn-Phong shading. MirrorReflectance represents the degree of the mirrorness of the material. If this value is non-zero, when a ray hits the object you must cast a new ray from the object and scale the resulting color value with the parameters of MirrorReflectance.

h VertexData: Each line contains a vertex whose x, y, and z coordinates are given, respectively.

i Mesh: Each mesh is composed of several faces. A face is actually a triangle which contains three vertices. When defining a mesh, each line in Faces attribute defines a triangle. So, each line is represented by three vertex IDs given in counter-clockwise order (see below). Material attribute represents the Material ID.

j Triangle: A triangle is represented by Material and Indices attributes. Material attribute represents the Material ID. Indices are the vertex IDs of the vertices which are the corners of the triangle. Vertices are given in counter-clockwise order, which is important when you want to calculate the normals of the triangles. Counter-clockwise order means that if you close your right-hand following the order of the vertices, your thumb will point in the direction of the surface normal. Material attribute represents the Material ID.

k Sphere: A sphere is represented by Material, Center, and Radius attributes. Material attribute represents the material ID. Center represents the vertex ID of the point which is the center of the sphere. Radius attribute is the radius of the sphere.

4 Hints & Tips

a Start early. It takes time to get a ray tracer up and running.

b You may use the -O3 option while compiling your code for optimization. This itself will provide a huge performance improvement.

c Try to pre-compute anything that would be used multiple times and save these results. For instance, you can pre-compute the normals of the triangles and keep it in your triangle data structure when you read the input file.

d We will not test your code with strange configurations such as the camera or a point light being inside a sphere or with objects in front of the image plane. If you doubt whether a special case in addition to these will be tested you may ask this in the forum.

e For debugging purposes, consider using low resolution images. Also, it may be necessary to debug your code by tracing what happens for a single pixel (always simplify the problem when debugging).

f If you see generally correct but noisy results (black dots), it is most likely that there is a floating-point precision error (you may be checking for exact equality of two FP numbers instead of checking if they are within margin of a small epsilon like 0.0001). Another thing you can try is, keeping track of the source of the ray and preventing the ray from intersecting with it.

5 Bonus

a You can get 5 points bonus if your ray tracer is among the fastest N ray tracers in our benchmarks. Number N is to be determined experimentally but you can expect it to be a small number such as 3 or 5. Of course, your outputs should be correct to be eligible for this bonus.

b You can get 5 points bonus if you create and share interesting XML scene files. These should not be very simple scenes as we already share such scenes with you. Interesting scenes to get bonus will be determined by the 477 instructors and assistants.

6 Regulations

a **Programming Language:** C/C++. You also must use gcc/g++ for the compiler. You are free to choose your compile options. Only under special cases other PLs may be allowed (for instance being an Erasmus student and not having learned these languages in your program). Ask your instructor if you are in doubt.

b **Changing the Code Template:** You are free to edit, rename or delete any file from the given code template as long as you comply the submission rules below. But keep in mind that any error introduced by the changes you made is your responsibility.

c **Submission:** Submission will be done via ODTÜClass. To submit you must select an option in "HW1 Groups" page in ODTÜClass according to the criterias in the Groups section below. Create a "tar.gz" file named "raytracer.tar.gz" that contains all your files necessary to compile. You must also create a Makefile since your code will be compiled using make command. The executable should be named as "raytracer" and should be able to be run using the command "./raytracer scene_file_name.xml". So your submission should be compiled and executed after the following commands:

```
tar -xf raytracer.tar.gz
make
./raytracer scene.xml
```

Any error in these steps will cause 5 points penalty.

d **Groups:** You can team-up with another student. To form a group, both students must select the same group with two people capacity in "HW1 Groups" page on ODTÜClass. If you want to do the homework by yourself, in the same page select any "Individual Work" option which only have one person capacity.

e **Late Submission:** You can submit your codes up to 3 days late. For the semester and all 4 homeworks, you can submit your homework late for total of 10 days. However, if you fail to submit even after 3 days, you will get 0 regardless of how many late credits you may have left. Every late submission will deduce from the total number of late days including the ones get 0.

f **Cheating:** We have zero tolerance policy for cheating. People involved in cheating will be punished according to the university regulations and will get 0 from the homework. You can discuss algorithmic choices, but sharing code between groups or using third party code is strictly forbidden. To prevent cheating in this homework, we will also compare your codes with online ray tracers and previous years' student solutions. In case of a match is found, this will also be considered as cheating. Even if you take only a "part" of the code from somewhere or somebody else, this is also cheating. Please be aware that there are very advanced tools that detect if two codes are similar.

g **Forum:** Any updates/corrections and discussions regarding the homework will be on ODTÜClass. So don't forget to check it on daily basis.

h **Evaluation:** Your codes will be evaluated on Department Inek Machines; based on several input files including, but not limited to the sample scenes given to you. Rendering all scenes correctly within the time limit will get you 100 points.

7 Sample Scene File

```
<Scene>
        <BackgroundColor>R G B</BackgroundColor>
    <ShadowRayEpsilon>X</ShadowRayEpsilon>
        <MaxRecursionDepth>N</MaxRecursionDepth>
        <Cameras>
        <Camera id="Cid">
            <Position>X Y Z</Position>
            <Gaze>X Y Z</Gaze>
            <Up> X Y Z </Up>
             <NearPlane>Left Right Bottom Top</NearPlane>
            <NearDistance>X</NearDistance>
            <ImageResolution>Width Height</ImageResolution>
            <ImageName>ImageName.ppm</ImageName>
        </Camera>
    </Cameras>
    <Lights>
        <AmbientLight>X Y Z</AmbientLight>
        <PointLight id="Lid">
            <Position>X Y Z</Position>
            <Intensity>X Y Z</Intensity>
        </PointLight>
    </Lights>
    <Materials>
        <Material id="Mid">
            <AmbientReflectance>X Y Z</AmbientReflectance>
            <DiffuseReflectance>X Y Z</DiffuseReflectance>
            <SpecularReflectance>X Y Z</SpecularReflectance>
            <PhongExponent>X</PhongExponent>
            <MirrorReflectance>X Y Z</MirrorReflectance>
         </Material>
     </Materials>
    <VertexData>
        V1X V1Y V1Z
        V2X V2Y V2Z
        ..................
    </VertexData>
        <Objects>
        <Mesh id="Meid">
            <Material>N</Material>
            <Faces>
                F1_V1 F1_V2 F1_V3
                F2_V1 F2_V2 F2_V3
                ..............
            </Faces>
        </Mesh>
        <Triangle id="Tid">
            <Material>N</Material>
            <Indices>
                V1 V2 V3
            </Indices>
        </Triangle>
        <Sphere id="Sid">
            <Material>N</Material>
            <Center>N</Center>
            <Radius>X</Radius>
        </Sphere>
        </Objects>
</Scene>
```