

Search Topics

Overview

Bookmarks

Course Schedule

Table of Contents

Syllabus

Policies

Assignment 1

Assignment 2


Assignment 3

Study/Practice Material

Midterm Exam

Final Exam

Assignment 3

 Due Dec 8, 2022 11:59 PM

100 % 1 of 1 topics complete

## Assignment 3

Assignment



 Due Dec 8, 2022 11:59 PM

For this assignment, you will be building a Ray Tracer using C/C++, Python, or Java. The choice of which of these languages is up to you, but keep in mind that language-specific support is not provided (you should use the language you can locally support and are comfortable coding in). The system only needs to handle the rendering of ellipsoids, with a fixed camera situated at the origin in a right-handed coordinate system, looking down the negative z-axis. Local illumination, reflections, and shadows will also need to be implemented. The program should take a single argument, which is the name of the file to be parsed. Make sure your executable has the name "RayTracer.exe" (or equivalent for the language you choose) and that we can run it as in the following examples:

```
> RayTracer.exe testCase1.txt
```

or

```
> java RayTracer testCase1.txt
```

or

```
> python RayTracer.py testCase1.txt
```

We will use a script to generate the outputs for the set of posted test cases. You will get zero marks if we cannot compile your program and if we cannot run this script because your project does not implement the required specifications above.

*Make sure you carefully read the **Instructions and Clarifications** below as they contain important information on the lighting, scene setup, tracing depth etc.*

### Input File Format

The content and syntax of the file is as follows:

#### Content

- The near plane\*\*, left\*\*, right\*\*, top\*\*, and bottom\*\*
- The resolution of the image nColumns\* X nRows\*
- The position\*\* and scaling\*\* (non-uniform), color\*\*\*,  $K_a^{***}$ ,  $K_d^{***}$ ,  $K_s^{***}$ ,  $K_r^{***}$  and the specular exponent  $n^*$  of a sphere
- The position\*\* and intensity\*\*\* of a point light source
- The background colour \*\*\*
- The scene's ambient intensity\*\*\*
- The output file name (you should limit this to 20 characters with no spaces)

\* *int*      \*\* *float*      \*\*\* *float between 0 and 1*

**Syntax**

NEAR &lt;n&gt;

LEFT &lt;l&gt;

RIGHT &lt;r&gt;

BOTTOM &lt;b&gt;

TOP &lt;t&gt;

RES &lt;x&gt; &lt;y&gt;

SPHERE <name> <pos x> <pos y> <pos z> <scl x> <scl y> <scl z> <r> <g> <b> <K<sub>a</sub>> <K<sub>d</sub>> <K<sub>s</sub>>  
 <K<sub>r</sub>> <n>

... // up to 14 additional sphere specifications

LIGHT <name> <pos x> <pos y> <pos z> <l<sub>r</sub>> <l<sub>g</sub>> <l<sub>b</sub>>

... // up to 9 additional light specifications

BACK <r> <g> <b>

AMBIENT <l<sub>r</sub>> <l<sub>g</sub>> <l<sub>b</sub>>

OUTPUT <name>

*All names should be limited to 20 characters, with no spaces. All fields are separated by spaces. There will be no angle brackets in the input file. The ones above are used to indicate the fields.*

**Output File Format**

You should output your images in PPM format. You can choose binary (P6) or text-based ppm (P3). The text-based format is very simple, but it is important to understand the assumptions about which pixel is where (where (0,0) is for example). Example code for these has been provided in C++ attached to the assignment. We must be able to open your output files in a viewer that is freely available see clarifications below.

<https://netpbm.sourceforge.net/doc/ppm.html>

**MARKING SCHEME**

26 Total Marks

- [2] Coding Style (i.e. well designed, clean, & commented code)
- [2] x 12 For each of the given test cases.
- [-4 Marks if not] Provide a readme.txt that describes what you have done, what you have omitted, and any other information that will help the grader evaluate your work, including what is stated below.
- There will be no partial marks given if your program fails to parse an input file or if it does not produce the correct output.
- Make sure you submit all the required files so we can compile and build your program. If there are missing libraries, you will get zero marks.
- **We will use a script to generate the outputs for a set of test cases. You will get zero marks if we cannot run this script because your program does not adhere to the given requirements. (Repeated for emphasis)**

## Requirements/Policies

### Collaboration

None. If you discuss this assignment with others, you should submit their names along with the assignment material.

## Original Work

The assignment must be done from scratch. Apart from the code provided, you should not use code from any other source, including the previous offering of the class. (**see clarifications below**)

Note there are many shader sources on the internet. Please develop your own first! You can seek inspiration but do not use other's code.

## Zero Mark

If the code does not run, or no objects appear in the window, or only the template code is running properly, no partial marks will be given.

## INSTRUCTIONS AND CLARIFICATIONS

- *Start working on it early. You will not have time to do it at the last minute.*
- Your submission should include ALL of the code necessary to compile and run the program, and should not contain any additional functionality besides what is described here. It should not contain any OpenGL API calls. You may use OpenGL for displaying the results during your debugging.
- You may use vector and matrix libraries, i.e. any vector matrix libraries for C/C++ (e.g. GLM), Java (JAMA, EJML), or Python (NumPy). You cannot use library code that solves any part of the raytracing problem on its own, you must write this code yourself. It must be clear how to set up your code, and, if possible, all libraries *\*must\** be included. If we cannot setup your code we cannot mark it.
- In the assignment attachments, you will find two pieces of example c++ code. One inverts a 4x4 matrix, and the other writes a char buffer to a ppm image, which is the expected output of this program.
- The code that inverts a 4x4 matrix expects two 4x4 matrices to be passed in as arguments. The first matrix will be inverted and the result will be stored in the second matrix. Both matrices are row order storage, so you have `M[row][column]`.
- You may use the STL string and vector classes.
- The assignment must be done from scratch.
- Make sure that your parse routine does not crash based on where the EOF character is.
- Given a reasonable resolution (400x400) on a modern machine, your program should generally take no more than five to twenty seconds (probably less than that) to run when compiled in "release" mode. Interpreted languages may take longer, but if the files do not return within a reasonable time the case will fail. For interpreted languages, reasonable may be upwards of a minute.
- A sphere at position (0,0,0), with scaling parameters (1,1,1) should be centred at (0,0,0) with a radius of 1.
- If the eye ray is constructed using the convention described in class, then when intersecting it with an object, the closest object is the one with a minimum hit time greater than 1. A hit time between 0 and 1 falls between the eye and the near plane, and hence is not a part of the view volume.
- When creating rays from the closest hit point on an object, you need to start them at  $t = 0.000001$ , (try a larger number like  $t = 0.0001$  if this does not work), to avoid false

intersections due to numerical errors. In other words, you may not want to consider intersections at time=0.

- For the keys and results, we use the following convention: Rays from the eye that hit nothing return the colour of the background, while reflected rays that hit nothing return black (i.e. nothing).
- The template code for saving your image to disk uses the ppm image format. If you do not already have a program that can read images of this type, you can download, GIMP (recommended!), IfranView ([www.irfanview.com](http://www.irfanview.com)), or Xnviewgb. These are excellent and free programs for viewing images, and can, amongst others, read ppm files.
- The “NEAR” value is an absolute value and represents the distance along the negative z-axis.
- Your code may need to handle hollow spheres, which are “cut” open by the near plane.
- Your code may need to be able to handle lights inside spheres.
- You will be using the following local illumination model:
  - $\text{PIXEL\_COLOR}[c] = K_a * I_a[c] * O[c] +$   
     for each point light (p)  $\{ K_d * I_p[c] * (N \cdot L) * O[c] + K_s * I_p[c] * (R \cdot V)^n \} +$   
      $K_r * (\text{Color returned from reflection ray})$
  - O is the object color ( $\langle r \rangle \langle g \rangle \langle b \rangle$ )
  - [c] means that the variable has three different color component, so the value may vary depending on whether the red, green, or blue color channel is being calculated
  - The other components of this equation are explained in the lecture notes.
- You should not spawn more than three reflection rays for each pixel, i.e. stop the recursion after 3 bounces.
- When summing over all lights, it is possible for the PIXEL\_COLOR value to go above 1. In this case, the simplest solution is to clamp the value to 1.
- Do not forget to scale by 255 before creating the ppm image
  - (C/C++) using the given “save\_imageP6()” function in ppm.cpp. “save\_imageP3()” is provided for debugging reasons because it produces a text file that is human readable. The P6 version produces a binary file. You are welcome to produce the human readable format for any of the available languages, as long as it can be opened by an image reader listed above.
- Make sure you submit all the required files so we can compile and build your program. If there are missing libraries, you will get zero marks. (C/C++) On Linux, we should be able to compile by just typing make “make”, and, on Windows, with MS VS -> Build. Make it clear in your readme how your code is compiled.
- (C/C++) Make sure that your Visual Studio project is designed to create a “Console Application” or “Command Line Tool”.
- Make sure that everything works on a standard **Windows** machine as this is what the lab and TAs support. (However, you should avoid using platform-dependent code, like the Windows API. You do not need it here.)
- (C/C++) Make sure you “clean” your project before zipping it and submitting it, to remove all the large auxiliary files that VS creates.

- Make sure you check regularly the forum and the announcements in case the grading or submission instructions change.