Assignment due date: Friday, April 7, 11:59pm

Electronic submission on the CDF server by the above due date

Student Name (last, first):

Student number:

Student UtorID:

I hereby affirm that all the solutions I provide, both in writing and in code, for this assignment are my own. I have properly cited and noted any reference material I used to arrive at my solution, and have not shared my work with anyone else.

_____

    Signature

(note: -3 marks penalty for not completing properly the above section)

Assignments 3 and 4 will be dedicated to building a fairly advanced raytracer from scratch.

You may work in teams of two students. Both students will receive the same mark.

This assignment (#3) involves creating the basic raytracing framework, implementing simple geometric primitives, computing object/ray intersections, applying transformations, and using material properties together with the Phong illumination model to determine the colour of surfaces under point light source illumination.

Learning Objectives:

You will integrate all the knowledge you have acquired so far in this course, and apply it to the task of building a simple ray tracer.

You will improve your understanding of geometry, transformations, world-to-camera coordinate conversion, and illumination.

You will implement the entire image formation pipeline, in reverse! This will help you find and eliminate any gaps in your knowledge of the image formation process.

You will understand how material and light source properties affect visual appearance

You will practice working with a large code-base, and to focus on your programming task while understanding and using already implemented functionality.

Skills Developed:

Implementing geometry and geometric computations.

Implementing transformations and using them to generate complex scenes from simple objects.

Working with a reasonably complex starter code distribution. Quickly evaluating what is already Implemented and what you need to do.

Debugging software that renders images.

Reference material:

On-line lecture notes for all topics involved in ray tracing.

The detailed comments in the starter code. The starter code ***already provides a lot of the Simple building blocks you will need***. Make sure you understand what is provided, so you don't waste time implementing functionality already there.

Course instructors, tutorials, online forums

Assignments 4 brings together everything you have learned in the course in order to create a high quality, ray-traced scene.
Here you will add advanced features to your ray tracer from A3. You will then design and render a scene of your choice that demonstrates the capabilities of your rendering software.

You may work in teams of two students. Both students will receive the same mark.

Learning Objectives:

You will apply your knowledge of computer graphics (including geometry, transformations, Illumination, and texturing) to the task of writing an advanced ray tracer

You will achieve a thorough understanding of how the different components of the computer graphics rendering pipeline fit together.

You will understand how advanced visual effects such as transparency, soft shadows, and anti-aliasing work.

You will gain experience in the design of complex scenes. This will involve thinking about not only the geometry, but also the material properties, light-source properties, camera set-up, textures, and so forth.

Skills Developed:

Implementing advanced rendering techniques.

Designing and defining complex scenes.\|

Polishing your final render to make it look impressive.

Reference material:

On-line lecture notes for all topics involved in ray tracing.

The detailed comments in the starter code.

Your implementation of the basic ray tracer

Make sure you understand what is provided, so you don't waste time implementing functionality already there.

Course instructors, tutorials, online forums.

Build Your Own Ray Tracer!
Assignment due April 7 at 11:59pm
(electronic submission only)
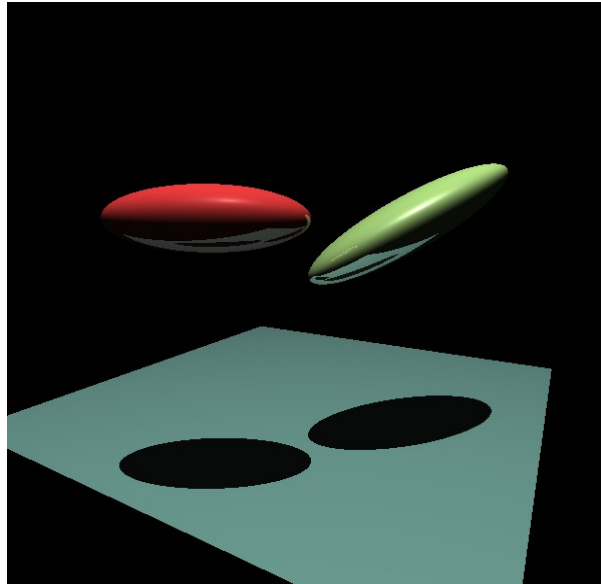
Output of an implemented solution for this assignment

Basic Ray Tracer [100 marks in total]

      Your task is to implement a basic ray-tracer and render a simple scene using ray casting and the Phong shading model as described in lecture. The starter code sets up a scene with two spheres and a plane as shown above, illuminated by a single point light source. Your job is to render the scene by implementing the code fragments missing for object intersections, Phong illumination, and recursive ray casting.

You will have to implement the following code fragments:

(a) *[10 marks] Ray casting* – computing a ray from the camera through each pixel and into the scene.
(b) *[15 marks] Intersection code for ray-sphere intersection* (it should also work for affinely deformed spheres).
(c) *[15 marks] Intersection code for ray-square intersection* (it should also work for affinely deformed squares).
(d) *[15 marks] Compute the correct normals* for affinely-deformed objects
(e) *[15 marks] Phong illumination* for a point light source (including ambient, diffuse, and specular components)

(f) *[15 marks] Shadows*
(g) *[15 marks] Reflection* by recursive ray-casting

To demonstrate the working of your program, you should generate three different types of renderings:

1. A scene signature where each pixel shows a unique color identifier for the first object hit (or back- ground). This gives you an impression of the relative positions of the camera and the objects. You can generate this by making the *ambient* albedo 1, and the *diffuse* and *specular* albedos zero.

2. A rendered scene with only the diffuse and ambient components of the Phong model.

3. A rendered scene with all three terms of the Phong model.

Compiling and running the starter code

Like all previous assignments. The raytracing code is designed to work on Linux. Therefore we expect you to do your work at the labs in Bahen. You can work remotely on CDF. This time the code will not produce a real-time visual display, so you do not have to worry about the graphical issues involved in working remotely over ssh. However, note that the code is fairly computationally intensive. You will be able to work faster if you are coding on a separate machine, as opposed to sharing the CPU with all other CDF users.

Use the included *compile.sh* script to compile the code.

Your final submission must compile and run without any form of glitch on CDF

Note: To encourage good coding practice, we will deduct up to

[10 marks] penalty marks - for badly designed code, that is, code lacking comments, confusing or unstructured code, or repetitive (not modular) code design.

Starter code in detail

utils.h, utils.c : Provide functions to manipulate points, vectors, matrices, and objects. This includes operations like dot products, matrix inversion, matrix-vector product, object creation and object transformations. There are parts in utils.c that you need to complete. Furthermore, you need to know the code in these functions since you will use these utilities extensively while building the ray tracer.

RayTracer.h, RayTracer.c: These comprise the main code for the ray tracer. They follow the structure and flow of the ray tracer pseudo-code from the lecture notes. You need to add code here to complete the ray tracer functionality.

svdDynamic.h, svdDynamic.c: This is a library used for matrix manipulation. You do not need to look at the code in any of these files.

Read carefully all the comments in the starter code – they describe in many places what you need to do. Parts of the code where work is needed are marked ***"TO DO"***.

What to submit:

- ***ALL*** your code. That means all .h and .c files as well as the compilation script.

- The three images requested above (scene signature, diffuse+ambient, full Phong)

- A completer REPORT file and a completed CHECKLIST (please create the REPORT file yourself)

The report for this assignment can be fairly short, but it should explain:

What was the process you used to complete the assignment. That is, how did you organize your work, go about understanding the starter code, and build a solution.

Brief description of the code you implemented and how it works. Particularly if you did anything that is not straightforward to understand by looking at your code.

What parts of the assignment were easy/hard.

Whether the assignment helped you understand better the following topics:

Geometry (points, vectors, planes, spheres, intersections, normals, etc.)
Object transformations
Projection and coordinate frames
Illumination (the Phong model, basically)

The role of each member on the project.

You will also be marked on the clarity and quality of your writing.

Submitting your work

From just outside the directory containing the starter code, do:

(see instructions for submitting A4. This assignment will be bundled with it)

General advice

- ***Start early:*** This is a fairly complex programming project, and requires both good C programming skills and understanding of the course material.
- ***Ask questions:*** Don't wait if you have problems. Ask your instructor or TA any questions related to the course material that arise while doing your work.
- ***Think, then code:*** The solution is reasonably short, but figuring out what the correct thing to do is requires thought and understanding. Do not start cobbling code together until you have understood what you should be doing.
- ***Have fun:*** Ray tracers are cool. After this assignment, you will understand how they work inside, and you will have your very own ray tracer to show off!

Advanced Ray Tracing
Assignment due April 7 at 11:59pm
(electronic submission only)

Same scene from A3 after applying some of
*the advanced rendering techniques*
from this assignment

Advanced Ray Tracer [100 marks in total]

For this assignment you will extend the basic ray tracer you build for assignment 3.
You will implement several extensions that result in much more realistic and impressive
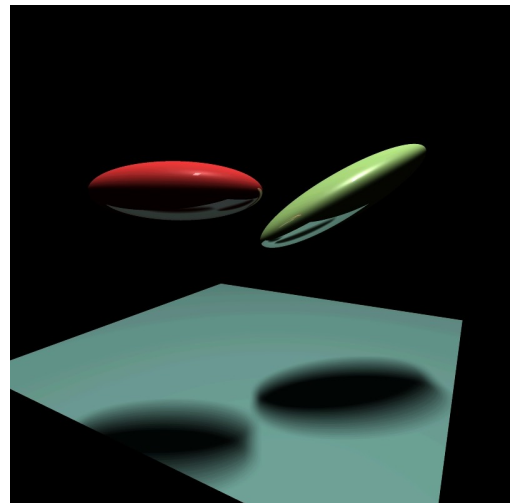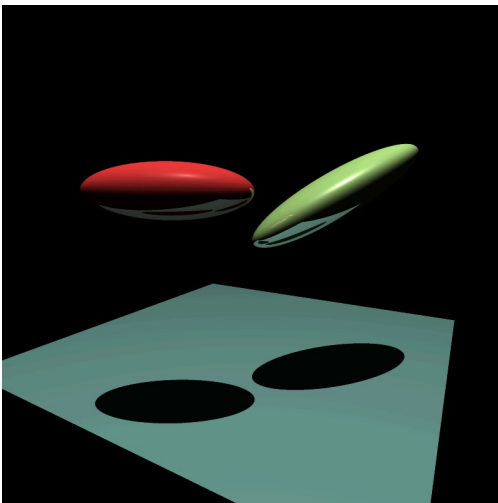images.

Extensions to implement:

(a) [15 marks] Anti-aliasing – Implement anti-aliasing by super-sampling as discussed in lecture.
(b) [20 marks] Area light sources – Implement a function to create planar area light sources by
representing the area light source as a collection of point light sources. Use proper plane equation
and plane transformations.
(c) [25 marks] Implement a neat scene – Use your creativity and imagination to create a cool-looking
scene that showcases what your ray tracer can do.
(d) [10 marks] Report – Write a clear one page report as described in the A3 submission information

Advanced Ray Tracing
Anti-aliasing example
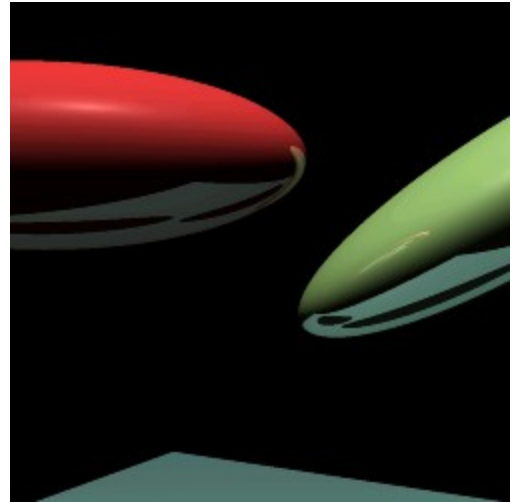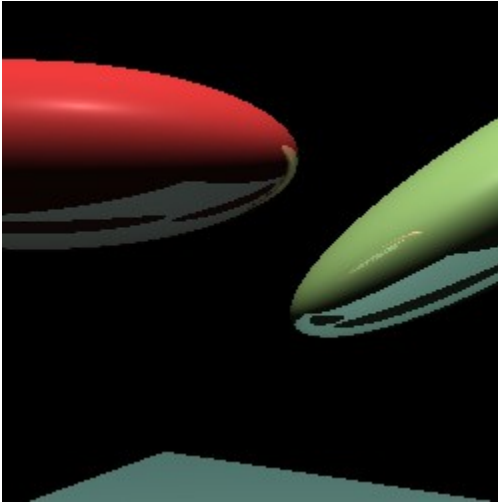
The image on the left was produced by the original, basic ray tracer. The image on the right shows the same scene after implementing anti-aliasing by super-sampling

Area light-sources

The image on the left was produced using a single point light-source. The image on the right shows the same scene under a single, rectangular, area light-source.

(e) *[up to 55 marks] Advanced techniques* – Implement one (or more) of the options below. You will need
To implement three features from here to possibly get full marks.
Note: A4 will be marked out of 100 marks. You could\In practice end up with a mark of 125% by doing all\I
the advanced work! Any excess marks will be applied to your other three assignments. This is a good
chance to recover marks lost through the term!

- *Handling a non-trivial compound object, containing quadratic surfaces (e.g., a cone or cylinder). Spheres, ellipses, planes and patches thereof do not count.*
- *Handling arbitrary surface mesh geometry.*
- *Glossy reflections*
- *Depth of field*
- *Texture-mapping (an interesting procedural texture is also acceptable).*
- *Adding environment mapping*
- *Motion Blur*
- *Multi-threading (to make use of available resources by multithreading to gain a performance boost)*
- *Refraction – Implement the code needed to correctly implement refraction and allow for transparent or semi-transparent objects.*

What to hand in:

- **ALL** your code. That means all .h and .c files as well as the compilation script

- Your rendered scene at a good resolution (i.e. no smaller than 1024x1024)

- A completed **CHECKLIST**

Submitting your work

From just outside the directory containing the starter code, do:

tar -cvfz a34_solution_studentNumber.tgz ./starter
submit -c csc418h -a A34 -f a34_studentNo.tgz (if registered in CSC418)
submit -c csc2504h -a A34 -f a34_studentNo.tgz (if registered in CSC2504)

General advice

Fix your ray tracer. You must have a working ray tracer from A3 to successfully complete A4.
You must absolutely not use any code other than your own.

Ask questions. Don't wait if you have problems. Ask your instructors or TA any questions related to
the course material or about the starter code.

Take time to think about your scene. Scene design is tricky, time spent designing your scene will pay
off at the end when you have a very nice image to show.

Have fun. That's it for ray tracers! After this assignment you should understand how they work
and be able to add additional features extensions on your own.

Previous Years Renders
These are some of the images created by students in previous years