Assignment #3 – Advanced Ray Tracing

Assignment due date: Wed, Nov. 16, 11:59pm

Electronic submission on Quercus
This assignment can be completed individually, or by a team of 2 students

Student Names (Last, First)

Student #1:

Student #2:

Student numbers

Student #1:

Student #2:

Student UtorIDs

Student #1:

Student #2:

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

_____ _____

Student 1 signature ***Student 2 signature***

(note: 3 marks penalty if any of the above information is missing)

Assignment #3 – Advanced Ray Tracing

Assignments 3 brings together everything you have learned up to this point in order to create a high quality, ray-traced scene.

Here you will add advanced features to your ray tracer from A2. You will then design and render a scene of your choice that demonstrates the capabilities of your rendering software.

Learning Objectives – after completing this assignment you will be able to:

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

Explain and implement a complete graphics rendering pipeline. From object definitions to pixel colours on screen. By doing this you will have strengthened your understanding of each step of the image rendering pipeline.

You will be able to handle complex illumination and lighting. You will have written code to deal with area light sources, smooth shadows, and transparent objects.

You will be able to set up the material properties of objects in a scene that will result in specific visual appearance in the image. This will require you to think about how object materials interact with light, and how light reflected or refracted through them will affect the appearance of nearby scene elements.

Skills Developed:

Implementing advanced rendering techniques.

Thinking about multiple light paths shining light on objects within the scene.

Designing and defining complex scenes, including composite objects, procedural geometry, and textured surfaces.

Thinking in terms of light, how it will interact with objects in the scene, and how it will affect the final rendered scene.

Reference material:

Lecture notes for all topics involved in ray tracing.

The detailed comments in the starter code. And your implementation of the basic ray tracer.

Your course instructor! Do not wait if you run into trouble. Come to my office with any questions you may have as you work on this assignment.

Advanced Raytracing

A scene with two spheres and a plane, similar to that of A2, but using advanced rendering techniques

Advanced Ray Tracer [50 marks in total]

For this assignment you will extend the basic ray tracer you built for assignment 2. You will implement several extensions that result in much more realistic and impressive images. You will then use your advanced ray tracer to build a cool raytraced scene.

Required features:

(a) [2.5 marks] Antialiasing – Implement anti-aliasing by super-sampling as discussed in lecture.

(b) [5 marks] Texture mapping – Modify the intersection computation functions so that they return texture coordinates, and implement the texture mapping.

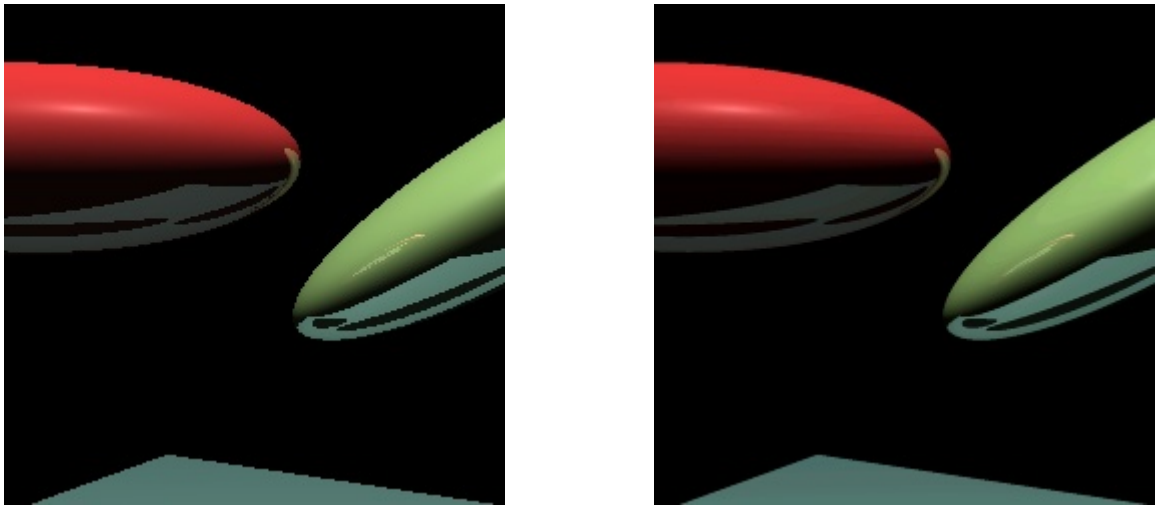
(c) [5 marks] Area light sources – For full marks, your ray tracer needs to be able to produce and handle area light sources of any of these shapes: Plane, sphere, cylinder. The light Sources should be visible as appropriate, and a clean pattern of soft shadows should be generated on objects where shadows occur.

(d) [15 marks] Implement a cool scene – Use your imagination and give us a cool looking scene that showcases what your raytracer can do. I expect creative use of geometry, textures, hierarchical transformations, and Illumination.

(e) [5 - 10 marks] Refractive objects – Implement the code within rtShade() that will handle refractive objects. To achieve 10 marks, your code should handle arbitrarily nested refractive objects.

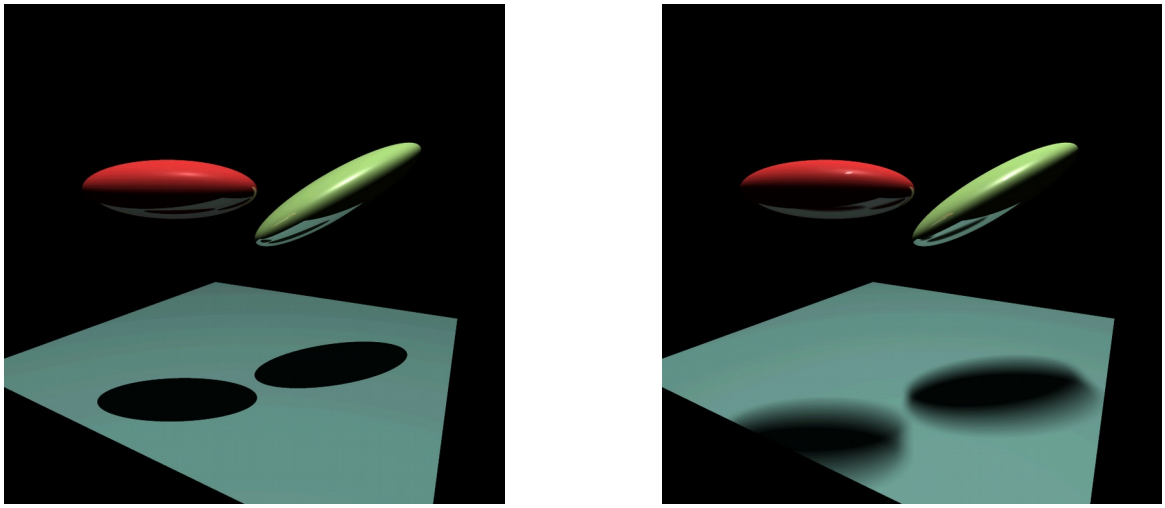
Advanced Raytracing

Anti-aliasing example



The image on the left was produced by the original, basic raytracer. 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. Notice the reflected area light source on the red object.

(e) Advanced techniques – Implement as many as you want! Note that you ***have to implement at least a few below to reach 50 marks***. Choose whatever seems more interesting to you, and what will let you create the most amazing scene!

[2 marks] Multi-threading – Ray tracing is computationally expensive. Modify your code so that it makes use of available resources by multithreading. If done carefully, this will result in a large performance boost.

[5-10 marks] CSG or hierarchical objects – Add to your scene complex objects created via constructive solid geometry, or use hierarchical objects and algorithmic geometry generation (e.g. plant life). The challenge here is keeping track of transformations and figuring out how to handle intersection tests for composite (CSG) objects.
For 10 marks, CSG should support all possible operations (intersect, union, and difference) and you should show complex shapes (e.g. a tea mug or something interesting).

[5 marks] Normal mapping – Implement the code needed to apply normal maps to existing objects to give the impression of surface detail. For full marks this must work on all basic shapes: Planes, spheres, cylinders.

[2.5 marks] Other mappings – *If you have texture and normal maps*, it should not be too hard to add an alpha map, or to create maps for any of the material properties in the basic ray-tracing illumination model.

[7 marks] Depth-of-field – Modify the raytracer to support the thin-lens model, so as to achieve photographic depth-of-field effects including blur away from the perfect focus distance. For full marks, this should have adjustable focal length, aperture, and focusing distance.

[12 marks] Photon mapping – Modify your code so that is supports photon mapping to create illumination effects such as caustics around refracting objects. For full marks this has to be demonstrated on a complex scene where caustics can be caused by multiply-bounced light rays.

[5 marks] Dispersion – Add light dispersion to your photon mapping to simulate the way refracting objects bend light of different colors by different amounts. If you did this for A1 now you will benefit from your effort!

[12 marks] Acceleration – As you add objects, rendering may become very slow due to the sheer number of intersection tests required for each ray. Implement raytracing acceleration either with octrees or BVHs.

[7 marks] Ray marching – Implement ray-marching for arbitrary implicit surfaces. For 10 marks your code should be able to easily add new implicit surfaces (i.e. your implementations should be as generic as possible).

Come and talk with me if you want to implement any of these features and would like a bit of help or a hint; or if you run out of features to implement!

Scene showcasing advanced raytracing features

This was rendered from my solution to A3

Antialiasing and area light sources

Hierarchical, procedurally generated objects (trees and butterflies)

Texture, alpha, and normal mapping

Refraction (supporting nested objects)

Depth of field with configurable aperture and focus distance

Photon mapping (supports both reflected and refracted photons)

What to hand in:

- ***ALL*** your code. That means all .h and .c files as well as the compilation script
- Scripts to ***compile and run ALL of your test images showcasing the features you added***
  * given you are free to modify the raytracer, we require you to provide the scripts to compile and run it with the right scene definitions and features, and with the proper command line call that generates each image we should look at.
- A 1024x1024 render of your final, awesome scene!
- The completed ***feature checklist form***.

Submitting your work

Create a single compressed tar file with the name

Advanced_Raytracer_studentNumber1_studentNumber2.tgz

From just outside your starter code directory:

```
tar -cvfz Advanced_Raytracer_11223344_55667788.tgz ./starter
```

Submit this file on Quercus as usual.

General advice

You must have a fully working ray tracer from A2 to successfully complete A3. You can talk with your peers about parts of A2 that were problematic. See your TA, or come to office hours if you are stuck. However, ***you must absolutely not use any code other than your own in Implementing raytracer features.*** You are ***allowed to use libraries to implement data structures or math functions not available to you,*** but these should be included with your solution and your code must compile/run without us having to tweak the Linux system at the lab.

Ask questions. Don't wait if you have problems. You can drop by for hints and suggestions on how to implement any of the features of the advanced raytracer.

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. Try to create functions that draw interesting objects, and then replicate these instead of assembling everything from primitives.

Marking Scheme	
Advanced Raytracer Implementation	50 marks
Crunchy bonus!	Up to you!

Past Renders

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