Assignment #3 – How to render amazing scenes

Assignment due date: Monday, Nov. 13, 8:30am

Hand-in to be submitted at the start of lecture,
code to be submitted on the mathlab server by the above due date
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 – How to render amazing scenes

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) [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.

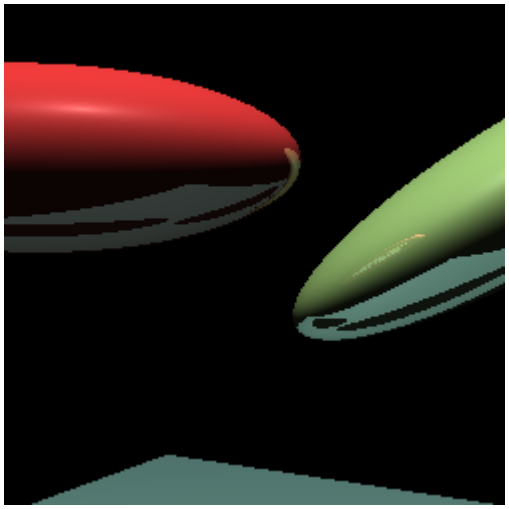
(c) [10 marks] Area light sources – Implement a function to create ***planar*** area light sources. The light source should be visible. You can either insert a set of point-light sources covering the ALS, or you can do random sampling within rtShade(). For full marks, your area light sources must not produce obvious artifacts.

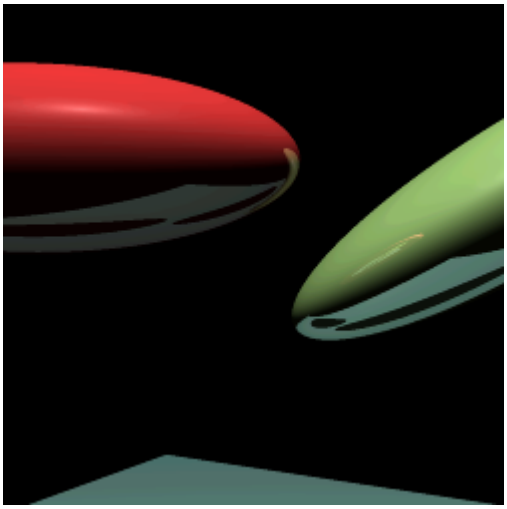
(d) [10 marks] Implement a cool scene – Use your imagination and give us a cool looking scene that showcases what your raytracer can do. I expect creative use of geometry, textures, hierarchical transformations, and Illumination.

(e) [10 marks] Refractive objects – Implement the code within rtShade() that will handle refractive objects. For refractive objects, compute a reflected ray and a refracted ray, remember your first assignment! You've already done this in 2D! However, here you must use the vector formulations to obtain reflection/refraction directions.

Advanced Raytracing

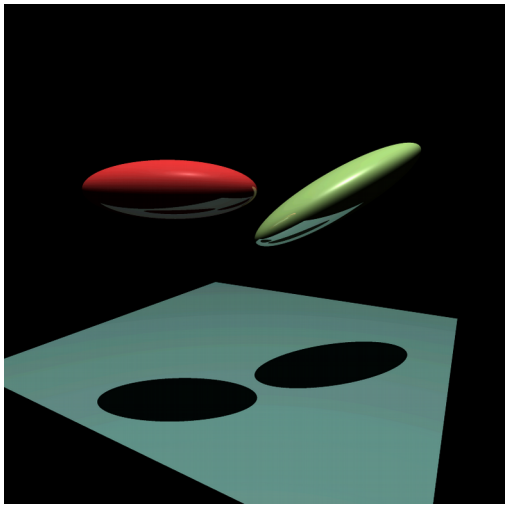
Anti-aliasing example

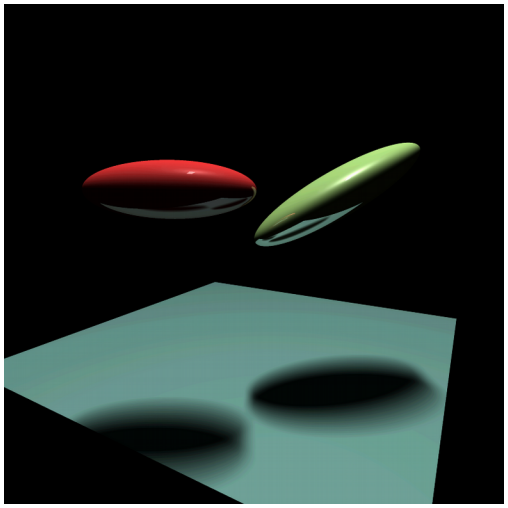




The image on the left was produced by the original, basic raytracer. The image on the right shows the same scene after implementing anti-aliasing by super-sampling

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) [up to 50 marks] Advanced techniques – Implement as many as you want! Choose whatever seems more interesting to you, and what will let you create the most amazing scene!

Note: A3 will be marked out of 50 marks. Compulsory features give you 40. You must earn at least 10 by implementing features below to get 100% in A3. Anything beyond that is bonus marks!

[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.

[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.

[10 marks] Normal mapping – Implement the code needed to apply normal maps to existing objects to give the impression of surface detail. For full marks the effect must work for global reflection and refraction.

[5 marks] Alpha mapping – *If you have texture and normal maps*, it should not be too hard to add an alpha map to your objects so you can control transparency at the pixel level. This will allow you to render objects with holes and create interesting visual effects. Alpha maps are grayscale .pgm images.

[15 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.

[20 marks] Photon mapping – Modify your code so that is supports photon mapping to create illumination effects such as caustics around refracting objects.

[10 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!

[20 marks] Scene octree – As you add objects, rendering may become very slow due to the sheer number of intersection tests required for each ray. Implement an octree-based rendering acceleration framework to greatly reduce the computational cost of ray tracing. Note that for this to make any sense your scene should be complex enough that the benefits of acceleration are perceptible.

Come and talk with me if you want to implement any of these features but need a few hints about how to do this, or when you have questions as you're working on them.

Scene showcasing advanced raytracing features

This was rendered from my solution to A3

Antialiasing and area light sources

Hierarchical, proceurally 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
- Your rendered scene at a good resolution (i.e. no smaller than 1024x1024)
- Your final scene should be named '***Final_Render.ppm***'

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 –c cscd18f17 –a A3 –f Advanced_Raytracer_11223344_55667788.tgz
```

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.***

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
Quiz/Interview	50 marks

Past Renders

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