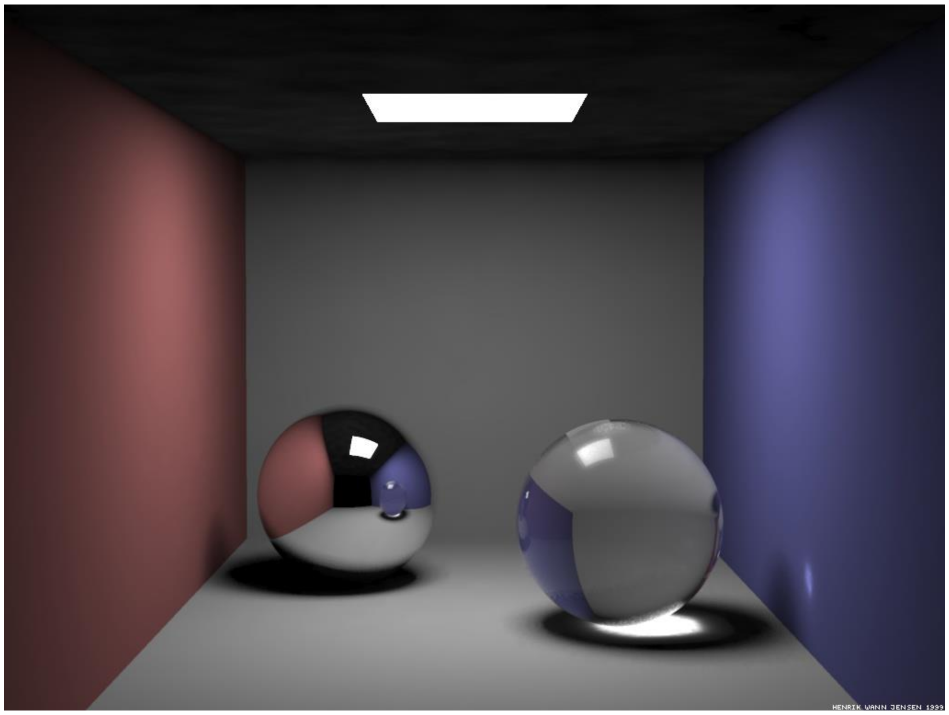
CSC 305 Assignment 1 – Ray Tracer

Due Monday May 23rd, 11:55pm



An example of a high-quality ray-traced image, slightly more advanced than the requirements of this assignment. Image credit: Dr. Henrik Wann Jensen, UCSD. 1999.

Overview

For this assignment, you will write a simple ray tracer using C++. The goal of this assignment is to familiarize yourself with the C++ programming language, and to learn the fundamental aspects of synthesizing a digital image from a three dimensional scene.

The *minimum requirement* of this assignment is to write a ray tracer that renders a sphere from a fixed point of view. The sphere should cast a hard shadow to a floor plane beneath it.

To improve your grade for this assignment, you can add *advanced features*. A list of possible improvements is listed later in this document. You are encouraged to propose and implement your own advanced rendering features, and marks will be given according to the difficulty of your enhancements. Please discuss your proposal with the instructor/TA before implementing it.

Unless you have extraordinary previous background knowledge, it can be very difficult to work on this assignment. Get yourself a good book (see “Additional Resources” section), chat with your instructor/TA/peers to improve your understanding, and get started as early as possible to have time to deal with any unexpected blocking issues.

Minimum Requirements (70%)

Features listed below are worth **70%** of the mark for the first assignment. Everyone in the class is expected to submit this part.

The minimum requirements are as follows:

- The program must compile and run without crashing
- The program outputs an image that renders:
    - one sphere
    - a floor (a plane) beneath the sphere
    - diffuse surface shading on both the sphere and the floor
    - illumination by a single point light source
    - a camera with a fixed point of view
    - a hard shadow casted by the sphere onto the floor

The quality of your submitted code will also be considered, including:

- Appropriate choices of data structures and algorithms
- Efficient use of computational resources
- Ease of reading (useful comments, intuitive naming, non over-engineered design)

Please provide a README file explaining features implemented in your program with your submission.

Advanced Requirements (30%)

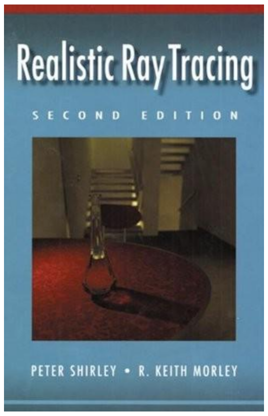
Features listed below are worth **30%** of the mark for the first assignment. Therefore, if you implement all features listed in the minimum requirements and the advanced requirements section, you can get up to a perfect grade in this assignment. If you want to substitute some features with those of your own choice, please consult your instructor/TA.

The advanced requirements are as follows:

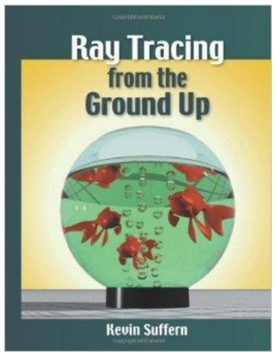
- Render multiple spheres in the scene. **(2%)**
- Implement instancing of objects using a transformation matrix. **(3%)**
- Render a triangle mesh like a cube. **(8%)**
- Implement supersampling anti-aliasing. **(2%)**
- Light the scene with multiple point light sources and cast shadows from all light sources. **(5%)**
- Render a reflective surface, like a mirror ball. **(10%)**

Additional Resources

You can find explanations for how to implement the features of this assignment in many books and on many web sites. This section contains some recommended additional resources, on top of the course textbook and lecture notes.



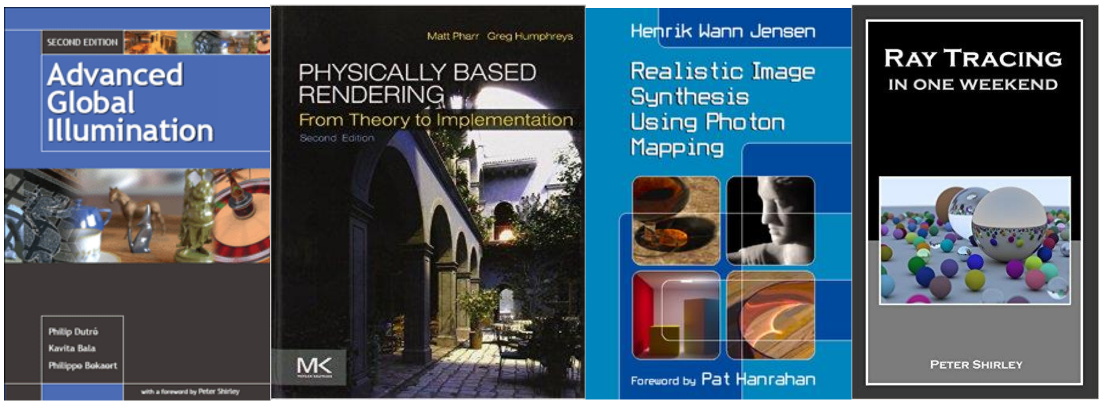
"**Realistic Ray Tracing**" by Peter Shirley is a classic. It's a bit old-school at this point, but still a good reference. It has chapters on ray-object intersections, sampling/filtering, viewing, triangle meshes, instancing with transformation matrices, and lighting methods. It includes example C++ code too, though it is quite old-school C++. There's a copy of this book in the UVic library, and we also have a copy in the graphics lab (ECS 354). Peter Shirley very recently released a mini-series called "Ray Tracing in One Weekend", which might be very helpful for this assignment.



"**Ray tracing from the ground up**" by Kevin Suffern is also a great book to learn how to build a ray tracer. It's thorough, illustrated well, and also has example C++ code. You can find it in the UVic library.

For more information on ray tracing, consider the following texts:

- "Advanced Global Illumination" by Philip Dutré, Kavita Bala, and Philippe Bekaert
- "Physically Based Rendering, From Theory to Implementation" by Matt Pharr and Greg Humphreys
- "Realistic Image Synthesis Using Photon Mapping" by Henrik Wann Jensen
- "Ray Tracing in One Weekend" by Peter Shirley


SECOND EDITION
Advanced Global Illumination
Philip Dutré
Kavita Bala
Philippe Bekaert
with a foreword by Peter Shirley
Matt Pharr Greg Humphreys
PHYSICALLY BASED RENDERING
From Theory to Implementation
Second Edition
MK
Henrik Wann Jensen
Realistic Image Synthesis Using Photon Mapping
Foreword by Pat Hanrahan
RAY TRACING
IN ONE WEEKEND
PETER SHIRLEY

Useful Third Party Libraries

You may find the following third party libraries useful. You don’t have to use them.

- GLM: A C++ library for common vector/matrix operations.
  Please don’t use its built-in ray intersection functions, you have to write those yourself.
  http://glm.g-truc.net/
- “stb_image_write”: An easy-to-use C library for writing PNG/BMP/TGA images.
  http://nothings.org/stb/stb_image_write.h
- “stb_image”: An easy-to-use C library for reading images.
  http://nothings.org/stb_image.h

Submission

Please submit your assignment through Connex. In your submission, upload a zip file that contains the code for your assignment project, with a README file that very briefly explains all features you have implemented.

Do not submit any compilation artifacts, such as “.o” files, “.pdb” files, or “.exe” files.

Marking

The marking of all assignments in CSC305 will be **presentation-based**. Every student is expected to give a short one-to-one/face-to-face presentation of their work to the TA.

At the demo session, please bring a list of features that you have completed, so we can go through it.

The TA may ask questions to test a student's understanding of their solution, and may change parameters of the student's program to test its correctness and robustness.

The presentation will take place in the teaching lab (ECS 354), but the student is free to use their own laptop to demonstrate the assignment.

The TA will not compile and run your code. Therefore, a major portion of the grade is marked during the face-to-face presentation.