

Homework 4: Ray Tracing

Introduction to Computer Graphics and Imaging (Summer 2016), Stanford University
Due Thursday, July 28, 11:59pm

This homework is more open-ended than the other homeworks so far. So, be sure to get an early start to make sure everything compiles and to reserve enough time to add all the features we're requesting.

In this homework, you will be adding some features to a barebones ray tracer using techniques we have discussed in lecture. Unlike our other assignments, we will be leaving nearly all of the software engineering, implementation, and structuring decisions to you.

The ray tracer provided in `hw4.zip` provides only the simplest functionality. Be sure to read through the following files before beginning the assignment:

- `Image.h/cpp`: Provides simple functionality for writing `.png` images; requires `libpng` to compile.
- `Scene.h/cpp`: Stores the objects in a scene and performs basic ray tracing.
- `Shape.h/cpp`: An abstract class for representing a shape; also defines `Ray` and `HitRecord` structs.
- `Sphere.h/cpp`: An example child class of `Shape`.
- `main.cpp`: The entry point for the program. Sets up a scene, initiates ray tracing, and saves the output.

For convenience, we have included the Eigen library for linear algebra. You can see samples of the library in use for processing three-dimensional vectors within the code; visit eigen.tuxfamily.org for full documentation. Eigen is used in many important libraries, so it's worth exploring a bit!

As you work on this project, you will be adding several classes that do not currently exist. You should add appropriate files in the `src/` and `include/` directories and modify the `Makefile` accordingly. This assignment has been tested on the Myth machines and on the CS148 VM (see the `handout.docx` for instructions on installing `libpng` on the VM). Although the code is simple enough that it should work in most environments with little modification; please check that it works on the VM before submitting.

For the assignment, please make the following changes to the ray tracer:

Problem 1 (15 points). *The ray tracer currently only can draw spheres. Add at least one more shapes. Potential options include other primitives, smooth surfaces with closed form equations $f(\vec{x}) = 0$, etc.; remember that all you need to add a shape to your ray tracer are methods for ray intersection and computing normals.*

Problem 2 (15 points). *Right now our output looks pretty boring since there is no color and all the textures are Lambertian. Add support for Phong shading as discussed in the last assignment, and make sure each shape can have its own unique colored material.*

Problem 3 (15 points). *Add support for casting shadows.*

Problem 4 (15 points). *Add support for mirrored surfaces that reflect rays of light according to the law of reflection discussed in class. Be sure to limit recursion depth so that your ray tracer doesn't get stuck!*

Problem 5 (15 points). *Use distribution ray tracing within each pixel to implement simple antialiasing.*

Problem 6 (25 points). *Include a short write-up of your ray tracer in a file entitled `writoup.pdf`. This write-up should include at least three images produced by your ray tracer demonstrating the features you have added.*

Problem 7 (Extra Credit upto 15 points). *Add additional features to your ray tracer; be sure to discuss them in `writoup.pdf` and have them clearly labeled as extra credit. You're welcome to come up with your own ideas; potential extensions include:*

- *More complex materials*
- *Multiple lights or different light shapes*
- *Another application of distribution ray tracing as discussed in class*
- *Instancing*
- *Distance-based falloff attenuating lighting based on how far it has traveled*
- *Refractive surfaces, lenses*
- *Interesting shapes, CSG*
- *Acceleration techniques*

The best ray-traced final images will be shown in class.