CSE 167: Assignment 4—Simple Raytracer

Ravi Ramamoorthi

1 Goals and Motivation

This assignment asks you to write a first simple raytracer. Raytracers can produce some of the most impressive renderings, with high quality shadows and reflections. In fact, you will be using much the same file format as for assignment 2 on the scene viewer, allowing you to make direct comparisons of images produced using standard OpenGL rasterization and with raytracing.

Raytracers are conceptually very simple. However, the actual implementation effort can be considerable. Therefore, you should start early and proceed through the assignment strictly in the order of the specifications provided. For this reason, we also include a milestone. The assignment can be fairly hard to debug. You should strive to make progress incrementally. Start with the simplest functionality (can you render an image with one triangle on the screen?), debug that fully and then proceed. Trying to write the whole thing at once will lead to a mess of undebuggable code.

In this assignment, we recommend but do not require you to work in a group of two (the requirements are only slightly reduced if you work alone, hence the strong recommendation). In general, if you have a partner you would work well with, please work together. However, if you are unable to find a suitable partner and are sure you can handle the workload, it is ok to go it alone.

The instructions for this homework are on edX edge as is the image-grader and feedback server (no code-grading or skeleton code for this assignment). The purpose of this PDF is to document a few additional topics, including the milestone submission, acceleration structures and extra credit.

2 Logistics

We do provide some test scene files. Download the file *testscenes.zip* which has three test scenes, that are documented and have multiple camera positions you should try. There are also images of these scenes with different camera specifications. Note that these images were created in an OpenGL previewer and are a useful guide, but do not have the sophisticated shading, shadows and reflections, that your raytracer will provide for the same images. Beyond these initial examples, we do provide an image-based feedback server for this assignment through edX. As with previous assignments, you are required to provide a link to the feedback server output in your submission. Please also note that the feedback server sends rays through the center of a pixel (i.e., at locations 0.5, 1.5, 2.5 and so on rather than at integer values). This is useful for getting an exact match. Please follow the file format and other instructions on edX edge.

3 Submission

Submission will be using the standard mechanism in the assignment submission (both partners should follow the official CSE 167 submission procedures). In addition, your submission can optionally include the link to a website which has images and documentation of your raytracer (but *please do not post code.*). This website is optional, but is recommended if you want to show off additional examples, and *is required if you want extra credit (in which case, it should document and show example images that justify the extra credit).* Your source code submission should also include a *README* as always. In this case, your *README* should also briefly explain what you did for your acceleration structure (if you implemented one), and point to any images or documentation/timings showcasing that it works.

For this homework, we require that you submit a milestone to remain on track. Milestone submission will follow the same procedure as above, except that you would of course have done much less of

the assignment and may not yet have a link to feedback server results. You also need only include a link to the website or a PDF document, not the source code. *In particular, I expect that you have completed at least the camera routine, so you can make an image of something, at least a quad.* You should also go further, either in beginning to develop the system and parser for the scene geometry, or beginning to code up the ray-surface intersection tests. Your submission should include at least one example image, and a paragraph (in your PDF or website or separately) that also briefly describes how/timeframe you intend to complete the remaining assignment. If you're unable to do this or can render only a black screen, please speak to the instructors or teaching assistants for additional help. No feedback server links are required for the milestone. The milestone will count for about 15% of the total grade in the homework, and it is compulsory to submit a milestone.

4 Acceleration Structure (not required for students working alone)

Please note that the acceleration structure is a required part of the assignment for students working in groups of two, and is documented here, rather than on the edX site.

Ray tracing has historically been a slow process, and the scenes you are provided therefore have pretty simple object geometry. To get raytracing to work on more complicated scenes, some type of ray tracing acceleration structure is required. Therefore, this assignment requires that you implement some sort of acceleration scheme. Note that this refers to a conventional geometric acceleration structure (such as axis-aligned bounding boxes or kd-trees or uniform/non-uniform grids), not optimizations such as parallelization or using hardware.

(This can be difficult; at least 90% of the credit will be given for the core components above, but if you want a perfect score, you must do this part; however, please do it last and you will lose only a very few points if you do not do it. This part is also only required for groups of two; not for students working alone.).

We do not specify what acceleration scheme you should use, and any method is fine. Perhaps the simplest is to just implement a regular uniform grid; each grid cell is a small cube and holds any geometry that intersects that grid cell (so some geometry could be duplicated). When tracing a ray, you go to each grid cell in order, and intersect only with the geometry in that cell. If a ray doesn't intersect some grid cells, they can be avoided altogether. The trick is to find out for each sphere/triangle, which grid cells it intersects, and store them appropriately, and then to augment the ray marcher to know when it hits the next grid cell. You will need to play with grid resolution, perhaps starting with something like $5 \times 5 \times 5$.

If you do get this working, please document the algorithm and speedup in your writeup *README*. You may need new test scenes with thousands of objects to really document and test the speedup (the feedback scenes also include one or two examples for this purpose). Perhaps it is simplest to find *.off* files online which have a very simple geometry format. You could either convert them to your raytracer format, or get the the raytracer to read that format directly.

Again, note that this part will not be judged too harshly, and will lead to only a small deduction if not implemented. However, it is important to do a complete raytracer, and I hope that at least some groups will go forward. We will not be picky; we just require some effort at acceleration (it is also worth a small amount of extra credit for students working alone).

5 Extra Credit

Please do the extra credit only after completing the regular assignment. The number of points of extra credit are small, especially relative to effort required. Up to 10 points of extra credit will be given for additional features and/or the best images produced by your ray tracer. Feel free to exercise your creativity here and go overboard. Obvious ideas are more primitives, better acceleration structures, soft shadows from area lights, Monte Carlo sampling for handling complex lighting and materials, and so on. If you are submitting for extra credit, your submission must include a link to a website, where you document these features and show images demonstrating the extra credit.