
Foreword



Computer graphics involves simulating the distribution of light in a 3D environment. There are only a few fundamentally different algorithms that have survived the test of time. They can be loosely classified into projective algorithms and image-space algorithms. The former class projects each geometric primitive onto the image plane, with local shading taking care of the appearance of objects. This class of algorithm is still widely used because it is amenable to pipeline processing and therefore to hardware implementation as evidenced by all modern graphics cards.

Image-space algorithms compute the color of each pixel by figuring out where the light came from for that pixel. Here, the basic operation is to determine the nearest object along a line of sight. Following light back along a line has given this basic operation and the associated image-synthesis algorithm their name: ray tracing.

In 1980, ray tracing was at the forefront of science. The quality of the images that can be computed with ray tracing was an eye opener, as it naturally includes light paths such as specular reflection and transmission, which are difficult to compute with projective algorithms. Some shapes are easier to intersect rays with than others, and in those early days, spheres featured heavily in ray-traced images. Hence, old images often contained shiny spheres to demonstrate the power of ray tracing.

A vast amount of research was then expended to make ray tracing both more tractable and to include more features. Variants were introduced, for instance, that compute diffuse inter-reflection, caustics, and/or participating media. To speed up image generation, many data structures were developed that spatially sort the 3D geometry. Spatial subdivision algorithms allow a very substantial reduction of the candidate set of objects that need to be

intersected to find the nearest object for each ray. Ray tracing is also amenable to parallel processing and has therefore attracted a substantial amount of research in that area.

All of this work moved ray tracing from being barely tractable, to just about doable for those who had state-of-the-art computers and plenty of time to kill. High-quality rendering tends to take a whole night to complete, mostly because this allows artists to start a new rendering before going home, to find the finished image ready when they arrive at work the next day. This, by the way, still holds true. For many practical applications, hardware and algorithmic improvements are used for rendering larger environments, or to include more advanced shading, rather than to reduce the computation time.

On the other hand, more than 25 years after its introduction, ray tracing has found a new lease on life in the form of interactive and real-time implementations. Such rendering speeds are obtained by using a combination of super-fast modern hardware, parallel processing, state-of-the-art algorithms, and a healthy dose of old-fashioned low-level engineering. Recent advances have enabled ray tracing to be a useful alternative for real-time rendering of animated scenes, as well as huge scenes that do not fit into main memory. In addition, there is a trend towards the development of dedicated hardware for ray tracing.

All of this research has helped to push ray tracing from an interesting esoteric technique for image synthesis to a seriously viable algorithm for practical applications. If necessary, ray tracing can operate in real time. If desired, ray tracing can be physically based and can therefore be used in predictive lighting simulations. As a result, ray tracing is now used in earnest in the movie industry, but also, for instance, in the automotive industry and in scientific visualization. In addition, it forms the basis for several other graphics algorithms, including radiosity and photon mapping.

The practical importance of ray tracing as a lighting-simulation technique means that ray tracing needs to be taught to students, as well as to practitioners in industry. In addition, ray tracing is sufficiently multifaceted that teaching students all aspects of the algorithm will give them all manner of additional benefits: 3D modeling skills, mathematics skills, software engineering skills (writing a ray tracer is for many students the first time that they will have to manage a sizeable chunk of code), hands-on experience in object-oriented programming, and deeper insights into the physics of light, as well as knowledge of the behavior of materials.

It would be ideal to present a ray-tracing course to students at the undergraduate level for all of the above reasons, but also because a deep understanding of ray tracing will make it easier to grasp other image-synthesis algorithms.

For this, a book is required that explains all facets of ray tracing at the right pace, assuming only a very moderate amount of background knowledge.

I'm positively delighted that such a book now exists. *Ray Tracing from the Ground Up* not only covers all aspects of ray tracing, but does so at a level that allows both undergraduate and graduate students to appreciate the beauty and algorithmic elegance of ray tracing. At the same time, this book goes into more than sufficient detail to deserve a place on the bookshelves of many professionals as a reference work.

Kevin was gracious enough to let me read early drafts of several chapters when I was teaching a graduate-level ray-tracing course at the University of Central Florida. This has certainly taught me many of the lesser-known intricacies of ray tracing. Kevin himself has taught ray tracing to undergraduate students for many years, and it shows. This book, which grew out of his course notes, is remarkably easy to follow, especially given the complexity of the subject matter.

As such, I can heartily recommend this book to both professionals as well as students and teachers. Whether you are only interested in rendering a collection of shiny spheres or want to create stunning images of highly complicated and realistic environments, this book will show you how. Whether its intended use is as a ray-tracing reference or as the basis of a course on ray tracing, this book is essential reading.

Erik Reinhard
University of Bristol
University of Central Florida