Ray Tracing Goes Mainstream

Introduction

Whether used to generate images in computer games, your favorite movie, or professional-level Computer-Aided Design (CAD), the ultimate goal for computer-generated graphics is the creation of *real-time* computer-generated images indistinguishable from those of photo-realistic imagery. The more realistic the rendered image, the greater the ability of the artist to conceptualize their idea and improve people's experiences.

Today, professional-level graphics (such as Hollywood films) are rendered offline (as opposed to real-time) using powerful workstations or render farms (many, many computers dedicated to generating images). However, in the last few years, techniques have been developed for real-time rendering of higher quality images through the use of dedicated accelerators using a rendering technique known as *rasterization*. This has enabled a new level of graphics realism. As rasterization evolved, photo-realistic effects such as physically correct lighting, accurate shadows, reflections, and refractions have required programming tricks in order to fit in the computational constraints of rasterization technology.

Figure 1. *Ray tracing automatically generates subtle lighting effects such as light passing through or reflecting off glass.*

An alternate approach to rasterization is ray tracing. This rendering technique generates photo-realistic images and animations by simulating the actual physics of light. Rendering, however, often takes hours or days to calculate. Fortunately, advances in computing architecture, such as multi-core processors, are helping to bring the vast amounts of parallel processing needed for real-time ray tracing into the mainstream. As a result, interest in ray tracing for photo-realistic animation is increasing.

For independent software vendors (ISVs), adding ray tracing to their rendering toolbox represents a huge step forward. It will free them from both the frustrating limitations of today's raster-based approach, as well as provide them with a new competitive advantage. Many consider real-time ray tracing to be critical for enabling ultra-realistic virtual environments, not just for gaming, but also for learning and collaboration.

A Look at How Raster Graphics and Ray Tracing Differ

Raster graphics and ray tracing take different approaches to modeling a scene in 3D.

The rasterization pipeline processes pixels based on the geometry the eye can see on the scene at any given point in time. The raster process's first step involves deciding which geometry the eye can actually see and what is hidden from the eye angle of view. Once the raster pipeline can determine which geometry is visible (a time-consuming part of the rendering process), it reads the remaining geometry of the scene in terms of the smallest possible denominator: triangles. It then applies lighting and shading to determine how each triangle should be colored. Textures are often applied on top of these triangles, and dedicated hardware is able to map the pixels of a 2D texture image onto the appropriate coordinate in 3D space. Every triangle is treated independently of other triangles, and there's no consideration for how one triangle might affect another. Therefore, casting a shadow or reflecting a ray of light requires separate computation in the form of a shader. However, programming these shaders adds significant complexity to the scene, both for the artist, and the computational abilities of the hardware accelerator. It also often requires multiple passes through the rendering pipeline before the image is suitable for display. To get the best image, ISVs must continually learn new techniques for producing ever-increasing complex images using what are essentially tricks and shortcuts meant to fool the eye into believing that physically incorrect approximations are in fact modeling the real world. These tricks and shortcuts take many hours to implement, requiring additional software developers and longer development times. Additionally at a later date, updates in the graphics hardware or software can thwart all the software developers' hard work, causing the desired visual results to break or look grossly inaccurate.

Ray tracing, conversely, attacks the problem much more directly and, from the software developers' point of view, more efficiently and dependably. It models light interactions on a physically correct simulation that tracks the path that light rays travel as they bounce around the scene. Unlike raster graphics, which concentrates solely on the current state of each triangle, ray tracing maintains random access to the entire geometry of the scene. By relying on what is called an "acceleration structure," in which any given object is decomposed into hierarchic regions of filled or empty space, ray tracing is able to efficiently determine the path a ray would take through space and test it only against those triangles that would be in the vicinity of the ray's trajectory. If the scene is dynamic, then the acceleration structure may have to be rebuilt on-the-fly for each new frame of animation. In an early research project, Intel researchers found several solutions for improving performance for dynamic scenes. In this way the computer automatically generates lighting effects such as shadows and reflections based on the laws of physics in a single pass. Since no tricks are required, compositing the scene becomes much more straightforward, allowing software developers to achieve compelling realism in a scene with less time and resources than with raster techniques.

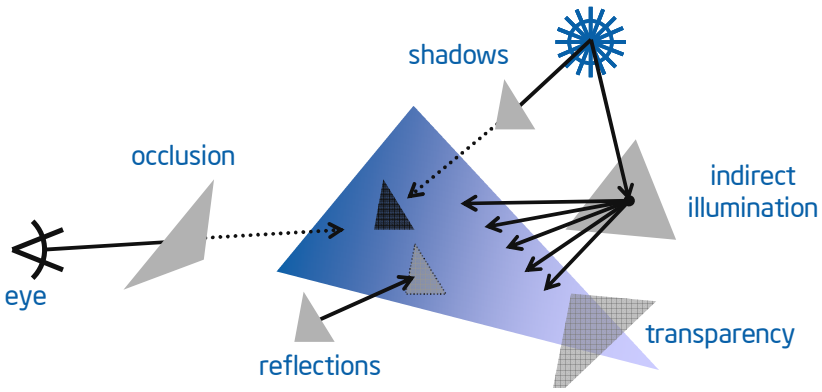

Figure 2. *Ray tracing engines consider the optical interactions between nearby triangles.*

A Clear Advantage in Scaling

One very promising advantage of ray tracing over raster graphics is application scaling. Ray tracing application performance is strongly affected by the number of rays shot through and allowed to bounce around a scene, but is only loosely affected by increasing the complexity and detail of the scene. This is very different from raster graphics, where performance scales with the number of triangles and pixels rendered, and complex scenes constitute a large workload for the computer and/or the graphics hardware. For a fixed resolution image, this workload roughly doubles as the complexity of the scene doubles. For ray tracing, however, you would have to increase the *visible* scene complexity by up to *ten times* to see an equivalent increase in workload for the computer.

Intel researchers have also found that ray tracing lends itself exceptionally well to multi-processing, and the performance of ray tracing scales almost linearly with the number of processing cores. This was demonstrated by performance data collected on symmetric multiprocessing (SMP) systems with 2, 4 and 8 total cores. Our data shows that multi-core processors will become increasingly well-suited to applications using ray tracing techniques as additional cores are added.

Implications for ISVs and Their Customers

Intel's research shows that the time is nearing when hardware innovations, as well as new advances in ray tracing engines, will allow this technology to finally become mainstream and useful in everyday applications for everything from home entertainment to business. Ray tracing will eliminate many of the "no win" compromises made in computer graphics, such as having to choose performance over geometric complexity, or having to fake the modeling of certain special effects using rasterized shaders, which often result in shadows with jaggy outlines which require additional graphics hardware processing to fix, or reflections that do not accurately model the objects being reflected. Using ray tracing, ISVs can simply cast more rays to turn on these extended features, and achieve greater amounts of physically correct photo-realism. What's more, a scene's computational effort can be calculated very accurately, and parameters can be adjusted to hit the desired frame rates to make the animation in the game, movie or other application smooth and enjoyable.

The good news is the move to ray tracing doesn't require starting from ground zero. Everything that gets used in raster graphics, such as multiple texture mapping for simulation of fine detail surface texture, still works, as well as all the sampling and filtering that goes with it. Ray tracing will benefit everything from video gaming and future blockbuster movies to professional-level CAD applications. ISVs in all disciplines should be ready to put ray tracing on their technology roadmaps. Future computer architectures with tens or hundreds of energy-efficient cores and teraflops of computation will enable real-time ray tracing for photo-realistic rendering.