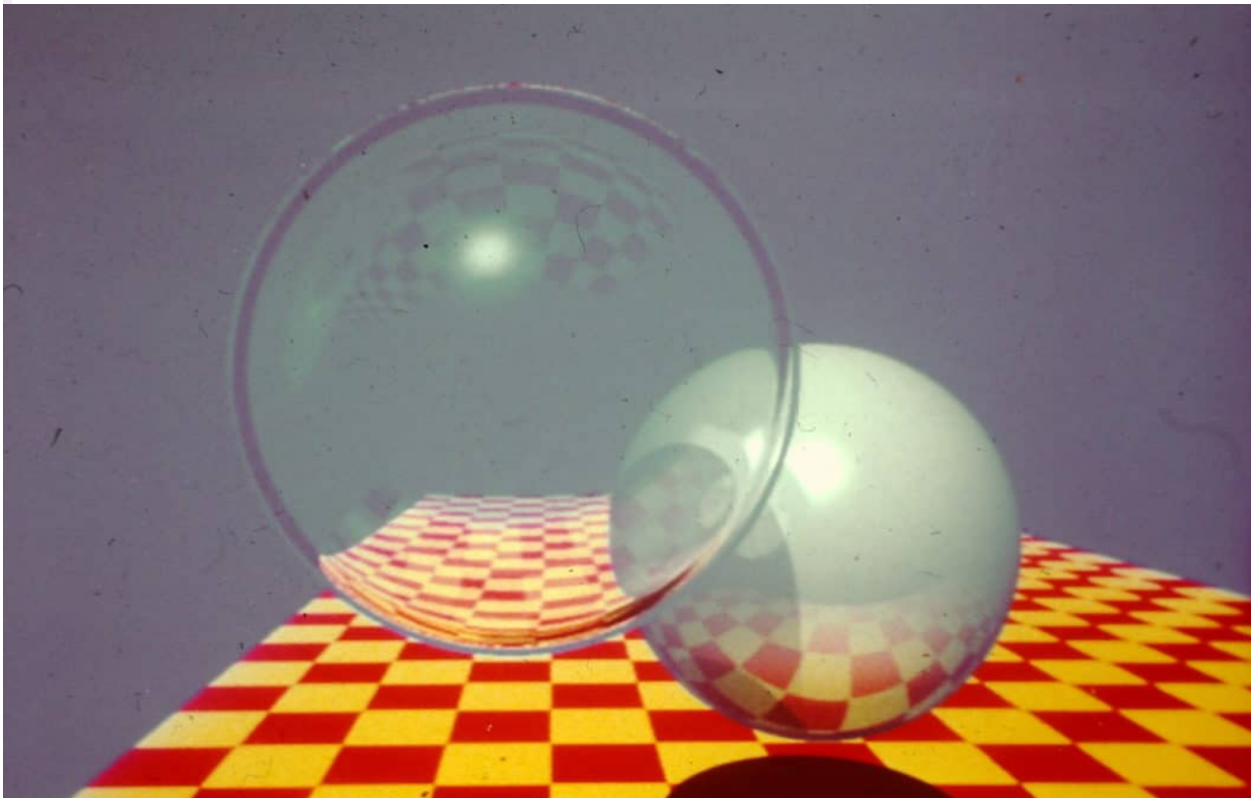
A Ray-Tracing Pioneer Explains How He Stumbled into Global Illumination

August 1, 2018 by J. Turner Whitted

Share



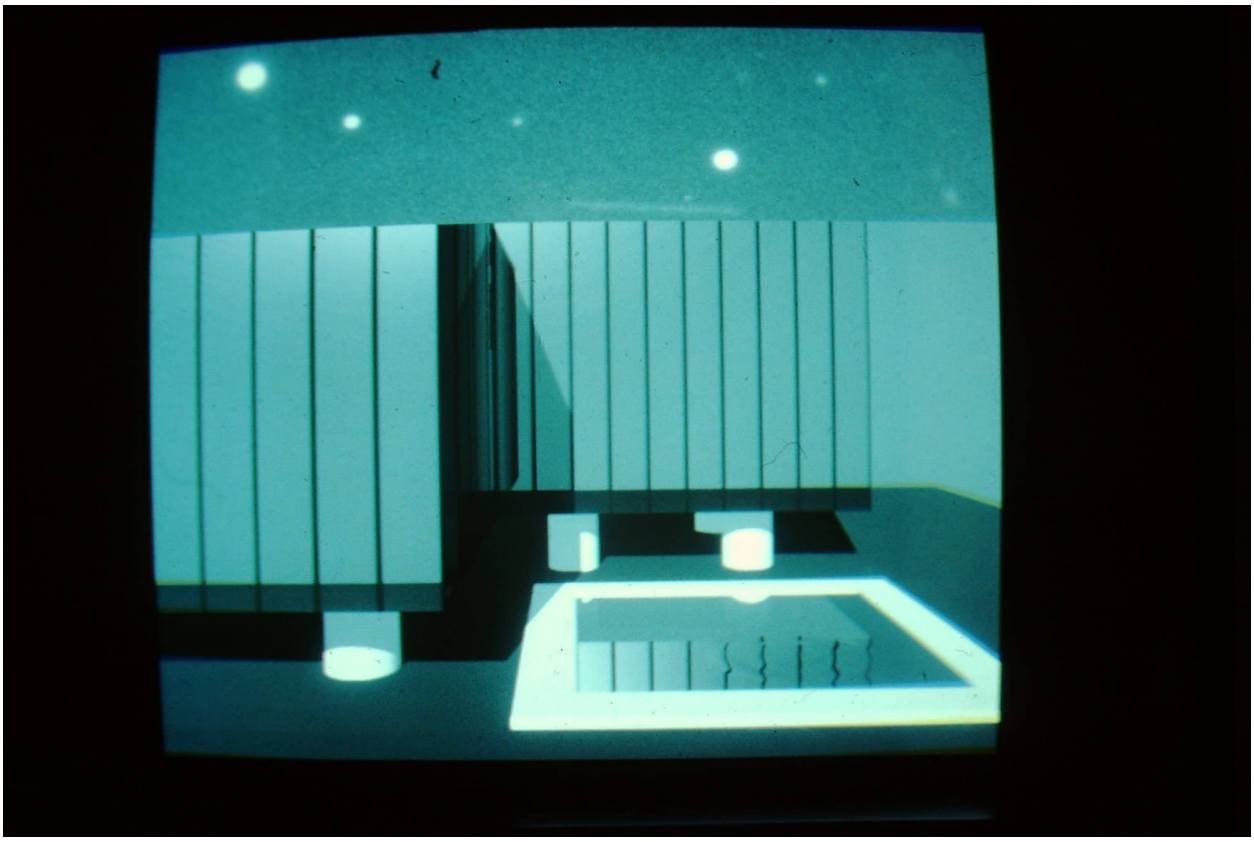
Editor's note: *You might not have heard of ray tracing, but you've certainly seen it. The graphics rendering technique mimics the way light works in the real world to create the special effects woven into practically ever modern movie. With the arrival of our RTX real-time ray-tracing technology, we're working to make this sophisticated technique faster and more accessible, promising to bring cinematic quality to everything from games to product design. Today, we're sharing a big part of the story of how modern ray tracing came to be from one of its pioneers, J. Turner Whitted. Now with NVIDIA Research, Whitted wrote the seminal paper, "An Improved Illumination Model for Shaded Display," which serves as part of the foundation for modern ray-traced computer graphics.*

Ray Tracing and Global Illumination

As most of us learned in high school physics, ray tracing is an idea that has been around for centuries. Global illumination in computer graphics has a somewhat shorter history.

Early computer graphics shading models proposed by Henri Gouraud and Bui Tuong Phong, among others, computed reflections based on the spatial relationship between light sources and a point on a surface. Somewhat later, Jim Blinn developed a reflection model derived from Ken Torrance's work describing inter-reflections of the microscopic structures of surfaces. (That model was later expanded by Rob Cook and Ken Torrance himself into the widely used Cook-Torrance shading model.) Shortly thereafter, Blinn and Martin Newell published a paper describing environment mapping, which replaced light sources with a 360-degree texture map of the surrounding environment.

It was this development of environment mapping that caught my attention. The rendering of a shiny teapot with doors and windows reflected from its surface yielded a level of realism that I had never imagined for computer-generated imagery. I stared at the image for hours and wondered how one could ever improve it. Environment mapping did have one huge limitation of not being able to accurately render reflections of objects close to the object being rendered.



An image from Turner Whitted's 1979 SIGGRAPH paper featured a structure resembling the Bell Labs building in Holmdel, New Jersey, where Whitted's work was done.

Clearly the effects of Phong's model are local and Torrance's model and its derivatives are microscopic in scale. It seemed natural to label Blinn and Newell's model as "global" in scope. Hence the term "global illumination." The question then became how to implement it without the limitations of the environment map?

Why Ray Tracing Is a Natural Choice for Global Illumination

Because ray tracing is so incredibly simple, it should have been an obvious choice for implementing global illumination in computer graphics. Ray casting for image generation had been pioneered by Arthur Appel, at IBM, and commercialized by Robert Goldstein and associates, at MAGI. MAGI had originally utilized multi-bounce ray tracing to track radiation within tanks. In my own early career, I had been involved with ocean acoustics and remembered a diagram of ray-traced sound being refracted through varying depths of the ocean and reflected from the surface. Eventually a memory of that diagram popped into my head and it then became clear to me how to improve upon the global illumination method that Blinn and Newell had initiated.

My own hesitation about using ray tracing was simply concern about performance. There has always been a bit of a divide between computer graphics for real-time

interaction and graphics for film. General Electric's lunar landing simulator obviously had to run in real time to train astronauts. This constraint continued with David Evans and Ivan Sutherland's flight simulators as well as neighboring research at the University of Utah. Henri Gouraud's smooth shading ran in real time at essentially no cost. A little known chapter in Bui Tuong Phong's dissertation includes a description of circuitry for real-time shading even though most implementations of Phong shading did not run in real time.

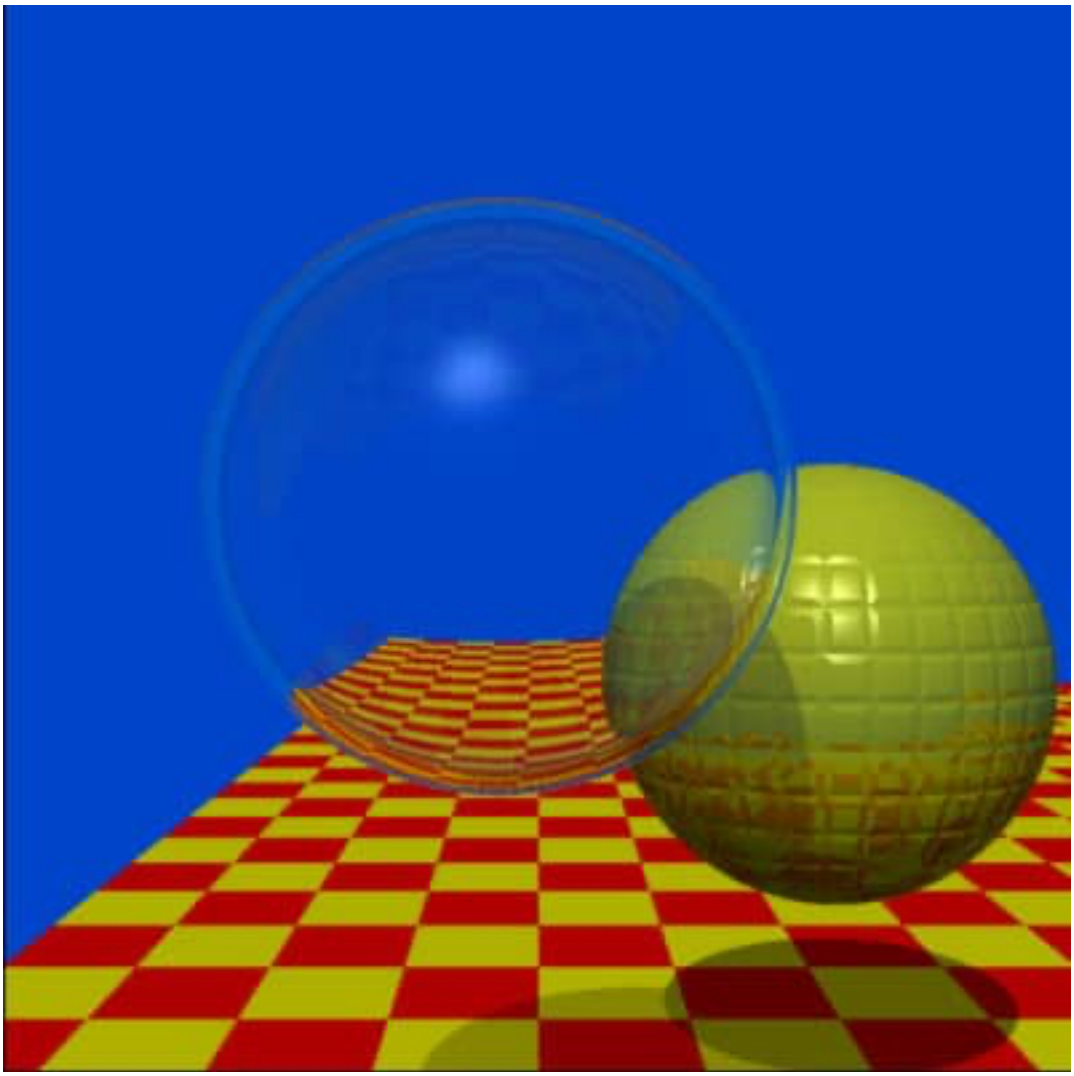
The availability of frame buffers changed the computer graphics landscape dramatically. Rather than attempting to render images at the refresh rate of CRTs, it became possible to render at any speed and view a static image on the screen. Ed Catmull's subdivision algorithm coupled to a z-buffer could render arbitrarily complex curved surfaces into a frame memory in a matter of minutes rather than milliseconds. Blinn and Newell's environment mapping required tens of minutes per frame. This trend toward higher realism at slower rates gave me the courage to try something even slower.

Adding Shadows, Refraction

In 1978, the Computer Systems Research Laboratory at Bell Labs was equipped with a Digital Equipment PDP-11/45 minicomputer and a 9-bit per pixel frame buffer. Those resources along with the C programming language were the perfect environment to attempt a ray-tracing improvement to Blinn and Newell's scheme. (I should've read more of Brian Kernighan and Dennis Ritchie's programming manual before attempting to write my first C program. As a hard-wired assembly language programmer, I didn't realize that C allowed recursion. Instead, I laboriously instituted my own stack and manually wrote a multi-bounce recursive ray-tracing program without using the recursion provided by the language itself.)

Eventually an image of inter-reflecting spheres showed up on the screen. Both Newell and Frank Crow had included checkerboards in some of their early images and I felt a need to copy them. So far, so good. At this point, tracing additional rays to the light sources was a no-brainer. The resulting shadows cost a bit of time, but the additional coding was minor and the result was worth it.

The minor coding change was to turn a list of rays into a tree of rays. As an electrical engineer and hardware designer with no computer science background, that was a data structure that I had never dealt with. Nevertheless, it worked.



In 1979, Whitted's groundbreaking, computer-generated image of inter-reflecting spheres illustrated the value of ray tracing for global illumination.

Well, if we could get that far, why not go for broke and add refraction just like we learned in high school physics? That took some work, but the result was stunning.

Adaptive Super-Sampling to the Rescue

As one might imagine, the execution times of these incremental features got out of hand. To estimate the rendering time for each image, I reduced the 512×512 image resolution by a factor of 64 and timed it with a watch. Multiplying the time by 64 gave me a fairly accurate guess for full-resolution rendering. But the jaggies visible at full resolution needed to be eliminated and super-sampling was the only practical approach. The estimated rendering time then expanded by an additional factor of 16 — and panic set in.

I had written a draft of a SIGGRAPH conference submission and was rendering illustrations to be included in the paper. The submission deadline was near, but with 16x super-sampling, the estimated rendering times extended beyond the submission deadline. The spontaneous idea of adaptively super-sampling was a life saver because it only added additional samples where needed. It was implemented within a couple of hours and the paper was edited to include this new idea while the illustrations were being rendered.

This efficiency became even more critical when it came time to create the sequence of images seen here in this animation:

The Compleat Angler – First demonstration of recursive ray tracing in computer graphics animation
https://youtu.be/0KrCh5qD9Ho

At lunch a few months later, a Bell Labs executive asked what it would take to run ray tracing in real time. At the time our new VAX-11/780 was computing each pixel of a simple scene in 1/60th of a second, so I proposed a vast array of Cray supercomputers, one per pixel, with a red, green and blue light bulb on top of each one. That was meant as a joke. It turned out not to be.

Real-Time Ray Tracing

In my original 1979 ray-tracing paper, there is mention of partitioning of tasks among multiple processors. Sometime afterwards I began to hear the term “embarrassingly parallel” in the context of ray tracing. In the original paper there is also a hint of

acceleration structures so long as a sphere happens to be contained within another sphere. Within a couple of years, I teamed up with Steve Rubin to apply his idea of hierarchical bounding volumes to ray tracing. It worked nicely and we were able to render scenes more complex than spheres floating over a checkerboard, but it still wasn't in real time.

Light and shadows: This bottle in a window is taken from a 1980 SIGGRAPH paper that introduced bounding volume hierarchy structures to accelerate ray tracing.

In years to follow, I wrote a couple of illustrative ray tracers as a teaching aid, but basically moved on to other topics. Other researchers, however, had taken up the slack with acceleration techniques that made an impractical algorithm practical. That was accompanied by additional advances that achieved image quality far beyond that demonstrated with spheres and checkerboards.

All of this happened on top of the progression of Moore's law and the availability of parallel processing. Like a Rip van Winkle in computer graphics, I can wake up nearly 40 years later and see real-time ray tracing as a commodity. How did that happen? Obviously some very bright people put a lot of effort into it.