

*The quality of 3-D computer graphics is poised for a quantum jump forward, thanks to speedier ways to simulate the flight of light*

By W. Wayt Gibbs

**F**or those of us who frittered our formative years away blasting blocky space invaders, video games today can widen the eyes and slacken the jaw. The primitive pixelated ape of Donkey Kong has evolved into a three-dimensional King Kong of startling detail. Some newer Xbox 360 games render their lead characters from an intricate mesh of more than 20,000 polygons, each tiny patch drawn dozens of times a second with its own subtle texture, shading and gloss.

Beyond the booming game industry, the evolution of graphics has lifted interactive software for design, engineering, architecture, medical imaging and scientific visualization to new heights of performance. Much of the credit belongs to advances in graphics processing units (GPUs), the microchips at the heart of computer video cards that transform 3-D scenes into 2-D frames at speeds faster than a trigger twitch. As the rendering capabilities of GPUs soared, so did the revenues of ATI, NVIDIA and Intel, which make the most popular models.

A wide gulf of verisimilitude, however, still separates interactive graphics from feature film effects and photography. And some experts say the only way that personal computers will ever cross that gulf—to reach the nirvana of computer graphics in which synthetic scenes display all the fluid motion and subtle shadings of reality—is through a basic change in how machines render 3-D models.

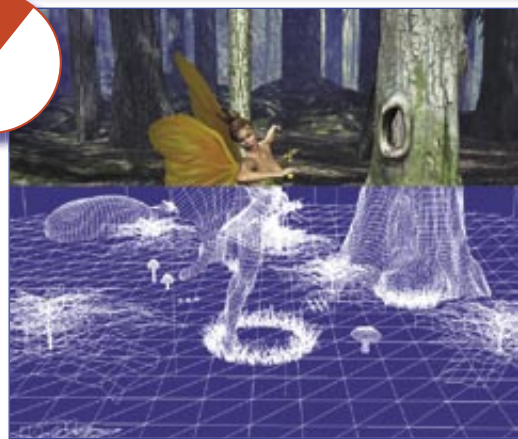
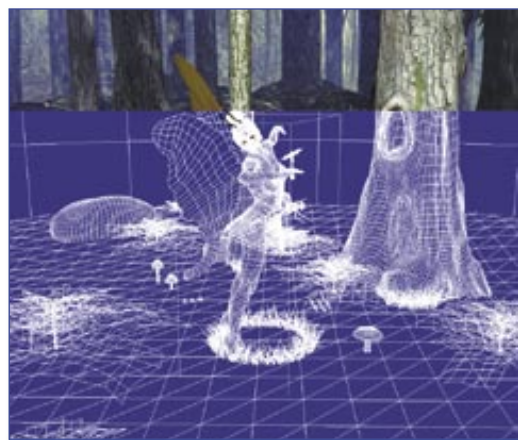
This change, from the so-called rasterization method that GPUs use to a more scientific approach known as ray tracing, was long dismissed as infeasible for interactive, rapidly changing scenes. But advances in both software and hardware have recently propelled ray tracing to within range of the consumer PC market.

Ray tracing, which originated with Renaissance painter Albrecht Dürer and

# A Great Leap in Graphics



W. WAYT GIBBS; MODELS COURTESY OF DAZ PRODUCTIONS AND INGO WALD



was first reduced to computer code in the 1970s, is now in the early stages of its own renaissance, says Philipp Slusallek, a computer science professor who leads a ray-tracing research group at Saarland University in Germany. The timing is fortuitous. “GPUs are running up against the wall,” he asserts.

## Ray Tracing Reborn

THREE PROBLEMS VEX GPUS as they try to render a 3-D scene by rasterizing it. All three weaknesses stem from the initial step in rasterization: dissecting the virtual world into a montage of flat polygons such as triangles, each of which is treated as if it were independent of the rest. The first issue is that most real objects have curves and look unnatural when approximated by facets.

When modelers can afford the labor involved, they add more polygons to round out bulges and concavities. But that exacerbates the second problem, which is that raster systems are designed to process every polygon in the model, even those hidden from view. “Until it has rendered the very last polygon, the system won’t know whether that final shape covers up every other object,” explains Gordon Stoll, a graphics researcher at Intel. The computational cost of a raster rendering thus rises in direct proportion to the geometric complexity of the scene. So every time the amount of detail doubles, the rate at which a GPU churns out frames of video falls by about half.

A third and more important problem, Slusallek argues, is that “shadows, reflections and other effects simply cannot be done right on GPUs.” The reason, Stoll explains, is that “it is just physically incorrect to assume that polygons are independent. The appearance of every object in a scene depends on every other object, because light bounces around.” Computer scientists refer to this phenomenon as the problem of global illumination. Newer GPUs can reprocess the scene multiple times to approximate indirect lighting. But that work-around chews up memory, clogs internal data channels and still falls far short of photorealism.

Ray tracing avoids these problems by simulating the ricochets of light rays throughout a scene. Where raster engines rely on tricks, approximations and hand-tweaked artistry, ray-tracing programs instead lean on the laws of optics—and with physics comes fidelity. “Rays of light really *are* independent,” Stoll notes, so in ray-traced images, reflections, shadows and smoke appear as they should.

“And when I turn on these effects in a ray tracer, they automatically compose properly, so I can get a reflection of a shadow of a puff of smoke—that’s not true for raster graphics,” he adds. So far only ray tracers have been able to arrive at near-perfect solutions to the problem of global illumination.

When Pixar Animation Studios set out to make *Cars*, an animated film that opened in June, artists there found that the metallic bodies of the vehicular characters did not shine quite right unless they added ray tracers to the usual raster rendering system. On all its previous movies, the studio had resisted using ray tracing for the same reason that makers of games and other interactive software have: the intensity of computation that the physics demands has always brought microprocessors to their knees. Even with Pixar’s fast network of 3,000 state-of-the-art computers, each second of finished film took days to render. Film producers may tolerate such delays, but gamers, engineers and doctors generally will not.

Ray tracing now seems set to leap the speed barrier, however. Better algorithms and custom-designed hardware have accelerated ray-tracing renderers by more than two orders of magnitude. As ray tracing goes “real time,” fast-moving computer imagery should become both dramatically more realistic and easier to create.

By 2003, Slusallek was so convinced real-time ray tracing was ready for mar-

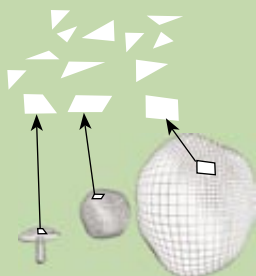
FLEET-FOOTED ALGORITHMS and custom microchips for ray-tracing 3-D models can now render hundreds of frames of a complex and rapidly changing scene [opposite page] in less time than conventional ray-tracing software takes to draw a single frame [this page].



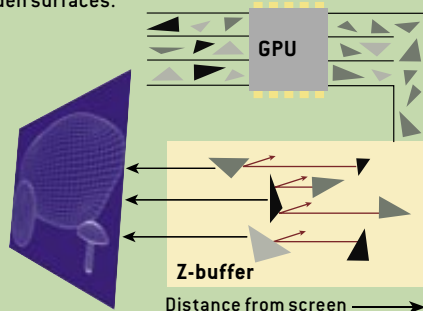
## FROM FAST RASTERS TO REAL-TIME RAYS

### RASTER GRAPHICS

Raster graphics systems in today's computer video cards and home console game machines start with a 3-D scene that the software has divided into objects and then subdivided into polygons (typically triangles).



The polygons stream through "pipelines" in the graphics processor [GPU], which work on many pieces in parallel to transform their geometry and to compute their shading. A "z-buffer" then sorts polygons by distance from the viewpoint, selects the frontmost for display, and tosses out the computed results for hidden surfaces.

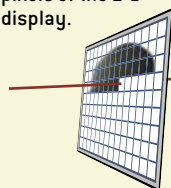


The polygons often must pass through the pipelines multiple times for each frame of video so that the system can calculate the shading, textures, translucency, reflections and other effects that add realism to the scene.



### TRADITIONAL RAY TRACING

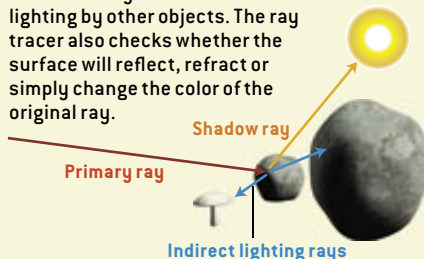
Ray tracing renders a 3-D scene by shooting virtual rays through the pixels of the 2-D display.



The scene is stored as a database of objects, which can include curved as well as flat surfaces.



When a ray hits an object, the system launches "shadow" rays toward each light source in the scene and rays to test for indirect lighting by other objects. The ray tracer also checks whether the surface will reflect, refract or simply change the color of the original ray.



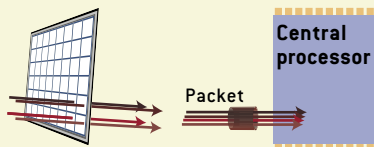
Thanks to its recursive nature, this method can render a scene accurately in just one pass.



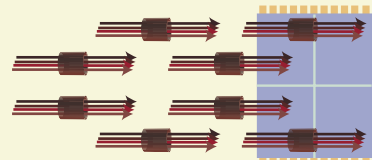
**REAL-TIME RAY TRACING** Real-time ray tracing is already in commercial use on high-end servers and is becoming increasingly feasible on consumer-level computers. Three kinds of advances have cut rendering time from hours to a fraction of a second per frame.

**1** Running rays together in parallel now happens at several levels within real-time ray tracers on desktop PCs.

Programs group similar rays into "packets," then march all the rays in a packet in lockstep through the same set of computations.

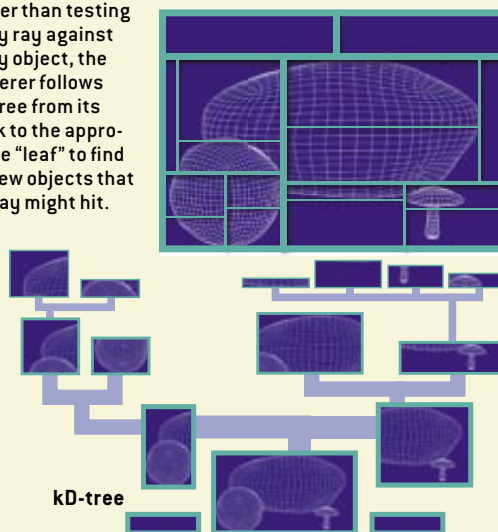


Each packet is processed by a separate program thread. Newer multicore CPUs can run more than a dozen such threads concurrently.



**2** Acceleration structures split the 3-D scene into a hierarchy, called a kD-tree, organized so that each section carries roughly equal computational cost.

Rather than testing every ray against every object, the renderer follows the tree from its trunk to the appropriate "leaf" to find the few objects that the ray might hit.



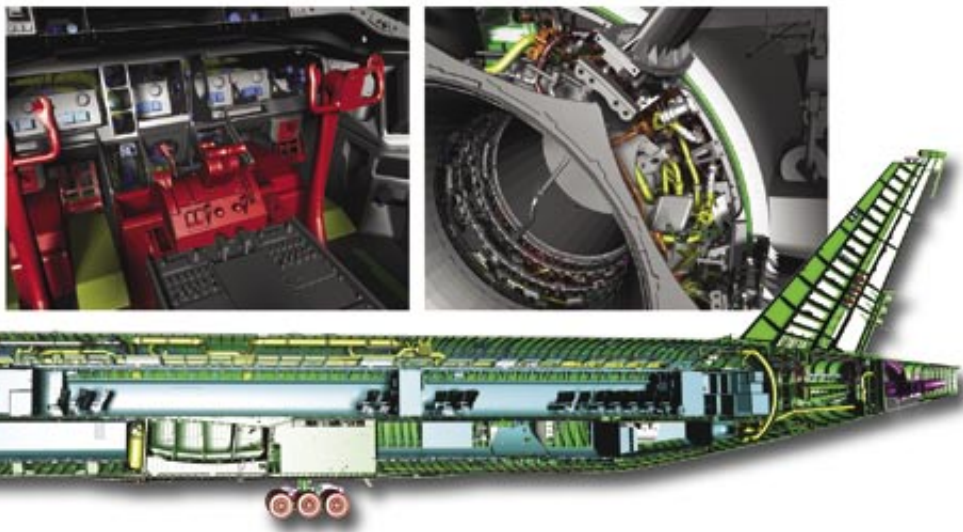
**3** Customized microchips built last year at Saarland University run at a mere 66 megahertz in prototype versions yet can render some ray-traced scenes more quickly than a 2,600-megahertz Pentium-4 system.

Once it is commercialized, the "ray processing unit," or RPU, should run roughly 50 times as fast—more than speedy enough for interactive software.



Prototype RPU is tested by its creators at Saarland University.

EVERY BOLT, KNOB AND CABLE is included in Boeing's 3-D model of its 777 jet. A scene of this complexity overwhelms traditional raster rendering approaches, because they attempt to process every one of its 350 million polygons when drawing a frame of video. Ray-tracing systems, which by their nature ignore any objects that are hidden from view, handle such detail with aplomb.



ket that he and his colleagues at Saarland spun off a company to commercialize the technology. They founded inTrace, using first-generation software that required a cluster of high-powered servers to render full-resolution, photorealistic images at 10 frames a second or higher. BMW, Volkswagen, Airbus and other firms snapped up inTrace systems despite the price and now use them to evaluate the sight lines, interior reflections and curb appeal of vehicle designs still on the drawing boards.

## Boosting the Speed of Virtual Light

RAY-TRACING ALGORITHMS have meanwhile improved so much that they can achieve interactive speeds on a single high-end PC. In 2004 Slusallek and his co-workers Carsten Benthin and Ingo Wald demonstrated a way to use ray tracing to rapidly render scenes crowded with curved, free-form shapes, without first carving them into polygons the way GPUs and "classic" ray-tracing programs do. And within the past two years Slusallek, Stoll and Wald (who is now at the University of Utah) have each demonstrated significantly faster ways to identify which object, if any, a virtual photon hits as it traverses a scene from screen to light source. (To save on computation, the rays are traced in reverse; the physics is the same.)

Testing a ray for collisions with an object, Slusallek says, is like finding a specific book at a library. "You don't start looking at the top of the first shelf each time. You use an index." Building an index to the database of 3-D objects is easy. The hard part is figuring out how to rebuild it within a few milliseconds every time something changes in the model. "It's as if the shelves in the library are constantly moving," he says.

Stoll and his Intel colleague Jim Hurley are working on a system that improves on the classical approach, which is an index known as a kD-tree. A kD-tree chops the 3-D space into pieces of various sizes and then arranges them into a treelike 3-D hierarchy. To save time, Intel's Razor system selects the branches of the tree it needs for each frame and rebuilds only those. Razor's kD-tree also represents the scene at multiple levels of detail, so that it can quickly draw a distant castle, for example, without having to fetch data on every one of its bricks.

"Razor is our most ambitious, over-the-horizon work," Hurley says. "It can handle explosions, splashes and the full range of lighting effects. It's not very fast yet, but by optimizing the code, we expect to boost performance by a factor of 50 to 100." The optimizations will exploit the ability of the latest "multicore" central processors to run a dozen or more program threads simultaneously.

Last year the Saarland group designed a new chip that can run many ray-tracing calculations in parallel. In tests the "RPU"—or ray processing unit—is able to churn out tens of ray-traced frames a second. To demonstrate its abilities, Slusallek had two students create a virtual island with 40 million polygons and ocean waves that reflect stars and fires on the beach. "We did all this in a couple of months," he reports. "With ray tracing, you don't need all the artistic work that goes into making rasterized environments look realistic; you just make the model, press a button, and it looks great." Slusallek is now lining up investors to commercialize the design as his team crafts software drivers and compilers for the RPU.

"The technology is finally there to handle highly detailed, realistic environments in real time," Slusallek says. "It isn't clear yet which platform will be the one that allows ray tracing to really take off into the mainstream: it could be big multicore CPUs, customized processors like our RPU, or ray-tracing features added incrementally by the GPU makers." What does seem clear is that a disruptive shift has begun that signals a new burst of evolution in computer graphics. SA

W. Wayt Gibbs is senior writer.

## MORE TO EXPLORE

**Ray Tracing Goes Mainstream.** Jim Hurley in *Intel Technology Journal*, Vol. 9, Issue 2; May 2005.

**RPU: A Programmable Ray Processing Unit for Realtime Ray Tracing.** Sven Woop, Jörg Schmittler and Philipp Slusallek in *Proceedings of ACM SIGGRAPH*, 2005.

**Razor: An Architecture for Dynamic Multiresolution Ray Tracing.** Gordon Stoll et al. University of Texas at Austin, Department of Computer Sciences Technical Report #TR-06-21, 2006.