# Ray Tracing in Industry

## An up-to-date review of industrial ray tracing applications and academic contributions

Hugo Pacheco

Departamento de Informática
Universidade do Minho, Braga, Portugal,
{hpacheco}@di.uminho.pt

**Abstract.** Ray tracing is a rendering method for generating realistic 3D computer graphics, better in a wide scope than the currently popular rasterization techniques, but is also labeled to be much more computationally expensive and for such a reason it is generally delegated to offline high-quality renderings.
In a world where processing power increases "faster than the speed of light", the walls that part ray tracing from online real-time applications are starting to thicken at an exciting pace and unveiling a whole new domain of interactive scenarios.
In this survey, we debate the pros and contras of ray tracing in comparison to rasterization methods and emphasize some of the possible application scenarios that are slowly convincing the graphics industry of ray tracing as a worthy mid-term investment.

*Keywords* ray tracing, rasterization, realistic rendering, interactive rendering

## 1 Introduction

Ray tracing has its roots in geometric optics and exists for as long as mirrors and optical instruments such as telescopes are known. It conceives a method for calculating the path of light as straight lines that is very precise for the design of optical lens systems[9].

In computer graphics, ray tracing can be seen as a recursive algorithm to compute the color of a pixel[23]. Individual light rays are shot from the viewer to the light sources that originated them (Figure 1). When a ray hits a surface, it can suffer up to three phenomena: reflection (the ray direction is changed so that it returns to the medium from which it originated, for example, for mirrors), refraction (the angled change of direction does not prevent the ray from traversing the object, for example, if it is translucent) and or shadow. To further avoid tracing all rays in a scene, a shadow ray is used to test if a surface is visible to a light (trace a direct line to each light source and test intersection with opaque objects in the scene). In case it is, the light contributes to the object's color, otherwise the object is in shadow for that light source.

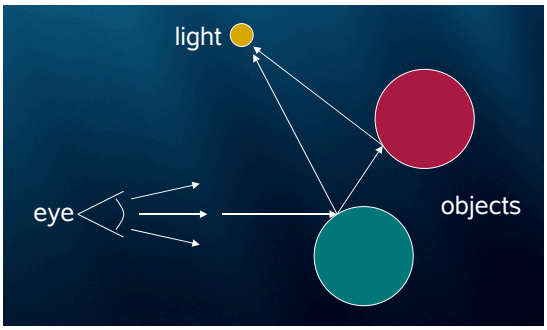


**Fig. 1.** The path of rays for a ray tracing algorithm.

Since ray tracing models the true physics and mathematics behind light propagation, it is able to calculate correct shadows and reflections, resulting in images that are stunningly more life-like to the human eye, namely "photo-realistic".

In addition, the method is output sensitive (you only pay for the things you see), meaning that the complexity of the scene does not affect the rendering times as much as in today's rasterization-based rendering systems.

This survey starts with a more detailed discussion on the advantages and disadvantages of ray tracing versus nowadays' rasterization techniques (Section 2) and leans on these arguments to justify professional application scenarios for which ray tracing is best suited (Section 3). After, we present a list of interactive ray tracing systems being made possible with the recent advances in computer graphics hardware (Section 4). Finally, we remember the readers that ray tracing is a far more general technique than it's computer graphics component and outline alternative applications in other scientific areas (Section 5), followed by some conclusions on the future of the computer graphics industry towards ray tracing rendering systems.

## 2 Ray Tracing versus Rasterization

Ray tracing excels at modeling a physical environment, but demands high processing power. Rasterization, on the other hand, is faster and excels at splicing many different algorithms together, but achieves results that may not match the reality (note the more enlightened objects with specular reflections in Figure 2).

Hardware-wise, rasterization (typically depth buffering, is primarily a visibility technique) relies on GPU performance to improve the polygon counts and quality of shading or bump-mapping techniques, whilst ray tracing (a visibility technique but also a catch-all to describe global illumination techniques) requires great CPU performance and dedicated ray-tracing hardware to reduce rendering delays and achieve more interactive framerates.

In the last years, as GPU and CPU processing power are constantly increasing, ray tracing implementations are beginning to reach acceptable interactive

**Fig. 2.** Difference in quality between a rasterized (left) and a ray traced (right) image.

framerates and dividing opinions around the currently most "sexy" topic that is likely to revolutionize graphics industry: wether ray tracing is potentially superior, if it shall supplant todays rasterization hardware, and when.

In a shared article[1], Slusallek defends his preference for ray tracing according to five key benefits that we debate as follows.

*Flexibility*

Ray tracing, in a broad sense, is nothing less than a perfect light simulation. In nature, the propagation of light corresponds to emitting a possibly infinite number of rays from light sources that are continuously bouncing in an open environment and can be perceived by the human eye. The only difference is that ray tracing algorithms follow the path of light in the reverse order, by shooting a finite number of rays from the viewer into the light sources, allowed to bounce an also finite number of times, in order to guarantee a deterministic solution.

Opposingly, raster environments are known for a direct forward mapping from polygons to screen pixels, incorporating layer upon layer of texture-based techniques for increased realism that, notwithstanding, are highly limited in what is efficient to compute. However, the cost of reliable visual results comes with ray tracing at a much lower price in implementation, without requiring any great skill or effort from the developer/artist.

An example of this flexibility is the handling of transparent objects in an interactive scenario. A rasterizer needs to process all the potentially visible polygons in a scene one by one, demanding a pre-processing of the geometry that will fit in the graphics card's memory. Therefore, if the geometry is changed by altering a transparent or occluding object, the whole scene would need to be recomputed, while with ray tracing naturally copes with it. Such a reason makes ray tracing a much more flexible rendering algorithm for interactive purposes.

*Generality*

Due to its physical accuracy and implementation flexibility, ray tracing algorithms are the premier choice for correctly simulating many real-life lighting situations, without compromising their simplicity and elegance.

Such advanced rendering effects are very hard to compute in a raster environment, or even too complex, too time consuming to be successfully implemented.

*Preciseness*

As we have seen before, rasterized graphics can only approximate reality, either for geometry and lighting.

In a typical 3D scene, objects are sculpted as polygons, that comprise the basic building block of a triangle. From triangles, we can build the most complex geometric forms such as cones, spheres, or even people.

In a raster pipeline, each triangle is processed one by one and painted on the screen, resulting in a complete 3D scene. During the process, very specific hacks can be evaluated for achieving special rendering effects, namely "shaders", by changing the color, light, displacement or texture of particular triangles. Such special effects are many times not intuitive nor trivial to implement and sum to the overall complexity of the algorithm.

On the other hand, ray tracing analyses a scene in terms of the rays of light that intersect each object, abstracting from the its geometric structure. Consequently, it can render curved surfaces directly, making it ideal for spline models, expressed by a continuous function (Figure 3). This accurate way of calculating reflections across surfaces provides an innate heuristic for evaluating the smoothness and curvature of the models for aesthetic purposes[10].

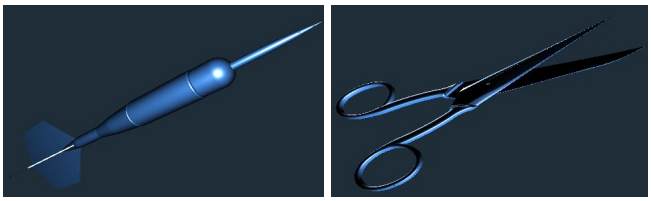

**Fig. 3.** Two images rendered directly from the surfaces' spline models.

*Plug & Play*

Rasterized graphics can only approximate reality and are based on particular hacks for particular rendering effects. These approximations are generally not intuitive nor trivial to implement and can prove difficult or even impossible to combine.

Ray tracing, conversely, directly employs high-level shaders that can be combined with arbitrary complexity. Even large scenes with many, complex shaders simply work as expected without further work or tweaking of parameters.

*Scalability*

The probably most important advantage of ray tracing is that it has great scalability properties both in terms of scene size and in terms of the processing units in use.

For the first argument, a ray tracer works by shooting rays from a point of view and seeing what it hits; what we can see is distinguishable from what what is hidden. Thus, ray tracing scales logarithmically in proportion to the scene size (we only need to render what is visible), whereas rasterization algorithms scale linearly. With rasterization, we feed feed triangles into a rasterizer and it processes them in order and the relationship of the triangles is not taken into account. Doing things over and over again in multiple passes feels like using brute force.

For the second argument, ray tracing is an highly parallelizable algorithm that gets near perfect scaling with the number of processing cores. Intel's technology strategist Jeffrey Howard writes in his weblog[15]: "*We have simulated with up to 16 cores, and we've already seen more than 15x scaling. With future platforms and additional optimizations, this may scale even better.*"

## 3 Application scenarios

Continuing the story form the previous section, ray tracing provides better image quality but is much computationally expensive and is, therefore, relegated to offline renderings whose scenes demand global illumination techniques. In this section we enumerate some of those applications for industrial contexts. Some of them had already been suggested by David Rogers back to 1987[11]:

### 3.1 Entertainment industry and animated movies

We can say that the first great example of an animated movie remounts back to 1995 with the release of Toy Story. The script was about a group of toys, mostly modeled with an opaque round geometry, with simple color textures and, thus, easy to render.

Since then, filmmakers have gradually became more ambitious, powered by the advances in computer technology, developing more complex stories, characters and scenes, but still relying on rasterization methods.

In a more recent past, although ray tracing was still expensive (compilation power and time cost money), animation houses such as Dreamworks and Pixar have started to integrate it in their movies to approximate global illumination. In films such as Shrek 2 and Finding Nemo, they have started to use pseudo ray tracing, in the sense that they generally rasterized the scenes, unless there was a polygon that needed complex reflections and justified to be ray traced. Currently, most digital animated films combine multiple techniques, including both ray tracing and rasterization, to create the widest possible variety of effects as efficiently as possible.

The first animated movie to extensively use ray tracing is Pixar/Disney's Cars, were cars have very curved surfaces, are very shiny and reflective and cast many shadows into a rather soft, natural environment (remark the reflections and sharp/soft shadows in Figure 11). For the production of the movie, Pixar have augmented their RenderMan scanline Interface Specification[17] with ray tracing[3].


**Fig. 4.** Images from the movie cars: rasterized image (upper left); ray traced image (upper right); final image with many cars and shadows (down).

### 3.2 Global illumination scenarios

Global illumination describes a group of algorithms used in computer graphics to add more realistic lighting to 3D scenes. It considers not only the light which comes directly from a light source (direct illumination), but also the cases when the light rays are reflected by a object toward other objects in the scene (indirect illumination).

Due to the very accurate and realistic light calculations, global illumination software such as Radiance[16] is typically used by architects and engineers to design innovative design spaces, predict their illumination and estimate their visual quality. It can also be used by researchers to study new lighting scenarios and daylighting technologies.

Notwithstanding, production quality rendering applications such as mentalray[19] and PhotoRealistic Renderman[17] employ global illumination techniques involving ray tracing to blend high quality and very realistic computer graphics effects

into real images. Motion pictures such as Spiderman 3, The Hulk and The Matrix Reloaded & Revolutions are known to have used mental ray.

**Fig. 5.** Examples of global illumination in building design (left) and motion picture's special effects (right)

However, we have not yet discussed what does the term global illumination comprise. When a ray finds an object, its reflection depends on the properties of the object's material: perfectly specular surfaces reflect or refract rays in a single direction; diffuse surfaces reflect the incoming light in all directions, which is equivalent to firing an infinite number of rays. The goal is to compute all possible light interactions in the scene to obtain the most photo-realistic image. All combinations of diffuse and specular reflections and transmissions must be accounted, included effects such as color bleeding and caustics.

Ray tracing, as a global illumination algorithm, must able to compute the effects of indirect lighting. It is very good at simulating specular reflections and transparency, since the rays that are traced through the scenes can be easily bounced at mirrors and refracted by transparent objects, but it requires an huge number of rays to be shot for the calculation of diffuse inter-reflexions.

For this reason, radiosity methods (and alike) that simulate the diffuse propagation of light were created, based on concepts from the heat transfer engineering field. The diverging idea is that the light is propagated starting from light sources and its contribution to the color of the objects shall converge to some value after a finite number of passes (one light bounce for each pass).

Since firing rays from light sources is a much more expensive task, two-pass solutions can be used to combine radiosity with ray tracing: the first pass involves a view independent radiosity solution; while the second pass is view-dependent ray tracing solution using the results of the first pass to avoid sending many

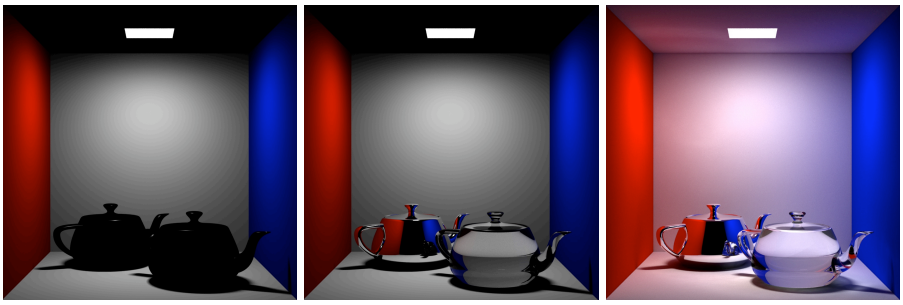

**Fig. 6.** Different lighting scenarios: direct lighting (left); direct and specular indirect lighting (middle); direct, specular and diffuse indirect lighting (right).

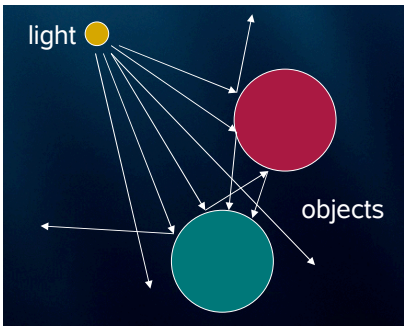


**Fig. 7.** The path of rays for a radiosity algorithm.

rays at diffuse reflections. An alternative two-pass global illumination algorithm is photon mapping, that explores similar concepts.

## 4 Real-time ray tracing

Though ray tracing has been extensively studied in computer graphics for more than three decades, the exponential growth of processing power and the embarrassingly parallel nature of ray tracing algorithms have renewed the interest in real-time rendering.

Among the first real-time ray tracers are the solutions developed at the Saarland University (inTrace, that originated a company with the same name[18]) and at the University of Utah (Manta[2]).

Years of research led to OpenRT[20], a real-time tray tracing standard library that can be easily integrated in existing applications. The OpenRT project provides an highly-optimized software core for ray tracing along with an OpenGL-like API that offers an alternative to the current rasterization-based approaches for interactive 3D graphics. It claims to be the fastest available ray tracing library (at least open source) and is designed to bring interactive performance to

both a single commodity PCs and distributed network solutions with multiple CPUs.

Interactive ray tracing is becoming one of the major rendering techniques. Its versatility, reliability, and its veridic simulation of lighting situations have found many applications amongst industrial projects, endowing designers, architects and engineers with fast and reliable reviews of their work.

### 4.1 Massive model visualization

The occlusion (hidden objects are not rendered) and scalability (it is sub-linear with respect to model size) properties of ray tracing make it appropriate for fully rendering very large CAD data sets that were otherwise impossible with rasterization techniques (would require simplification of the 3D model or selective rendering of parts of the model at different moment). Real world industrial examples involve the visualization of airplanes, submarines, power plants or oil tankers[4]. With the recent advances in multi-core ray tracing systems, reasonable framerates between 15 and 20 frames/sec can be achieved.

In [5], a solution is described for an aircraft manufacturing scenario, more specifically, a complete model of the Boeing 777 with a 35GB data set constituted by more than 3.000.000 individual pieces and 350 million polygons.

Although positional preciseness of the rendered objects is highly desired, the goal of massive model visualization is not the generation photo-realistic scenes. Instead, the visualization system shall take advantage of the flexibility and power of surface shading effects (for better distinction of the components) such as color mapping, shadowing and translucency control (Figure 8) or scene manipulation operations (for better inspecting the model) such as selective hiding of components and the definition of cutting planes (Figure 9). Also, the distance between two points in the scene is easily measured by shooting two additional rays to each extremity and trigonometrically calculate the resultant distance based on the distance of each extremity to the viewpoint.

### 4.2 Volume graphics

Volume graphics[8] are a recent subfield of computer graphics originated from the research efforts on volume visualization for scientific visualization needs in general, such as digital reconstruction of historical artifacts, medical imaging and stress simulations (crack propagation, fire simulation).

As such, volume visualization is a method for extracting meaningful information from volumetric data sets through the use of interactive graphics and imaging and encompasses the manipulation, representation and rendering of volume data. A typical 3D data set is a group of 2D slice images, possibly acquired by a CT or MRI scanner, that can be interpolated to generate highly dense polygon meshes.

The ability to render large volumetric data sets quickly is an essential requirement in most scientific applications. This is a challenging demand, since

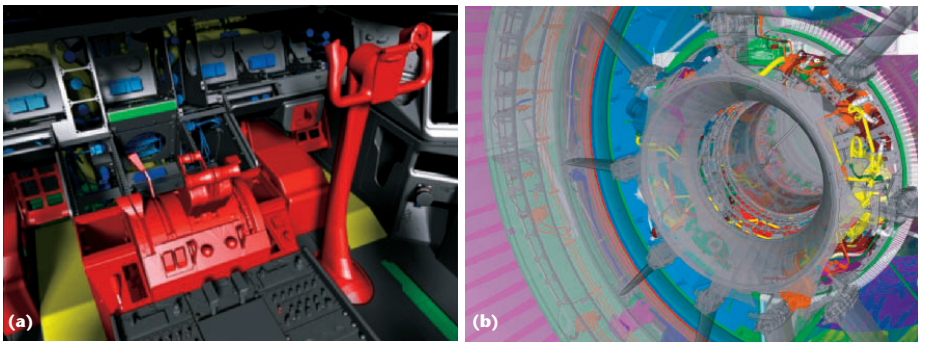


**Fig. 8.** Surface shading effects: soft shadows to amplify the relative placement (left); translucent object to better understand the spacial arrangement (right).

the scientific user is typically unwilling to sacrifice the accuracy of the final image in order to decrease projection times and rasterization algorithms are not best suited for this task for the approximations interpolation comprises and other reasons similar to massive model visualization.

An alternative to rendering millions of triangles is to support isosurface rendering. Rather than computing a triangulated mesh and rendering the resulting geometry, we can instead find the implicit surface defined by the volume. By interpolating between the data values (the isovalues) we can find a (nearly) continuous function describing the surface with this specific value. One way to achieve direct volume rendering is to shoot a ray from the camera point into the volume and finding the intersection of this implicit surface with ray. This technique eliminates the intermediate geometry producing a view dependent representation of the isosurface. Since the surface is recomputed each frame as part of the intersection phase, manipulation is made easier: we can interactively change isosurface values.

In contrast to hardware projection algorithms, a volumetric ray tracing algorithm[13,14,12] is generally slower, but much more accurate and flexible: it has greater scalability by factors of scene and object complexity and is view independent. Additionally, the use of volumetric ray tracing algorithms naturally eases common visualization techniques such as cut and segmentation of volumes, translucency control and relative distance measurements already provided for massive model visualization.

### 4.3 Virtual prototyping

A more balanced scenario than the visualization of massive 3D data models is to directly render not so critical models such as automobiles, with tens of millions of polygons (Figure 11).

In this setup, however, the main goal is to review a design and or look at different variants of it as soon as possible. When referring to the automotive

**Fig. 9.** Visualization of a full model in memory: with hidden components (top); by applying a cutting plane (bottom).

engineers of companies such as Volkswagen and Audi, Slusallek[22] states that: "*They need to reliably visualize the final look of the cars in order to make far-reaching decisions as early as possible in the design process. They don't need "nice images", they need correct images they can trust. With rasterization this has not been possible even after they tried hard for many years.*"

### 4.4 Computer games

Take a look, for example, at what Hollywood uses when they make special effects for films and in-game movies. They already use ray-tracing engines, but it takes hours to fully compute each individual frame. These offline computations are very exact and time consuming, but breakthroughs in the way Intel has designed our ray-tracing engine may allow some of these special effects to happen in real time, while playing a video game.

We can say that ray tracing has already convinced the computer games industry of its' superior quality, since they already use ray tracing engines to generate the in-game movies for their video games, though in offline and vert time consuming computations. At the higher end, we want these special effects to happen in real-time, while playing the video game, but PC nowadays clusters' solutions are definitely not acceptable for gaming.

For filling the gap with the development of computer games based on ray tracing, the RTGames working group at the Saarland University has been created

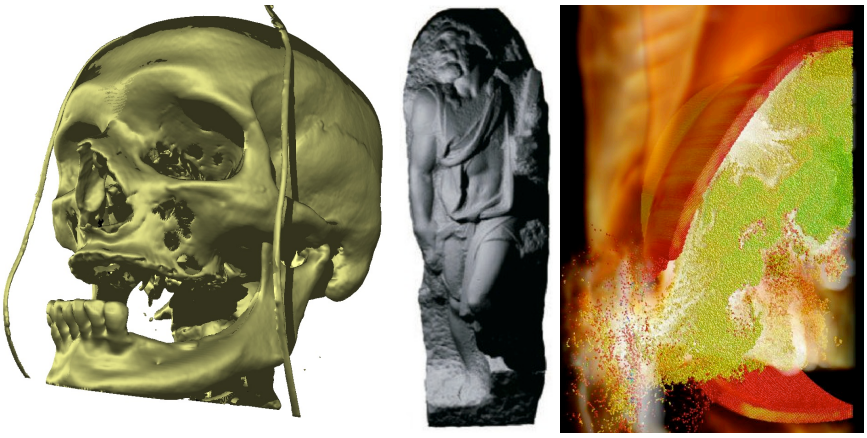

**Fig. 10.** Volumetric visualization scenarios: isosurface rendering of an human skull (left); a scanned model of Michelangelo's St. Matthew statue (middle); rupture simulation of a steel container filled with a plastic bonded explosive, modeled as millions of spheres, and heated by a fire (right).

to study two open questions: how to adapt existing rasterization-based games to ray tracing by reusing part of their original content (models, sounds, etc); and how computer games would feel like if they were directly developed to use ray tracing.

The first example of a ray traced game was a port of the Quake 3 engine coded by Daniel Pohl (Figure 12). He replaced the old rasterization engine with a new ray tracing engine (the ray tracing algorithm is handled by the OpenRT ray tracing library). The game engine comprised the usage of a PC cluster, a requirement that is certainly not acceptable for gaming, but alternative configurations could be explored by using one of the ray tracing dedicated hardware chips developed by the RTGames group. So far they have published two designs: the initial SaarCOR and the most recent RPU[24].

Intel has also made an effort towards ray tracing by employing Daniel Pohl in a "sequel" port of the Quake 4 engine (Figure 13). The ray traced version of the game runs on an Intel's eight-core ray tracing dedicated machine, and provides an important breakthrough towards demonstrating the potential of interactive real-time ray tracing for computer games.

Among the results, Intel publicizes the great scalability of the algorithm in proportion to the number of cores, and defend that the nearly linear gain factor is sufficient to draw ray tracing into handled consoles such as the successors of the PSP and Nintendo DS (since the rendering times are proportional to the number of pixels) whenever it becomes possible for commodity PCs.

**Fig. 11.** Virtual prototypes of automobiles: car interior components (upper left); model detail and image quality (upper right); full car overview (down)).

## 5 Applications outside computer graphics

The ray tracing general procedure works by assuming that waves can be represented as a large number of straight rays, very narrow beams that are sufficiently small to approximate a line. A ray tracer is responsible for the iterative process of advancing the ray over a predefined distance, testing if it collides with solid objects, and calculate the ray's new direction. The process finishes when a complete path is generated.

### 5.1 Optical design

The origins of ray tracing date back to the initial designs of lenses and optical systems, such as telescopes, microscopes, cameras and binoculars. Initially

**Fig. 12.** Screenshot of a Quake 3 ray traced implementation.

calculated by hand by hundreds of collaborative workers, geometric ray tracing can now be solved by computer applications[21] and is used to describe the propagation of light rays through a lens, allowing the image-forming properties of the system to be modeled.

In a related field, ray tracing can also be used for the simulation of real-world phenomena for vision research. As an example, reverse ray tracing (rays are shot from a laser diode as a light source) is used to study the optical system of the human eye[6].

### 5.2 Seismic tomography

Seismic tomography is a major research topic on geophysics and concerns the reconstruction of the Earth's interior.

The underlying principle is that seismic wave velocity varies within and beneath Earth's crust, as the waves pass through different layers, causing them to bend and or reflect.

**Fig. 13.** Screenshot of a Quake 4 ray traced implementation.

By modeling a seismic wave as a set of rays, ray tracing can be used to trace earthquake waves back to their source and to deduce the physical rock properties of the intervening material (Figure 15). In [7], a software application for a fast seismic ray tracer in an Earth mesh is implemented.

### 5.3 Others

Other adaptations of ray tracing algorithms can be found in the co-related areas of ocean acoustics (calculate the propagation of sound waves for different sea depths in order to discover the best underwater channels for communication), radio signals (study the propagation of electromagnetic waves to improve radio communications), mechanical engineering, impact and penetration studies and collision detection.

## 6 Conclusions

Rasterization has come a long way, and with today's shaders and scalable graphics processing units brought to commodity PCs, programmers are unlocking many new and convincing special effects. Yet, it relies on the creativity and cleverness of engineers in the face of exponentially increasing complexity.

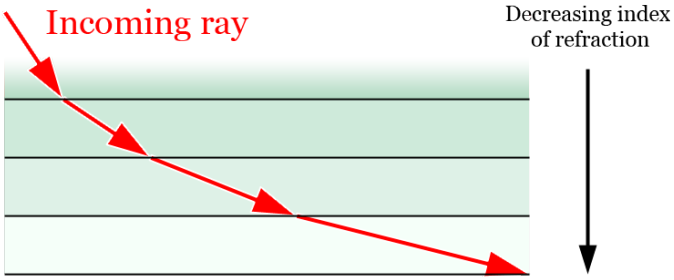


**Fig. 14.** General ray tracing procedure for a beam of light.

On the other side, ray tracing is modeled in the correct physical way and has benefited from improvements in computer hardware. The strong effort made by Intel to advocate the superiority of ray tracing is certainly generating apprehension among graphics cards manufacturers that defend their investments in rasterization as the ultimate method for increasing graphics performance.

Although existing real-time ray tracing solutions are still too restrictive to be taken as a replacement of rasterization, according to the Moore's law, hardware solutions will continue to evolve, both CPU and GPU based, and ray tracing will inevitably arrive, sooner or later, to commodity PCs.

What remains is the question about hardware: will ray tracing run on the CPU (as it currently is), on the GPU or, most likely, on some new special-purpose hybrid platform? GPUs rely on brute-force bandwidth, but ray tracing can run efficiently on a CPU because of the large caches. While the GPU and CPU guys equally strive for the best hardware support for ray tracing, it will be interesting to see when the two philosophies converge, if they will, and who wins this multibillionaire race for the future of 3D graphics. However, one thing is certain: rasterization may not, but ray tracing, or any derived method, will play a role in it.

## Acknowledgments

Thanks to all the bloggers and internet enthusiasts whose webpages are not reported in this document. Most pictures were taken from the referenced articles. Also, my apologies to the authors for not requiring their permission.

## References

1. Kurt Akeley, David Kirk, Larry Seiler, Philipp Slusallek, and Brad Grantham. When will ray-tracing replace rasterization? In *SIGGRAPH '02: ACM SIGGRAPH 2002 conference abstracts and applications*, pages 86–87, New York, NY, USA, 2002. ACM.

[htb]

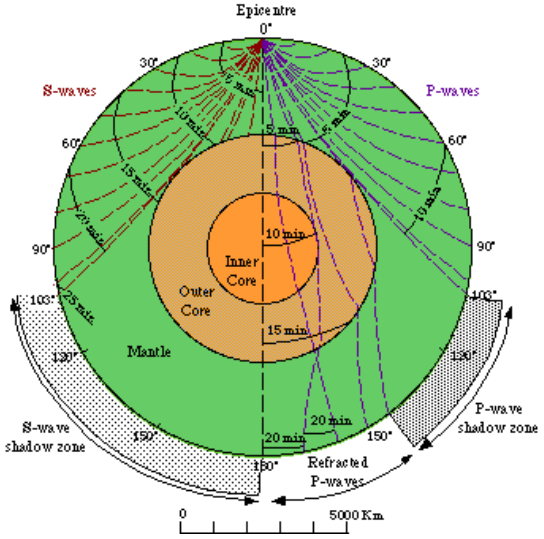


**Fig. 15.** Example of the propagation of seismic waves through the interior of the Earth for an earthquake.


2. J. Bigler, A. Stephens, and SG Parker. Design for Parallel Interactive Ray Tracing Systems. *Interactive Ray Tracing 2006, IEEE Symposium on*, pages 187–196, 2006.
3. P. Christensen and P.A. Studios. Ray Tracing for the Movie" Car. *Interactive Ray Tracing 2006, IEEE Symposium on*, 2006.
4. A. Dietrich, E. Gobbetti, and S.E. Yoon. Massive-Model Rendering Techniques: A Tutorial. *IEEE Computer Graphics and Applications*, 27(6):20–34, 2007.
5. A. Dietrich, A. Stephens, and I. Wald. Exploring a Boeing 777: Ray Tracing Large-Scale CAD Data. *IEEE Computer Graphics and Applications*, 27(6):36–46, 2007.
6. A.V. Goncharov, M. Nowakowski, M.T. Sheehan, and C. Dainty. Reconstruction of the optical system of the human eye with reverse ray-tracing. *Optics Express*, 16(3):1692–1703, 2008.
7. M. GRUNBERG, S. GENAUD, and C. MONGENET. Seismic Ray-Tracing and Earth Mesh Modeling on Various Parallel Architectures. *The Journal of Supercomputing*, 29:27–44, 2004.
8. A. Kaufman, D. Cohen, and R. Yagel. Volume graphics. *Computer*, 26(7):51–64, 1993.
9. M. MURTY. General ray tracing procedure. *1962.*, 1962.
10. S. Parker, M. Parker, Y. Livnat, P.P. Sloan, C. Hansen, and P. Shirley. Interactive ray tracing for volume visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 5(3):238–250, 1999.
11. D.F. Rogers and R.A. Earnshaw. Techniques for Computer Graphics. *Techniques for Computer Graphics*, 1987.
12. L.M. Sobierajski and R.S. Avila. A Hardware Acceleration Method for Volumetric Ray Tracing. *Proceedings of the 6th conference on Visualization'95*, 1995.
13. L.M. Sobierajski and A.E. Kaufman. Volumetric ray tracing. *Proceedings of the 1994 symposium on Volume visualization*, pages 11–18, 1994.

14. M. Sramek and A. Kaufman. Fast Ray-Tracing of Rectilinear Volume Data Using Distance Transforms. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*, pages 236–252, 2000.
15. http://blogs.intel.com/research/2007/10/real_time_raytracing_the_end_o.php. Research@intel: Jeffrey howards' weblog.
16. http://radsite.lbl.gov/radiance. Radiance synthetic imaging system.
17. https://renderman.pixar.com. Pixar's renderman ®.
18. http://www.inTrace.com. intrace: Realtime ray-tracing technologies.
19. http://www.mentalimages.com. mental images.
20. http://www.openRT.de. The openrt real-time ray-tracing project.
21. http://www.optics-lab.com. Optics lab optical ray tracing software.
22. http://www.taronga.com/~peter/io/raytracing-vs-rasterization.html. Raytracing vs Rasterization.
23. T. Whitted and N.J. Holmdel. An Improved Illumination Model for Shaded Display. *Communications*, 1980.
24. S. Woop, J. Schmittler, and P. Slusallek. RPU: a programmable ray processing unit for realtime ray tracing. *International Conference on Computer Graphics and Interactive Techniques*, pages 434–444, 2005.