GPU RAY TRACING FOR HIGH-FIDELITY ACOUSTIC SIMULATION

DJ Pate Georgia Tech Research Institute, Atlanta, Georgia, USA

1 INTRODUCTION

Ray tracing in acoustics typically refers to long distance low frequency propagation; the focus of the computation is wavefront refraction and spreading, and so the main computation is the interaction with the sound speed profile. In computer graphics ray tracing, however, the goal is global illumination and the main computation is the interaction of light rays and the surfaces in a scene. The camera and light source are analogous to a transmitter and receiver.

In computer graphics, the goal of the global illumination problem is to compute the radiance on every surface in a scene due to every source of light, and, further, to compute the intensity of light reaching a camera lens [1,2]. This is directly analogous to acoustic propagation and scattering, in which the desired quantity is acoustic pressure at a receiver due to the scattering in a scene from a transmission source. Similarly, this is also analogous to electromagnetic scattering in which the electric current density vector is needed at every surface and the electric field vector is needed at the receiver.

There is a significant potential to apply the methods of computer graphics to acoustic propagation and scattering simulation, especially for synthetic aperture sonar (SAS).

2 THE GLOBAL ILLUMINATION PROBLEM

In the field of computer graphics, the global illumination problem is expressed via the rendering equation, which is a recursive integral equation where the spectral radiance at a point on a surface is the sum of all contributions emitted from all other surfaces and sources [3]. It is generally expected that light may take an intricate path bouncing off of multiple surfaces before reaching the camera. Similarly, in the sonar scattering problem, a wavefront may interact with multiple surfaces (e.g. multipath, reverberation, etc.) on its way from a transmitter to a receiver, whether it is large-scale forward scattering from the sea surface, or small-scale reflections from a corner reflector on a man-made object.

Solving the global illumination problem by integrating the emitted and scattered light on every surface recursively ad infinitum is nearly intractable with conventional numerical integration. Alternatively, Monte Carlo sampling is an effective way of solving difficult numerical integration problems. Moreover, it simplifies the descriptions of the physics: a ray interacts with the surfaces in the scene via the bidirectional reflectance distribution function (BRDF), which is what would be called a directional scattering strength in acoustics or a directional radar cross section in electromagnetics. The BRDF describes the intensity in a given outward direction given a particular incident direction. The BRDF can be used as a probability density function to randomly generate rays in the most probable directions. This technique is called importance sampling [1,2].

2.1 Evolution of Ray Tracing

As told by Christensen et al. [1,4], ray tracing for computer graphics begins in 1968 when Appel introduced ray casting [5]. Though, this is less about lighting and more about projecting a three dimensional

object or scene onto a two dimensional plane. As depicted in Figure1, which is adapted from Christensen[1], given a scene, an eye point, and an image plane, rays are cast out from the eye point, through the pixels in the image plane, and to the scene. The colors of the pixels are exactly determined from the color at the point touched by the ray. Multiple rays per pixel can be used for antialiasing. Note that *aliasing* is an effect in which an edge or line appears jagged due to pixelation, especially when a diagonal line is crossing pixel rows or columns. Ray casting does not account for shadows, reflections, or other lighting effects.

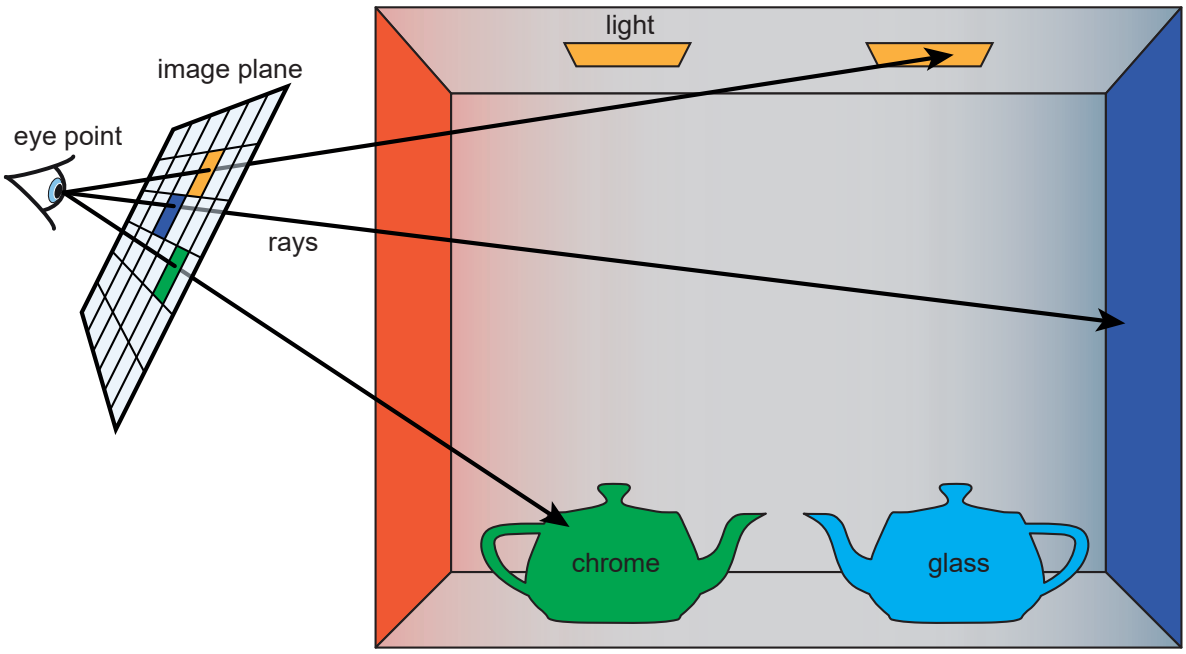

Figure 1: An example of ray casting. Diagram is copied and modified from Christensen and Jarosz[1] figure 3.1.

Next, recursive ray tracing was introduced by Whitted in 1980[6]. As described by Christensen[1], "at each eye ray intersection point, a shadow ray is traced to each light source, and recursive reflection and refraction rays are spawned." Therefore, each starting ray produces a full tree of rays, and the color of the pixel is determined by the total effect of this tree. An example of this process is presented in Figure 2, in which a ray is initially reflected by the chrome teapot and moves on to impact the blue wall. At each intersection point, shadow connections to *every* light source are established. If a shadow ray is unobstructed, then the light contributes to the color. One drawback of this method is that as a tree grows, the rays it is spawning are less important to the color of the image.

Subsequently, in 1984 Cook et al. developed distributed ray tracing[7]. They nicely state "ray tracing is one of the most elegant techniques in computer graphics. Many phenomena that are difficult or impossible with other techniques are simple with ray tracing, including shadows, reflections, and refracted light." As described by Christensen[1], random sampling is used for camera shutter time, lens position, and area light sources, and this produces the effects of motion blur, depth of field, and soft shadows, respectively.

Then, in 1986 Kajiya[3] introduced *path tracing* and the *rendering equation*, which formulated global illuminate as an integral equation than can be effectively solved with Monte Carlo integration. Christensen[1] describes the path tracing process: "When a ray intersects a surface, the direct illumination from the light sources is calculated for the intersection point…in addition, a new ray is spawned to calculated indirect illumination. The direction of the new ray is stochastically chosen based on the light scattering properties of the surface material: specular or matte, reflective or refractive." This is

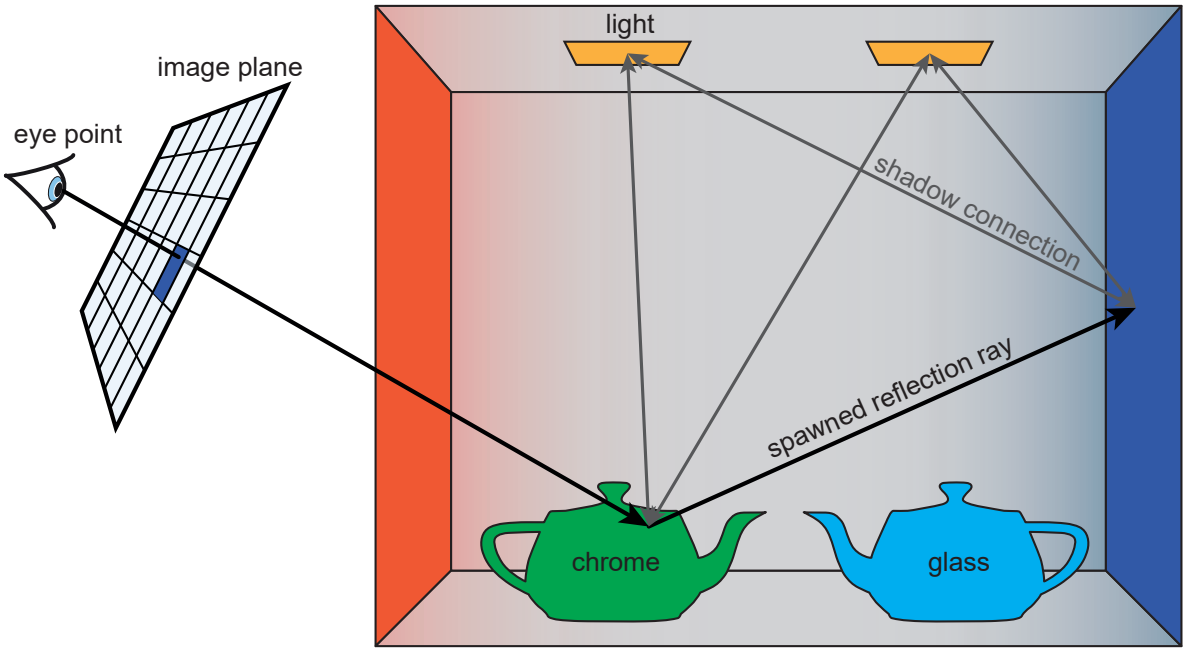

Figure 2: An example of recursive ray tracing. Diagram is copied and modified from Christensen and Jarosz[1] figure 3.1.

demonstrated in Figure 3. When a ray hits the chrome or glass teapot, the next ray proceeds in the reflection or refraction direction, respectively. When a ray hits a wall that is a matte surface with diffuse scattering, the next ray proceeds in a random direction, and also a shadow connection to a light is made. Note that at each intersection point, a shadow ray is connected to only one of the lights, and at a random point within its boundary.

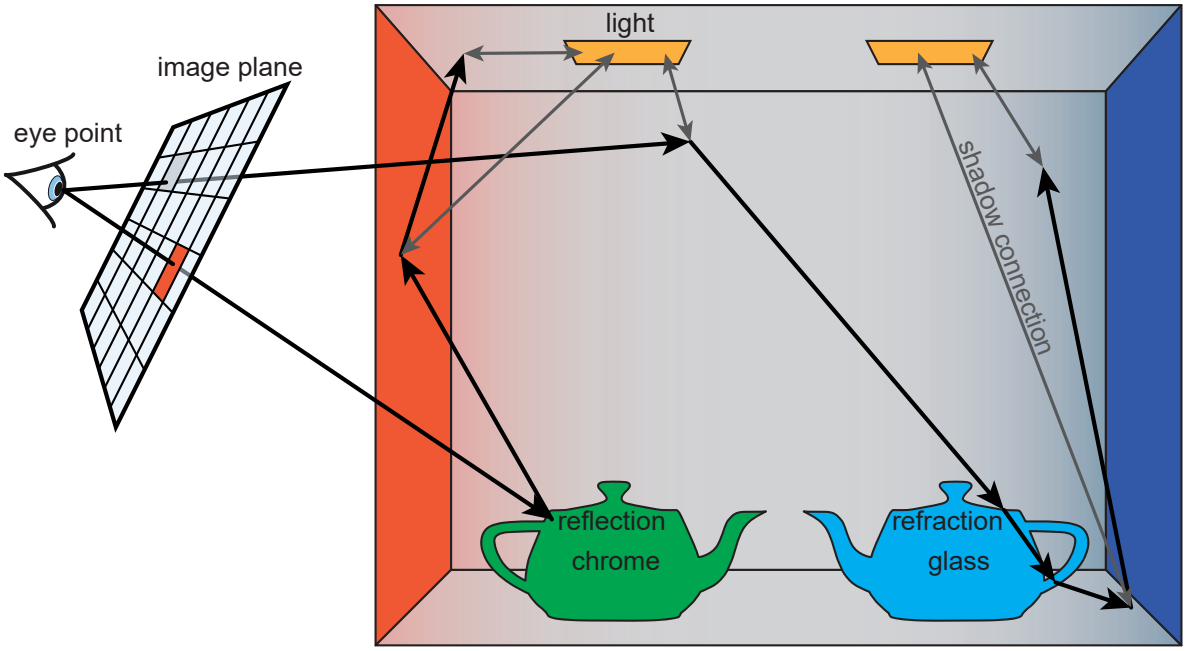

Figure 3: An example of path tracing. Diagram is copied and modified from Christensen and Jarosz[1] figure 3.1.

Lastly, *bidirectional path tracing* traces a pair of rays: one from the camera and one from a light, connecting them with shadow rays along the way as they move through the scene a la path tracing[2]. This can improve the convergence rate for scenes with significant indirect lighting.

Path tracing is both powerful and versatile—effort need only be placed in writing the rules for the behavior of light, and the Monte Carlo integration does all the work. The drawback is that, by relying on random sampling, a sufficiently large number of rays are needed to converge the simulation and eliminate the noise in the resulting image. In computer graphics, denoising filters are used to mitigate this image noise.

2.2 Additional Physical Effects

The versatility of path tracing allows for many additional physical effects to be modeled. Subsurface scattering is important for capturing the appearance of skin and other translucent materials[4]. The most direct approach is to apply path tracing in a random walk within the material volume below the surface, as demonstrated on the left in Figure 4. Though, the added ray steps make these an expensive option. Instead, a shortcut can be taken by statistically sampling the average displacement distance, as depicted on the right in Figure 4. This is referred to as a diffusion model, and it requires assuming a semi-infinite solid[8].

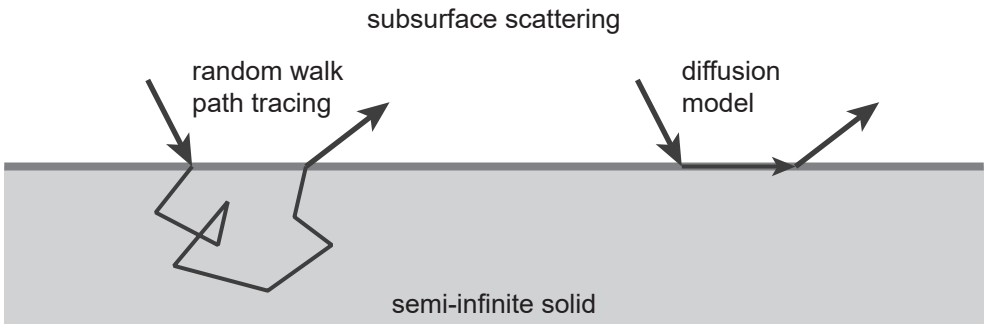

Figure 4: Demonstration of random walk path tracing (left) and a diffuse model (right) for subsurface scattering. Adapted from Burley[9].

In computer graphics, hair and fur are especially challenging. Christensen and Jarosz note “one trick is to not model individual hairs, but render planes with textures of hair (and transparent space between hairs). Another trick is to widen the hairs, but at the same time make them more transparent. The quantity and thinness of hairs make them expensive for ray intersection computations. Instead of axis-aligned bounding boxes, it can be better to use locally oriented bounding boxes.

Volume scattering is important for effects such as smoke and clouds. In homogeneous volumes, volume scattering is relatively easy: “repeatedly choose a random scattering distance (with exponentially decreasing probability), and at that distance choose between absorption or scattering…if scattering is chosen…generate a new scattering direction” states Christensen et al.[4]. This approach covers both single and multiple scattering. For heterogeneous volumes, the ray marching algorithm[10] steps through the space at a fixed interval to sample the local properties. The Beer–Lambert law is use to model absorption[11].

2.3 Analogy to Acoustics and Synthetic Aperture Sonar

The analogy between light transport in computer graphics and acoustic propagation and scattering for synthetic aperture sonar is rather straight forward:

- The camera and lights match to the sensor receiver and transmitter elements. Rectangular transducer elements can be modeled analytically with a beampattern, of more generally (but more computationally expensive) with actual sampling.
- Diffuse and specular reflections are directly applicable to surfaces such as the sea surface, seafloor, targets, and other objects.
- Subsurface scattering for skin and translucent materials can match to seafloor scattering.
- Surface textures are directly applicable to such effects as biofouling on man-made objects or capillary waves on the sea surface.
- The solutions for modeling hair and fur could be used for modeling a sandy bottom type. Just as it would be too much to express every strand of hair, it would be *way* too much to express every grain of sand.
- Volume scattering for smoke and clouds is directly relevant for acoustic volume scattering from particulates in the water column.

Then, to apply path tracing to simulate a SAS ping, the algorithm would be: a ray would be cast from a receiver element. It would hit a surface, and a shadow ray would be cast to a randomly chosen transmitter element. The original ray could then scatter in a new random direction, and this process would continue until the exit criteria are met.

Of course, there are some significant differences between light transport and SAS. One difference is that light transport deals with bulk energy, rather than coherent phase. To simulate acoustic propagation and scattering to a high fidelity, the phase of the signal will need to be properly captured.

Another important difference is the time in which the propagation proceeds is much slower for SAS, and so time of arrival is important to track. Likewise, Doppler shift is also relevant.

Lastly, the wave nature of sound propagation is more applicable at SAS frequencies than the wave nature of light. Depending on the scene, diffraction may be important.

3 AURALIZATION

Auralization is the corresponding sound rendering for computer-generated scenes, such as for movies and games. Naturally, similar techniques are used for sound propagation and scattering as for light. Cao et al.[12] applied bidirectional path tracing (see 2.1) to sound propagation in the name of the bidirectional sound transport algorithm. This uses *geometric acoustics* paired with the same path tracing computations used for illumination. The algorithm produces the impulse response of the scene from the acoustic sources, that is then convolved with the source to produce the final waveform.

Schissler and Manocha[13] simulate the sound propagation for scene with many source. To accommodate large quantities of acoustic sources, the algorithm clusters nearby sources together. Doppler shifts are produced by sorting the arrivals based on relative speed and then applying fractional delay interpolation. Additionally, Schissler et al. extend geometric optics to include diffraction around wall corners[14].

Similar works include Taylor et al.[15] and Mo et al.[16,17,18].

4 BOUNDING VOLUME HIERARCHY FOR FAST RAY INTERSECTION COMPUTATION

The fundamental computation of ray tracing is the ray–triangle intersection test. To determine at which point a ray will first hit a surface, it needs to be checked against every triangle in the scene. Scenes often comprise millions or many millions of triangles, and even more rays. It would be exceptionally expensive to check every ray against every triangle, even for a GPU. Instead, efficient data structures can be used. A common data structure that is used in ray tracing is the bounding volume hierarchy (BVH)[19,20].

The process for constructing a BVH is

1. start with a list of primitives (triangles)
2. fit an axis-aligned bounding box tightly around these objects
3. determine the longest dimension of the box and split it at the midpoint. If there are only a small number of objects, stop instead.
4. in memory, sort the objects across the midpoint dividing line to partition the set of triangles into two sets.
5. for each of these two sets of objects, repeat back to 1.

This process is demonstrated in Figure 5 for an example object.

The main advantage of a BVH is that if it can be determined that a ray does not pass through a particular bounding box, then it necessarily does not pass through any of the triangles within it, and so they do not need to be checked. By culling the tree this way, sub-linear computation scaling is achieved. The process for traversing a BVH tree to find the closest intersection point with a ray is

1. consider a ray with a starting point and direction
2. check if the ray passes through the root node (the outer-most bounding box). If it does not, exit. If it has already hit an object that is closer than this box, exit.
3. if it passes through this box, and the node is a leaf node, check all of the triangles inside to see if the ray hits them. If the node is not a leaf node, check the two child boxes

Checking if the ray passes through a box is very fast thanks to the axes of the rectangular prism being in the same coordinate system as the position and direction vectors of the ray. Additionally, checking if the ray hits a triangle is very fast thanks to the Möller–Trumbore algorithm[21]. Then, bringing together a GPU, a BVH, fast bounding box intersection test, and a fast ray–triangle intersection test, ray tracing can be computed quickly and efficiently.

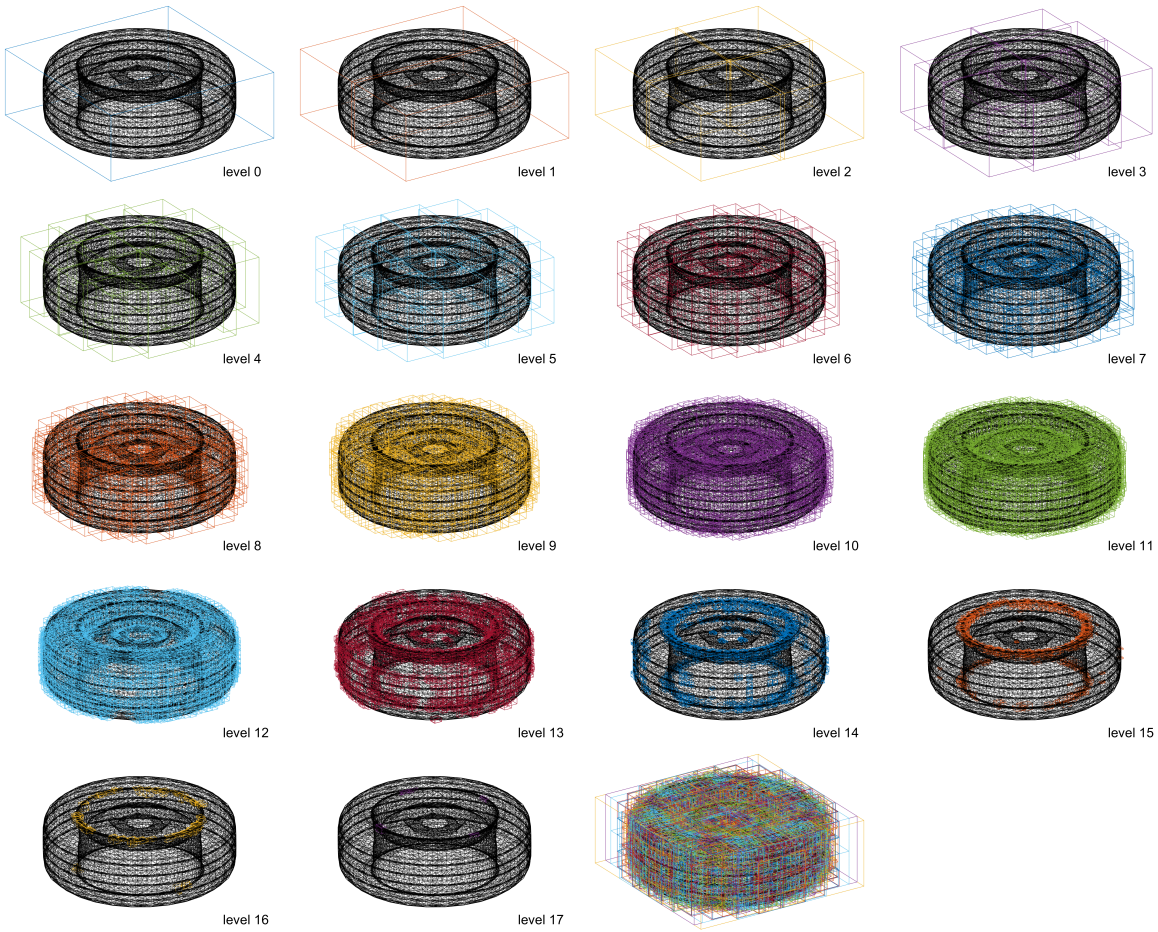

Figure 5: Bounding volume hierarchy for a tire. Tire model attribution: https://skfb.ly/oxWQn

5 REFERENCES

[1] Per H Christensen, Wojciech Jarosz, et al. The path to path-traced movies. *Foundations and Trends® in Computer Graphics and Vision*, 10(2):103–175, 2016.

[2] Michal Vlnas. Bidirectional path tracing. In *Proceedings of the 22nd Central European Seminar on Computer Graphics*, volume 1, pages 9–11, 2022.

[3] James T Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, 1986.

[4] Per Christensen, Julian Fong, Jonathan Shade, Wayne Wooten, Brenden Schubert, Andrew Kensler, Stephen Friedman, Charlie Kilpatrick, Cliff Ramshaw, Marc Bannister, et al. Renderman: An advanced path-tracing architecture for movie rendering. *ACM Transactions on Graphics (TOG)*, 37(3):1–21, 2018.

[5] Arthur Appel. Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, pages 37–45, 1968.

[6] Turner Whitted. An improved illumination model for shaded display. In *ACM Siggraph 2005 Courses*, pages 4–es. 2005.

[7] Robert L Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 137–145, 1984.

[8] Per H. Christensen and Brent Burley. Approximate reflectance profiles for efficient subsurface scattering. techreport 15-04, Pixar Animation Studios, July 2015.

[9] Brent Burley. Extending the disney brdf to a bsdf with integrated subsurface scattering. In

SIGGRAPH Course: Physically Based Shading in Theory and Practice, volume 19, page 9. Association for Computing Machinery, 2015.

[10] Ken Perlin and Eric M. Hoffert. Hypertexture. *Computer Graphics*, 23(3):253–262, July 1989.

[11] Konstantinos Vardis. *Efficient Illumination Algorithms for Global Illumination In Interactive and Real-Time Rendering*. phdthesis, Athens University of Economics and Business, December 2016.

[12] Chunxiao Cao, Zhong Ren, Carl Schissler, Dinesh Manocha, and Kun Zhou. Interactive sound propagation with bidirectional path tracing. *ACM Transactions on Graphics (TOG)*, 35(6):1–11, 2016.

[13] Carl Schissler and Dinesh Manocha. Interactive sound propagation and rendering for large multi-source scenes. *ACM Transactions on Graphics (TOG)*, 36(4):1, 2016.

[14] Carl Schissler, Gregor Mückl, and Paul Calamia. Fast diffraction pathfinding for dynamic sound propagation. *ACM Transactions on Graphics (TOG)*, 40(4):1–13, 2021.

[15] Micah Taylor, Anish Chandak, Qi Mo, Christian Lauterbach, Carl Schissler, and Dinesh Manocha. Guided multiview ray tracing for fast auralization. *IEEE Transactions on Visualization and Computer Graphics*, 18(11):1797–1810, 2012.

[16] Qi Mo, Hengchin Yeh, and Dinesh Manocha. Tracing analytic ray curves for light and sound propagation in non-linear media. *IEEE transactions on visualization and computer graphics*, 22(11):2493–2506, 2016.

[17] Qi Mo, Hengchin Yeh, Ming Lin, and Dinesh Manocha. Analytic ray curve tracing for outdoor sound propagation. *Applied Acoustics*, 104:142–151, 2016.

[18] Qi Mo, Hengchin Yeh, Ming Lin, and Dinesh Manocha. Outdoor sound propagation with analytic ray curve tracer and gaussian beam. *The Journal of the Acoustical Society of America*, 141(3):2289–2299, 2017.

[19] Ingo Wald, Solomon Boulos, and Peter Shirley. Ray tracing deformable scenes using dynamic bounding volume hierarchies. *ACM Transactions on Graphics (TOG)*, 26(1):6, 2007.

[20] Alfonso Breglia, Amedeo Capozzoli, Claudio Curcio, and Angelo Liseno. Comparison of acceleration data structures for electromagnetic ray-tracing purposes on gpus [em programmer's notebook]. *IEEE Antennas and Propagation Magazine*, 57(5):159–176, 2015.

[21] Tomas Möller and Ben Trumbore. Fast, minimum storage ray/triangle intersection. In *ACM SIGGRAPH 2005 Courses*, pages 7–es, 2005.