A Beam Tracing Approach to Acoustic Modeling for Interactive Virtual Environments

Thomas Funkhouser*, Ingrid Carlbom, Gary Elko,
Gopal Pingali, Mohan Sondhi, and Jim West
Bell Laboratories

Abstract

Virtual environment research has focused on interactive image generation and has largely ignored acoustic modeling for spatialization of sound. Yet, realistic auditory cues can complement and enhance visual cues to aid navigation, comprehension, and sense of presence in virtual environments. A primary challenge in acoustic modeling is computation of reverberation paths from sound sources fast enough for real-time auralization. We have developed a system that uses precomputed spatial subdivision and "beam tree" data structures to enable real-time acoustic modeling and auralization in interactive virtual environments. The spatial subdivision is a partition of 3D space into convex polyhedral regions (cells) represented as a cell adjacency graph. A beam tracing algorithm recursively traces pyramidal beams through the spatial subdivision to construct a beam tree data structure representing the regions of space reachable by each potential sequence of transmission and specular reflection events at cell boundaries. From these precomputed data structures, we can generate high-order specular reflection and transmission paths at interactive rates to spatialize fixed sound sources in real-time as the user moves through a virtual environment. Unlike previous acoustic modeling work, our beam tracing method: 1) supports evaluation of reverberation paths at interactive rates, 2) scales to compute high-order reflections and large environments, and 3) extends naturally to compute paths of diffraction and diffuse reflection efficiently. We are using this system to develop interactive applications in which a user experiences a virtual environment immersively via simultaneous auralization and visualization.

Key Words: Beam tracing, acoustic modeling, auralization, spatialized sound, virtual environment systems, virtual reality.

1 Introduction

Interactive virtual environment systems combine graphics, acoustics, and haptics to simulate the experience of immersive exploration of a three-dimensional virtual world by rendering the environment as perceived from the viewpoint of an observer moving under real-time control by the user. Most prior research in virtual environment systems has focused on visualization (i.e., methods for rendering more realistic images or for increasing image refresh rates), while relatively little attention has been paid to auralization (i.e., rendering spatialized sound based on acoustical modeling). Yet, it is clear that we must pay more attention to producing realistic sound in order to create a complete immersive experience in which aural cues combine with visual cues to support more natural interaction within a virtual environment. First, qualitative changes in sound reverberation, such as more absorption in a room with more lush carpets, can enhance and reinforce visual comprehension of the environment. Second, spatialized sound can be useful for providing audio cues to aid navigation, communication, and sense of presence [14]. For example, the sounds of objects requiring user attention can be spatialized according to their positions in order to aid object location and binaural selectivity of desired signals (e.g., "cocktail party" effect). The goal of this work is to augment a previous interactive image generation system to support real-time auralization of sound based on realistic acoustic modeling in large virtual environments. We hope to use this system to support virtual environment applications such as distributed training, simulation, education, home shopping, virtual meetings, and multiplayer games.

A primary challenge in acoustic modeling is computation of reverberation paths from a sound source to a listener (receiver) [30]. As sound may travel from source to receiver via a multitude of reflection, transmission, and diffraction paths, accurate simulation is extremely compute intensive. For instance, consider the simple example shown in Figure 1. In order to present an accurate model of a sound source (labeled 'S') at a receiver location (labeled 'R'), we must account for an infinite number of possible reverberation paths (some of which are shown). If we are able to model the reverberation paths from a sound source to a receiver, we can render a spatialized representation of the sound according to their delays, attenuations, and source and receiver directivities.

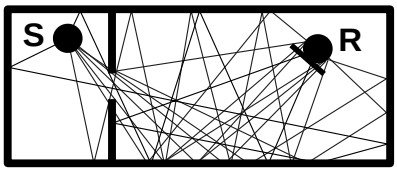


Figure 1: Example reverberation paths.

Since sound and light are both wave phenomena, acoustic modeling is similar to global illumination in computer graphics. However, there are several significant differences. First, the wavelengths of audible sound fall between 0.02 and 17 meters (20kHz to 20Hz), more than five orders of magnitude longer than visible light. As a result, though reflection of sound waves off large walls tends to be primarily specular, significant diffraction does occur around edges of objects like walls and tables. Small objects (like coffee mugs) have significant effect on the sound field only at frequencies beyond 4 kHz, and can usually be excluded from models of acoustic environments, especially in the presence of other significant sources of reflection and diffraction. Second, sound travels through air 10^6 times slower than light, causing significantly different arrival times for sound propagating along different paths, and the resulting acoustic signal is perceived as a combination of direct and reflected sound

* Princeton University

(reverberation). The time distribution of the reverberation paths of the sound in a typical room is much longer than the integration period of the perception of sound by a human. Thus, it is important to accurately compute the exact time/frequency distribution of the reverberation. In contrast, the speed of light and the perception of light is such that the eye integrates out the transient response of a light source and only the energy steady-state response needs to be calculated. Third, since sound is a coherent wave phenomenon, the calculation of the reflected and scattered sound waves must incorporate the phase (complex amplitude) of the incident and reflected wave(s), while for incoherent light, only the power must be summed.

Although acoustic modeling has been well-studied in the context of non-interactive applications [34], such as concert hall design, there has been relatively little prior research in real-time acoustic modeling for virtual environment systems [15]. Currently available auralization systems generally model only early specular reflections, while late reverberations and diffractions are modeled with statistical approximations [1, 25, 40, 53]. Also, due to the computational complexity of current systems, they generally consider only simple geometric arrangements and low-order specular reflections. For instance, the Acoustetron [17] computes only first- and second-order specular reflections for box-shaped virtual environments. Video games provide spatialized sound with ad hoc localization methods (e.g., pan effects) rather than with realistic geometrical acoustic modeling methods. The 1995 National Research Council Report on Virtual Reality Scientific and Technological Challenges [15] states that "current technology is still unable to provide interactive systems with real-time rendering of acoustic environments with complex, realistic room reflections."

In this paper, we describe a beam tracing method that computes high-order specular reflection and transmission paths from fixed sources in large polygonal models fast enough to be used for auralization in interactive virtual environment systems. The key idea behind our method is to precompute and store spatial data structures that encode all possible transmission and specular reflection paths from each audio source and then use these data structures to compute reverberation paths to an arbitrarily moving observer viewpoint for real-time auralization during an interactive user session. Our algorithms for construction and query of these data structures have the unique features that they scale well with increasing numbers of reflections and global geometric complexity, and they extend naturally to model paths of diffraction and diffuse reflection. We have incorporated these algorithms and data structures into a system that supports real-time auralization and visualization of large virtual environments.

2 Previous Work

There has been a large amount of work in acoustic modeling. Prior methods can be classified into four types: 1) image source methods, 2) radiant exchange methods 3) path tracing, and 4) beam tracing.

2.1 Image Source Methods

Image source methods [2, 6] compute specular reflection paths by considering *virtual sources* generated by mirroring the location of the audio source, S, over each polygonal surface of the environment (see Figure 2). For each virtual source, S_i, a specular reflection path can be constructed by iterative intersection of a line segment from the source position to the receiver position, R, with the reflecting surface planes (such a path is shown for virtual source S_c in Figure 2). Specular reflection paths can be computed up to any order by recursive generation of virtual sources.

The primary advantage of image source methods is their robustness. They can guarantee that all specular paths up to a given order or reverberation time will be found. The disadvantages of image source methods are that they model only specular reflections, and

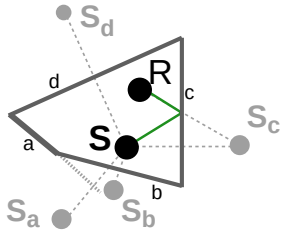


Figure 2: Image source method.

their expected computational complexity has exponential growth. In general, $O(n^r)$ virtual sources must be generated for r reflections in environments with n surface planes. Moreover, in all but the simplest environments (e.g., a box), complex validity/visibility checks must be performed for each of the $O(n^r)$ virtual sources since not all of the virtual sources represent physically realizable specular reflection paths [6]. For instance, a virtual source generated by reflection over the non-reflective side of a surface is "invalid." Likewise, a virtual source whose reflection is blocked by another surface in the environment or intersects a point on a surface's plane which is outside the surface's boundary (e.g., S_a in Figure 2) is "invisible." During recursive generation of virtual sources, descendents of invalid virtual sources can be ignored. However, descendents of invisible virtual sources must still be considered, as higher-order reflections may generate visible virtual sources (consider mirroring S_a over surface d). Due to the computational demands of $O(n^r)$ visibility checks, image source methods are practical only for acoustic modeling of few reflections in simple environments [32].

2.2 Radiant Exchange Methods

Radiant exchange methods have been used extensively in computer graphics to model diffuse reflection of radiosity between patches [21]. Briefly, radiosity methods consider every patch a potential emitter and reflector of radiosity. Conceptually, for every pair of patches, A and B, a form factor is computed which measures the fraction of the radiosity leaving patch A that arrives at patch B. This approach yields a set of simultaneous equations which are solved to obtain the radiosity for each patch.

Although this approach has been used with good results for modeling diffuse indirect illumination in computer graphics, it is not easily extensible to acoustics. In acoustics modeling, transport equations must account for phase, specular reflection tends to dominate diffuse reflection, and "extended form factor" computations must consider paths of diffraction as well as specular reflection. Furthermore, to meet error tolerances suitable for acoustic modeling, patches must be substructured to a very fine element mesh (typically much less than the acoustic wavelength), the solution must be computed for many frequencies, and the representation of the sound leaving an element must be very data intensive, a complex function of phase, direction, and frequency usually requiring thousands of bytes. As a result, direct extensions to prior radiosity methods [36, 39, 52] do not seem practical for large environments.

2.3 Path Tracing Methods

Ray tracing methods [33, 61] find reverberation paths between a source and receiver by generating rays emanating from the source position and following them through the environment until an appropriate set of rays has been found that reach a representation of the receiver position (see Figure 3).

Monte Carlo path tracing methods consider randomly generated paths from the source to the receiver [28]. For instance, the Metropolis Light Transport algorithm [54] generates a sequence of light transport paths by randomly mutating a single current path by

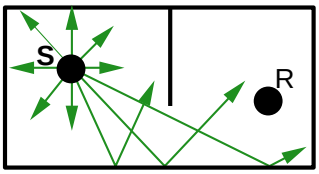


Figure 3: Ray tracing method.

adding, deleting, or replacing vertices. Mutated paths are accepted according to probabilities based on the estimated contribution they make to the solution. As contributing paths are found, they are logged and then mutated further to generate new paths in a Markov chain. Mutation strategies and acceptance probabilities are chosen to insure that the method is unbiased, stratified, and ergodic.

A primary advantage of these methods is their simplicity. They depend only on ray-surface intersection calculations, which are relatively easy to implement and have computational complexity that grows sublinearly with the number of surfaces in the model. Another advantage is generality. As each ray-surface intersection is found, paths of specular reflection, diffuse reflection, diffraction, and refraction can be sampled [10], thereby modeling arbitrary types of indirect reverberation, even for models with curved surfaces.

The primary disadvantages of path tracing methods stem from the fact that the continuous 5D space of rays is sampled by a discrete set of paths, leading to aliasing and errors in predicted room responses [35]. For instance, in ray tracing, the receiver position and diffracting edges are often approximated by volumes of space (in order to admit intersections with infinitely thin rays), which can lead to false hits and paths counted multiple times [35]. Moreover, important reverberation paths may be missed by all samples. In order to minimize the likelihood of large errors, path tracing systems often generate a large number of samples, which requires a large amount of computation. Another disadvantage of path tracing is that the results are dependent on a particular receiver position, and thus these methods are not directly applicable in virtual environment applications where either the source or receiver is moving continuously.

2.4 Beam Tracing Methods

Beam tracing methods [23] classify reflection paths from a source by recursively tracing pyramidal beams (i.e., sets of rays) through the environment. Briefly, a set of pyramidal beams are constructed that completely cover the 2D space of directions from the source. For each beam, polygons are considered for intersection in order from front to back. As intersecting polygons are detected, the original beam is clipped to remove the shadow region, a transmission beam is constructed matching the shadow region, and a reflection beam is constructed by mirroring the transmission beam over the polygon's plane (see Figure 4).

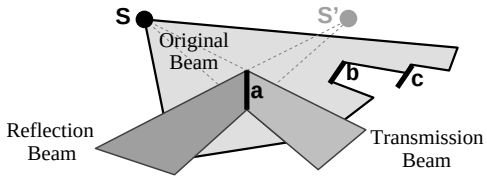


Figure 4: Beam tracing method.

As compared to image source methods, the primary advantage of beam tracing is that fewer virtual sources must be considered for environments with arbitrary geometry. Since each beam represents the region of space for which a corresponding virtual source (at the apex of the beam) is visible, higher-order virtual sources must be considered only for reflections of polygons intersecting the beam. For instance, in Figure 5, consider the virtual source S_a, which results from reflection of S over polygon a. The corresponding reflection beam, R_a, contains exactly the set of receiver points for which S_a is valid and visible. Similarly, R_a intersects exactly the set of polygons (c and d) for which second-order reflections are possible after specular reflection off polygon a. Other polygons (b, e, f, and g) need not be considered for second order specular reflections after a. Beam tracing allows the recursion tree of virtual sources to be pruned significantly. On the other hand, the image source method is more efficient for a box-shaped environment for which a regular lattice of virtual sources can be constructed that are guaranteed to be visible for all receiver locations [2].

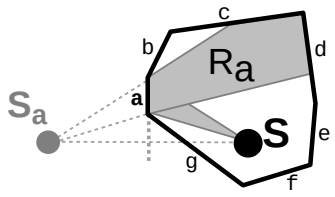


Figure 5: Beam tracing culls invisible virtual sources.

As compared to path tracing methods, the primary advantage of beam tracing is that it takes advantage of spatial coherence, as each beam-surface intersection represents an infinite number of ray-surface intersections. Also, pyramidal beam tracing does not suffer from sampling artifacts of ray tracing [35] or the overlap problems of cone tracing [3, 55], since the entire 2D space of directions leaving the source can be covered by beams exactly.

The primary disadvantage of beam tracing is that the geometric operations required to trace a beam through a 3D model (i.e., intersection and clipping) are relatively complex, as each beam may be reflected and/or obstructed by several surfaces. Another limitation is that reflections off curved surfaces and refractions are difficult to model.

Geometric beam tracing has been used in a variety of applications, including acoustic modeling [13, 38, 46, 57], illumination [9, 19, 20, 22, 23, 59], and radio propagation [16]. The challenge is to perform geometric operations (i.e., intersection, clipping, and mirroring) on beams efficiently as they are traced recursively through a complex environment.

Some systems avoid the geometric complexity of beam tracing by approximating each beam by its medial axis ray for intersection and mirror operations [36], possibly splitting rays as they diverge with distance [31, 42]. In this case, the beam representation is only useful for modeling the distribution of rays/energy with distance and for avoiding large tolerances in ray-receiver intersection calculations. If beams are not clipped or split when they intersect more than one surface, significant reverberation paths can be missed.

Heckbert and Hanrahan [23] described an algorithm for illumination in which pyramidal beams represented by their 2D polygonal cross-sections are traced recursively either forward from a viewpoint or backward from a point light source. For each beam, all polygons are processed in front to back order. For each polygon intersecting the beam, the shadow region is "cut out" of the original beam using a polygon clipping algorithm capable of handling concavities and holes. The authors describe construction of an intermediate "light beam tree" data structure that encodes the beam tracing recursion and is used for later evaluation of light paths. Their implementation does not scale well to large environments since its computational complexity grows with $O(n^2)$ for n polygons.

Dadoun et al. [12, 13] described a beam tracing algorithm for acoustic modeling in which a hierarchical scene representation (HSR) is used to accelerate polygon sorting and intersection testing. During a preprocessing phase, a binary space partition (BSP) tree

structure is constructed and augmented with storage for the convex hull for each subtree. Then, during beam tracing, the HSR is used to accelerate queries to find an ordered set of polygons potentially intersecting each beam. As in [23], beams are represented by their 2D polygonal cross-sections and are updated using the Weiler-Atherton clipping algorithm [60] at polygon intersections.

Fortune [16] described a beam tracing algorithm for indoor radio propagation prediction in which a spatial data structure comprising "layers" of 2D triangulations is used to accelerate polygon intersection testing. A method is proposed in which beams are partitioned into convex regions by planes supporting the edges of occluding polygons. However, Fortune expects that method to be too expensive for use in indoor radio propagation (where attenuation due to transmission is small) due to the exponential growth in the number of beams. Instead, he has implemented a system in which beams are traced directly from the source and along paths of reflection, but are not clipped by occluding polygons. Instead, attenuation due to occlusion is computed for each path, taking into account the attenuation of each occluding polygon along the path. This implementation trades-off more expensive computation during path generation for less expensive computation during beam tracing.

Jones [27] and Teller [51] have described beam tracing algorithms to compute a "potentially visible set" of polygons to render from a particular viewpoint in a computer graphics scene. These algorithms preprocess the scene into a spatial subdivision of cells (convex polyhedra) and portals (transparent, boundaries between cells). Polyhedral beams are traced through portals to determine the region of space potentially visible from a view frustum in order to produce a conservative and approximate solution to the hidden surface problem.

In this paper, we describe beam tracing data structures and algorithms for real-time acoustic modeling in interactive virtual environment applications. Our method is most closely related to work in [23] and [51]. As compared to previous acoustic modeling methods, the unique features of our method are the ability to: 1) generate specular reflection and transmission paths at interactive rates, 2) scale to support large virtual environments, 3) scale to compute high-order reflection and transmission paths, and 4) extend to support efficient computation of diffraction and diffuse reflection paths. We have included these algorithms and data structures in an interactive virtual environment system that supports immersive auralization and visualization in complex polygonal environments.

3 System Organization

Our virtual environment system takes as input: 1) a description of the geometry and visual/acoustic surface properties of the environment (i.e., sets of polygons), and 2) a set of anechoic audio source signals at fixed locations. As a user moves through the virtual environment interactively, the system generates images as seen from a simulated observer viewpoint, along with a stereo audio signal spatialized according to the computed reverberation paths from each audio source to the observer location.

In order to support real-time auralization, we partition our system into four distinct phases (see Figure 6), two of which are preprocessing steps that execute off-line, while the last two execute in real-time as a user interactively controls an observer viewpoint moving through a virtual environment. First, during the *spatial subdivision phase*, we precompute spatial relationships inherent in the set of polygons describing the environment and represent them in a cell adjacency graph data structure that supports efficient traversals of space. Second, during the *beam tracing phase*, we recursively follow beams of transmission and specular reflection through space for each audio source. The output of the beam tracing phase is a beam tree data structure that explicitly encodes the region of space reachable by each sequence of reflection and transmission paths from each source point. Third, during the *path generation* phase, we compute reverberation paths from each source to the receiver via lookup into the precomputed beam tree data structure as the receiver (i.e., the observer viewpoint) is moved under interactive user control. Finally, during the *auralization phase*, we spatialize each source audio signal (in stereo) according to the lengths, attenuations, and directions of the computed reverberation paths. The spatialized audio output is synchronized with real-time graphics output to provide an immersive virtual environment experience.

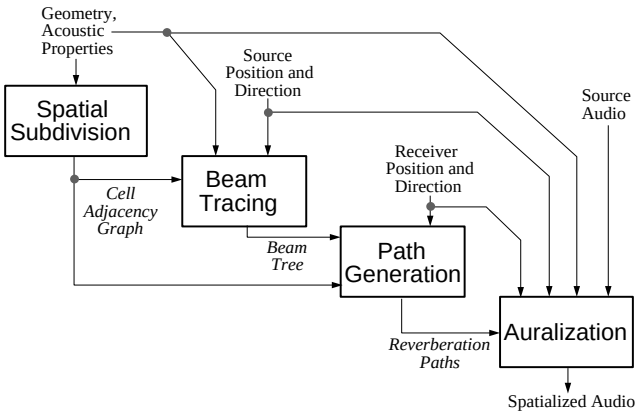


Figure 6: System organization.

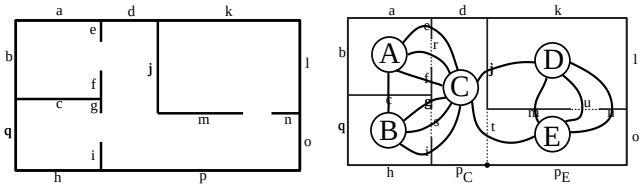
3.1 Spatial Subdivision

Our system preprocesses the geometric properties of the environment and builds a spatial subdivision to accelerate beam tracing. The goal of this phase is to precompute spatial relationships inherent in the set of polygons describing the environment and to represent them in a data structure that supports efficient traversals of space. The spatial subdivision is constructed by partitioning 3D space into a set of convex polyhedral regions and building a graph that explicitly represents the adjacencies between the regions of the subdivision.

We build the spatial subdivision using a Binary Space Partition (BSP) [18], a recursive binary split of 3D space into convex polyhedral regions (*cells*) separated by planes. To construct the BSP, we recursively split cells by candidate planes selected by the method described in [41]. The binary splitting process continues until no input polygon intersects the interior of any BSP cell. The result of the BSP is a set of convex polyhedral cells whose convex, planar boundaries contain all the input polygons.

An adjacency graph is constructed that explicitly represents the neighbor relationships between cells of the spatial subdivision. Each cell of the BSP is represented by a node in the graph, and two nodes have a *link* between them for each planar, polygonal boundary shared by the corresponding adjacent cells in the spatial subdivision. Construction of the cell adjacency graph is integrated with the binary space partitioning algorithm. If a leaf in the BSP is split into two by a plane, we create new nodes in the graph corresponding to the new cells in the BSP, and we update the links of the split leaf's neighbors to reflect the new adjacencies. We create a separate link between two cells for each convex polygonal region that is entirely either transparent or opaque along the cells' shared boundary.

A simple 2D example model (on left) and its cell adjacency graph (on right) are shown in Figure 7. Input "polygons" appear as solid line segments labeled with lower-case letters ($a - q$); transparent cell boundaries introduced by the BSP are shown as dashed line segments labeled with lower-case letters ($r - u$); constructed cell regions are labeled with upper-case letters ($A - E$); and, links are drawn between adjacent cells sharing a convex "polygonal" boundary.



(a) Input model. (b) Cell adjacency graph.

Figure 7: Example spatial subdivision.

3.2 Beam Tracing

After the spatial subdivision has been constructed, we use it to accelerate traversals of space in our beam tracing algorithm. Beams are traced through the cell adjacency graph via a recursive depth-first traversal starting in the cell containing the source point. Adjacent cells are visited recursively while a beam representing the region of space reachable from the source by a sequence of cell boundary reflection and transmission events is incrementally updated. As the algorithm traverses a cell boundary into a new cell, the current convex pyramidal beam is "clipped" to include only the region of space passing through the convex polygonal boundary polygon. At reflecting cell boundaries, the beam is mirrored across the plane supporting the cell boundary in order to model specular reflections. As an example, Figure 8 shows a sequence of beams (green polyhedra) traced up to one reflection from a source (white point) through the spatial subdivision (blue 'X's are cell boundaries) for a simple set of input polygons (red surfaces).

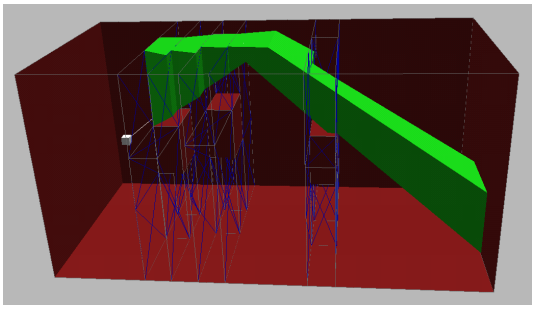


Figure 8: A beam clipped and reflected at cell boundaries.

Throughout the traversal, the algorithm maintains a *current cell* (a reference to a cell in the spatial subdivision) and a *current beam* (an infinite convex pyramidal beam whose apex is the source point). Initially, the *current cell* is set to be the cell containing the source point and the *current beam* is set to cover all of space. During each step of the depth-first traversal, the algorithm continues recursively for each boundary polygon, P, of the *current cell*, C, that intersects the *current beam*, B. If P does not coincide with an opaque input surface, the algorithm follows a transmission path, recursing to the cell adjacent to C across P with a *transmission beam*, B_t, constructed as the intersection of B with a pyramidal beam whose apex is the source point and whose sides pass through the edges of P. Likewise, if P coincides with a reflecting input surface, the algorithm follows a specular reflection path, recursing in cell C with a *specular reflection beam*, B_r, constructed by mirroring the *transmission beam* over the plane supporting P. The depth-first traversal along any path terminates when the length of a path exceeds a user-specified threshold or when the cumulative absorption due to transmission and reflection exceeds a preset threshold. The traversal may also be terminated when the total number of reflections or transmissions exceeds a third threshold.

Figure 9 contains an illustration of the beam tracing algorithm execution for specular reflections through the simple 2D example model shown in Figure 7. The depth-first traversal starts in the cell (labeled 'D') containing the source point (labeled 'S') with a beam containing the entire cell (shown as dark green). Beams are created and traced for each of the six boundary polygons of cell 'D' (j, k, l, m, n, and u). For example, transmission through the cell boundary labeled 'u' results in a beam (labeled T_u) that is trimmed as it enters cell 'E.' T_u intersects only the polygon labeled 'o,' which spawns a reflection beam (labeled $T_u R_o$). That beam intersects only the polygon labeled 'p,' which spawns a reflection beam (labeled $T_u R_o R_p$). Execution continues recursively for each beam until the length of every path exceeds a user-specified threshold or when the absorption along every path becomes too large.

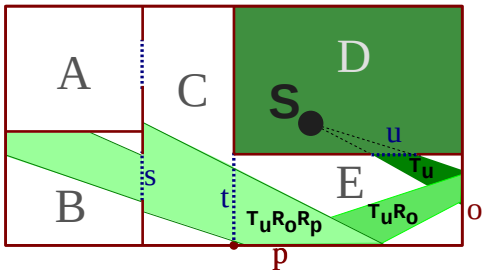


Figure 9: Beam tracing through cell adjacency graph.

While tracing beams through the spatial subdivision, our algorithm constructs a *beam tree* data structure [23] to be used for rapid determination of reverberation paths from the source point later during the path generation phase. The beam tree corresponds directly to the recursion tree generated during the depth-first traversal through the cell adjacency graph. It is similar to the "stab tree" data structure used by Teller to encode visibility relationships for occlusion culling [51]. Each node of the beam tree stores: 1) a reference to the cell being traversed, 2) the cell boundary most recently traversed (if there is one), and 3) the convex beam representing the region of space reachable by the sequence of reflection and transmission events along the current path of the depth-first traversal. To further accelerate reverberation path generation, each node of the beam tree also stores the cumulative attenuation due to reflective and transmissive absorption, and each cell of the spatial subdivision stores a list of "back-pointers" to its beam tree nodes. Figure 10 shows a partial beam tree corresponding to the traversal shown in Figure 9.

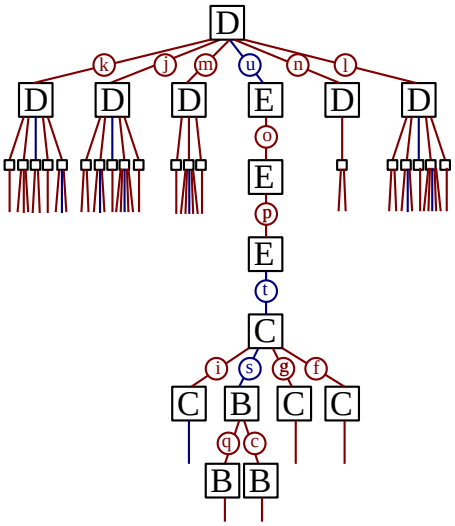


Figure 10: Beam tree.

3.3 Path Generation

During an interactive session in which the user navigates a simulated observer (receiver) through the virtual environment, reverberation paths from a particular source point, S, to the moving receiver point, R, can be generated quickly via lookup in the beam tree data structure. First, the cell containing the receiver point is found by logarithmic-time search of the BSP. Then, each beam tree node, T, associated with that cell is checked to see whether the beam stored with T contains the receiver point. If it does, a viable ray path from the source point to the receiver point has been found, and the ancestors of T in the beam tree explicitly encode the set of reflections and transmissions through the boundaries of the spatial subdivision that a ray must traverse from the source to the receiver along this path (more generally, to any point inside the beam stored with T).

The attenuation, length, and directional vectors for the corresponding reverberation path can be derived quickly from the data stored with the beam tree node, T. Specifically, the attenuation due to reflection and transmission can be retrieved from T directly. The length of the reverberation path and the directional vectors at the source and receiver points can be easily computed as the source's reflected image for this path is stored explicitly in T as the apex of its pyramidal beam. The actual ray path from the source point to the receiver point can be generated by iterative intersection with the reflecting cell boundaries stored with the ancestors of T. For example, Figure 11 shows the specular reflection path to a particular receiver point (labeled 'R') for the example shown in Figure 9.

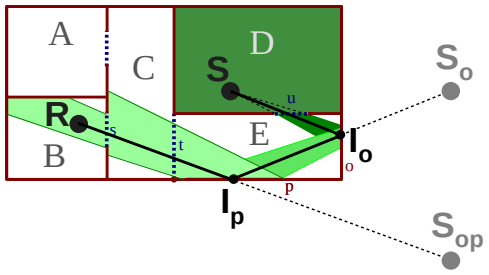


Figure 11: Reverberation path to receiver point ('R') computed via lookup in beam tree for source point ('S').

3.4 Auralization

Once a set of reverberation paths from a source to the receiver has been computed, the source-receiver impulse response is generated by adding one pulse corresponding to each distinct path from the source to the receiver. The delay associated with each pulse is given by L/C, where L is the length of the corresponding reverberation path, and C is the speed of sound. Since the pulse is attenuated by every reflection and dispersion, the amplitude of each pulse is given by A/L, where A is the product of all the frequency-independent reflectivity and transmission coefficients for each of the reflecting and transmitting surfaces along the corresponding reverberation path.

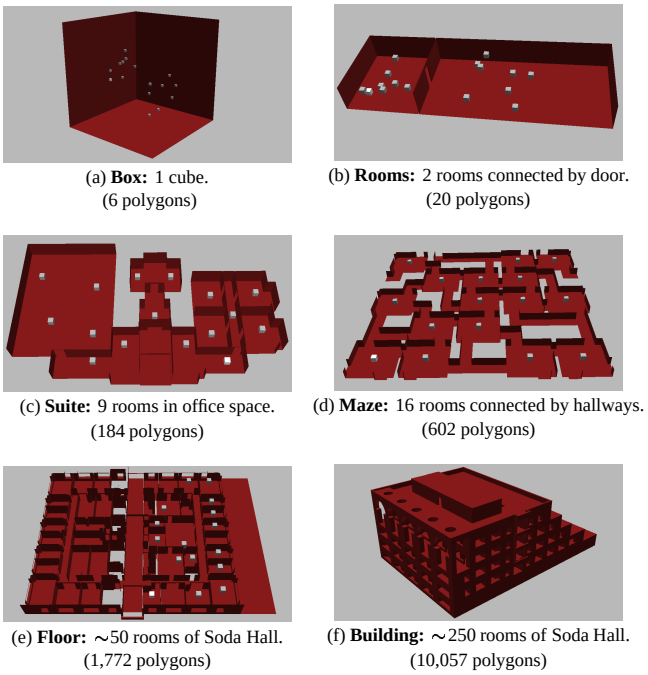
At the receiver, the binaural impulse responses (response of the left and right ears) are different due to the directivity of each ear. These binaural impulse responses are generated by multiplying each pulse of the impulse response by the cardioid directivity function $(1/2(1+\cos(\theta))$, where θ is the angle of arrival of the pulse with respect to the normal vector pointing out of the ear) corresponding to each ear. This rough approximation to actual head scattering and diffraction is similar to the standard two-point stereo microphone technique used in high fidelity audio recording. Finally, the (anechoic) input audio signal is auralized by convolving it with the binaural impulse responses to produce a stereo spatialized audio signal. In the future, we intend to incorporate source directivity, frequency-dependent absorption [34], and angle-dependent absorption [11, 43] into our acoustic models.

A separate, concurrently executing process is spawned to perform convolution of the computed binaural impulse responses with the input audio signal. In order to support real-time auralization, transfer of the impulse responses from the path generation process to the convolution process utilizes double buffers synchronized by a semaphore. Each new pair of impulse responses is loaded by the path generation process into a "back buffer" as the convolution process continues to access the current impulse responses stored in the "front buffer." A semaphore is used to synchronize the processes as the front and back buffer are switched.

4 Results

The 3D data structures and algorithms described in the preceding sections have been implemented in C++ and run on Silicon Graphics and PC/Windows computers.

To test whether the algorithms scale well as the complexity of the 3D environment and the number of specular reflections increase, we executed a series of experiments with our system computing spatial subdivisions, beam trees, and specular reflection paths for various architectural models of different complexities. Our test models ranged from a simple box to a complex building, Soda Hall, the computer science building at UC Berkeley (an image and description of each test model appears in Figure 12). The experiments were run on a Silicon Graphics Octane workstation with 640MB of memory and used one 195MHz R10000 processor.



(a) **Box:** 1 cube. (6 polygons)

(b) **Rooms:** 2 rooms connected by door. (20 polygons)

(c) **Suite:** 9 rooms in office space. (184 polygons)

(d) **Maze:** 16 rooms connected by hallways. (602 polygons)

(e) **Floor:** ∼50 rooms of Soda Hall. (1,772 polygons)

(f) **Building:** ∼250 rooms of Soda Hall. (10,057 polygons)

Figure 12: Test models (source locations are gray dots).

4.1 Spatial Subdivision Results

We first constructed the spatial subdivision data structure (cell adjacency graph) for each test model. Statistics from this phase of the experiment are shown in Table 1. Column 2 lists the number of input polygons in each model, while Columns 3 and 4 contain

the numbers of cells and links, respectively, generated by the spatial subdivision algorithm. Column 5 contains the wall-clock time (in seconds) for the algorithm to execute, while Column 6 shows the storage requirements (in MBs) for the resulting spatial subdivision.

Model Name	# Polys	# Cells	# Links	Time (sec)	Storage (MB)
Box	6	7	18	0.0	0.004
Rooms	20	12	43	0.1	0.029
Suite	184	98	581	3.0	0.352
Maze	602	172	1,187	4.9	0.803
Floor	1,772	814	5,533	22.7	3.310
Bldg	10,057	4,512	31,681	186.3	18.694

Table 1: Spatial subdivision statistics.

Empirically, we find that the numbers of cells and links created by our spatial subdivision algorithm grow linearly with the number of input polygons for typical architectural models (see Figure 13), rather than quadratically as is possible for worst case geometric arrangements. The reason for linear growth can be seen intuitively in the two images inlaid in Figure 13, which compare spatial subdivisions for the Maze test model (on the left) and a 2x2 grid of Maze test models (on the right). The 2x2 grid of Mazes has exactly four times as many polygons and approximately four times as many cells. The storage requirements of the spatial subdivision data structure also grow linearly as they are dominated by the vertices of link polygons.

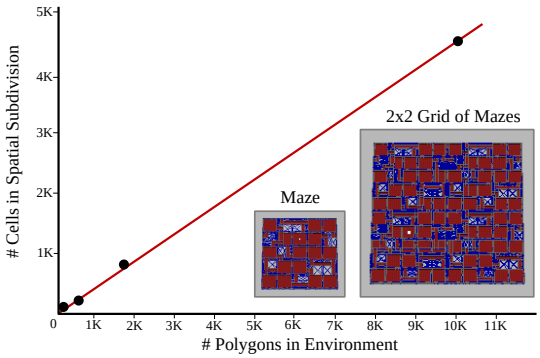


Figure 13: Plot of subdivision size vs. polygonal complexity.

The time required to construct the spatial subdivisions grows super-linearly, dominated by the code that selects and orders splitting planes during BSP construction (see [41]). It is important to note that the spatial subdivision phase must be executed only once off-line for each geometric model, as its results are stored in a file, allowing rapid reconstruction in subsequent beam tracing executions.

4.2 Beam Tracing Results

We experimented with our beam tracing algorithm for sixteen source locations in each test model. The source locations were chosen to represent typical audio source positions (e.g., in offices, in common areas, etc.) – they are shown as gray dots in Figure 12 (experiments with the Building test used the same source locations as are shown in the Floor model). For each source location, we traced beams (i.e., constructed a beam tree) five times, each time with a different limit on the maximum number of specular reflections (e.g., up to 0, 1, 2, 4, or 8 reflections). Other termination criteria based on attenuation or path length were disabled, and transmission was ignored, in order to isolate the impact of input model size and maximum number of specular reflections on computational complexity.

Table 2 contains statistics gathered during the beam tracing experiment – each row represents an execution with a particular test model and maximum number of reflections, averaged over all 16 source locations. Columns 2 and 3 show the number of polygons describing each test model and the maximum number of specular reflections allowed in each test, respectively. Column 4 contains the average number of beams traced by our algorithm (i.e., the average number of nodes in the resulting beam trees), and Column 5 shows the average wall-clock time (in milliseconds) for the beam tracing algorithm to execute.

Model Name	# Polys	# Rfl	Beam Tracing # Beams	Beam Tracing Time (ms)	Path Generation # Paths	Path Generation Time (ms)
Box	6	0	1	0	1.0	0.0
		1	7	1	7.0	0.1
		2	37	3	25.0	0.3
		4	473	42	129.0	6.0
		8	10,036	825	833.0	228.2
Rooms	20	0	3	0	1.0	0.0
		1	31	3	7.0	0.1
		2	177	16	25.1	0.3
		4	1,939	178	127.9	5.2
		8	33,877	3,024	794.4	180.3
Suite	184	0	7	1	1.0	0.0
		1	90	9	6.8	0.1
		2	576	59	25.3	0.4
		4	7,217	722	120.2	6.5
		8	132,920	13,070	672.5	188.9
Maze	602	0	11	1	0.4	0.0
		1	167	16	2.3	0.0
		2	1,162	107	8.6	0.1
		4	13,874	1,272	36.2	2.0
		8	236,891	21,519	183.1	46.7
Floor	1,772	0	23	4	1.0	0.0
		1	289	39	6.1	0.1
		2	1,713	213	21.5	0.4
		4	18,239	2,097	93.7	5.3
		8	294,635	32,061	467.0	124.5
Bldg	10,057	0	28	5	1.0	0.0
		1	347	49	6.3	0.1
		2	2,135	293	22.7	0.4
		4	23,264	2,830	101.8	6.8
		8	411,640	48,650	529.8	169.5

Table 2: Beam tracing and path generation statistics.

Scale with Increasing Polygonal Complexity

We readily see from the results in Column 4 that the number of beams traced by our algorithm (i.e., the number of nodes in the beam tree) does *not* grow at an exponential rate with the number of polygons in these environments (as it does using the image source method). Each beam traced by our algorithm pre-classifies the regions of space according to whether the corresponding virtual source (i.e., the apex of the beam) is visible to a receiver. Rather than generating $O(n)$ virtual sources (beams) at each step of the recursion as in the image source method, we directly find only the potentially visible virtual sources via beam-polygon intersection and cell adjacency graph traversal. We use the current beam and the current cell of the spatial subdivision to find the small set of polygon reflections that admit visible higher-order virtual sources.

The benefit of this approach is particularly important for large environments in which the boundary of each convex cell is simple, and yet the entire environment is very complex. As an example, consider computation of up to 8 specular reflections in the Building test model (the last row of Table 2). The image source method must consider approximately 1,851,082,741 virtual sources ($\sum_{r=0}^{8}(10,057/2)^r$), assuming half of the 10,057 polygons are front-facing to each virtual source. Our beam tracing method con-

siders only 411,640 virtual sources, a difference of four orders of magnitude. In most cases, it would be impractical to build and store the recursion tree without such effective pruning.

In "densely-occluded" environments, in which all but a little part of the environment is occluded from any source point (e.g., most buildings, cities, etc.), the number of beams traced by our algorithm does not even grow linearly with the total number of polygons in the environment (see Figure 14). In these environments, the number of boundaries on each cell is nearly constant, and a nearly constant number of cells are reached by each beam, leading to near-constant expected-case complexity of our beam tracing algorithm with increasing global environment complexity. As an example, the two images inlaid in Figure 14 show that the number of beams (green) traced in the Maze test model (left) does not increase significantly if the model is increased to be a 2x2 grid of Maze models (right). *The beam tracing algorithm is impacted only by local complexity, and not by global complexity.*

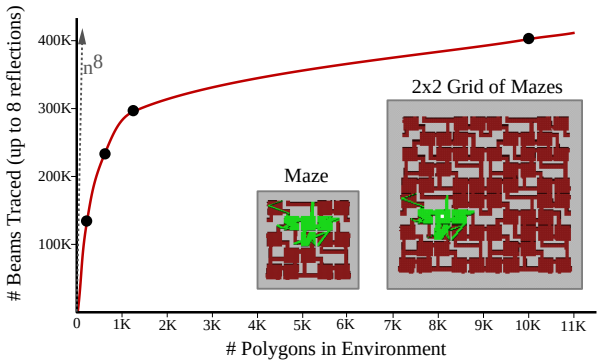


Figure 14: Plot of beam tree size vs. polygonal complexity.

Scale with Increasing Reflections

We see that the number of beams traced by our algorithm grows exponentially, but far slower than $O(n^r)$, as we increase the maximum number of reflections. Figure 15 shows a logscale plot of the average number of beams traced in the Building model with increasing numbers of specular reflections. The beam tree growth is less than $O(n^r)$ because each beam narrows as it is clipped by the cell boundaries it has traversed, and thus it tends to intersect fewer cell boundaries (see the example beam inlaid in Figure 15). In the limit, each beam becomes so narrow that it intersects only one or two cell boundaries, on average, leading to a beam tree with a small branching factor (rather than a branching factor of $O(n)$, as in the image source method).

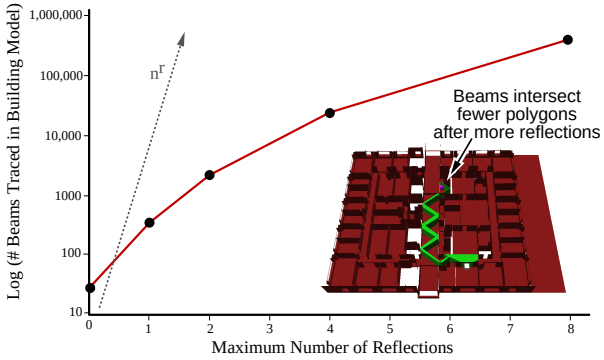


Figure 15: Plot of beam tree size with increasing reflections.

Tree Depth	Total Nodes	Interior Nodes	Leaf Nodes	Branching Factor
0	1	1	0	16.0000
1	16	16	0	6.5000
2	104	104	0	4.2981
3	447	446	1	2.9193
4	1,302	1,296	6	2.3920
5	3,100	3,092	8	2.0715
6-10	84,788	72,469	12,319	1.2920
11-15	154,790	114,664	40,126	1.2685
>15	96,434	61,079	35,355	1.1789

Table 3: Example beam tree branching statistics.

As an example, consider Table 3 which shows the average branching factor for nodes at each depth of the beam tree constructed for up to 8 specular reflections in the Building model from one source location. The average branching factor (Column 5) generally decreases with tree depth and is generally bounded by a small constant in lower levels of the tree.

On the other hand, if a beam is trimmed by many cell boundaries and becomes too narrow, the advantages of beam tracing over ray tracing are diminished. This observation suggests a possible future hybrid approach in which medial rays are used to approximate intersections for beams whose cross-sectional area falls below a threshold.

4.3 Path Generation Results

In order to verify that specular reflection paths can be computed from fixed sources at interactive rates as the receiver moves, we conducted experiments to quantify the complexity of generating specular reflection paths to different receiver locations from precomputed beam trees. For each beam tree constructed in the previous experiment, we logged statistics during generation of specular reverberation paths to 16 different receiver locations. Receivers were chosen randomly within a two foot sphere around the source to represent a typical audio scenario in which the source and receiver are in close proximity within the same "room." We believe this represents a worst-case scenario as fewer paths would likely be found to more remote and more occluded receiver locations.

Columns 6 and 7 of Table 2 contain statistics gathered during path generation for each combination of model and termination criterion averaged over all 256 source-receiver pairs (i.e., 16 receivers for each of the 16 sources). Column 6 contains the average number of reverberation paths generated, while Column 7 shows the average wall-clock time (in milliseconds) for execution of the path generation algorithm. Figure 16 shows a plot of the wall-clock time required to generate up to eighth-order specular reflection paths for each test model.

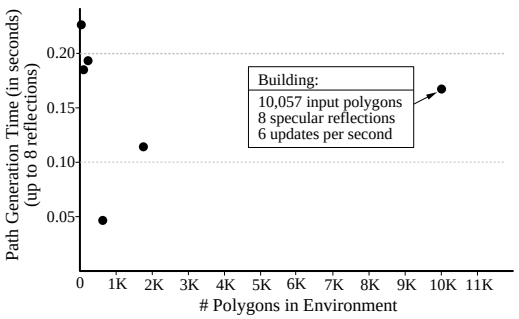


Figure 16: Path compute time vs. polygonal complexity.

We find that the number of specular reflection paths between a source and receiver in close proximity of one another is nearly constant across all of our test models. Also, the time required by our path generation algorithm is generally *not* dependent on the number of polygons in the environment (see Figure 16), nor is it dependent on the total number of nodes in the precomputed beam tree. This result is due to the fact that our path generation algorithm considers only nodes of the beam tree with beams residing inside the cell containing the receiver location. Therefore, the computation time required by the algorithm is *not* dependent on the complexity of the whole environment, but instead on the number of beams that traverse the receiver's cell.

Overall, we find that our algorithm supports generation of specular reflection paths between a fixed source and any (arbitrarily moving) receiver at interactive rates in complex environments. For instance, we are able to compute up to 8th order specular reflection paths in the Building environment with more than 10,000 polygons at a rate of approximately 6 times per second (i.e., the rightmost point in the plot of Figure 16).

4.4 Auralization Results

We have integrated the acoustic modeling method described in this paper into an interactive system for audio/visual exploration of virtual environments (e.g., using VRML). The system allows a user to move through a virtual environment while images and spatialized audio are rendered in real-time according to the user's simulated viewpoint. Figure 17 shows one application we have developed, called VirtualWorks, in which a user may interact with objects (e.g., click on them with the mouse) in the virtual environment to invoke behaviors that present information in various media, including text, image, video, and spatialized audio. For instance, if the user clicks on the workstation sitting on the desk, the application invokes a video which is displayed on the screen of that workstation. We are using this system to experiment with 3D user interfaces for presentation of multimedia data and multi-user interaction.

We ran experiments with this application using a Silicon Graphics Octane workstation with 640MB of memory and two 195MHz R10000 processors. One processor was used for image generation and acoustic modeling (i.e., reverberation path generation), while the second processor was dedicated solely to auralization (i.e., convolution of the computed stereo impulse responses with audio signals).

Due to the differences between graphics and acoustics described in Section 1, the geometry and surface characteristics of the virtual environment were input and represented in two separate forms, one for graphics and another for acoustics. The graphical model (shown in Figure 17) was represented as a scene graph containing 80,372 polygons, most of which describe the furniture and other small, detailed, visually-important objects in the environment. The acoustical model contained only 184 polygons, which described the ceilings, walls, cubicles, floors, and other large, acoustically-important features of the environment (it was identical to the Suite test model shown in Figure 12*c*).

We gathered statistics during sample executions of this application. Figures 17*b-c* show an observer viewpoint path (red) along which the application was able to render between eight and twelve images per second, while simultaneously auralizing four audio sources (labeled 1-4) in stereo according to fourth-order specular reflection paths updated during each frame. While walking along this path, it was possible to notice subtle acoustic effects due to reflections and occlusions. In particular, near the viewpoint labeled 'A' in Figure 17*b*, audio source '2' became very reverberant due to reflections (cyan lines) in the long room. Likewise, audio source '3' suddenly became much louder and then softer as the observer passed by an open doorway near the viewpoint labeled 'B' in Figure 17*c*.

Throughout our experiments, the auralization process was the bottleneck. Our C++ convolution code running on a R10000 processor could execute fast enough to output 8 KHz stereo audio for a set of impulse responses cumulatively containing around 500 non-zero elements. We are planning to integrate DSP-based hardware [37] with our system to implement real-time convolution in the near future.

5 Discussion

5.1 Geometric Limitations

Our method is not practical for all virtual environments. First, the geometric input must comprise only planar polygons. Each acoustic reflector is assumed to be locally reacting and to have dimensions far exceeding the wavelength of audible sound (since initially we are assuming that specular reflections are the dominant components of reverberation).

Second, the efficiency of our method is greatly impacted by the complexity and quality of the constructed spatial subdivision. For best results, the polygons should be connected (e.g., without small cracks between them) and arranged such that a large part of the model is occluded from any position in space (e.g., like most buildings or cities). Specifically, our method would not perform well for geometric models with high local geometric complexity (e.g., a forest of trees). In these cases, beams traced through boundaries of cells enclosing free space would quickly become fragmented into many smaller beams, leading to disadvantageous growth of the beam tree. For this reason, our method is not as well suited for global illumination as it is for acoustic modeling, in which small objects can be ignored and large surfaces can be modeled with little geometric surface detail due to the longer wavelengths of audible sound.

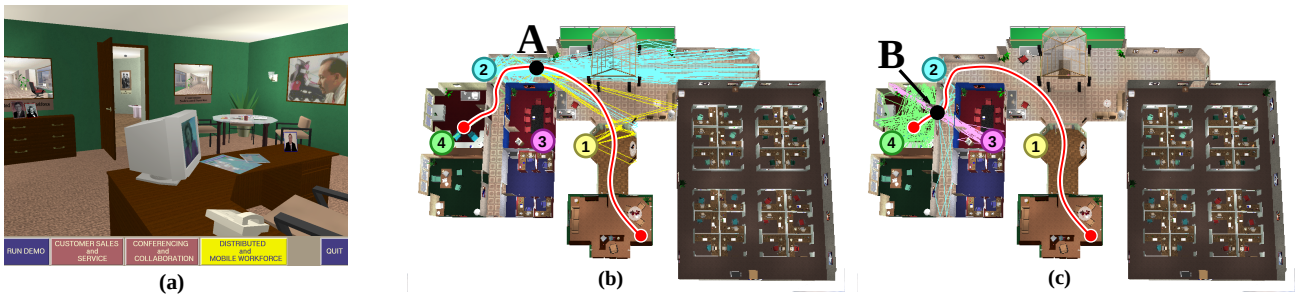
Third, the major occluding and reflecting surfaces of the virtual environment must be static during interactive path generation and auralization. If any acoustically significant polygon moves, the spatial subdivision and every beam tree must be recomputed.

The class of geometric models for which our method does work well includes most architectural and urban environments. In these models, acoustically significant surfaces are generally planar, large, and stationary, and the acoustical effects of any sound source are limited to a local region of the environment.

5.2 Diffraction and Diffuse Reflection

Our current 3D implementation traces beams only along paths of specular reflection and transmission, and it does not model other scattering effects. Of course, paths of diffraction and diffuse reflection are also important for accurate acoustic modeling [34, 26]. Fortunately, our beam tracing algorithm and beam tree representation can be generalized to model these effects. For instance, new beams can be traced that enclose the region of space reached by diffracting and diffuse reflection paths, and new nodes can be added to the beam tree representing diffractions and diffuse reflection events at cell boundaries. For these more complex scattering phenomena, the geometry of the beams is most useful for computing candidate reverberation paths, while the amplitude of the signal along any of the these paths can be evaluated for a known receiver during path generation. We have already included these extensions in a 2D beam tracing implementation, and we are currently working on a similar 3D implementation.

First, consider diffraction. According to the Geometrical Theory of Diffraction [29], an acoustic field that is incident on a discontinuity along an edge has a diffracted wave that propagates into the shadow region. The diffracted wave can be modeled in geometric terms by considering the edge to be a source of new waves emanating from the edge. Higher order reflections and diffractions occur as diffracted waves impinge upon other surfaces and edge discontinuities. By using edge-based adjacency information in our



(a) (b) (c)

Figure 17: VirtualWorks application. User's view is shown in (a), while a bird's eye view of fourth-order reverberation paths (color coded lines) from four sources (numbered circles) to Viewpoints 'A' and 'B' (black circles) are shown in (b) and (c), respectively.

spatial subdivision data structure, we can quickly perform the geometric operations required to construct and trace beams along paths of diffraction. For a given beam, we can find edges causing diffraction, as they are the ones: 1) intersected by the beam, and 2) shared by cell boundaries with different acoustic properties (e.g., one is transparent and another is opaque). For each such edge, we can determine the region of space reached by diffraction at that edge by tracing a beam whose "source" coincides with the portion of the edge intersected by the impinging beam, and whose extent is bounded by the solid wedge of opaque surfaces sharing the edge. For densely-occluded environments, each such diffraction beam can be computed and traced in expected-case constant time.

Second, consider diffuse reflection. We may model complex reflections and diffractions from some highly faceted surfaces as diffuse reflections from planar surfaces emanating equally in all directions. To compute the region of space reached by such a reflection using our approach, we can construct a beam whose "source" is the convex polygonal region of the surface intersected by an impinging beam and whose initial extent encloses the entire halfspace in front of the surface. We can trace the beam through the cell adjacency graph to find the region of space reached from any point on the reflecting part of the surface (i.e., the anti-penumbra [50]).

We have implemented these methods so far in 2D using a planar winged-edge representation [5] for the spatial subdivision and a bow-tie representation [49] for the beams. Unfortunately, tracing 3D "beams" of diffraction and diffuse reflection is more complicated. First, the source of each diffraction beam is no longer a point, but a finite edge, and the source of each diffuse reflection beam is generally a convex polygon. Second, as we trace such beams through the spatial subdivision, splitting and trimming them as they passes through multiple convex polygonal cell boundaries, their bounding surfaces can become quadric surfaces (i.e., reguli) [50]. Finally, evaluation of the amplitude of the signal along a "path" of diffraction or diffuse reflection requires integration over (possibly multiple) edges and polygons. We are currently extending our 3D data structures and algorithms to model these effects. Initially, we are planning to trace polyhedral beams that conservatively over-estimate the region covered by an exact, more complex, representation of the scattering patterns. Then, as each reverberation path to a particular receiver is considered, we will check whether it lies within the exact scattering region, or whether it should be discarded because it lies in the over-estimating region of the polyhedral beam.

5.3 Visualization

In order to aid understanding and debugging of our acoustic modeling method, we have found it extremely valuable to use interactive visualization. So far, we have concentrated on visualization of our data structures and algorithms. Our system provides menu and keyboard commands that may be used to toggle display of the: 1) input polygons (red), 2) source point (white), 3) receiver point (purple), 4) boundaries of the spatial subdivision (gray), 5) pyramidal beams (green), 6) image sources (cyan), and 7) reverberation paths (yellow). The system also supports visualization of acoustic metrics (e.g., power, clarity, etc.) computed for a set of receiver locations on a regular planar grid displayed with a textured polygon. Example visualizations are shown in Figures 18-20.

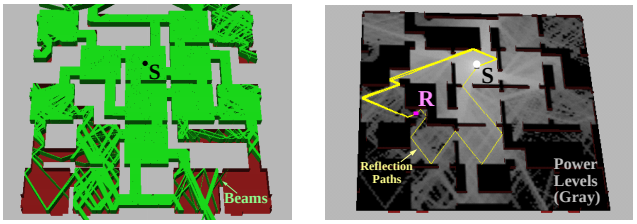


Figure 18: Eighth-order specular reflection beams (left) and predicted power levels (right) in Maze model.

Of course, many commercial [7, 8, 40] and research systems [38, 47] provide elaborate tools for visualizing computed acoustic metrics. The critical difference in our system is that it supports continuous interactive updates of reverberation paths and debugging information as a user moves the receiver point with the mouse. For instance, Figures 18 and 20 show eighth-order specular reflection paths (yellow lines) from a single audio source (white points) to a receiver location (purple points) which can be updated more than six times per second as the receiver location is moved arbitrarily. The user may select any reverberation path for further inspection by clicking on it and then independently toggle display of reflecting cell boundaries, transmitting cell boundaries, and the associated set of pyramidal beams for the selected path.

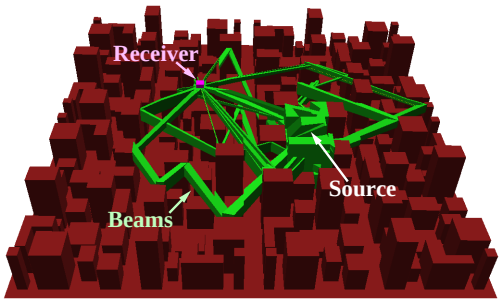


Figure 19: Beams (green) containing all eighth-order specular reflection paths from a source to a receiver in City model.

Separate pop-up windows provide real-time display of other useful visual debugging and acoustic modeling information. For instance, one popup window shows a diagram of the beam tree

data structure. Each beam tree node is dynamically colored in the **d**iagram according to whether the receiver point is inside its associated beam (white) or cell (green). Another popup window shows a plot of the impulse response representing the reverberation paths from source to receiver (see Figure 20). A third popup window shows values of various acoustic metrics, including power, clarity, reverberation time, and frequency response. All of the information displayed is updated in real-time as the user moves the receiver interactively with the mouse.

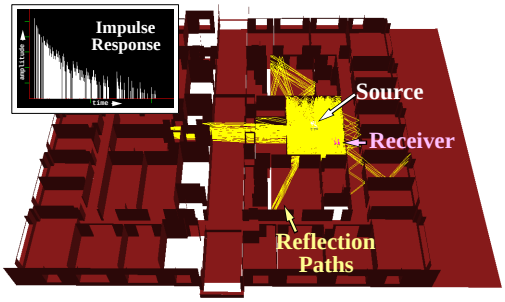


Figure 20: Impulse response (inset) derived from eighth-order specular reflection paths (yellow) in Floor model.

6 Future Work

6.1 System Extensions

Our system could be extended in many ways. For instance, the beam tracing algorithm is well-suited for parallelization, with much of the previous work in parallel ray tracing directly applicable [4]. Also, the geometric regions covered by each node of the beam tree could be stored in a single hierarchical spatial structure (e.g., a BSP), allowing logarithmic search during path generation, rather than linear search of the beams inside a single cell. HRFT (Head-Related Transfer Functions) directional filtering, angle-dependent and frequency-dependent acoustic properties of absorption, and source directivity should be included in our acoustical models. Of course, we could also use beam trees to allow a user to manipulate the acoustic properties of individual surfaces of the environment interactively with real-time feedback, like parameterized ray tracing [44].

6.2 Moving Sources

In order to support acoustic modeling in real-time, our current approach is to fix the position of each sound source and to precompute and store potential reverberation paths from that position to all points in space (i.e., the beam tree) so that reverberation paths to a specific receiver can be generated quickly. This method achieves interactive performance by trading real-time processing for storage and precomputation time. Yet, it requires that each sound source be stationary, which is not adequate to support all virtual environment applications (e.g., multi-user chat). In order to extend our method to support real-time acoustic modeling for virtual environments with moving sound sources, one approach is to precompute and store beam trees for a finite number of source locations (e.g., on a grid), and then derive the impulse response for any arbitrary source location via interpolation.

A second approach is to rework our beam tracing algorithm (i.e., the second phase, which currently executes as a preprocessing step) to execute in real-time at interactive rates. Although real-time beam tracing requires improvement of one or two orders of magnitude (beam tracing times for our test models ranged from 0.8 to 49 seconds for eigth-order specular reflections), we are optimistic that this is possible. In contrast to our current beam tracing precomputation, which must consider potential receivers at any point in space, *a real-time beam tracing algorithm must compute reverberation paths only to a specific set of known receiver positions* (e.g., the locations of other avatars in a multi-user chat). Therefore, we can implement a far more efficient beam tracing algorithm by employing aggressive path pruning methods and importance heuristics [45] to trace only beams that represent (psychoacoustically) significant reverberation paths between some source-receiver pair. Bi-directional beam tracing (computing kth-order reflections by combining beams traced up to $k/2$ reflections from both the source and the receiver positions [23, 24]) should also improve performance. We plan to experiment with these techniques and to incorporate moving sound sources into our system in the near future.

6.3 Simulation Verification

Verification of our simulation results by comparison to measured data is an important topic for further discussion. Unlike sound rendering systems for animation in virtual environments [48, 53], we aim to simulate room impulse responses accurately enough to be used also for architectural and concert hall design applications.

Although we do not present verification results in this paper due to space limitations, it is useful to note that our current system computes (more efficiently) the same specular reflection paths as the source image method, for which verification results have been published [56]. We are currently making impulse response measurements for verification of our simulations in the Varechoic Chamber, a specially constructed acoustics facility that allows one to vary the reverberation time by more than a factor of 10 by adjusting the acoustic reflection coefficient of 384 individually computer controllable acoustic panels [58].

6.4 Psychoacoustics Experiments

Perhaps the most interesting direction of future work is to investigate the possible applications of *interactive* acoustic modeling. What can we do with interactive manipulation of acoustic model parameters that would be difficult to do otherwise?

As a first application, we hope to build a system that uses our interactive acoustic simulations to investigate the psychoacoustic effects of varying different acoustic modeling parameters. Our system will allow a user to interactively change various acoustics parameters with real-time auralization and visualization feedback. With this interactive simulation system, it may be possible to address psychoacoustic questions, such as “how many reflections are psychoacoustically important to model?,” or “which surface reflection model provides a psychoacoustically better approximation?" Moreover, we hope to investigate the interaction of visual and aural cues on spatial perception. We believe that the answers to such questions are of critical importance to future designers of 3D virtual environment systems.

7 Conclusion

We have described a system that uses beam tracing data structures and algorithms to compute high-order specular reflection and transmission paths from static sources to a moving receiver at interactive rates for real-time auralization in large virtual environments.

As compared to previous acoustic modeling approaches, our beam tracing method takes unique advantage of *precomputation* and *convexity*. Precomputation is used twice, once to encode in the spatial subdivision data structure a depth-ordered sequence of (cell boundary) polygons to be considered during any traversal of space, and once to encode in the beam tree data structure the region of space reachable from a static source by sequences of specular reflections and transmissions at cell boundaries. We use the convexity

of the beams, cell regions, and cell boundary polygons to enable efficient and robust computation of beam-polygon and beam-receiver intersections. As a result, our method is uniquely able to: 1) support evaluation of reverberation paths at interactive rates, 2) scale to compute high-order reflections in large environments, and 3) extend to compute paths of diffraction and diffuse reflection.

Our virtual environment system integrates real-time auralization with visualization of large virtual environments. Based on our initial experiences with this system, we believe that accurately spatialized audio is a very important cue for experiencing and navigating virtual environments. We are continuing this research in order to further investigate the perceptual interaction of visual and acoustics effects and to better realize the opportunities possible with interactive acoustic modeling.

Acknowledgements

The authors thank Arun C. Surendran and Michael Gatlin for their valuable discussions and contributions to the project. We are also grateful to Bob Kubli who helped record audio for the accompanying video tape.

References

[1] Ahnert, Wolfgang. EARS Auralization Software. *J. Audio Eng. Soc.*, 41, 11, November, 1993, 894-904.

[2] Allen, J.B., Berkley, D.A. *Image Method for Efficiently Simulating Small-Room Acoustics*, J. Acoust. Soc. Am., 65, 4, April, 1979, 943–951.

[3] Amanatides, J. Ray Tracing with Cones. *Computer Graphics* (SIGGRAPH 84). 18, 3, 129-135.

[4] Arvo, J. and D. Kirk. A Survey of Ray Tracing Acceleration Techniques. in *An Introduction to Ray Tracing*, Andrew Glassner editor, Academic Press, San Diego, CA, 1989.

[5] Baumgart, Bruce G. *Winged Edge Polyhedron Representation.* Ph.D. Thesis, Computer Science Department, Stanford University, 1972.

[6] Borish, Jeffrey. Extension of the Image Model to Arbitrary Polyhedra. *J. Acoust. Soc. Am.*, 75, 6, June, 1984, 1827-1836.

[7] *Bose Modeler*, Bose Corporation, Framingham, MA. http://www.bose.com.

[8] *CATT-Acoustic*, CATT, Gothenburg, Sweden, http://www.netg.se/ catt.

[9] Chuang, J.H. and S.A. Cheng. Computing caustic effects by backward beam tracing. *The Visual Computer*, 11, 3, 1995, 156–166.

[10] Cook, Robert, L., Thomas Porter, and Loren Carpenter. Distributed Ray Tracing. *Computer Graphics* (SIGGRAPH 84). 18, 3, 137-146.

[11] D'Antonio, Peter, and John Konnert. The Directional Scattering Coefficient: Experimental Determination. *J, Audio Eng. Soc.*, 40, 12, December, 1992, 997-1017.

[12] Dadoun, N., D.G. Kirkpatrick, and J.P. Walsh. Hierarchical Approaches to Hidden Surface Intersection Testing. *Graphics Interface '82*, Toronto, Canada, May, 1982, 49-56.

[13] Dadoun, N., D.G. Kirkpatrick, and J.P. Walsh. The Geometry of Beam Tracing. *Proceedings of the Symposium on Computational Geometry*, Baltimore, June, 1985, 55-71.

[14] Durlach, N.I., R.W. Pew, W.A. Aviles, P.A. DiZio, and D.L. Zeltzer. *Virtual Environment Technology for Training (VETT)*. Report No. 7661, Bolt, Beranek, and Newmann, Cambridge, MA, 1992.

[15] Durlach, N.I, and A.S. Mavor, editors, *Virtual Reality Scientific and Technological Challenges*, National Research Council Report, National Academy Press, Washington, D.C., 1995.

[16] Fortune, Steve. Algorithms for Prediction of Indoor Radio Propagation. *Technical Memorandum*, Document #11274-960117-03TM, Bell Laboratories, 1996. A partial version of this paper appears in *Applied Computational Geometry, Towards Geometric Engineering*, proceedings of the *FCRC '96 Workshop* in conjunction with *WACG '96*, Philadelphia, PA, May, 1996, 157-166.

[17] Foster, S.H., E.M. Wenzel, and R.M. Taylor. Real-time Synthesis of Complex Acoustic Environments. *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, 1991.

[18] Fuchs, H. Kedem, Z., and Naylor, B. On Visible Surface Generation by a Priori Tree Structures. *Computer Graphics* (Proc. SIGGRAPH '80), 124-133.

[19] Fujomoto, Akira. Turbo Beam Tracing - A Physically Accurate Lighting Simulation Environment. *Knowledge Based Image Computing Systems*, May, 1988, 1-5.

[20] Ghazanfarpour, G. and J. Marc Hasenfratz. A Beam Tracing with Precise Antialiasing for Polyhedral Scenes. *Computer & Graphics*, 22, 1, 1998.

[21] Goral, Cindy M., Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modeling the Interaction of Light Between Diffuse Surfaces. *Computer Graphics* (Proc. SIGGRAPH '84), 18, 3, July, 1984, 213-222.

[22] Haines, Eric A. Beams O' Light: Confessions of a Hacker. *Frontiers in Rendering Course Notes, SIGGRAPH '91*, 1991.

[23] Heckbert, Paul, and Pat Hanrahan. Beam Tracing Polygonal Objects. *Computer Graphics* (SIGGRAPH 84), 18, 3, 119-127.

[24] Heckbert, Paul. Adaptive Radiosity Textures for Bidirectional Ray Tracing. *Computer Graphics* (SIGGRAPH 90), 24, 4, 145-154.

[25] Heinz, R. Binaural Room Simulation Based on an Image Source Model with Addition of Statistical Methods to Include the Diffuse Sound Scattering of Walls and to Predict the Reverberant Tail *J. Applied Acoustics*, 38, 2-4, 1993, 145-160.

[26] Hodgson, M. Evidence of Diffuse Surface Reflections in Rooms. *J. Acoust. Soc. Am.*, 89, 1991, 765-771.

[27] Jones, C.B. A New Approach to the 'Hidden Line' Problem. *The Computer Journal*, 14, 3 (August 1971), 232-237.

[28] Kajiya, James T. The Rendering Equation. *Computer Graphics* (SIGGRAPH 86), 143-150.

[29] Keller, Joseph B. Geometrical Theory of Diffraction. *Journal of the Optical Society of America*, 52, 2, February, 1962, 116-130.

[30] Kleiner, Mendel, Bengt-Inge Dalenback, and Peter Svensson. Auralization – An Overview. *J. Audio Eng. Soc.*, 41, 11, November, 1993, 861-875.

[31] Kreuzgruber, P., P. Unterberger, and R. Gahleitner. A Ray Splitting Model for Indoor Radio Propagation Associated with Complex Geometries. *Proceedings of the 1993 43rd IEEE Vehicular Technology Conference*, 1993, 227-230.

[32] Kristiansen, U.R., A. Krokstad, and T. Follestad. Extending the Image Method to Higher-Order Reflections. *J. Applied Acoustics*, 38, 2-4, 1993, 195-206.

[33] Krockstadt, U.R. *Calculating the Acoustical Room Response by the Use of a Ray Tracing Technique*, J. Sound and Vibrations, 8, 18, 1968.

[34] Kuttruff, Heinrich *Room Acoustics*, 3rd Edition, Elsevier Science, London, England, 1991.

[35] Lehnert, Hilmar. Systematic Errors of the Ray-Tracing Algorithm. *J. Applied Acoustics*, 38, 2-4, 1993, 207-221.

[36] Lewers, T. A Combined Beam Tracing and Radiant Exchange Computer Model of Room Acoustics. *J. Applied Acoustics*, 38, 2-4, 1993, 161-178.

[37] McGrath, David, and Andrew Reilly. Convolution Processing for Realistic Reverberation. The 98th Convention of the Audio Engineering Society, February, 1995.

[38] Monks, Michael, Byong Mok Oh, and Julie Dorsey. Acoustic Simulation and Visualization using a New Unified Beam Tracing and Image Source Approach. *Meeting of the Audio Engineering Society*, November, 1996.

[39] Moore, G.R. *An Approach to the Analysis of Sound in Auditoria.* Ph.D. Thesis, Cambridge, UK, 1984.

[40] Naylor, G.M. ODEON - Another Hybrid Room Acoustical Model. *J. Applied Acoustics*, 38, 2-4, 1993, 131-144.

[41] Naylor, B.F. Constructing Good Partitioning Trees. *Graphics Interface '93*, Toronto, CA, May, 1993.

[42] Rajkumar, A., B.F. Naylor, and L. Rogers. Predicting RF Coverage in Large Environments using Ray-Beam Tracing and Partitioning Tree Represented Geometry. *Wireless Networks*, 1995.

[43] Rindel, J.H. Modelling the Angle-Dependent Pressure Reflection Factor. *J. Applied Acoustics*, 38, 2-4, 1993, 223-234.

[44] Sequin, Carlo, and Eliot Smyrl. Parameterized Ray Tracing. *Computer Graphics* (SIGGRAPH 89), 23, 3, 307-314.

[45] Smits, Brian, James R. Arvo, and David H. Salesin. An Importance-Driven Radiosity Algorithm. *Computer Graphics* (SIGGRAPH 92), 26, 2, 273-282.

[46] Stephenson, U., and U. Kristiansen. Pyramidal Beam Tracing and Time Dependent Radiosity. *Fifteenth International Congress on Acoustics*, Tapir, June, 1995, 657-660.

[47] Stettner, Adam, and Donald P. Greenberg. Computer Graphics Visualization for Acoustic Simulation. *Computer Graphics* (SIGGRAPH 89), 23, 3, 195-206.

[48] Takala, Tapio, and James Hahn. Sound Rendering. *Computer Graphics* (SIGGRAPH 92), 26, 2, 211-220.

[49] Teller, Seth J., and Carlo H. Séquin, Visibility Preprocessing for Interactive Walkthroughs. *Computer Graphics* (SIGGRAPH 91), 25, 4, 61-69.

[50] Teller, Seth J. Computing the Antiumbra Cast by an Area Light Source. *Computer Graphics (Proc. SIGGRAPH '92)*, 26, 2 (August 1992), 139-148.

[51] Teller, Seth J. *Visibility Computations in Densely Occluded Polyhedral Environments.* Ph.D. thesis, Computer Science Division (EECS), University of California, Berkeley, 1992. Also available as UC Berkeley technical report UCB/CSD-92-708.

[52] Tsingos, Nicolas, and Jean-Dominique Gascuel. A General Model for Simulation of Room Acoustics Based On Hierarchical Radiosity. Technical Sketches, *SIGGRAPH 97 Visual Proceedings*, 1997.

[53] Tsingos, Nicolas, and Jean-Dominique Gascuel. Soundtracks for Computer Animation: Sound Rendering in Dynamic Environments with Occlusions. *Graphics Interface '97*, Kelowna, May 21-23, 1997, 9-16.

[54] Veach, Eric, and Leonidas J. Guibas. Metropolis Light Transport. *Computer Graphics* (SIGGRAPH 97), 65-76.

[55] Vian, J.P. and D. van Maercke. Calculation of the Room Response Using a Ray Tracing Method. *Proceedings of the ICA Symposium on Acoustics and Theater Planning for the Performing Arts*, Vancouver, CA, 1986, 74-78.

[56] Vorlander, M. International Round Robin on Room Acoustical Computer Simulations. *Proceedings of the 15th International Congress of Acoustics*, Trondheim, Norway, June, 1995.

[57] Walsh, John P., and Norm Dadoun. What Are We Waiting for? The Development of Godot, II. *presented at the 103rd Meeting of the Acoustical Society of America*, Chicago, April, 1982.

[58] Ward, William C., Gary, W. Elko, Robert A. Kubli, and W. Craig McDougald. The New Varechoic chamber at AT&T Bell Labs. *Proceeding of Wallace Clement Sabine Centennial Symposium*, Acoustical Society of America, New York, June, 1994, 343-346.

[59] Watt, Mark. Light-Water Interaction Using Backward Beam Tracing. *Computer Graphics* (SIGGRAPH 90), 24, 377-385.

[60] Weiler, K. and P. Atherton. Hidden Surface Removal Using Polygon Area Sorting. *Computer Graphics* (SIGGRAPH 77), 11, 2, 214-222.

[61] Whitted, Turner. An Improved Illumination Model for Shaded Display. *Communications of the ACM*, 23, 6, June, 1980, 343-349.