ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2020
J. Bender and T. Popa
(Guest Editors)

Volume 39 (2020), Number 8

Interactive sound propagation for dynamic scenes using 2D wave simulation

Matthew Rosen[1], Keith W. Godin[2], Nikunj Raghuvanshi[3]

[1]DigiPen Institute of Technology [2]Microsoft Mixed Reality [3]Microsoft Research

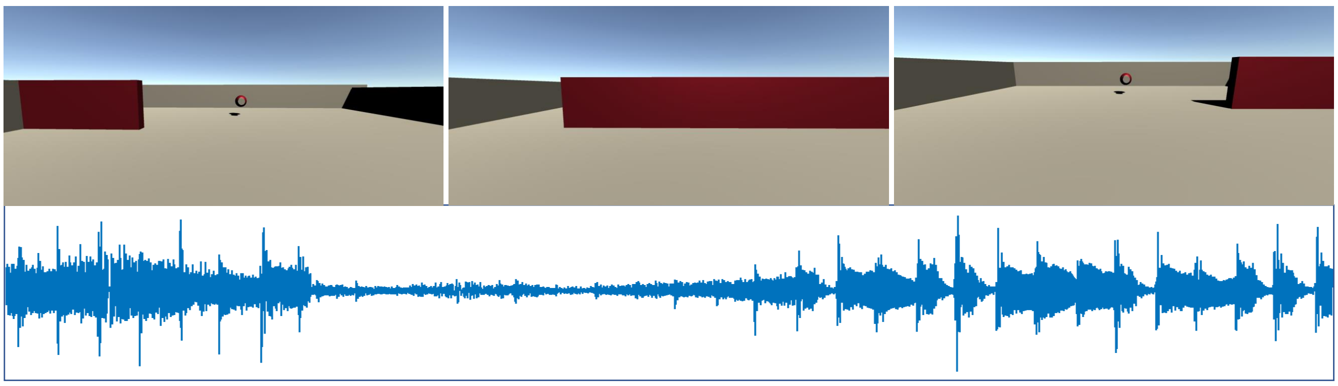

Figure 1: *Our system performs live 2D wave simulation at interactive rates to capture variation in acoustic effects from arbitrary dynamic changes in scene geometry. In the above scene, the moving wall shown in red moves from left to right, obstructing the source in the center of the image. Our system renders the smooth variation in loudness, as seen in the waveform at bottom.*

Abstract

We present a technique to model wave-based sound propagation to complement visual animation in fully dynamic scenes. We employ 2D wave simulation that captures geometry-based diffraction effects such as obstruction, reverberation, and directivity of perceptually-salient initial sound at the source and listener. We show real-time performance on a single CPU core on modestly-sized scenes that are nevertheless topologically complex. Our key ideas are to exploit reciprocity and use a perceptual encoding and rendering framework. These allow the use of low-frequency finite-difference simulations on static scene snapshots. Our results show plausible audio variation that remains robust to motion and geometry changes. We suggest that wave solvers can be a practical approach to real-time dynamic acoustics. We share the complete C++ code of our "Planeverb" system.

CCS Concepts

• ***Computing methodologies*** → ***Physical simulation; Virtual reality;*** • ***Applied computing*** → ***Sound and music computing;***

1. Introduction

Immersion in interactive experiences requires sound propagation rendering that is consistent with visual animation. In game and VR applications, a typical scene can consist of many sound sources. Sources close to the listener should be heard clearly, while those in adjacent rooms or other spaces should be faint, heard redirected around obstructions or through portals, and with increased reverberance. Such effects are important for grounding the player by conveying events in the world that persist beyond line of sight, similar to everyday experience. Unlike visuals, audible sounds do not cut off abruptly when obstructed, owing to their large wavelength, instead *diffracting* (bending) around obstructions and portal edges. In interactive applications, scenarios are commonplace where high-order diffraction around numerous scene features is the dominant mode of direct and indirect energy transport from source to listener.

Interactive sound propagation must thus model diffraction in order to produce a plausible rendering: one that exhibits smooth spatial variation in acoustic effects as many sources, listener, and scene geometry move, and robustly avoids jarring loudness jumps on visual (dis-)occlusion. And it must do so in a single CPU core or less. These challenges are quite distinct from room acoustics research where using all resources of one or a few machines for a walk-through rendering is acceptable [Sch11]. Further, the scene topology is simple: a single hall, with source and listener in line of sight, or nearly so. Therefore, absence of diffraction does not prohibit plausible rendering in room acoustics. Rather, it is desired to boost accuracy [BAA*19]. With these practical considerations, ray-tracing methods that start with a diffraction-less geometric approximation are standard in room acoustics [Vor07]. Our concerns are in sharp contrast: we require diffraction for plausibility in visually occluded cases, while trading off accuracy for performance.

Wave field (Eulerian) solvers are thus a natural starting point for our problem: they model diffraction of all orders while remaining robust and computationally insensitive to scene or path complexity. But they are costly, scaling as $D^{\eta}\nu_m^{\eta+1}$ where D is the scene diameter, η is the number of spatial dimensions, and ν_m is the maximum simulated frequency. Precomputed techniques [RS14, RS18] move expensive simulation to an offline stage (with $\nu_m \sim 500-1000$Hz) which allows fraction-of-a-core CPU usage at runtime, enabling adoption in games and VR [RTS17, GRSR18]. But dynamic scenes are ruled out, which is our focus.

Dynamic wave simulation approaches in the past have treated near-field propagation [WQLJ18], 2D wind instruments [AR15], and mid-frequency room acoustics [Sav10]. These systems are either not real-time or consume an entire GPU, far beyond our budget. Importantly, they directly solve for the complete sound field, including the signals emitted by sound sources. This requires expensive full-bandwidth simulations with $\nu_m = 22$kHz, otherwise rendered audio lacks high frequencies, sounding muffled. Further, geometry must be smoothly updated at every solver step to avoid audible clicks. The required careful numerical treatment increases solver complexity and incurs large CPU overhead as typical solver update rate is orders-of-magnitude larger than visual frame rate.

To address these computational challenges, we borrow ideas from precomputed parametric systems [RS14, RS18, CRG*20]. Rather than simulating wideband sound fields with immersed dynamic geometry, we take static snapshots of the scene configuration at interactive rates (10Hz). Each tick, we perform low-frequency ($\nu_m = 275$Hz) pulsed simulation using a second-order finite-difference time-domain (FDTD) solver for the geometry snapshot. This yields the impulse response for each source, which is converted to a set of perceptual acoustic parameters that compactly encode acoustic transport properties, independently from source signals. These parameters are: propagation delay, directivity of initial sound at source and listener, loudness of initial (direct) and reflected (indirect) sound, and reverberation decay time.

The parameters for each source are then applied to its radiated signal using a fast multi-source rendering pipeline borrowing from [RS14, CRG*20]. In particular, we incorporate source directivity and perform temporal interpolation of parameters that ensures artifact-free output on geometry motion. Frequency extrapolation is performed implicitly by taking parameters computed over limited bandwidth and applying them over the full audible bandwidth. This has the effect of applying transport behavior of low frequencies to high frequencies. While this sacrifices accuracy, it retains fast computation of plausible, smooth output as we show in our results. A further practical benefit of our parametric approach is that sound designers can modify the acoustic parameters on the fly to exert artistic control on simulated output [GGR19].

To afford scalable rendering for many sound sources, we avoid performing per-source simulation by exploiting acoustic reciprocity. We show that a single monopole simulation from the listener location suffices. Extracting listener directivity in a reciprocal framework requires additional dipole simulations [RS18], which quadruples simulation cost. We avoid this cost by proposing a fast technique that performs gradient descent on the simulated time-delay field. While it cannot model directional reverberation effects of the dipole approach [RS18], we are able to extract the perceptually-dominant [LCYG99] direction of initial sound.

Finally, we restrict to 2D, by taking a horizontal slice of the 3D scene at the current height of the listener's head. During rendering, we perform 2D-to-3D correction for distance attenuation. Although our current system cannot model elevation effects such as propagation between building floors, many practical scenarios such as indoor floor-plans and outdoor scenes lend well to our approach.

Overall, we demonstrate the first technique to capture salient 2D wave-diffraction effects like smooth obstruction and source/listener directivity on dynamic scenes within a single-core CPU budget. Our results show that wave-based methods, traditionally thought too expensive for real-time dynamic acoustics, may in fact be a promising avenue for robust immersive audio cues in games and VR in the future. A complete C++ implementation of our system is available as "Project Planeverb": `https://bit.ly/planeverb`.

2. Related Work

We target fully dynamic scenes which places significant constraints on auralization techniques [Sch11]. Precomputing data structures such as beam trees [FCE*98], edge-visibility graphs [SMM14], or impulse response fields [RS14] accelerates runtime computation, allowing moving sources and listener, but cannot be applied to fully dynamic geometry. The computational challenge has motivated heuristic methods in games [NL16].

Room acoustics research extensively employs ray-based geometric approximation of acoustic transport for providing reverberation cues. Recent approaches [PASV14] allow reverberation responsive to dynamic room size. However, our emphasis on obstruction requires diffraction modeling. Robust and fast diffraction modeling in geometric methods is a challenging open problem [SS15]. A recent approach to modeling diffraction within geometric acoustics with some dynamics is presented in [RSR*18]. Although generality is suggested, the technique is limited to modeling dynamic but separable objects, such as people or cars, whose individual diffraction response ("kernel") is pre-computed and composed in real-time via ray-tracing. The approach requires dynamic objects to be in each other's far-field so their diffraction effects may be accounted separately. For instance, dynamic portals cannot be modeled. Our approach avoids both the constraint to separable objects and the requirement for pre-computation, instead allowing arbitrary dynamic changes to the global scene. But our generality comes with the current computational limitation to two dimensional simulation.

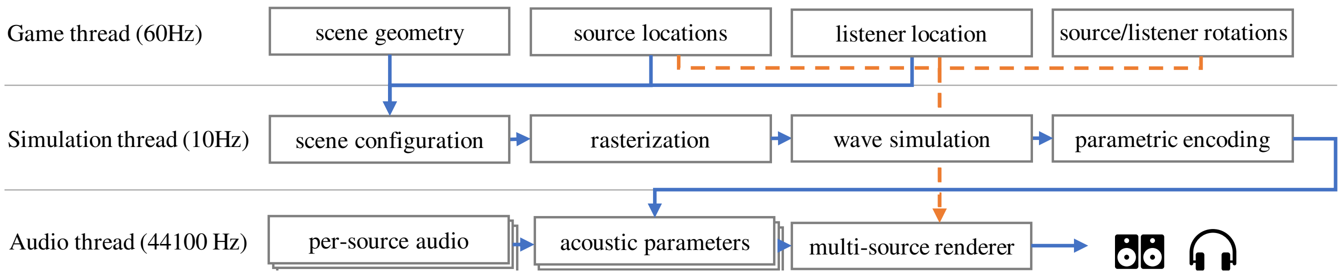

Figure 2: *System architecture. Execution flow is shown with blue lines. Acoustic simulation updates at 10Hz, taking scene configuration from game thread, performing wave simulation to compute acoustic parameters, that are interpolated and rendered in the audio thread. Dashed lines indicate data sent directly from game thread to audio thread, namely, source and listener locations and rotations.*

Real-time wave based approaches in 3D have been proposed but require significant system resources. [Sav10] demonstrated a wave-based auralization in 3D that made full use of a GPU, and [SCM18] demonstrated a wave simulation in 3D that relied on distributed computing. We assume that only a single core is available for acoustic simulation. This is representative of a typical game or VR experience; system resources must be shared with other compute.

In contrast to these approaches, our system meets the real-time threshold through two optimizations: impulse response parametrization, and constraining the simulation to 2D. Two-dimensional wave simulations have been used as proxy for 3D simulations to reduce computational cost [KVS06, OM14], but for offline room acoustic analysis rather than real-time dynamic acoustics we consider. Real-time 2D simulations have been applied to synthesis of wind instrument sounds [AR15] and speech [ZVA*16] but require full-bandwidth simulations consuming a full GPU.

3. System Architecture

Our system architecture is sketched in Fig. 2. The acoustics system (middle row) runs in a background thread allocated to a single core that is constantly looping. In each iteration, it accesses the game thread to collect the scene configuration: current scene geometry, source locations, and listener head location. In our implementation, geometry is represented with axis-aligned bounding boxes. This is done as a practical expedient rather than being a fundamental limitation of our technique. For instance, one could in principle use the object collision volumes, or even visual meshes if the copying from game to simulation thread does not get prohibitive.

The geometry is then rasterized onto a 2D slice of the scene whose height is at the listener's head. Each occupied cell is filled with the corresponding material's wide-band pressure reflectivity. The grid resolution is determined by the solver, as described later. The grid dimensions are fixed at 25x25m in our tests as that spans our test scenes and allows the simulation thread to update at a mean update rate of 10Hz on a single core. For handling larger scenes, our implementation could be easily changed to move the simulation domain horizontally as well to keep the listener at its center. This would come at negligible additional cost. We note that although we restrict to one core for the intended application, we have observed close to linear scaling up to 8 cores.

The next steps are described in the following sections. As detailed in Section 4, acoustic simulation is performed from the listener on the grid. Then, perceptual acoustic parameters are extracted from the wave field at each source location, discussed in Section 5. The updated parameters are sent to the audio thread for rendering which additionally incorporates the current source and listener rotations, detailed in Section 6.

Latency Note that our architecture ensures that the audio system latency is unaffected by acoustic simulation, as the audio thread operates as usual without blocking on the simulation. For example, audio-visual lip synchronization of a speaking character will be maintained. Rather, the response of propagation effects, such as loudness variation from moving geometry, will have a latency of about 100ms as that is how long it takes us to update each source's acoustic parameters. This is quite close to the perceptual threshold of 70ms for noticing latency in acoustic effects [BSM*04].

4. Wave simulation

We solve the wave equation as a coupled first-order system relating pressure $p(x,t)$ and normalized particle velocity $v(x,t)$:

$$\frac{\partial p}{\partial t} + c\nabla\cdot\vec{v} = 0 \tag{1}$$

$$\frac{\partial \vec{v}}{\partial t} + c\nabla p = 0, \tag{2}$$

where $c = 343m/s$ is the speed of sound. We normalize so that $\vec{v} = \rho c\vec{u}$ where $\vec{u}$ is the physical particle velocity, and $\rho = 1.2041Kg/m^3$ is the mean density of air. This brings $\vec{v}$ into the same scale-space as p: for a plane wave $|p| = |\vec{v}|$.

We solve using a rectilinear finite-difference time-domain (FDTD) scheme in two dimensions. At each solver invocation, the simulation thread provides the solver with a 2D occupancy grid at the listener's head height on which the solver operates. Each grid cell contains the tuple $\{p,\vec{v},\beta,R\}$. Yee's staggered grid is employed: p samples the cell's center, v_x its left edge (smaller X), and v_y its bottom edge (smaller Y). Only the pressure and velocity update each time-step, as the world geometry is held static during simulation. Following [AR15], β is an indicator function which is 1 if the cell is open and 0 if occupied. Lastly, R is the acoustic reflectivity of the cell. It is 0 for air and at open boundaries at the edges of the simulated domain.

4.1. Discretization and performance

The key free parameter in our approach to control accuracy-performance trade-off is the maximum modeled frequency ν_m. The practical question is whether it can be made high enough for plausible results, while being low enough to allow real-time performance. The minimum modeled wavelength is then $\lambda_m = c/\nu_m$. The spatial discretization cell size is then determined as: $\Delta x = \lambda_m/s$ where s is the number of spatial samples per smallest wavelength, or, "points per wavelength" with Nyquist limit at $s = 2$. In practice this value needs to be larger. FDTD schemes exhibit dispersion as the primary numerical error. The practical impact is that pulse responses contain significant ringing, which can be conflated with reverberation. Values of $s = 4-8$ are common for accurate simulation. We use a slightly smaller value of $s = 3.5$ as it was found to produce similar encoded output as larger values but at a significant acceleration.

Finally, being an explicit scheme, the time-step must be determined as $\Delta t = \mathcal{C}\Delta x/c$, where the CFL criterion: $\mathcal{C} < 1/\sqrt{2}$ in two dimensions provides an upper bound on Δt to ensure stability. In practice, including boundary handling, we found our solver implementation required only a slightly smaller value of $\mathcal{C} = 1/1.5$. In all our tests, we set $\nu_m = 275$Hz, which results in cell size $\Delta x = 0.36$m and update rate $1/\Delta t = 1443.75$Hz. The frequency was determined as follows. We first fixed domain size as $D = 25$m, to ensure participating geometry in a modestly sized region around the player is represented. We then tuned ν_m so that computational time was limited to $\sim$100ms. An important consideration was also that the resulting cell size should be small enough to resolve typical scene features like obstructions and portals. Reducing ν_m much further resulted in unconvincing results due to cells becoming too large.

Simulation duration is determined based on the required duration of impulse response for extracting decay time as discussed later. A duration of 0.25s was found sufficient in practice to give converged estimates in our tests. Accounting for propagation delay to the corners of the domain, we set the simulated duration as: $T_{sim} = 0.25 + D/(\sqrt{2}c)$.

Source pulse To compute a band-limited impulse response field, we introduce a Gaussian pulse at the grid cell occupied by the listener as a "soft" source by adding the following to the cell pressure at each time-step:

$$P_s(t) = a_s \exp[-(t-2\sigma)^2/\sigma^2],\ \sigma = 2/(\pi\nu_m) \tag{3}$$

This results in a wideband Gaussian magnitude spectrum with peak at 0Hz, falling off smoothly to -35dB at ν_m. Smaller values of σ were also tested, but resulted in stronger high-frequency content beyond ν_m which caused significant dispersion errors. The amplitude a_s is fixed empirically to ensure initial wavefront energy $E_{DS} = 1$ (defined in Section 5.2) at a distance of 1m in free field.

4.2. Update equations

The solver's discrete update equations are:

$$\begin{aligned}
p^+ &= \beta\left(p - \mathcal{C}\tilde{\nabla}\cdot\vec{v}\right)\\
v_x^+ &= \beta\beta_<(v_x - \mathcal{C}\tilde{\nabla}_x p^+) + (\beta_< - \beta)(\beta Y_< + \beta_< Y)(\beta p^+ + \beta_< p_<^+)\\
v_y^+ &= \beta\beta_\vee(v_y - \mathcal{C}\tilde{\nabla}_y p^+) + (\beta_\vee - \beta)(\beta Y_\vee + \beta_\vee Y)(\beta p^+ + \beta_\vee p_\vee^+)
\end{aligned} \tag{4}$$

where $*^+$ denotes updated values for next time-step, the spatial position, $\vec{x}$ is suppressed, the subscripts $\{<,\vee\}$ denote accessing the corresponding field at locations $\{(x-\Delta x, y), (x, y-\Delta y)\}$ respectively. Numerical gradient is denoted by $\tilde{\nabla}$, using backward finite difference so that $\tilde{\nabla}_x(p^+) = p^+ - p_<^+$. Numerical divergence using forward finite difference is denoted '$\tilde{\nabla}\cdot$'. The indicator function β and relative acoustic admittance Y are sampled at cell centers providing all necessary geometric information. For an air cell, $\beta = 1, Y = 1$. For a solid cell, $\beta = 0, Y = (1-R)/(1+R)$ where R is the pressure reflectivity of the material [Kut00]. We use $R = 0.97$ corresponding to rough concrete in all our test examples.

We simplify the fast, branch-free method of [AR15], observing that $\beta \in \{0, 1\}$ without intermediate values in our case, as geometry is held fixed during every solver run. Equation 4 reduces to the standard FDTD update equations for air cells and approximates the impedance boundary condition $\vec{v} = -Yp\hat{n}$ otherwise, where $\hat{n}$ is the (axis-aligned) surface normal pointing from surface into domain. To see this, first observe that the pressure update reduces to the usual FDTD update for air ($\beta = 1$) while inside solids it sets $p^+ = 0$ (any value can be set without affecting results). Next, consider v_x. Velocities are sampled on cell edges so one must account for the two cells sharing the edge, and consider all four combinations of air and solid. If both are air, the first FDTD update term is in effect, as desired. The second term handles all other cases. If both cells are solid, we safely set $v_x = 0$. That leaves the two boundary cases. The first factor of $(\beta_< - \beta)$ accounts for normal direction, the second selects the admittance of the solid cell, and the last factor selects pressure in the air cell. Similar holds for v_y.

5. Parametric Encoding

The simulation outputs a wave field $\{p(\vec{x}, t), \vec{v}(\vec{x}, t)\}$ that we then encode in terms of perceptual parameters. Due to our use of reciprocity, the field represents the response at the listener at $\vec{x}_l$ due to any possible source location, $\vec{x}$. The game thread provides a list of current source locations $\{\vec{x}_i\}$ that are projected onto the simulation grid plane. The following describes the encoding process for each of these source locations. Our current implementation performs this encoding for the entire wave field. However, as will be seen shortly, in principle only the propagation delay field is required for extracting listener directivity. The remaining parameters could be extracted only at the source locations, $\{\vec{x}_i\}$. For notational convenience, we use continous time below, actual operations are performed on band-limited signals sampled at the solver time-step with integrals implemented with rectangular quadrature.

5.1. Propagation delay, τ

The propagation delay $\tau(\vec{x}_i)$ is the time in seconds at which the first wavefront arrives at the listener from source at $\vec{x}_i$. In the following, the source position $\vec{x}_i$ is assumed unless noted otherwise. Numerical noise in FDTD travels faster than sound, so propagation delay estimation must discriminate between numerical noise and a (possibly highly attenuated) first wavefront. Following [RS14], for a measured pressure function $p(t)$ at the source location (due to reciprocity) we found a simple threshold detector sufficed:

$$\tau = \arg\min_t \left(10\log_{10} p^2(t) > G_\tau\right) \tag{5}$$

We use $G_\tau = -110$ dB. More robust detectors can be easily substituted if required, such as in [RS18].

5.2. Direct sound energy, E_{DS}

While we use the term 'direct sound' as it is common in room acoustics, it is better understood as the first-arriving sound, which

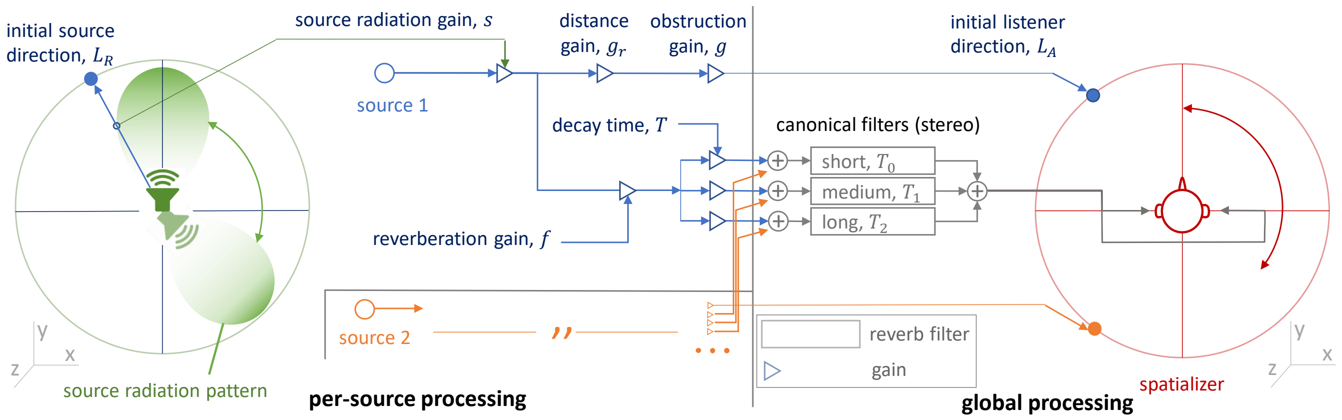

Figure 3: *Run-time signal processing. We borrow the scalable multi-source rendering from [RS14] via "canonical filters" for reverberation, and model source directivity on direct sound as in [CRG*20].*

may arrive attenuated by a diffracted path. We estimate in a fixed window after first arrival:

$$E_{DS} = \int_{\tau}^{\tau_{DS}} p^2(t)\, dt, \tag{6}$$

with $\tau_{DS} = \tau + 10\,\text{ms}$. This integration interval is based on the best-case echo fusion threshold for human listening [LCYG99].

5.3. Reflections energy, E_{RS}

As with E_{DS} we use a fixed window:

$$E_{RS} = \int_{\tau_{DS}}^{\tau_{RS}} p^2(t)\, dt, \tag{7}$$

$\tau_{RS} = \tau_{DS} + 80\,\text{ms}$, a typical value from room acoustics [Gad07].

5.4. Decay time, T_{60}

We estimate the reverberation decay time, T_{60} using linear regression on the log-backward Schroeder integral [Sch65]:

$$l(t) = 10\log_{10}\int_{t}^{T_{sim}} p^2(t')\, dt' \tag{8}$$

In practice $l(t)$ can be evaluated efficiently by accumulating back-to-front in the pressure array sampling $p(t)$. Linear regression is performed on the restricted signal $l(t), t \in (\tau_{DS}, T_{sim} - 10ms)$. Recall that T_{sim} is the duration of the simulated impulse response. We discard the last 10ms of the integral which dips towards $-\infty$, and also remove the direct peak.

5.5. Arrival direction, $\vec{L_A}$

Recall that using reciprocity, the pulse is emitted from the listener location, $\vec{x_l}$. Having first calculated the propagation delay field $\tau(\vec{x})$, we start at the source location $\vec{x_i}$ and perform gradient descent on propagation delay by following the eikonal, $-\nabla\tau(\vec{x})$. The descent terminates when the current cell, $\vec{x_e}$, has line-of-sight to the listener: $c\tau(\vec{x_e}) - d_{los} < 0.3\lambda_m$, where the line-of-sight distance, $d_{los} = |\vec{x_e} - \vec{x_l}|$. Recall that λ_m is the shortest modeled wavelength. The arrival direction at the listener is then calculated as: $\vec{L_A} = (\vec{x_e} - \vec{x_l})/d_{los}$. Terminating descent on line-of-sight saves compute, especially for unoccluded sources. It also allows fast descent that only hops to a cell's immediate neighbors based on largest component of the gradient rather than following the true gradient direction which can go off-grid, requiring interpolation. We employ the former, showing that it works well in practice.

5.6. Radiation direction, $\vec{L_R}$

The shortest path leaves the source in a radiation direction before arriving at listener. We borrow the technique from [CRG*20]. Due to reciprocity, $\vec{L_R}$ can be estimated as opposite to the direction from which the first wavefront radiating from the listener during simulation arrives at the source location, $\vec{x_i}$. The wavefront direction can be estimated using energy flux density. Thus,

$$\vec{L_R} = -\frac{\vec{I_R}}{\|\vec{I_R}\|}, \vec{I_R} = \int_{\tau}^{\tau_R} p\vec{v}, \quad \tau_R = \tau + 5\,\text{ms}. \tag{9}$$

6. Rendering

Figure 3 shows the signal flow of our rendering system. Each sound source's processing is split into rendering its direct and reflected sound which are summed at the end.

6.1. Parameter conversion

The dynamically computed acoustic parameters E_{DS}, E_{RS}, $\vec{L_R}$, $\vec{L_A}$, and T_{60} capturing acoustic transfer from the source to listener are first used to derive intermediate signal processing parameters g, f, s, and T as defined below.

Obstruction gain, g is computed from the direct sound energy E_{DS} as:

$$g = \max\left(\sqrt{rE_{DS}}, D_g\right) \tag{10}$$

where r is the Euclidean distance between source and listener and D_g is a threshold to prevent totally occluded sources from being unheard. We use a value of $D_g = 0.0316$ ($-30\,\text{dB}$). This compensates for a deficiency of 2D wave simulation: in outdoor scenes, paths going over walls aren't captured at all. At the same time D_g is conservatively small, so that the results remain reasonable in indoor scenes. Multiplying by r removes the $1/r$ energy attenuation of 2D propagation. As a result, this gain captures the energy attenuation due to presence of geometry, and $g = 1$ throughout space in the absence of geometry.

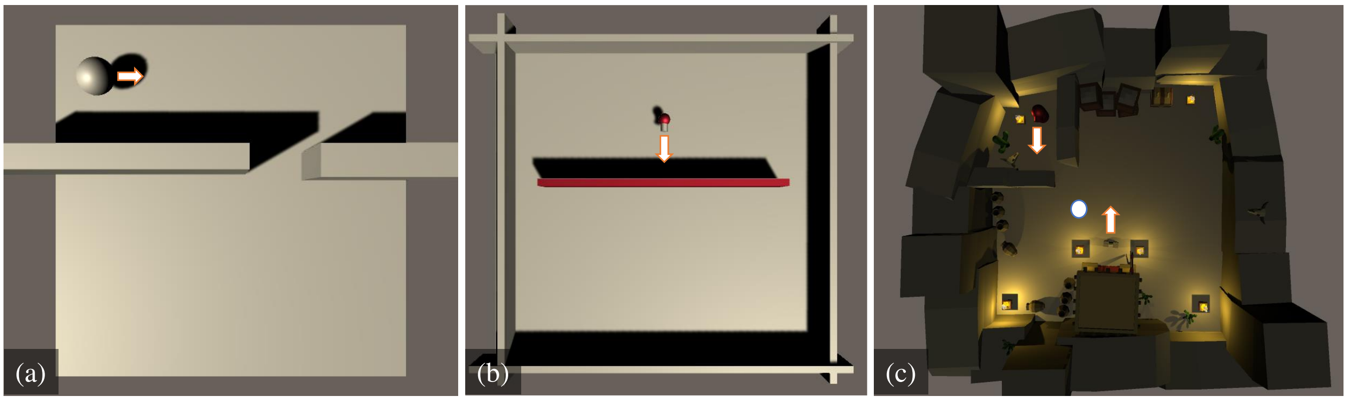

Figure 4: *Top views of evaluation scenes with arrows marking sound source directions: (a) Static walls with portal (b) Four static walls with dynamic segment (in red) (c) Game scene with dynamic door.*

Reflections gain, f is computed from reflections energy E_{RS} as:

$$f = \sqrt{E_{RS}}. \tag{11}$$

In some cases, especially with small spaces, f can be quite large with our 2D simulation, with unrealistic reverberance: in two dimensions, reverberant energy is unable to travel vertically to reflect from a high ceiling, or escape entirely to the sky in outdoor scenes. Investigating accurate scene-dependent 2D-to-3D correction for reverberation, perhaps parameterized on the ceiling's height and reflectivity (with 0 "ceiling" reflectivity outdoors) is an important avenue for future work.

The gains g and f are linearly interpolated over a 10.6 ms window (512 samples at a 48 kHz sampling rate) to scale the source's signal, yielding smooth renderings in our tests.

Source radiation direction, $\vec{L_R}$ is then used to sample a source radiation pattern to compute a broadband source radiation gain s. In the general case this radiation pattern can be specified by a designer or drawn from measurements or simulations of radiation objects; for this work we use a cardioid pattern:

$$s = \frac{1}{2}\left(1 + \vec{L_R} \cdot \vec{L_S}\right) \tag{12}$$

where L_S is the forward pointing vector of the source provided by the game thread. General frequency-dependent radiation patterns could be straightforwardly included in our approach by a combination of lookup tables and fast equalization filters as in [CRG*20].

Rendered decay time, T is computed from physical decay time, T_{60} by clamping to the bounds of the reverberation bank described in Section 6.3 as:

$$T = \min(\max(T_{60}, T_0), T_{max}) \tag{13}$$

with $[T_0, T_{max}]$ spanning the bank's decay times.

Design control could be exercised at this stage by performing in-place transformations that modify g, f, and T, as described in [GGR19]. Such aesthetic control can have significant importance in practical applications like games and can also help mitigate some of the issues with 2D simulation discussed above.

6.2. Direct sound rendering

As illustrated at the top of Fig. 3, having removed the effect of 2D free-field decay in computing g, we now apply 3D free-field amplitude decay as gain $g_r = 1/r$, then we apply source directivity gain, s and obstruction gain, g. The resulting signal is then spatialized with Vector Base Amplitude Panning (VBAP, [Pul01]) in the world direction $\vec{L_A}$ incorporating the current head rotation of the player. HRTF (head-related transfer function) based binaural rendering could also be easily employed instead. This processing is repeated for each sound source.

6.3. Reflections rendering

To render reflections and reverberation, we borrow the scalable multi-source rendering method from [RS14] which avoids expensive per-source convolutions. In this method, a fixed bank of reverberation ("canonical") filters is shared across sources, with increasing decay times, $\{T_i\}$. Each filter is scaled such that the early portion of the impulse response (of duration $\tau_{RS} - \tau_{DS} = 80$ms in our case) has unit energy.

The source signal is scaled by two gains $\{\alpha_j, \alpha_{j+1}\}$ and mixed into the input of two of the filters whose decay times immediately bracket the dynamic decay time for the source T, i.e., $T_j \leq T \leq T_{j+1}$. The filter mix gains are computed as [RS14]:

$$\alpha_j = f \frac{10^{-3t^*/T_{j+1}} - 10^{-3t^*/T}}{10^{-3t^*/T_{j+1}} - 10^{-3t^*/T_j}} \tag{14}$$

$$\alpha_{j+1} = f - \alpha_j \tag{15}$$

with matching parameter $t^* = 100$ ms.

Final mix: The stereo output from all reverberation filters is summed, which is then added to the per-source spatialized stereo output from direct sound processing. This produces the final signals played over headphones.

7. Results

7.1. Performance

Our system was implemented in C++ with simulation running on a single thread. For all evaluations we used a $D = 25$m simulation region with maximum frequency $\nu_m = 275$Hz and resulting cell size $\Delta x = 0.36$m. The simulation and encoding time bench-marked on a high end desktop CPU core was below 100ms. Simulation cost was ∼65% of the compute time, with the rest spent in parameter extraction. Geometry rasterization cost was negligible because of our use of axis-aligned box representation of geometry - with triangulated geometry or k-dop collision volumes, this can be expected to increase. But simulation cost is insensitive to geometric complexity, an important trait of the branch-free and uniform per-cell cost of our FDTD update in (4).

7.2. Walk-throughs

We integrated our proposed system into the Unity game engine (`http://www.unity3d.com`). We constructed three scenes in Unity, shown in Figure 4, and recorded walk-throughs. Please consult the supplementary video to view results that are discussed in the following.

Static walls with portal. The first scene, shown in Figure 4(a), comprises a single sound source and a wall with a portal. This case shows two effects. Firstly, as the listener walks away from the source past the portal, diffracted obstruction results in a smooth change in loudness, with sound direction shifting towards the door. A simple ray-based occlusion query would result in odd jumps in loudness when the source became obstructed. Secondly, notice that the loudness change on obstruction is plausibly synchronized in time with listener motion, showing that the end-to-end latency of our system is tolerable.

Moving obstruction. The second scene, shown in Figure 4(b), adds complexity with a moving obstruction between the source and listener. Live wave simulation affords the necessary diffraction effects to produce smooth results. Also note that when the source is obstructed, reverberated sound is still heard from the surrounding walls as expected. We then demonstrate source directivity with a rotating source, with the initial sound loudness depending both on the pose of the source relative to the listener, and the position of the obstruction. Even though a simple scene, such real-time directional, diffraction-based effects for dynamic general scenes have not been possible before.

Game scene: multi-source, dynamic portal. The third scene, shown in Figure 4(c), reflects part of a typical game scene, with static and dynamic scene elements showing multiple sound sources: torches, a boombox, and speech. As the boombox is rotated, its directivity is clearly audible. Next the player picks dynamic geometry (a box) and we show dynamic obstruction from it as the player crouches and jumps on top of the box. This results from our use of a horizontal slice through player head height. When the slice intercepts the box, waves propagate in 2D around the box cross section, rendering obstruction. When the player jumps onto the box, the slice does not contain a box, and the source location as projected onto the simulation plane has line of sight to the player's head. Near-hit or miss of geometry sliced by the listener plane can be a practical issue but in our tests we didn't note any perceptually jarring difficulties. We believe 3D simulation in a few-meter vertical span around the player's head could be a practical approach in the future to avoid such issues if they arise.

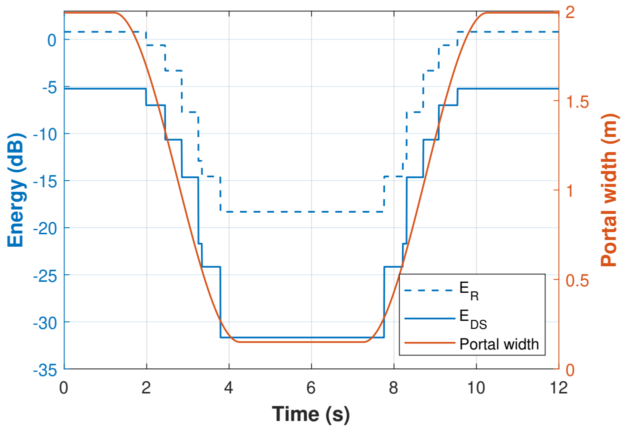

Figure 5: *Direct sound energy E_{DS} and reflections energy E_{RS} for a source and listener in the scene in Figure 4(c). A portal separating the source and listener shrinks and enlarges. The corresponding rendered gains $\{g, f\}$ employ linear interpolation over audio samples to result in smooth renderings.*

We then show the effect of a dynamic portal that causes smooth loudness changes. Figure 5 shows the direct sound energy E_{DS} and reflections energy E_{RS} for the source in the upper left of Figure 4 and listener location at the circle, just on the opposite side of the dynamic wall from the source. The energy levels are shown as a function of time while the portal opens and closes according to a scripted action. The energy curves show stair-stepping because we simulate on static snapshots of geometry at 10Hz and also due to the discrete simulation grid. Because the corresponding gains, $\{g, f\}$ are linearly interpolated over audio sample rates, the resulting rendering is smooth in our tests. This illustrates the essence of our approach: rather than expensive modeling of geometry changing at audio rates within a live solver, we do fast simulations on *static* geometry snapshots at visual rates with the resulting parameters interpolated at audio rates. Because the motion of geometry is reasonably well-sampled at 10Hz, interpolating the resulting parameters yields natural results with audio-visual synchronization, as can be heard in the supplementary video.

8. Conclusion

We have demonstrated the first technique to capture wave-diffraction effects like smooth obstruction and source and listener directivity on general dynamic scenes within a single-core CPU budget. By leveraging reciprocity, using perceptual parametrization, and constraining the wave simulation to 2D, our method scales to many sound sources while supporting integration into standard audio DSP and designer tools. Our results show that wave-based methods, traditionally thought too expensive for real-time dynamic acoustics, can in fact be a promising avenue for robust immersive audio cues in games and VR.

Much remains to be done. As discussed in Section 6.1, 2D-to-3D correction for reflections and reverberation is an important area for future improvement. Optimizations could provide computational headroom for progressing beyond 2D, such as 2.5D simulation in a constrained height around the listener, or even a full 3D simulation. By our estimate, code optimizations such as vectorization and restricting encoding to source locations could significantly accelerate our current implementation by 5 times or more. Investigating existing fast wave solvers [RNL09] or inventing new ones tailored for dynamic scenes also offer an exciting avenue. Finally, although we restricted to a single CPU core, using a small fraction of a GPU is another possible approach to practicability.

As game engines move towards increasingly dynamic worlds, we hope that our results and code motivate others to pursue wave solvers for real-time dynamic acoustic simulation.

9. Acknowledgements

The first author would like to thank Prof. Matthew Klassen of DigiPen Inst. of Tech. for leading the course "Audio Design Project III/IV" under which this project was completed.

References

[AR15] Allen A., Raghuvanshi N.: Aerophones in Flatland: Interactive Wave Simulation of Wind Instruments. *ACM Trans. Graph. 34*, 4 (July 2015). 2, 3, 4

[BAA*19] Brinkmann F., Aspöck L., Ackermann D., Lepa S., Vorländer M., Weinzierl S.: A round robin on room acoustical simulation and auralization. *J. Acoustical Soc. Am. 145*, 4 (2019), 2746–2760. 2

[BSM*04] Brungart D., Simpson B. D., McKinley R. L., Kordik A. J., Dallman R. C., Ovenshire D. A.: The interaction between head-tracker latency, source duration, and response time in the localization of virtual sound sources. In *ICAD 2004: 10th Mtg. Intl. Conf. Auditory Display* (2004), Barrass S., Vickers P., (Eds.). 3

[CRG*20] Chaitanya C. R. A., Raghuvanshi N., Godin K. W., Zhang Z., Nowrouzezahrai D., Snyder J. M.: Directional sources and listeners in interactive sound propagation using reciprocal wave field coding. *ACM Trans. Graph. 39*, 4 (July 2020). URL: https://doi.org/10.1145/3386569.3392459. 2, 5, 6

[FCE*98] Funkhouser T., Carlbom I., Elko G., Pingali G., Sondhi M., West J.: A Beam Tracing Approach to Acoustic Modeling for Interactive Virtual Environments. In *Proc. 25th Annual Conf. Comp. Graph. Interactive Techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 21–32. 2

[Gad07] Gade A.: Acoustics in Halls for Speech and Music. In *Springer Handbook of Acoustics*, Rossing T., (Ed.), 2007 ed. Springer, May 2007, ch. 9. URL: http://www.worldcat.org/isbn/0387304460. 5

[GGR19] Godin K., Gamper H., Raghuvanshi N.: Aesthetic modification of room impulse responses for interactive auralization. In *AES Intl. Conf. Immersive and Interactive Audio* (March 2019), Audio Engineering Society. 2, 6

[GRSR18] Godin K. W., Rohrer R., Snyder J., Raghuvanshi N.: Wave acoustics in a mixed reality shell. In *2018 AES Intl. Conf. Audio for Virtual and Augmented Reality (AVAR)* (August 2018). 2

[Kut00] Kuttruff H.: *Room Acoustics*, 4 ed. Taylor & Francis, 2000. 4

[KVS06] Kelloniemi A., Valimaki V., Savioja L.: Simulation of Room Acoustics using 2-D Digital Waveguide Meshes. In *IEEE Intl. Conf. Acoustics, Speech and Sig. Proces. (ICASSP)* (2006). 3

[LCYG99] Litovsky R. Y., Colburn S. H., Yost W. A., Guzman S. J.: The precedence effect. *J. Acoustical Soc. Am. 106*, 4 (1999), 1633–1654. 2, 5

[NL16] Neumann T., Lawlor S.: Overwatch - the elusive goal: Play by sound. Talk at Game Developer's Conference (GDC) 2016, 2016. URL: https://www.gdcvault.com/play/1023317/Overwatch-The-Elusive-GoalPlay. 2

[OM14] Oxnard S., Murphy D.: Achieving realistic auralisations using an efficient hybrid 2d multi-plane fdtd acoustic model. In *Proc. of the EAA Joint Symposium on Auralization and Ambisonics* (2014). 3

[PASV14] Pelzer S., Aspöck L., Schröder D., Vorländer M.: Interactive real-time simulation and auralization for modifiable rooms. *J. Bldg. Acoustics* (2014). 2

[Pul01] Pulkki V.: *Spatial Sound Generation and Perception by Amplitude Panning Techniques*. PhD thesis, Helsinki U. of Tech., 2001. 6

[RNL09] Raghuvanshi N., Narain R., Lin M. C.: Efficient and Accurate Sound Propagation Using Adaptive Rectangular Decomposition. *IEEE Trans. Vis. and Comp. Graphics 15*, 5 (2009), 789–801. 8

[RS14] Raghuvanshi N., Snyder J.: Parametric Wave Field Coding for Precomputed Sound Propagation. *ACM Trans. Graph. 33*, 4 (July 2014). URL: http://dx.doi.org/10.1145/2601097.2601184, doi:10.1145/2601097.2601184. 2, 4, 5, 6

[RS18] Raghuvanshi N., Snyder J.: Parametric directional coding for precomputed sound propagation. *ACM Trans. Graph. 37*, 4 (July 2018), 108:1–108:14. URL: http://doi.acm.org/10.1145/3197517.3201339, doi:10.1145/3197517.3201339. 2, 4

[RSR*18] Rungta A., Schissler C., Rewkowski N., Mehra R., Manocha D.: Diffraction kernels for interactive sound propagation in dynamic environments. *IEEE Trans. Vis. and Comp. Graph. 24*, 4 (April 2018), 1613–1622. doi:10.1109/TVCG.2018.2794098. 2

[RTS17] Raghuvanshi N., Tennant J., Snyder J.: Triton: Practical pre-computed sound propagation for games and virtual reality. *J. Acoustical Soc. Am. 141*, 5 (2017), 3455–3455. URL: https://doi.org/10.1121/1.4987164, arXiv:https://doi.org/10.1121/1.4987164, doi:10.1121/1.4987164. 2

[Sav10] Savioja L.: Real-Time 3D Finite-Difference Time-Domain Simulation of Mid-Frequency Room Acoustics. In *13th Intl. Conf. Digital Audio Effects (DAFx-10)* (Sept. 2010). 2, 3

[Sch65] Schroeder M. R.: New method of measuring reverberation time. *J. Acoustical Soc. Am.* (1965). 5

[Sch11] Schröder D.: *Physically Based Real-Time Auralization of Interactive Virtual Environments*. Logos Verlag, Dec. 2011. URL: http://www.worldcat.org/isbn/3832530312. 2

[SCM18] Saarelma J., Califa J., Mehra R.: Challenges of distributed real-time finite-difference time-domain room acoustic simulation for auralization. In *Conf. on Spatial Reproduction* (2018). 3

[SMM14] Schissler C., Mehra R., Manocha D.: High-order Diffraction and Diffuse Reflections for Interactive Sound Propagation in Large Environments. *ACM Trans. Graph. 33*, 4 (July 2014). URL: http://dx.doi.org/10.1145/2601097.2601216, doi:10.1145/2601097.2601216. 2

[SS15] Savioja L., Svensson U. P.: Overview of geometrical room acoustic modeling techniques. *J. Acoustical Soc. Am. 138*, 2 (Aug. 2015), 708–730. URL: http://dx.doi.org/10.1121/1.4926438, doi:10.1121/1.4926438. 2

[Vor07] Vorländer M.: *Auralization: Fundamentals of Acoustics, Modelling, Simulation, Algorithms and Acoustic Virtual Reality (RWTHedition)*, 1 ed. Springer, Nov. 2007. URL: http://www.worldcat.org/isbn/3540488294. 2

[WQLJ18] Wang J.-H., Qu A., Langlois T. R., James D. L.: Toward wave-based sound synthesis for computer animation. *ACM Trans. Graph. 37*, 4 (July 2018), 109:1–109:16. URL: http://doi.acm.org/10.1145/3197517.3201318, doi:10.1145/3197517.3201318. 2

[ZVA*16] Zappi V., Vasuvedan A., Allen A., Raghuvanshi N., Fels S.: Towards real-time two-dimensional wave propagation for articulatory speech synthesis. In *Proc. Mtgs. Acoust.* (2016). 3