3D Geometrical Acoustics Simulation Final Report

Maj Bregar
Faculty of Computer and Information Science
Večna pot 113, 1000 Ljubljana
Email: mb95331@student.uni-lj.si

Abstract—**This report presents a real-time, web-based geometrical acoustics simulation implemented using WebGPU. Sound propagation is modeled as energy-carrying rays in a voxelized environment, accounting for frequency-dependent attenuation, reflection, refraction, transmission and diffraction. A GPU-based simulation is coupled with an audio processing pipeline that filters direct sound and delayed reflections from accumulated energy histograms, while visual feedback is provided through voxel face heat maps and energy graphs. Evaluation results demonstrate energy conservation and interactive performance on consumer hardware.**

I. Introduction

Geometrical acoustics models sound propagation as rays of acoustic energy traveling along straight paths. While inherently approximate, this approach enables efficient simulation of complex sound behavior with relatively low computational cost. The objective of this work is to develop an intuitive, real-time sound ray simulation capable of running at interactive frame rates (at least 30 FPS) in a web browser using WebGPU.

The proposed system models frequency-dependent sound attenuation during propagation through materials, as well as delayed reflections arising from late ray arrivals. Users may select or upload an audio source, interactively reposition the sound emitter and listener within a voxelized simulated room, and visualize energy absorption on surfaces using a heat-map representation. The input audio energy at the emitter is displayed alongside the rendered scene, while per-band gain-loss coefficients of the room are presented for inspection. The input signal is filtered according to the simulated acoustic environment and played back to the user.

II. Related Work

Several studies have explored GPU-based implementations of geometrical acoustics for sound propagation.

Ray Acoustics Using Computer Graphics Technology [1] presents a non-real-time ray acoustics system in which acoustic energy is represented using octave-band vectors. Rays are emitted from the listener and traced through the scene using the texels of a cube map. This work forms the primary conceptual foundation of our approach, particularly with respect to ray-based energy representation and frequency-band modeling.

Guided Multiview Ray Tracing for Fast Auralization [2] introduces a guidance algorithm that dynamically steers geometric acoustic ray tracing toward perceptually relevant sound propagation paths. The method efficiently computes early specular reflections and first-order diffraction on the GPU.

A web-based implementation of geometric acoustics is presented in *Web-based Geometric Acoustic Simulator* [3], demonstrating the feasibility of running ray-based acoustic simulations in a browser environment.

III. Methodology

The project pipeline consists of a preprocessing step and three repeating sequential steps. First, the user-uploaded audio file is preprocessed into energy vectors and stored for use during runtime. The energy vectors are individually passed to the compute shader every frame for use in the simulation. The output data from the shader is then passed to the audio engine, which computes the room loss coefficients and ray delays then uses those to filter the original audio. The absorbed energy information of each face is passed to the visualization pipeline, where it is drawn to the screen.

A. Simulation

Before running the simulation, the audio signal is preprocessed using a short-time Fourier transform (STFT). A time-indexed sequence of octave-band energy vectors is derived by integrating the magnitude-squared spectrum over predefined octave-band frequency ranges. During runtime, the appropriate energy vector is selected from this precomputed sequence based on the current simulation time. For our implementation, we used the FFT library by Fedor Indutny [4].

To execute the simulation on the GPU, a WebGPU compute shader is employed in which acoustic rays are initialized at the emitter position. Initial ray directions are generated deterministically using a Fibonacci lattice to achieve quasi-uniform spherical sampling. The initial energy vector assigned to each ray is obtained from the total energy vector and normalized by the total number of deployed rays. Ray propagation through the voxelized environment is simulated using a voxel-based digital differential analyzer (DDA), which advances the ray incrementally through the grid while testing for material boundary crossings at each step.

1) **Attenuation.** Ray energy attenuation is computed independently for each frequency band using the sound absorption formula 1. We compute per-band attenuation using stored material attenuation coefficients $\mu(i)$.

$$P(x + \Delta x) = P(x)\, e^{-\mu(i)\, \Delta x} \tag{1}$$

All material properties are stored in a JSON file and bound to the compute shader during pipeline initialization. Air voxel attenuation coefficients were computed for each frequency band using the atmospheric absorption data specified in *ISO 9613-1* [5]. Concrete voxel attenuation coefficients were obtained using a rough approximation derived from the experimental data reported in *Frequency-dependent stress wave attenuation in cement-based materials* [6]. The resulting energy attenuation coefficients for both materials are summarized in Table I.

Band Center (Hz)	α^{air} (m^{-1})	$\alpha^{concrete}$ (m^{-1})
31.11	1.8×10^{-5}	3.2237×10^{-2}
62.23	2.8×10^{-5}	6.4475×10^{-2}
124.80	9.8×10^{-5}	1.2932×10^{-1}
250.32	2.72×10^{-4}	2.5937×10^{-1}
500.28	5.32×10^{-4}	5.1836×10^{-1}
999.85	9.35×10^{-4}	1.0360
1999.70	2.19×10^{-3}	2.0720
3999.75	6.98×10^{-3}	4.1443
8000.21	2.49×10^{-2}	8.2894
16000.06	3.80×10^{-2}	1.6578×10^{1}

TABLE I: Energy attenuation coefficients for air and concrete as a function of band center frequency.

2) **Material boundaries.** When a ray intersects a material boundary, it is recursively split into four new rays representing specular reflection, diffuse reflection, refraction, and transmission.

   Specular reflection is computed using the law of reflection,

$$\hat{\mathbf{d}}_{\text{refl}} = \hat{\mathbf{d}} - 2\left(\hat{\mathbf{d}} \cdot \hat{\mathbf{n}}\right)\hat{\mathbf{n}}, \tag{2}$$

   where $\hat{\mathbf{d}}$ denotes the ray direction, $\hat{\mathbf{d}}_{\text{refl}}$ is the resulting unit reflected direction, and $\hat{\mathbf{n}}$ is the surface normal at the point of intersection.

   We calculate a random direction for the diffusely reflected ray by sampling a direction using a cosine-weighted hemisphere around the face normal.

   The ray refraction direction is computed according to Snell's law expressed in terms of the speed of sound,

$$\frac{\sin\alpha}{c_1} = \frac{\sin\beta}{c_2}, \tag{3}$$

   where α and β are the angles of incidence and refraction, and c_1 and c_2 denote the speed of sound in the incident and transmitted media, respectively.

   As WebGPU compute shaders do not support recursion, ray generation is implemented using a preallocated recursion stack. The maximum recursion depth and stack size are configurable parameters, allowing the user to balance simulation accuracy and performance. In cases where the recursion stack cannot contain all newly created rays, only the available number of rays are pushed, where the ones with the most energy take priority. This design enables real-time execution across a range of hardware configurations.

3) **Diffraction.** In cases where a ray intersects a material boundary near a voxel edge, an additional check is performed for neighboring air voxels adjacent to the intersection point. If an air voxel is detected, a new diffracted ray is cast at the air–material interface by sampling a cosine-weighted hemisphere oriented along the incident ray direction. The diffracted ray is assigned per-band energy, which is subtracted from the energy of the original ray. The original ray then continues its standard energy exchange computation at the material boundary.

4) **Energy exchange.** The distribution of acoustic energy among the new rays is governed by impedance-based reflection and transmission coefficients. The acoustic impedance of a material is defined as

$$Z = \rho c, \tag{4}$$

   where ρ is the material density and c is the speed of sound inside it. The energy reflection coefficient is given by

$$R = \left(\frac{Z_2 - Z_1}{Z_2 + Z_1}\right)^2, \tag{5}$$

   with the corresponding transmission coefficient defined as

$$T = 1 - R. \tag{6}$$

   For each frequency band i, the incident ray energy E_i is partitioned into reflected and transmitted components using the broadband reflection and transmission coefficients R and T. The total reflected and transmitted energies are given by

$$E_i^{\text{refl}} = R\,E_i, \qquad E_i^{\text{trans}} = T\,E_i. \tag{7}$$

   The reflected energy is further subdivided into specular and diffuse components according to the frequency-dependent diffusion coefficient d_i,

$$E_i^{\text{spec}} = (1 - d_i)\,E_i^{\text{refl}}, \qquad E_i^{\text{diff}} = d_i\,E_i^{\text{refl}}, \tag{8}$$

   where E_i^{spec} and E_i^{diff} denote the energies carried by the specularly and diffusely reflected rays, respectively.

   The transmitted energy is partitioned into refracted and direct transmission components using a frequency-dependent refraction coefficient r_i. If refraction is physically admissible, the transmitted energies are defined as

$$E_i^{\text{refr}} = r_i\,E_i^{\text{trans}}, \qquad E_i^{\text{trans,direct}} = (1-r_i)\,E_i^{\text{trans}}. \tag{9}$$

   This formulation ensures strict energy conservation for each frequency band,

$$E_i = E_i^{\text{spec}} + E_i^{\text{diff}} + E_i^{\text{refr}} + E_i^{\text{trans,direct}}. \tag{10}$$

5) **Audio output.** At the beginning of each ray step, rays are tested for proximity to the listener position. Rays that fall within a small predefined radius are considered audible. Their corresponding traveled path lengths are discretized

and accumulated into an energy histogram. Both the listener energy histogram and the voxel face absorbed energy buffer are transferred back to the CPU after each simulation frame.

B. Audio Processing

The received listener energy histogram is divided into two sections within the audio engine.

1) **Direct audio.** The first section consists of a 30 ms interval following the earliest received energy and is used to estimate the direct energy transmission of the room. For each frequency band i, a band-limited amplitude loss coefficient is computed as

$$c_i = \sqrt{\frac{E_i^{\text{input}}}{E_i^{\text{output}}}}, \tag{11}$$

where E_i^{input} denotes the input energy of band i and E_i^{output} is the energy sum received within the time window. This formulation follows from the assumption that signal amplitude is proportional to the square root of acoustic energy. The resulting coefficient c_i therefore represents an amplitude factor that accounts for energy loss during propagation.

2) **Reflected audio.** The remaining portion of the histogram is used to extract delayed reflections by selecting a finite set of bins with the highest energy. Reflections are subsequently represented as a list of tuples of gain and delay.

Sound processing is implemented using the Web Audio API AudioWorkletProcessor [7]. The implementation employs a bank of 10 biquad band-pass filters to decompose the input signal into octave-band amplitude components A_i. Each band is scaled by the computed frequency-dependent loss coefficient c_i, and the filtered bands are summed to produce the direct output amplitude according to Eq. (12).

$$A_{\text{direct}} = \sum_{i=1}^{10} A_i\, c_i, \tag{12}$$

Reflected sound is synthesized by writing gain-scaled direct amplitude A_{direct} into a delay buffer, which is sampled and combined with the direct output during audio synthesis.

Since new simulation outputs are computed at every frame, it is necessary to apply smoothing to both the loss coefficients and the reflection parameters within the audio engine. This smoothing prevents audible artifacts such as zipper noise and clicks that may arise from listener or emitter movement, as well as discontinuities caused by inconsistent ray arrivals.

C. Visualization

During the final stage of the pipeline, the face color buffer is updated by mapping the absorbed energy of each face to a corresponding color value as shown in Fig. 1. The simulation room is rendered using rasterization, with lighting evaluated via per-fragment Lambertian diffuse shading and shadow mapping. The spheres representing the listener and the emitter are rendered using GPU instancing. All matrix computations before rendering are performed using the glMatrix library by Brandon Jones and Colin MacKenzie IV [8].

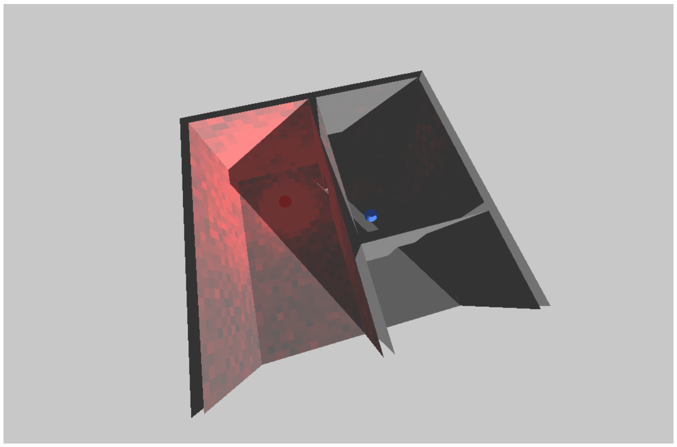
Fig. 1: Visualization of the simulated room geometry and absorbed surface energy.

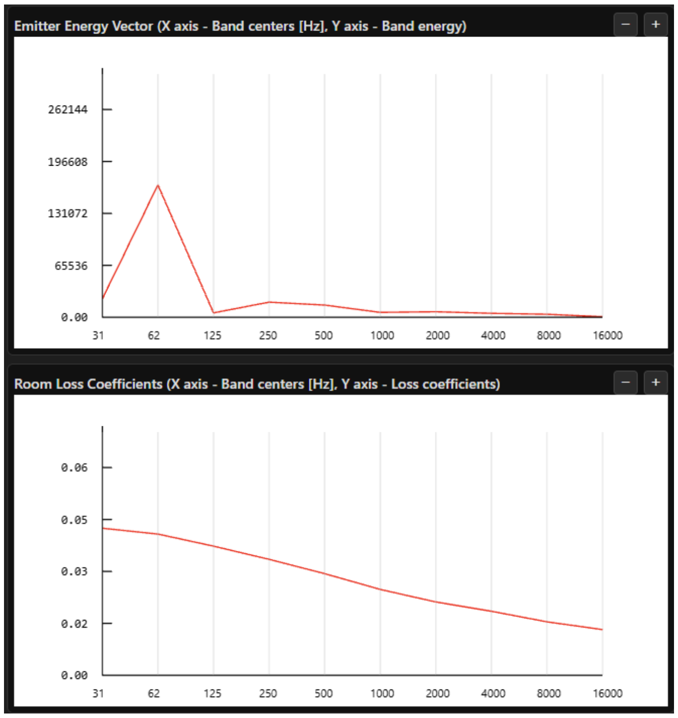

Fig. 2: Visualization of the sound energy input vector and room loss coefficients.

Finally, diagnostic plots of the input energy vectors and the computed loss coefficients are updated and rendered using the webgl-plot library by Danial Chitnis [9] as shown in Fig. 2.

IV. Discussion and Evaluation

To evaluate the proposed system, we assessed energy conservation, real-time performance relative to the initial constraints, and perceived audio realism.

1) **Performance.** We evaluated the real-time performance of the system on a high-end laptop equipped with an NVIDIA

RTX 4080 Laptop GPU, an AMD Ryzen 9 7945HX CPU, and 64 GB of memory. Because ray traversal time strongly depends on the surrounding scene geometry, the measured frame rate varied with the emitter position. With a maximum recursion depth of 3 and a recursion stack size of 12, the system achieved an average frame rate ranging from 12.3 FPS (minimum) to 56.4 FPS (maximum). Reducing the maximum recursion depth to 2 and the recursion stack size to 6 increased the minimum average frame rate to 30.1 FPS, thereby satisfying the real-time constraint of 30 FPS.

2) **Energy conservation.** During simulation runtime, we log all subtracted energy into an atomic buffer, including attenuation contributions from voxels without visible faces. We analyzed a short interval of recorded simulation frames and observed that the sum of the energy absorbed by the voxel geometry and the energy received at the listener never exceeds the input energy. This confirms that the simulation does not introduce artificial energy into the system. The sampled energy values are shown in Fig. 3. We further observe that the output energy is approximately one order of magnitude lower than the input energy, whereas an ideal system would conserve energy exactly. This discrepancy is primarily attributed to the limited recursion stack, which causes rays to be discarded once its capacity is reached. A secondary source of ray discarding is the maximum allowed step count. Increasing this limit did not lead to higher absorbed energy, indicating that recursion stack limitations are the dominant source of energy loss.

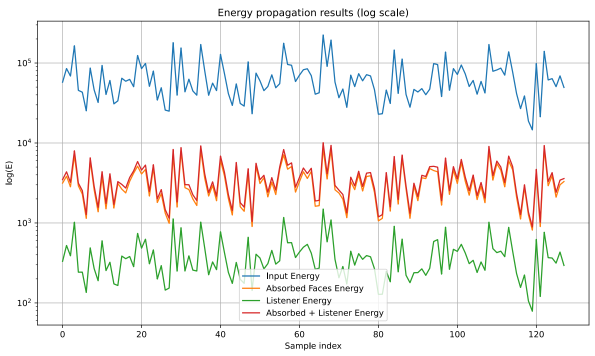

Fig. 3: Energy propagation results shown on a logarithmic scale.

3) **Audio realism.** Although the system models per-band energy loss and delayed reflection arrivals, it remains an inherently approximate and simplified representation of sound propagation. The use of rays introduces a discrete formulation, such that the received energy depends entirely on discrete ray arrivals. Our approximations of diffuse reflection and diffraction emit only a single additional ray, despite representing phenomena that are continuous in nature. Additionally, we do not model sound interference and assume total independence of each ray. Furthermore, the simulation discards all phase and angular information associated with ray arrivals. As a result, the system cannot reproduce physically accurate sound localization cues.

V. Improvements

Although this project represents a bare-bones implementation of an energy-based ray-acoustics simulation, several extensions could significantly improve both computational efficiency and physical realism.

First, a substantial performance improvement could be achieved by replacing the current single-voxel DDA traversal with a hierarchical octree-based traversal, such as the approach described in [10]. This would allow rays to skip large homogeneous regions, such as empty or uniformly filled volumes, thereby reducing the number of voxel boundary checks.

Second, the simulation could be extended to support ray casting from the listener rather than from the emitter. This modification would provide angular information at the listener position, enabling the incorporation of head-related transfer functions (HRTFs). Such an approach would also require tracking phase information throughout the simulation. The current choice of emitter-based ray casting was motivated by its intuitive visualization during the rendering phase, where absorbed energy naturally concentrates near the source.

Finally, several enhancements are possible without altering the core simulation architecture. The user interface could be extended with a room editing tool to allow interactive modification of the simulation environment. In addition, the existing JSON-based material database could be expanded to include a wider range of materials. With minimal additional changes, the system could also support a first-person navigation mode, enabling users to move through the environment in real time.

VI. Conclusion

This work presents a real-time, web-based implementation of an energy-based geometrical acoustics simulation using WebGPU. By modeling frequency-dependent sound behavior within a voxelized environment, the system enables interactive exploration and visualization of sound propagation at interactive frame rates.

Evaluation results demonstrate that the system conserves acoustic energy and can meet real-time performance constraints on consumer hardware, while highlighting the trade-offs between simulation accuracy and computational efficiency. Although the current implementation does not compete with more physically accurate sound propagation models, it provides an intuitive and extensible foundation for real-time acoustic browser-based applications.

References

[1] N. Röber, U. Kaminski, and M. Masuch, “Ray acoustics using computer graphics technology,” in *10th International Conference on Digital Audio Effects (DAFx-07), S*, 2007, pp. 117–124.

[2] M. Taylor, A. Chandak, Q. Mo, C. Lauterbach, C. Schissler, and D. Manocha, “Guided multiview ray tracing for fast auralization,” *IEEE transactions on visualization and computer graphics*, vol. 18, no. 11, pp. 1797–1810, 2012.

[3] M. Taylor and F. Meng, “Web-based geometric acoustic simulator,” in *Proceedings of the 23rd International ACM Conference on 3D Web Technology*, ser. Web3D ’18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3208806.3208817

[4] F. Indutny. Fft.js. [Online]. Available: https://github.com/indutny/fft.js

[5] *Acoustics — Attenuation of sound during propagation outdoors — Part 1: Calculation of the absorption of sound by the atmosphere*, International Organization for Standardization Std. ISO 9613-1, 1993.

[6] E. N. Landis and S. P. Shah, "Frequency-dependent stress wave attenuation in cement-based materials," *Journal of Engineering Mechanics*, vol. 121, no. 6, pp. 737–743, 1995.

[7] Audioworkletprocessor. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/AudioWorkletProcessor

[8] glmatrix. [Online]. Available: https://github.com/toji/gl-matrix

[9] webgl-plot. [Online]. Available: https://github.com/danchitnis/webgl-plot

[10] K. Sung, "A dda octree traversal algorithm for ray tracing," 1991.