*UDC* 004.94 *http://dx.doi.org/10.30970/vam.2024.32.12378*

# RAY TRACING BASED APPROACH FOR THE SOUND PROPAGATION SIMULATION

**O. Terletskyy, V. Trushevskyy**

*Ivan Franko National University of Lviv,*
*1, Universytetska str., 79000, Lviv, Ukraine,*
*e-mail: oleksandr.terletskyi@lnu.edu.ua, valeriy.trushevsky@lnu.edu.ua,*

Simulating sound propagation in complex 3D environments is a computationally intensive task due to the need to account for the interaction between sound waves and various materials. In this paper, a ray tracing method that breaks sound into multiple frequency bands to accurately simulate these interactions is proposed. The Unity engine is utilized to render the 3D environment and manage the physical properties of objects, where each material influences how sound rays at different frequencies reflect or absorb upon contact.

To enhance sound rendering, the CSound library to process and synthesize audio based on the results of the frequency-based ray tracing is integrated. Additionally, a diffraction binary search method and a frequency-based path caching technique are introduced to measure decibel levels efficiently across different frequency bands, reducing computational overhead while maintaining high accuracy. This approach provides a realistic simulation of acoustic behavior in virtual spaces and can be applied to a wide range of applications requiring detailed sound analysis and real-time performance.

*Key words*: sound propagation, ray tracing method, frequency bands, Unity Engine, wave simulation, sound reflection, CSound library.

## 1. Introduction

Simulating sound propagation in complex 3D environments is a computationally demanding task. The challenge lies in accurately modeling how sound interacts with different surfaces, materials, and objects while accounting for reflections, absorption, and frequency-dependent behavior. Traditional methods like finite element simulation [1] or adaptive rectangular decomposition [10] often struggle to balance the need for real-time performance with acoustic accuracy, particularly when dealing with the complexities of high-frequency sound interactions [2].

Several existing tools aim to address these challenges. Meta XR Audio SDK offers a robust framework for acoustic ray tracing in virtual environments, designed specifically for use with the Unity engine. This SDK allows developers to simulate realistic audio effects, such as sound reflections and diffraction, in VR applications [3]. Similarly, NVIDIA VRWorks Audio provides advanced solutions for real-time ray tracing, leveraging GPU acceleration to handle complex audio propagation in large scenes, which is crucial for applications requiring real-time feedback with minimal latency [4].

In this paper, a ray tracing method that builds upon these ideas by breaking down sound into multiple frequency bands is proposed. Each band is treated independently to better simulate how sound waves of different frequencies propagate and reflect in a virtual environment. The Unity engine is utilized to render the 3D environment, enabling real-time interactions between sound rays and scene geometry. Each object within the environment is assigned material properties that dictate how rays of various frequencies reflect, absorb, or transmit based on the material's acoustic characteristics.

To achieve high-quality audio rendering, the CSound library is integrated, which synthesizes sound based on the results of our frequency-based ray tracing. CSound allows for fine control over sound synthesis, making it possible to render the complex acoustic interactions that occur in the simulation. Additionally, a center ray method and a frequency-based path caching system are introduced that optimize the calculation of decibel levels across frequency bands. This approach significantly reduces computational overhead while maintaining a high degree of accuracy in simulating how sound behaves in different environments.

## 2. Ray Tracing

The propagation of sound waves in a medium is described by the Helmholtz equation, derived from the wave equation in the frequency domain [5]:

$$\nabla^2 p(\mathbf{r}) + k^2 p(\mathbf{r}) = 0, \tag{1}$$

where $p(\mathbf{r})$ is the sound pressure at position $\mathbf{r}$, and $k$ is the wave number, given by:

$$k = \frac{2\pi f}{c}. \tag{2}$$

Here, $f$ is the frequency of the sound wave, and $c$ is the speed of sound in the medium. For air, $c \approx 343\,\mathrm{m/s}$. The solution to the Helmholtz equation allows us to calculate the sound pressure field in the environment, but ray tracing simplifies this by treating sound as rays traveling through space.

Ray tracing is a technique used to simulate the propagation of sound in an environment by treating sound waves as rays [2]. These rays travel in straight lines and interact with surfaces and objects by reflecting, refracting, and diffracting. In this section, the mathematical principles governing these interactions are described, which allow us to simulate realistic acoustic behavior. As the complexity of the environment increases, maintaining high performance while accurately simulating these interactions requires the use of efficient computational techniques.

Monte Carlo integration is a statistical method used to estimate integrals [13], particularly in cases where the integral is too complex or expensive to compute analytically. In the context of acoustic ray tracing, Monte Carlo methods are used to estimate the contributions of sound rays by sampling multiple paths and averaging the results. This approach is particularly useful for handling complex environments where sound interacts with many objects, reflections, and diffractions, while balancing accuracy and performance.

Monte Carlo integration estimates the value of an integral by randomly sampling points in the integration domain and taking the average value of the function at those points. Given an integral of the form:

$$I = \int_{\Omega} f(x)\,dx, \tag{3}$$

where $\Omega$ is the domain of integration and $f(x)$ is the function being integrated, Monte Carlo integration approximates the integral as:

$$I \approx V(\Omega)\frac{1}{N}\sum_{i=1}^{N} f(x_i), \tag{4}$$

where $V(\Omega)$ is the volume of the domain $\Omega$, $N$ is the number of random samples, and $x_i$ are the randomly sampled points within $\Omega$.

In acoustic ray tracing, $f(x)$ represents the sound intensity contributed by a ray, and $x_i$ corresponds to a random path or ray emitted from the sound source. By averaging the contributions of many rays, Monte Carlo integration provides an approximation of the total sound intensity at a given point (such as the listener's position).

In acoustic simulations, the sound field at a listener's position is influenced by direct sound, reflections, and diffractions. The sound intensity at the listener's position $I_{\text{total}}$ can be modeled as an integral over all possible ray paths:

$$I_{\text{total}} = \int_{\Omega} I(\mathbf{r})\, d\mathbf{r}, \tag{5}$$

where $I(\mathbf{r})$ is the sound intensity along ray path $\mathbf{r}$, and $\Omega$ is the space of all possible ray directions and paths. Monte Carlo integration is used to approximate this integral by sampling a set of rays and calculating their contributions.

The accuracy of Monte Carlo integration improves as the number of samples $N$ increases. The error in the approximation decreases proportionally to $\frac{1}{\sqrt{N}}$, which means that the accuracy improves slowly with the number of samples. However, Monte Carlo integration is particularly useful in scenarios where exact calculations are too complex or costly, such as in environments with numerous sound reflections and obstacles [13].

To ensure the method converges to a realistic estimate of the sound field, the variance in the estimated sound intensity can be tracked as more rays are sampled. As the variance decreases, the estimate becomes more accurate.

## 2.1. Reflection, Absorption and Diffraction

When a ray encounters a surface, it can reflect according to the law of reflection [7]. Mathematically, the angle of incidence $\theta_i$ is equal to the angle of reflection $\theta_r$:

$$\theta_i = \theta_r. \tag{6}$$

The reflected ray direction $\mathbf{r}$ can be computed using the incident ray direction $\mathbf{i}$ and the surface normal $\mathbf{n}$:

$$\mathbf{r} = \mathbf{i} - 2(\mathbf{i} \cdot \mathbf{n})\mathbf{n}. \tag{7}$$

This equation (7) calculates the new direction of the ray after it reflects off a surface.

When sound encounters a surface, some of the sound energy is absorbed, reducing the intensity of the reflected sound [8]. The intensity of the reflected sound is determined by the absorption coefficient $\alpha$, which ranges from 0 (no absorption) to 1 (complete absorption):

$$I_{\text{reflected}} = (1 - \alpha) I_{\text{incident}}, \tag{8}$$

where $I_{\text{incident}}$ is the intensity of the incoming sound wave, and $I_{\text{reflected}}$ is the intensity after reflection. Surfaces with higher absorption coefficients, like carpets or curtains, absorb more sound energy.

Diffraction refers to the bending of sound waves around obstacles [9]. The intensity of the diffracted sound can be modeled using Maekawa's empirical formula for diffraction over barriers [12], which is widely used in environmental acoustics and noise control. According to Maekawa, the diffraction loss $L_d$ is influenced by the size of the obstacle and the wavelength of the sound.

$$L_d = 10 \log_{10} \left( 3 + 20 \left( \frac{\lambda}{D} \right) \right), \tag{9}$$

where $\lambda$ is the wavelength of the sound and $D$ is the effective size of the obstacle. This formula accounts for how the size of the obstacle relative to the wavelength influences the extent to which sound waves are bent around the barrier. Lower frequencies (longer wavelengths) are diffracted more easily, while higher frequencies experience greater attenuation when encountering large obstacles.

## 2.2. Frequency Banding

In acoustic simulations, sound waves consist of a wide range of frequencies, each interacting with the environment in a unique way. Frequency banding refers to the process of splitting the full frequency spectrum into discrete bands, allowing each band to be simulated independently. This approach is crucial for modeling the frequency-dependent behavior of sound, including how different frequencies reflect, absorb, and refract when encountering objects in a 3D environment. By dividing sound into frequency bands, it was possible to simulate more accurately how higher frequencies tend to be absorbed by materials, while lower frequencies are more likely to reflect and travel further distances. Each frequency band is assigned a distinct ray tracing model that accounts for the specific behavior of sound at that frequency.

## 2.3. Algorithm

The process of simulating sound propagation in complex 3D environments is outlined in Algorithm 1, which demonstrates the use of acoustic ray tracing with frequency banding and CSound for realistic sound synthesis, implemented within the Unity engine. This simulation is calculated for every frame of the real-time simulation, which is executed 60 times per second. In this method, sound is divided into multiple frequency bands, and rays are emitted from the sound source, interacting with objects in the environment through reflection, absorption, and diffraction. Rays have energy that starts from 1 and decays as they travel through the 3D environment, with a limited number of bounces before they are ignored in the simulation. By caching the paths of rays for each frequency band and synthesizing the sound with CSound, this approach efficiently calculates the decibel levels at the listener's position. The details of frequency banding and the integration of CSound will be further explored in the following sections.

**Algorithm 1** Acoustic Ray Tracing with Frequency Banding and CSound

**Input:** 3D environment with objects and materials, sound source, listener position, ray parameters (number of rays, frequency bands, etc.)
**Output:** Simulated sound at the listener's position
**Initialization:**
Initialize 3D environment and material properties
Set up sound source and listener position
Divide sound into frequency bands
Initialize ray parameters (number of rays, number of bounces, max ray length)
**for** each frame **do**
    **Ray Emission:**
    Emit rays from the sound source in all directions
    **for** each frequency band **do**
        Initialize path cache for the frequency band
        **for** each ray **do**
            **Ray Propagation:**
            Trace the ray through the environment
            **if** ray intersects with object **then**
                Calculate reflection, absorption, and transmission based on the material properties
                Update ray direction and energy based on frequency-specific properties
                **if** ray energy falls below threshold or max bounces exceeded **then**
                    Terminate the ray
                **end if**
            **end if**
        **end for**
    **end for**
    **Path Caching:**
    Store the calculated ray paths for each frequency band in the cache
    Retrieve cached paths for subsequent calculations
    **Sound Synthesis:**
    **for** each frequency band **do**
        Use CSound's filters to process the sound data for the frequency band
    **end for**
    **Decibel Measurement:**
    Calculate the decibel levels at the listener's position based on the accumulated rays from all frequency bands
    **Audio Output:**
    Output the synthesized audio using CSound
**end for**

## 2.4. Efficient Diffraction Modeling

To reduce the computational overhead of diffraction modeling, a binary search was implemented which it outlined in Algorithm 2. This algorithm accelerates the process of determining the best possible path for diffracted sound waves. The binary search efficiently narrows down potential diffraction paths by systematically dividing the search space into two halves, significantly reducing the number of calculations needed.

**Algorithm 2** Binary Search for Diffraction Modeling in Individual Ray with Obstacle Size

**Input:** Obstacle geometry, obstacle size, source position, listener position, initial ray direction, sound frequency
**Output:** Optimal diffraction angle for the ray
**Initialization:**
Set the minimum angle $\theta_{\min}$ (direct path) and maximum angle $\theta_{\max}$ (edge of the obstacle)
Set the error tolerance $\epsilon$ for convergence
Set maximum number of iterations $N_{\max}$
**Binary Search:**
**while** convergence not reached **and** iterations $< N_{\max}$ **do**
    Calculate midpoint angle: $\theta_{\text{mid}} = \frac{\theta_{\min}+\theta_{\max}}{2}$
    Calculate the diffraction path for angle $\theta_{\text{mid}}$
    Calculate listener's received sound energy for $\theta_{\text{mid}}$ based on sound frequency, obstacle size, and wavelength (9)
    **if** received sound energy for $\theta_{\text{mid}}$ is acceptable (above threshold equal of 5% of energy) **then**
        Set $\theta_{\max} = \theta_{\text{mid}}$ ▷ Move upper bound to the midpoint
    **else**
        Set $\theta_{\min} = \theta_{\text{mid}}$ ▷ Move lower bound to the midpoint
    **end if**
    Check for convergence: $|\theta_{\max} - \theta_{\min}| < \epsilon$
**end while**
**Output:** Optimal diffraction angle $\theta_{\text{opt}} = \theta_{\text{mid}}$

## 3. CSound

CSound is an open-source sound synthesis library that allows for precise control over sound generation and processing. In our acoustic ray tracing system, CSound is used to render the audio based on the simulation results of frequency-based ray tracing [15]. By leveraging CSound's flexible sound synthesis capabilities, it is possible to accurately reproduce the behavior of sound as it propagates through the environment, taking into account frequency-specific interactions with objects.

CSound provides a set of powerful tools for audio processing [16], including filters and reverberation effects, which can be tailored to match the ray tracing results for each frequency band. These tools allow for more realistic audio rendering, enhancing the overall quality of the acoustic simulation.

### 3.1. CSound filters

The `butterbp` function in CSound is a Butterworth bandpass filter that isolates specific frequency ranges from the audio signal. In our simulation, this filter is applied to the output of each frequency band, ensuring that only the relevant frequencies are processed and synthesized. The Butterworth filter is known for its flat frequency response within the passband, which helps maintain the integrity of the simulated sound.

For each frequency band, the `butterbp` filter is configured with the appropriate cutoff frequencies based on the band's range [6]. This allows for accurate filtering of the sound

data generated by the ray tracing system, preserving the specific acoustic characteristics of each frequency band.

The `alpass` function is an all-pass filter in CSound that introduces phase shifts to the audio signal without affecting its amplitude. This filter is particularly useful for simulating reverberation and delay effects, as it can create the complex phase relationships that occur when sound waves interact with surfaces and objects in an environment.

In our implementation, `alpass` is used to simulate the late reflections and reverberations that result from multiple ray interactions in the environment. By applying the all-pass filter to the synthesized audio, a realistic representation of how sound decays and reverberates over time can be generated. This enhances the spatial quality of the acoustic simulation, making it more immersive and lifelike.

## 3.2. Late Reverberations

Reverberation is an essential component of realistic acoustic simulations, representing the way sound waves persist in an environment after the original source has stopped emitting. While early reflections can be easily modeled using comb filters, late reverberations - the denser, more diffused sounds that decay gradually - require more sophisticated modeling. In our system, late reverberations are handled using a combination of all-pass filters and advanced reverb algorithms to simulate the decay of sound in a virtual environment.

Late reverberations occur after multiple reflections off surfaces in the environment. These reverberations build up as sound waves scatter across the room, gradually losing energy with each interaction. To model this, a combination of two key techniques is used:

- **Comb Filters:** Used to simulate early reflections by providing feedback loops that introduce short delays into the sound, mimicking the first few bounces of sound waves in the environment.
- **All-Pass Filters:** Applied after the comb filters, these filters modify the phase of the signal without affecting its amplitude, producing the diffused, smooth tail associated with late reverberations.

The decay of late reverberations is controlled using the `reverbTime` parameter. This value dictates how long the reverberation persists in the virtual environment, simulating the acoustic properties of different spaces:

- **Short Reverb Times:** Simulate smaller, more enclosed spaces like a room or small hall, where sound decays quickly.
- **Long Reverb Times:** Simulate larger environments like a cathedral or concert hall, where sound decays slowly over time, resulting in a longer reverb tail.

## 3.3. Csound processing pipeline

The Algorithm 3 describes the process of multiband audio processing using filters and reverberation effects to create a realistic audio environment. It processes an audio signal in real-time, adjusting each frequency band separately, applying reverberation and echo effects, and combining them to produce the final output.

The input variables include various audio parameters and the audio signal itself. The audio parameters control different aspects of the sound. Gains adjust the amplitude for each frequency band, while panning modifies the stereo balance between the left and

right channels. Frequencies define the center point of each band, and Q-factors influence the sharpness of the filters, determining how focused or broad the filtering is around the center frequency. Reverb time controls how long the reverberation effect lasts, echo density determines the intensity and frequency of echoes, and the reverb mix adjusts the balance between the original (dry) and reverberated (wet) signal. The audio signal is the sound that will be processed through these multiband and reverberation effects.

**Algorithm 3** Multiband Audio Processing and Reverberation Algorithm

**Input:**
- Set of audio parameters: gains, panning, frequencies, Q-factors (resonance), reverb time, echo density, and reverb mix.
- Audio signal for processing.

**Initialization:**
Define frequency bands for processing, ranging from low to high frequencies.
Initialize filters (low-pass, band-pass, high-pass) for each frequency band.

**Main Loop:**
**for** each audio frame **do**
    **Filtering and Frequency Band Processing:**
    **for** each frequency band (0 to 8) **do**
        Apply the appropriate filter (low-pass, band-pass, or high-pass) to the input signal.
        Adjust the amplitude of the filtered signal based on the gain value for that frequency band.
        Adjust the stereo panning of the filtered signal based on the pan value for that frequency band.
    **end for**
    **Combine Filtered Signals:**
    Sum the processed signals from all frequency bands to form a combined signal for the left and right audio channels.
    **Reverb and Echo Processing:**
    Apply a series of comb filters to the combined signal to simulate early reflections.
    Pass the result through two all-pass filters to simulate reverberation.
    Mix the reverberated signal with the original signal, adjusting the balance based on the reverb mix value.
    Add echo effects based on the echo density parameter.
    **Final Output:**
    Output the final processed signal for both the left and right audio channels.
**end for**
**End:**
Continue processing audio for the duration of the input signal.

## 4. Results

The primary goal of this work is to demonstrate the accuracy and efficiency of the proposed acoustic ray tracing system, particularly focusing on the impact of frequency banding, path caching, and late reverberations on sound simulation. The results presented in this section showcase how the system performs across various test environments, highlighting both the computational efficiency and acoustic realism.

### 4.1. Test Setup

To evaluate the performance of the system, several virtual environments with varying levels of complexity were created. These environments ranged from simple rooms with minimal geometry to complex spaces containing multiple reflective and absorptive surfaces. The tests were conducted with the following parameters:

- **Number of Rays:** 20 rays per frame.
- **Number of Bounces:** Maximum of 8 reflections per ray.
- **Frequency Bands:** Sound was divided into 9 frequency bands, each handled separately.
- **Reverb Settings:** Reverb time ranged from 0.5s (small room) to 5.0s (large hall), with varying echo densities.

The tests measured both the auditory output quality and the computational performance (e.g., frame rate and processing time) to assess the trade-off between sound realism and system load.

### 4.2. Frequency Banding Performance

One key feature of the system is its ability to simulate sound propagation across multiple frequency bands. By processing each frequency band independently, the system accurately simulates frequency-dependent behavior, such as:

- **Low-Frequency Propagation:** Lower frequencies tend to travel further and are less affected by absorption.
- **High-Frequency Reflection:** Higher frequencies are more easily absorbed and are more sensitive to reflections, creating distinct spatial audio effects.

**Observation:** For testing the acoustic properties of concrete were used to model how different surfaces affect sound behavior. The data for these coefficients were sourced from Acoustic Group's sound absorption data [17], providing reliable values for materials commonly used in acoustic simulations. This information allowed for accurate modeling of sound reflection, absorption, and transmission during the testing phase.

### 4.3. Reverberation Accuracy

Reverberation is critical for creating immersive audio environments, and our system accurately simulates both early reflections and late reverberations. The system was tested in environments of varying sizes to observe how well it could replicate realistic reverberation effects [11]. The reverberation time, $T_{60}$, is a measure of the time it takes for sound energy to decay by 60 dB in a room. It can be calculated using Sabine's formula [14]:

$$T_{60} = \frac{0.161 \cdot V}{A}, \tag{10}$$

where:

- $V$ is the volume of the room in cubic meters.
- $A$ is the total absorption of the room, calculated as the sum of the products of surface areas and absorption coefficients.

For a room with dimensions 10m x 20m x 20m:

- Volume $V = 10 \times 20 \times 20 = 4000\,\mathrm{m}^3$
- Assuming the average absorption coefficient of 0.05 for the walls and 0.1 for the ceiling and floor:

$$A = 2 \times (20 \times 20) \times 0.1 + 2 \times (20 \times 10) \times 0.05 + 2 \times (20 \times 10) \times 0.05 = 120\,\mathrm{m}^2$$

- The reverberation time is calculated as:

$$T_{60} = \frac{0.161 \times 4000}{120} \approx 5.37\,\text{seconds}. \tag{11}$$

An impulse response is a function that describes how a system reacts to a brief input signal, typically an idealized sound pulse, capturing how sound energy behaves in a space. The following Figure 1 shows an impulse response chart, which visually represents the decay of sound energy over time. The amplitude of the energy ranging from 0.0 (min intensity) to 1.0 (max intensity) is plotted against time where the initial amplitude of the energy is 1.0.

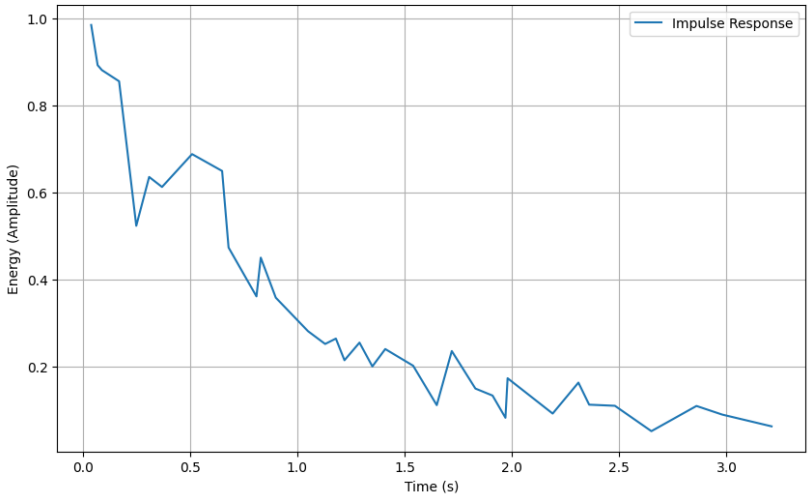


Fig. 1. Impulse Response (Amplitude) with 100 rays (less accurate)

Comparing the impulse response to the reverberation time:

- The reverberation time from the Sabine formula predicts that the sound energy decays to a negligible level in approximately 5.37 seconds (11).
- In the impulse response chart, we can observe that the sound energy amplitude decreases rapidly and continues to decay, with most of the energy dissipated by around 3 to 4 seconds, indicating a faster decay.
- The difference between the Sabine prediction and the actual impulse response could be attributed to various factors such as frequency-dependent absorption, room shape, or material properties not accounted for in the Sabine formula.
  The chart on Figure 3 compares the reverberation time calculated using Sabine's formula ($T_{60} = 5.37$ seconds) with results obtained through ray tracing simulations using different numbers of rays. The red dashed line represents the constant reverberation time predicted by Sabine's formula, while the blue line shows how the reverberation time varies as the number of rays changes.

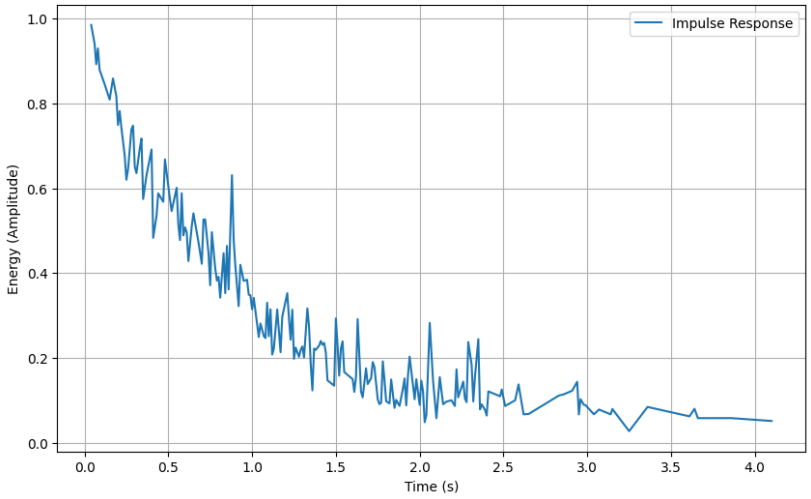


Fig. 2. Impulse Response (Amplitude) with 1000 rays

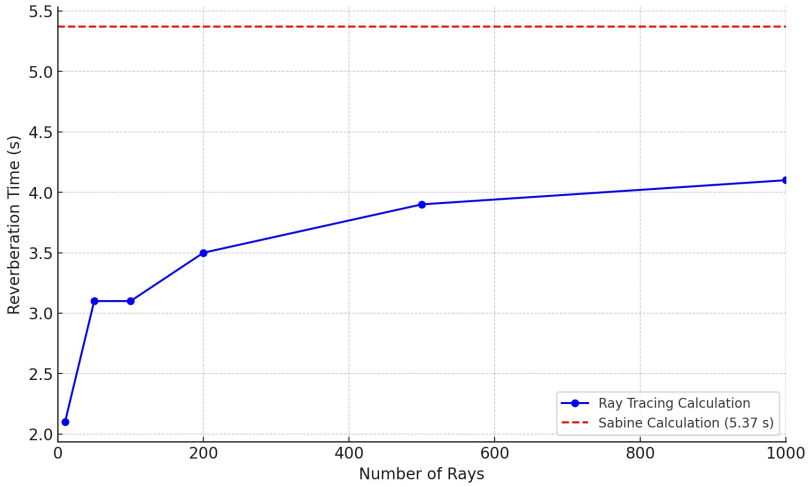


Fig. 3. Reverberation time for different number of rays

– As the number of rays increases, the accuracy of the ray tracing method improves, approaching the Sabine prediction.
– For a higher number of rays, such as 1000, the ray tracing result ($T_{60} = 4.1$) comes closer to the Sabine calculation.
– However, with fewer rays (e.g., 10 rays), the estimated reverberation time is significantly lower, indicating that a smaller number of rays leads to less accurate results.

This comparison highlights the trade-off between computational cost and accuracy in ray tracing methods for predicting room acoustics.

**Observation:** The reverberation simulation was found to be highly accurate, closely matching real-world acoustic behavior. Adjustments to the reverb time and echo den-

sity parameters allowed for flexibility in simulating environments of different sizes and acoustic properties.

### 4.4. Computational Performance

The performance of the system was evaluated by measuring the frame rate during real-time simulations and tested on AMD Ryzen 7800 X3D on a scene with 13900 triangles. The following factors were considered:

- **Ray Count:** Increasing the number of rays per frame naturally increased the computational load, but this was mitigated by the path caching system.
- **Complexity of the Environment:** Complex environments with more objects and reflective surfaces required more processing time, though the system maintained acceptable frame rates in most cases.
- **Real-Time Performance:** Even in large, complex environments, the system was able to maintain real-time performance with minimal latency which is shown on Figures 4, 5.

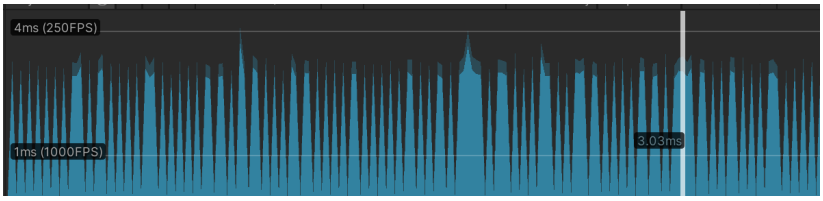


Fig. 4. 20 rays per frame 8 bounces (4ms computation time)

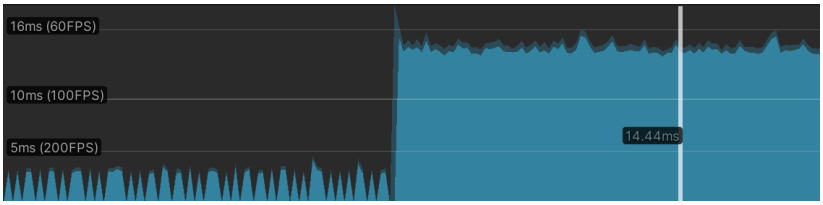


Fig. 5. Increasing rays count to 100 per frame with 8 bounces still took less than 16 ms of computational time

**Observation:** The system maintained an average frame rate of 30 to 60 frames per second, depending on the complexity of the environment. The combination of path caching and optimized ray tracing ensured that the system could handle real-time acoustic simulations without significant slowdowns.

## 5. Conclusions

This study explored sound propagation in a 3D environment using a ray tracing approach, enhanced by the integration of frequency banding and Monte Carlo methods. The approach aimed to provide a high performance simulation of acoustic behavior by simulating sound interactions with various materials and surfaces in a realistic manner.

By breaking down the sound into multiple frequency bands and utilizing path caching, computational overhead was reduced while maintaining high simulation accuracy.

The ray tracing method was compared with the Sabine formula for reverberation time, which predicts sound decay based on the total absorption in a room. For a room of dimensions 10m x 20m x 20m, Sabine's formula calculated a reverberation time of approximately 5.37 seconds. Ray tracing simulations demonstrated that with a higher number of rays, the results approached the Sabine prediction. However, with fewer rays, the reverberation time was significantly underestimated, highlighting the trade-off between computational cost and accuracy in ray tracing methods.

This comparison revealed that while ray tracing can be computationally intensive, especially with fewer rays, it offers more flexibility in handling complex environments and frequency-dependent behaviors, such as diffraction and absorption. The inclusion of CSound for sound synthesis further enhanced the realism of the simulation, making it suitable for a variety of real-time applications, including virtual reality and architectural acoustics.

## References

1. Kagawa Y. Finite Element Simulation of Non-linear Sound Wave Propagation / Y. Kagawa, T. Tsuchiya, T. Yamabuchi, H. Kawabe, T. Fujii // Journal of Sound and Vibration. – 1992. – Vol. 154. – P. 125–145.
2. Ouellet-Delorme E. Live Ray Tracing and Auralization of 3D Audio Scenes with vaRays / E. Ouellet-Delorme, H. Venkatesan, E. Durand, N. Bouillot // 18th Sound and Music Computing Conference. – 2021. – P. 1–10.
3. Meta. Acoustic Ray Tracing in Meta XR Audio SDK // 2023. – Retrieved from URL: https://developers.meta.com/horizon/documentation/unity/meta-xr-acoustic-ray-tracing-unity-overview.
4. NVIDIA. VRWorks Audio: Real-time Ray Tracing for Audio // 2023. – Retrieved from URL: https://developer.nvidia.com/rendering-technologies.
5. Kim Y.-H. Sound Visualization and Manipulation / Y.-H. Kim, J.-W. Choi. – Wiley, 2013. – 400 p.
6. Samsurya B. Realtime Audio Raytracing and Occlusion in Csound and Unity / B. Samsurya // ISCS. – 2022. – P. 1–15.
7. Kajiya J.T. The Rendering Equation / J.T. Kajiya // ACM SIGGRAPH Computer Graphics. - 1986. – Vol. 20. – P. 143–150.
8. Kuttruff H. Room Acoustics / H. Kuttruff // CRC Press. – 2009. – 302 p.
9. Pierce A.D. Acoustics: An Introduction to Its Physical Principles and Applications / A.D. Pierce // Acoustical Society of America. – 1989. – P. 1–10
10. Raghuvanshi N. Efficient and Accurate Sound Propagation Using Adaptive Rectangular Decomposition / N. Raghuvanshi, R. Narain, M.C. Lin // IEEE Transactions on Visualization and Computer Graphics. – 2009. – Vol. 15. – P. 789–801.
11. Krokstad A. Calculating the Acoustical Room Response by the Use of a Ray Tracing Technique / A. Krokstad, S. Strom, S. Sorsdal // Journal of Sound and Vibration. – 1968. – Vol. 8. – P. 118–125.
12. Maekawa Z. Noise Reduction by Screens / Z. Maekawa // Applied Acoustics. – 1968. – Vol. 1. – P. 157–173.
13. Abou Jaoude A. The Monte Carlo Methods - Recent Advances, New Perspectives and Applications / A. Abou Jaoude // IntechOpen. – 2022. – P. 2–6
14. Beranek L. Analysis of Sabine and Eyring Equations and Their Application to Concert Hall Audience and Chair Absorption / L. Beranek // The Journal of the Acoustical Society of America. – 2006. – Vol. 120. – P. 1399–1410.

15. Boulanger R. The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing, and Programming / R. Boulanger. – MIT Press, 2000. – 83 p.
16. The Csound Development Team. CSound Manual: Documentation for Csound // Retrieved from URL: https://csound.com/docs/manual/index.html.
17. Acoustic Group. Sound Absorption Data for Various Materials // Retrieved from URL: https://www.acoustic.ua/st/web_absorption_data_eng.pdf.

*Article: received* 08.10.2024
*revised* 24.10.2024
*printing adoption* 14.11.2024

# ПІДХІД НА ОСНОВІ ТРАСУВАННЯ ПРОМЕНІВ ДЛЯ СИМУЛЯЦІЇ ПОШИРЕННЯ ЗВУКУ

**О. Терлецький, В. Трушевський**

*Львівський національний університет імені Івана Франка,*
*вул. Університетська 1, Львів, 79000, Україна*
*e-mail: oleksandr.terletskyi@lnu.edu.ua, valeriy.trushevsky@lnu.edu.ua*

Моделювання поширення звуку в складних тривимірних середовищах є обчислювально інтенсивним завданням через необхідність врахування взаємодії звукових хвиль з різними матеріалами.Пропонуємо метод трасування променів, який розбиває звук на кілька частотних діапазонів для точного моделювання цих взаємодій. Двигун Unity використовується для рендерингу 3D-середовища та управління фізичними властивостями об'єктів, де кожен матеріал впливає на те, як звукові промені різних частот відбиваються або поглинаються під час контакту.

Для поліпшення рендерингу звуку інтегровано бібліотеку CSound для обробки та синтезу аудіо на підставі результатів трасування променів з розбиттям на частоти. Крім того, введено новий метод центрального променя та техніку кешування шляхів за частотами для ефективного вимірювання рівнів децибелів у різних частотних діапазонах, що знижує обчислювальне навантаження зі збереженням високої точності. Цей підхід забезпечує реалістичну симуляцію акустичної поведінки у віртуальних просторах і може бути застосований у різних сферах, де потрібен детальний аналіз звуку та робота в реальному часі.

*Ключові слова*: поширення звуку, трасування променів, частотні діапазони, Unity Engine, симуляція поширення хвиль, відбиття звуку, бібліотека CSound.