Physically Based Sound Simulation

JIEXIAO XU and NICHOLAS BATCHELDER, University of Washington

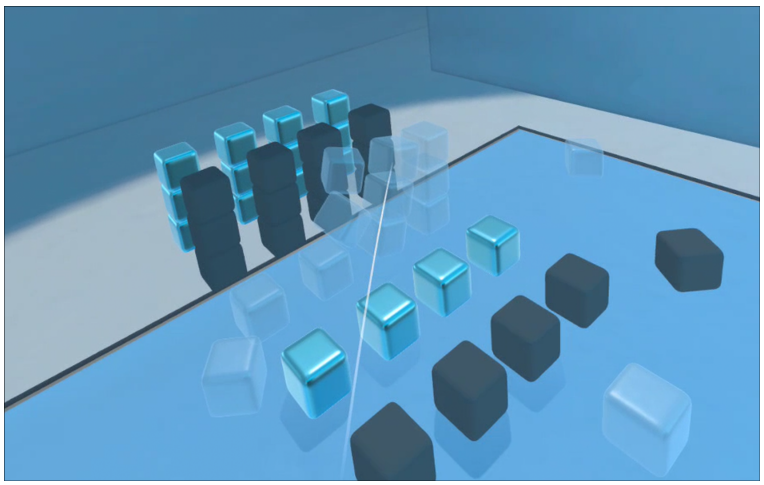


Fig. 1. Our Project implements physical sound simulation in realtime on the Meta Quest 3. Above is an image of our real-time testing environment: A room in Virtual Reality with an assortment of metal, glass, and plastic cubes. These cubes contain precomputed modal frequencies based on their material and shape, and dynamically create sounds based on these modal frequencies upon collision.

Pre-recorded audio is commonly used in interactive applications to provide sound effects. We propose an implementation of physically-based sound simulation that produces realistic acoustic simulation within an interactive environment. Using a Meta Quest 3 and the Unity Engine, we can record both the displacement and speed of our hands interacting with digital objects. Using an algorithm resolving the sound radiation, we used the pre-recorded data to simulate the sound of an interaction between two objects. For example, when a metal object hits the ground, a sound is produced based on the speed of the hit, the material of the object, and the shape of the object. Our implementation is capable of producing physically accurate sounds with regard to object material and listener position — in real-time.

1 INTRODUCTION

Pre-recorded audio is commonly used in most interactive applications to provide sound effects. However, this approach lacks scalability. Specifically, given a scene with complex geometries that are actively changing, reconstructing physically accurate sound with manual recording is nearly impossible. Therefore, we propose to implement a physics-based sound simulation prototype.

The process of sound simulation generally consists of three components: sound synthesis, sound propagation, and sound spatialization. Sound synthesis involves generating sound waves from fundamental physical principles, such as simulating vibrations of objects based on their material properties and shape. Sound propagation determines how these vibrations become audible sounds by modeling how waves propagate from the source into the surrounding medium, accounting for phenomena such as attenuation and refraction. Finally, sound spatialization focuses on how the listener perceives the sound.

Authors' address: Jiexiao Xu, jiexiao@cs.washington.edu; Nicholas Batchelder, nicbat@cs.washington.edu, University of Washington.

Our project will focus on the first two component. Rather than relying on the pre-recorded samples, we aim to construct an efficient physics-based sound model that dynamically responds to changes in the virtual environment. One major challenge in real-time sound simulation is the computational cost of solving wave propagation equations for complex scenes. Directly computing acoustic wave interactions with obstacles and materials at runtime is infeasible due to the high dimensionality of the problem. To address this, we leverage **precomputed acoustic transfer (PAT)** techniques [Doug L. James and Pai 2006], which allow us to preprocess sound propagation behavior offline and efficiently retrieve the results at runtime. Precomputed acoustic transfer involves **sampling the environment**, calculating impulse responses at various listener positions, and storing key acoustic parameters such as **early reflections, late reverberation, and diffraction effects**. These precomputed datasets are then compressed and stored in a format suitable for real-time retrieval. As a result, our implementation achieves real-time performance with physically accurate sound.

1.1 Contributions

In this project, we made the following contributions:

- Full Implementation of global sound radiation precomputation pipeline [Doug L. James and Pai 2006] for realtime audio synthesis.
- Real-time sound simulation in VR using precomputed data.

2 RELATED WORK

Pre-recorded audio is commonly used in most interactive applications to provide sound effects. However, this approach lacks scalability. Specifically, given a scene with complex geometries that are actively changing, reconstructing physically accurate sound with manual recording is nearly impossible. As a result, a few papers in the last two decades have proposed to implement a physics-based sound simulation system.

Frequency Domain Wavesolver There are plenty of studies on sound synthesis and sound propagation. The prior work [Doug L. James and Pai 2006] proposed a real-time sound simulation paradigm. It suggests a pre-computation scheme that preprocesses the global sound radiation effect before runtime. At runtime, it only computes the weighted contribution from sound sources at the listener's location, hiding the cost before interaction. Most sound studies approach the problem with the numerical method at the frequency domain. This is the principal paper that our implementation follows.

Time Domain Wavesolver [Wang et al. 2018] suggest a time-domain wavesolver that presents the better-simulated results. However, this method is not feasible on the fly, and uses extensive cloud computation resources.

3 METHOD

Modal Analysis: When the object surface oscillates, the sound wave will propagates through the air, and our ear membrane in turn mirrors the displacement that occurred on the sound source. As this process recurs in certain frequency, we can hear the sound. The wave phenomenon is periodic, so it can be decomposed into linear combination of trigonometric functions, or modes shapes u_k. More specifically, any complex sound can be represented as a combination of sinusoidal waves. The displacement of time can be described as

$$u(t) = \langle U, q(t) \rangle$$

where U denotes the vector of modes and $q(t)$ represents the corresponding amplitude coefficients. This constructs the basis of sound generation and will help us differentiate the objects by its sound in virtual environment. For example, we can perceive the difference between the plastic, metallic, or glassy objects as the materials produce different modes when vibrating. The modal analysis mature in engineering, as multiple pde solver exist. We are more interested about how the sound radiates and reaches the listener's ear after generation.

Sound Transfer: Wave equation addresses the problem of how the sound progress to the microphone through space and time.

$$\frac{\partial^2 p}{\partial t^2} = k^2 \Delta p$$

Δ is the Laplace operator based on position, k is wavenumber given by $\frac{2\pi f}{c}$, and $p(x,t)$ describes the sound pressure given position x and time t. However, in our implementation, we will simplify the simulation model by approximating the change in time with damping factor. Thus, our model of the pressure result become $p(x)e^{i\omega t}$, where ω is the frequency of the mode. In this case, Helmholtz equation, temporal Fourier transform of wave equation, is preferred in acoustics.

$$\Delta p(x) + k^2 p(x) = 0, x \in \Omega$$

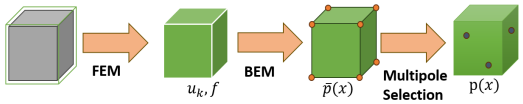
The evaluations of sound will based on the $p(x)$ at each mode, and the solution of $p(x)$ will depend on the boundary condition. Two type of boundary conditions exist. Dirichlet boundary condition indicates that the value of solutions is on the boundary, while the Neumann boundary condition specifies the normal derivative of solution resides on the boundary. Neumann boundary condition is favorable here because it models the sound-hard surfaces and rigid boundaries as it indicates that sound has constant normal velocity at the boundary. The condition is given by

$$\frac{\partial p}{\partial n} = -i\omega\rho v$$

ρ is the fluid density (in this case, $\rho_{air} = 1.21$). v is the normal velocity on the surface given by $v = i\omega\langle n, u\rangle$, where $\langle n, u\rangle$ is the normal displacement. Thus, we may simplify the condition to

$$\frac{\partial p}{\partial n} = -i\omega\rho(i\omega\langle n, u\rangle) = \rho\omega^2\langle n, u\rangle$$

Solving the radiation equation is usually done by Boundary Element Method, or BEM. As for each mode, we need to evaluate the sound pressure $p(x)$ for each vertex. Given N boundary elements and M modes, this operation will incur $O(NM)$ cost. This cost make real-time computation infeasible, which become the main challenge of real-time sound computation.



Approximating Acoustic Transfer: The simple intuition is that we do not need to evaluate the sound pressure on all vertices. If two vertices are closed enough, it is likely that they has similar contribution to the result sound. Therefore, we may achieve realistic sound simulation by picking m sound sources representing the N vertices while $m << N$. In this sense, we will off-load the computation before the runtime. The pre-computation consist of following steps.

(1) Compute modes (by FEM package)
(2) Compute pressure at vertices (by BEM package)
(3) Pick multipoles from candidates
(4) Compute Pressure Coefficient based on multipoles
(5) Export the coefficient for real-time computation

Our final sound is linear combination of spherical multipoles $\psi_{lm}(x - \bar{x})$, where $\bar{x}$ denotes the position of sound sources.

$$\psi_{lm}(x - \bar{x}) = h_l^{(2)}(kr)Y_{lm}(\theta, \phi)$$

r, θ, ϕ are the spherical coordinates of $x - \bar{x}$. The spherical harmonics $Y_{lm}(\theta, \phi)$ is widely used in the graphics with regarding to the variation of visual effects based on the angular position. It describe the angular relation between the listener and object in this case. Hankel Function represents the wave propagation,

$$h_l^{(2)}(kr) = j_l(kr) - iy_l(kr)$$

where j_l and y_l are spherical Bessel's function of the first and second kind.

Given m multipoles of order n in spherical harmonics, we will have

$$p(x) = \sum_{q=1}^{m} \sum_{l=0}^{n-1} \sum_{m=-l}^{l} c_{qlm} \psi_{lm}(x - \bar{x_q})$$

Plus, for each multipoles, we will have n^2. This gives us that $p(x) = \sum_{j=1}^{mn^2} c_j \psi_j(x)$, where j is generalized index for (q, l, m). At this stage, we could two problems in the formulation. First, we can not evaluate the sound pressure when $x - \bar{x} = 0$ due to the computation of spherical coordinates. Second problem is how to find the coefficient.

Offset Surface: It is straightforward to consider the option that we place the fictitious sound sources inside the object. However, if the object consist of thin shell such as a bell, the source points are likely to collide with its vertices, which cause the spherical harmonics broken. To avoid the singular value at the computation, we will create a shell that enclosed the original object. The offset surface will define clearly what is inside region. The other factor to consider is the offset distance δ. If δ is too small, it will cause overfitting issue, so we will choose δ to be multiple of the largest mesh's edge. In our prototype, as we experiment on a cube object, typical δ is around 10cm. The original work applied the marching cube and signed distance field method to construct the offset surface. After construction of the offset surface, we will use BEM to sample the ground truth pressure $\bar{p}(x)$ on the offset surface at N vertex sample position.

Weighted Least Square Now we need to handle the coefficients in the linear combination. After we set up the offset surface, if we place sources inside the volume inside the offset surface, as the Dirichlet BC is satisfied on surface, Helmholz Radiation problem is satisfied everything exterior to the surface. The approximation error is determined by how well we satisfy the boundary condition, $p(x) = \bar{p}(x)$. Suppose we have m unique sound sources, we can compute the expansion coefficients c_j by finding the minimizer of weighted least square as following.

$$WVc = W\bar{p}$$

W is the weight matrix that has shape N-by-N. It scales the pressure by the area for each vertices, where $W_{ii} = \sqrt{a_i}$ and a_i is given by the 1/3 of the adjacent triangle areas the vertex i. V is the multipole basis matrix with with $V_{ij} = psi(s_i)$ is the j^{th} multipole function evaluated at the i^{th} sample position, s_i. $\bar{p}_i = \bar{p}(s_i)$ is the BEM pressure evaluated at the s_i. To solve this problem, we will have $A = WV$ and $b = W\bar{p}$. The system equation is in our familiar format.

$$Ac = b$$

However, A is not always in good condition. To address poorly condition A matrix, we will solve the least square problem by truncated singular value decomposition with a singular value threshold 10^{-6}. More specifically, we will take the singular value decomposition of the matrix A. In the resulting singular value matrix, we will only use the singular value that is greater than the threshold. Then, the product is easy to invert the help us solve the system of linear equations.

Multipole Placement Algorithm Now we come to the most important part. How could we determine the multipole that minimize the difference between the ground truth sound pressure and approximation? The original paper first introduce the greedy algorithm of finding the multipole and another algorithm that accelerates the finding process with a more complicated strategy. In this place, we will focus on the first strategy.

We will incrementally add the source points. We will based on the residual $r = b - Ac$ to rank the candidates in the queue. From the perspective of mathematical formulation, we will use the basis

$$U_x = \text{basis}(WV_x)$$

The fitness can be determine by the norm of the projection from residual to the basis $||(U_x)^H r||_2^2$, where H is the matrix Hermitian conjugate. Thus, our best candidates is

$$x^* = \text{argmax}_{x \in X} ||(U_x)^H r||_2^2$$

To obtain the candidates, we use the rejection sampling. We first construct a bounding box that enclosed the offset surface. Then, we would randomly sample the points inside the bounding box. The candidates will be the random points that fall inside the offset surface.

With the new the multipole position, we then need to update our residual and unitary basis Q spanned by all the multipole selected. In the first place, we will apply the modified GramSchmidt to $[Q|WV_X]$, which gives us $[Q|Q_x]$. The resulting Q is orthogonal to the residual. Then we will update the residual with Q_x with $r = r - Q_x Q_x^H r$.

Now we have all the tools we need for the multipole placement algorithm. We will first initialize the residual $r = W\bar{p}$. While the residual is greater than the tolerance, in our case 10^{-4}, we will filling the sound sources from the candidates for each round. Finally, we will return all the multipole positions.

Based on the multiple positions, we will be able the compute the coefficient c we mentioned before. The last step in the pre-computation is exporting them for the real-time computation.

Real-Time: We will parse the binary file from the pre-computation, computing the sound by iterating on each frequency. We then apply a damping factor that will make sound diminish over the time.

4 IMPLEMENTATION DETAILS

Precomputation

We first create the mesh inside Blender. While the author from original paper chose to use marching cube and signed distance field to obtain the offset surface of the object, Blender provides solidify modifier, which is a handy approach to construct the offset surface.

The first step in our pre-computation is acquiring the modes and frequency of the object. Autodesk Fusion provides a convenient tool for conducting the modal synthesis. We first need to convert the mesh to solids and assign the physical material. The simulation tool in the Autodesk Fusion will solve the pde and output the modes and frequency. We will only take the mode that make sense in reality. For example, s-shape distortion of glass objects is not taken into account.

The majority of our code was in python and C#. Bempp is pythonic library that provides BEM solver. With the modes and frequency

as the input to the BEM package. We will obtain the pressure evaluated at each vertices. Then we will find the multipoles through aforementioned multipole placement algorithm.

As the result, we have the coefficient c and multipole positions. We will then encoded these parameter into binary file code respectively. For instance, we have K file saving the wavenumber k and SOURCE file saving the multipole positions and spherical coefficients.

On-Device Implementation Differences We implemented the real-time computation step of this algorithm on a Meta Quest 3 headset using Unity. When implementing real-time sound production on device, many optimizations had to be made. The pressure for each modal frequency was computed in parallel, and the individual waves for each frequency were stored in memory for efficient computation. Each object loaded its calculated source and k files in memory, and audio synthesis was done in a collision handler. We took advantage of the Unity game engine for physics and collision detection.

5 EVALUATION OF RESULTS

Due to the nature of sound-focused project, it is more favorable to examine the quality by ear. We have achieved real-time performance of computing sound. Given different material such bronze, glass, or plastic, our project could produce realistic sound while showing the clear difference between the objects with different material. We will still put the data we have for the operation.

Table 1. Computing Time for Different Materials

Material	time
Bronze	33.6235
Glass	10.9968
Plastic	11.3111

The table above shows the computation time including the BEM solver and multipole placement algorithm. The FEM will take around 3 minutes to each material with simple cube mesh. We did not directly test on object with more complex geometry because the Autodesk fusion does not directly support non-watertight meshes. We may need to find other tools with modal analysis. The bronze takes longer time because Bempp file take long time to set up the solver. The extra 20 seconds are the overheads of initialization. Beyond this, we were trying to obtain the latency of sound simulation in the real-time process and the error between our predicted sound and actual sound, but due to the time limitation, we could only give qualitative results. In terms of the former latency measurement, the computation is fast enough that we can not perceive any latency by our ear. The simulation result for bronze material and glass material is realistic based on our experience, while the plastic object does not fit our expectation. The potential issue is that we don't have enough familiarity to the FEM and BEM tools such that we may get flawed data from the pde solvers.

6 DISCUSSION OF BENEFITS AND LIMITATIONS

Our implementation successfully integrates a frequency-domain algorithm for real-time sound simulation in a sample VR environment. This demonstrates the feasibility of simulating physically accurate sound in interactive applications without pre-recorded audio. By leveraging Unity's engine, we significantly reduced the latency between simulated objects and static objects to near zero. Furthermore, latency between multiple interacting objects remains negligible, enabling a seamless user experience where objects produce distinct sounds based on their material properties and shape. The parallelism API of Unity's engine played a crucial role in optimizing computation, allowing for efficient real-time execution.

However, the complexity of the frequency-domain algorithm, combined with our initial unfamiliarity with Unity, resulted in a longer-than-expected implementation time. Consequently, we were unable to explore time-domain-based sound simulation, which could offer alternative advantages in fidelity and realism. Additionally, while our implementation supports smooth interactions in most cases, collisions between more than 20 objects simultaneously lead to a noticeable reduction in frame rate.

7 FUTURE WORK

Future work could investigate further optimization strategies for frequency-domain sound simulation, including enhanced parallelism and leveraging Unity shaders for audio synthesis to improve performance in complex scenes with orders of magnitude more objects. Optimizing computational efficiency would enable smoother real-time interactions, particularly in scenarios with high object collision counts, ensuring scalability for large-scale simulations.

Beyond performance improvements, physically-based sound simulation has potential applications in interactive environments, particularly in sandbox games. Unlike traditional pre-recorded audio systems, dynamically generated soundscapes respond to environmental changes, enhancing immersion by providing real-time auditory feedback based on object interactions and material properties.

Additionally, future research could explore real-time implementations of time-domain-based sound simulation algorithms [Wang et al. 2018]. While these approaches traditionally require significant computational resources, recent advancements in GPU optimization for large language models (LLMs) may provide new opportunities for on-device processing. Leveraging modern GPU architectures could enable real-time execution of time-domain sound simulation, bridging the gap between physical accuracy and interactive performance.

8 CONCLUSION

Our work demonstrates the feasibility of zero-latency real-time physics-based sound simulation in Virtual Reality using a frequency-domain approach. This implementation highlights the potential for more physically accurate and scalable soundscapes in AR/VR environments, moving beyond pre-recorded audio limitations. While challenges do remain, particularly in handling large-scale object interactions, further optimizations and GPU advancements could enable even more complex simulations. Physical sound simulation within AR/VR applications could redefine audio-visual immersion. Accurate audio for interactions between user-created objects could be produced without thousands of pre-recorded audio samples, allowing for more dynamic experiences. Increased research in this

Table 2. FEM result for Bronze, Glass, and Plastic

Material	Mode	Frequency (Hz)	X Participation	Y Participation	Z Participation
Bronze	1	101.171	61.245	0.000	0.000
	2	156.719	0.000	63.098	0.000
	3	431.000	22.349	0.000	0.0001
	4	450.849	0.000	0.000	78.859
	5	495.658	0.000	22.491	0.000
Glass	1	165.229	61.181	0.000	0.000
	2	259.775	0.000	62.378	0.000
	3	721.696	22.132	0.000	0.0001
	4	737.924	0.000	0.000	80.261
	5	837.943	0.000	22.922	0.000
Plastic	1	43.949	61.180	0.000	0.000
	2	67.372	0.000	63.468	0.000
	3	183.914	22.520	0.000	0.0002
	4	195.117	0.000	0.000	77.584
	5	210.158	0.000	22.242	0.000

area could lead to more computationally efficient solutions, paving the way for physically accurate, interactive audio in next-generation AR/VR applications.

ACKNOWLEDGMENTS

We would like to thank Trisha Binwade for countless hours of testing and helping us debug each iteration of our real-time simulation in VR.

REFERENCES

Jernej Barbič Doug L. James and Dinesh K. Pai. 2006. Precomputed Acoustic Transfer: Output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Trans. Graph.* (2006).

Jui-Hsien Wang, Ante Qu, Timothy R. Langlois, and Doug L. James. 2018. Toward Wave-based Sound Synthesis for Computer Animation. *ACM Trans. Graph.* 37, 4, Article 109 (July 2018), 16 pages. https://doi.org/10.1145/3197517.3201318