Inaudible - Visualising Sound for Deaf People in 3D Games

Julius Herrmann and Marin Prusac

Lund University
Sweden

February 8, 2026

Abstract

This computer graphics project explores the visualization of directional sound modelled by implementing audible ray tracing. To achieve this, the Unity game engine is employed and a 3D modelled lightweight environment is used to efficiently calculate sound ray collisions. Simulating the propagation of sound is done by approximating the soundwaves as rays which behave similar to light rays which can be reflected, absorbed and passed through. This enables the creation of an dynamic sound-based visual texture to indicate sound intensity and directions. The result show that that while providing a intuitive and immersive visualization of sound a few limitations regarding physical accuracy and occlusion handling remain.

1 Introduction

Many 3D applications rely on directional sound to portray essential information and guidance, especially in video games. While this is useful and immersive for hearing users, those who are deaf are unable to experience this. In competitive games this puts those players at a disadvantage. Current solutions for deaf people in popular competitive video games usually consist of a single visual indicating the direction to the sound; some also include the type of sound, but this is deemed as insufficient and unrealistic.

To approximately and accurately model directional sound, multiple approaches exist. [3] The first one is using wave simulation to approximate the propagation of sound. This approach, however, although very accurate, is also very resource-intensive and can run into issues with the complexity of digital environments. For this project, the focus lies on the second approach: using raytracing to model the different paths sound can take and thus creating an approximate solution for sound propagation that is not as resource-intensive. The following application presents our approach to modelling audible ray tracing using the Unity game engine.

2 Application

2.1 Tools

- Blender: Open-source, multi-platform 3D modelling software. It was used for modelling the 3D environment of the project.[1]
- Unity: Unity is a multi-platform game engine. It was used to essentially build the application and for its shader tools. [2]

2.2 Modelling (Julius)

The first step was to model the 3D environment for the project. This was done using Unity and Blender where 3D assets were first created and then exported to Unity using the FBX standard. When importing these into Unity problems were encountered because of Unity's behaviour to automatically delete objects with overlapping edges. To overcome this part of the modelling work was done in Unity directly. The environment was mostly modelled by axis-aligned boxes. The bounds of these boxes, represented by minimum coordinates (min x, min y, min z) and maximum coordinates (max x, max y, max z), were easily extracted and put into shader buffers, making them lightweight and easy to compute ray collisions with.

2.3 Raytracing

Boxes' bounds, camera position, camera-rendered texture and sound sources are all sent to the shader as uniforms and buffers. Now the raytracing is performed similarly to how its done with light sources [2], but with special considerations given to the wall (box) sound materials. When a ray hits a wall it has a chance to be absorbed, bounce off or pass through. Each time a wall is hit, a sound contribution is calculated based on the nearby unoccluded sound sources. The decreasing energy of each sound ray makes it gradually more likely to be absorbed.

3 Results

The world, visible in 1, is contained of white walls, floor and ceiling which build the three rooms, of which two are smaller and divided by door frames from the biggest, main room. In the scene, a sound source and a camera are present. Sound appears as a texture on top of the camera rendered world. It is noisy as sampling is not performed. Around the sound source and on the wall parts visible

[1] Blender: https://www.blender.org/, last retrieved on 26/12/25
[2] Unity: https://unity.com/, last retrieved on 26/12/25

from the sound source are where there's most of the sound indication 2. In essence, the sound source behaves similarly to a light source. The biggest difference from light is how it manifests on the outer parts of the walls, as seen on 3. The appearance is different when the indicator is a result of a sound bounce and when it's a result of passage of sound though a wall. Stepping farther from the sound source makes the noise sparser, while steeping closer makes it denser. Different sound sources produce differently coloured indicators. It is not possible to see the sound coming from behind, unless it bounces in the field of view. The noise intensity varies slightly with the volume of sound at the given moment. Audible music is played from the source to show the hearing people how sound is transformed into visual indicators, although it is not raytraced.

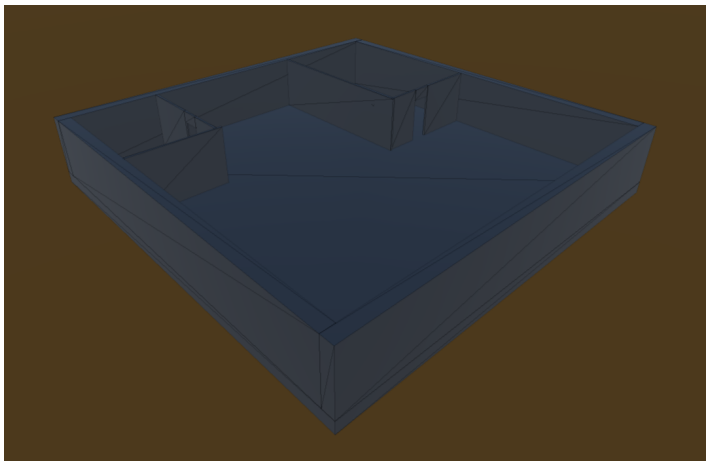

Figure 1: Modelled environment

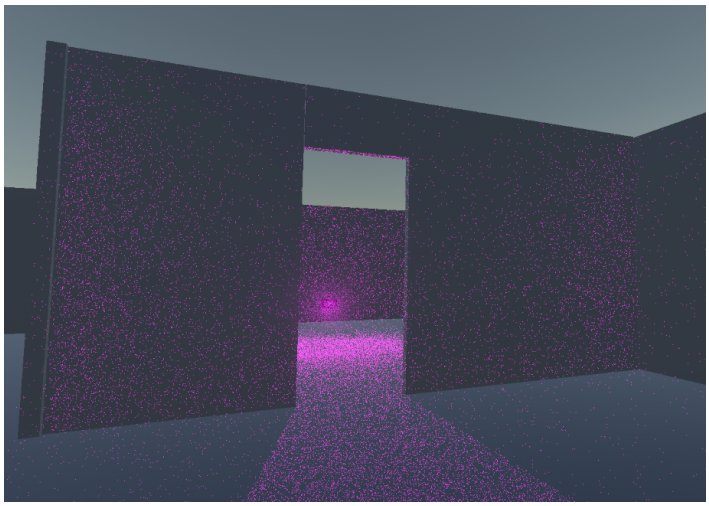

Figure 2: Direct view of sound source

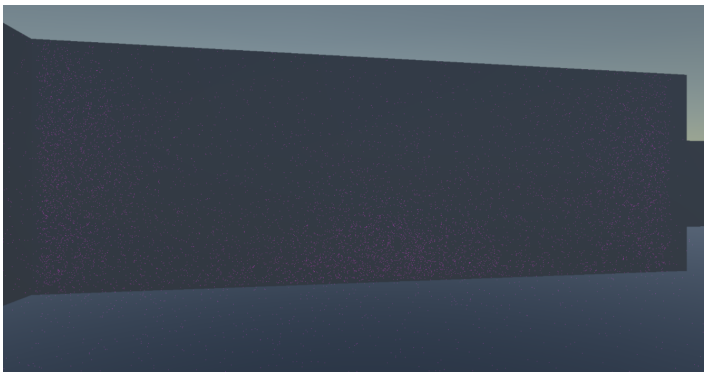

Figure 3: Sound behind a wall

4 Discussion

The produced noisy texture was graded as acceptable, even preferable, because it contrasts the visuals of the environment. Low volume of a sound source or large distance from the camera would make the sound noise sparse and harder to orient by - mimicking the perception of sound in real life. Being in a room with a loud sound source makes the noise very dense, making it harder to sense the quieter sounds from other rooms.

There are some features missing in the implementation. For one, a person cannot see the sound indicators if not looking at the wall from which sound rays bounce, but a hearing person would still hear the actual sound. An additional visual indicator should be added onto the screen, like a sphere at the bottom of the screen from which rays are shoot to the directions behind the player, on which the raycast values would be drawn.

The second issue is related to the physical accuracy. Changes made to the traditional light raytracer are not sufficient to imitate the behaviour of sound waves. [1] Sound should propagate through the air and go around the walls, not just bound from them, but that is currently not the case. While this issue should be resolved, the opinion is that the implementation is not needed to be fully physically accurate, but to produce results with the same advantages and disadvantages of directional sound.

The approximation that the environment is made from axis aligned boxes is also quite limiting. That can be fixed by adding new shapes and their buffers and then implement new ray-hit functions. A solution that was explored before is to model the sound-interacting world with voxels, but keep the actual world in the original mesh form.

Being close to a very loud sound source would completely colour the room. There should be adaptation to the sound to soften the scaling of visual indication from the perceived sound volume.

5 Conclusion

The project successfully implemented a functional prototype for rendering audible ray tracing using the unity game engine. This provides an alternative sensory experience conveying sound direction and intensity. The chosen approach balances performance and realism, simplifying simulation aspects while still producing

meaningful results. In the future development should address limitations such as the lack of back facing sound visualization and increasing the fidelity of the sound simulations. Implementing enhancements such as volumetric sound diffusion, adaptive noise scaling and enhancing the prototype to handle more complex geometry would improve user accessibility as well as realism. Overall this prototype demonstrates the potential of visual sound mapping as a valuable accessibility tool for 3D environments.

References

[1] E. Lakka et al. “Spatial Sound Rendering – A Survey”. In: *IJIMAI* 5.3 (2018), pp. 33–45. (Visited on 12/26/2025).

[2] SebLague. *GitHub - SebLague/Ray-Tracing at Episode01*. https://github.com/SebLague/Ray-Tracing. (Visited on 01/01/2026).

[3] *Vercidium*. https://vercidium.com/. (Visited on 01/01/2026).