Proceedings of the Seventeenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2021)

Efficient 2D Sound Propagation in Video Games

Nima Davari, Clark Verbrugge
McGill University, Montréal, Canada
nima.davari@mail.mcgill.ca, clump@cs.mcgill.ca

Abstract

Sound is well known for improving immersion in video games. In many games sound also serves as an important game mechanism; in stealth games, for example, sounds triggered by player actions may attract enemy agents, and sounds created by enemy agents give useful information on where to target or avoid combat. Modelling sound, however, is computationally complex, and thus sound propagation is often restricted to highly localized contexts, or based on trivial propagation models such as a fixed radius that ignores the impact of obstacles. In this work, we describe an efficient approach to propagating sound in a real-time 2D game level. We use a heuristic ray-based approach, adapted to include diffraction, reflection, and transmission. Our design allows for dynamic geometries, avoids sound discontinuities, and can be tuned in various ways to trade-off accuracy for computation cost.

Introduction

Sound is an important aspect of video games, well known for increasing the sense of immersion in virtual environments (Brown and Cairns 2004). Sounds made by enemies allow players to localize unseen opponents, but sound can also be a major factor in game-play decision-making and strategization: sounds made by a player agent can be used to alert or otherwise guide enemy NPCs toward the player, and are thus useful in AI systems.

Computing realistic sound propagation, however, is difficult. Sound is typically understood in terms of its wave-based properties, which allow it to bend around corners (diffraction), reflect off of a wide variety of surfaces, echo, reverberate, con/de-structively interfere, and even transmit through obstacles, resulting in a complex soundscape that strongly depends on the local geometry and materials. To reduce cost, many video games either use very simple models of sound propagation, such as room or radius-based approaches that limit sound travel and ignore obstacles entirely, or use pre-scripted volume settings determined manually or computed ahead of time based on a static geometry.

In this paper, we describe a heuristic, 2D geometry-based sound propagation design that can be efficiently computed at runtime to provide reasonable sound propagation in a real-time, dynamic environment. Our model builds upon ideas of visibility. Similar to light modelling, we use rays to propagate sound through the environment. Rays encountering obstacles diffract and/or reflect, resulting in additional virtual sound sources that extend the coverage of the primary source. Smoothly merging these sound sources eliminates discontinuities, and consideration of which sources reach a given point allows modelling echos or other forms of sound combination.

Specific contributions of our work include:

- We define a base sound computation that supports diffraction, reflection, and sound transmission. Secondary, virtual sound sources are generated and smoothly combined to present a cohesive sound experience.
- We support both static and dynamic contexts. The entire model can be computed at runtime, allowing for dynamic geometries and recomputation due to moving sound sources. In static contexts the model does not require a listener, and can thus be fully pre-computed.
- Our design is computationally efficient. We use a Unity3D implementation to demonstrate low runtime cost, which can be further tuned by reducing level detail and controlling the number of secondary sound sources, trading off quality for cost.

Related Work

(Funkhouser et al. 2004) divides sound simulation methods into two groups: numerical solutions to wave equations, and geometrical propagation-path approximations. (Savioja 1999) uses the terms wave-based models and ray-based models to refer to the two groups. Our work is a ray-based method that utilizes a small portion of the CPU to provide real-time sound modeling for dynamic environments.

Wave-based models can result in more accurate simulations, but they can also be much more computationally expensive; therefore, most wave-based solutions are not suitable for real-time environments such as video games. Finite-difference time-domain (FDTD) is a technique that is used for wave-based sound simulation (Botteldooren 1995). In this method, the derivatives in the wave equation are replaced by finite differences (Savioja 1999; Strikwerda 2004). (Zamith et al. 2010) uses such technique to simulate sound wave propagation on the GPU. Finite Elements Methods (FEM) and Boundary Element Methods (BEM) are wave-based techniques that solve the wave-equation using a partitioning scheme; however, they are only suitable for low frequencies due to high computational cost (Savioja 1999; Foale and Vamplew 2007).

Project Acoustics is a wave acoustics engine for 3D interactive experiences (Microsoft 2019). It precomputes the details of the sound model on the cloud, similar to how static lighting is often modelled. During the baking stage, a large number of impulse responses (terabytes of data in some cases) are processed, which lifts a lot of the computation away from the CPU at runtime, but comes with the disadvantage of not supporting dynamic geometries.

While wave-based models try to solve the wave equation, geometry-based (ray-based) methods model sound using rays and the geometry. They fire rays from the sound source, which will reflect off of surfaces in the environment and some will eventually reach the listener. Sound is then modelled using the propagation paths. There are a number of approaches that have been used for geometry-based sound modelling. Image-Source method is one of the first techniques that was used to simulate acoustics (Allen and Berkley 1979). This technique models specular reflection by recursively creating new virtual sound sources that are reflections of the sound source with respect to an edge in the geometry. The method has exponential complexity and is not practical for complex geometries.

Path tracing and ray tracing methods are very well-known in the field of Computer Graphics. The scalable nature of ray-based techniques and their simple computation make them good options for acoustics modeling as well (Mahjoob and Malakooti 2008; Krokstad, Strom, and Sørsdal 1968; Kulowski 1985). There are extensions to ray tracing, such as beam tracing, which uses volumetric elements instead of rays. Our model is also based on ray tracing; however, it prioritizes performance and user experience over accuracy, and therefore, it tries to simulate sound propagation using only a small number of rays.

(Tsingos et al. 2001) and (Funkhouser et al. 1998) use a similar approach to model the diffraction, reflection, and transmission properties of sound. They use beam tracing for efficiency; however, this technique is only practical for coarsely-detailed models, as beams would constantly get fragmented while travelling through a complex environment (Tsingos et al. 2001). In contrast, our model's complexity is primarily bound to the number of vertices in the geometry, and the shapes and sizes of obstacles do not impose strict limitations.

(Taylor et al. 2012) is another work that uses beam tracing. Similar to our method, it considers simplifications such as computing only the first few specular reflections, or considering diffraction only in shadow regions. The primary difference between this method and our technique is that for sound propagation, we do not aimlessly fire a large number of rays or beams, hoping to reach the listener through reflections. This means that our model does not need to be computed on the GPU, whereas the aforementioned model requires utilizing the GPU to be able to perform in real-time. (Cowan and Kapralos 2015) also use the GPU to provide an interactive experience. Their approach runs a shortest-path search on a grid representation of the geometry. Its complexity grows with the resolution of the grid, which limits the geometry's shape and size for a proper real-time experience.

In the context of video games, interactive sound propagation can benefit from being faster and simpler at the cost of some accuracy. (Boev 2015) describes the audio budget for Hitman (2016 stealth video game) to be 1ms per frame and up to 50 percent of a CPU core. They use a Rooms-and-Portals representation for the geometry and consider simplifications such as partially ignoring sound source position inside rooms, applying a muffling filter for listeners that are behind obstacles, having a single material for all walls of a room, and mostly relying on the level geometry to be static. While all these approximations are reasonable for the game, we believe that our techniques can provide more accurate results, while keeping the computation time low and supporting complex, dynamic geometries.

Tom Clancy's Rainbow Six Siege (2015 shooter video game) is more focused on dynamic geometry support, as walls can be shot to create holes in them in the middle of combat (Dion 2017). Their approach to sound propagation is to use strategically placed nodes in the map to calculate the shortest path. This method restricts the dynamic aspect of the geometry to some extent, and it requires the level layout to be determined offline. They also handle transmission and reflection through simulation tricks, which we try to more accurately model using rays.

Sound Propagation

Emitting from the sound source, sound signals travel in all directions and can reach their destination from many different paths. The signal caught by the listener can be computed using the following equation (Takala and Hahn 1992).

$$\mathit{received}(t) = \int_0^\infty w(t,\tau) \cdot \mathit{emitted}(t-\tau)d\tau \qquad (1)$$

Where $w(t,\tau)$ is the amount of signal received through all paths with delay τ.

The above model is not suitable for a real-time environment. We need a simplified version that is easier to compute, but also preserves reasonable sound propagation through the geometry. With that goal in mind, we assume the following properties for our sound propagation model:

- Sound amplitude decays in proportion to the distance travelled.
- Sticking to the visibility idea, sound can only move in a straight line; therefore, a listener will not receive any signal from a source that is not directly in its sight.
- Sound travels with no delay. We assume sound travels fast enough to be heard instantaneously in different locations of the environment. We later introduce an exception for this rule to model echos.
- Since we do not consider any delays, sound propagation can be formulated exclusively with attention to amplitude. We can therefore have a simple sound propagation equation such as the following.

$$A_d = \max\{A_s - \mathit{pdf} \times d,\ 0\} \qquad (2)$$

Where A_d is the amplitude created at a destination point, A_s is the amplitude at the source point, d is the distance between the source and the destination, and *pdf* is the sound propagation decay factor.

While we use a linear amplitude fall-off for simplicity, the equation can be trivially modified to represent other decay functions as well.

Proposed Model

We introduce an efficient ray-based sound model that attempts to simulate the following sound behaviors: diffraction, reflection, and transmission. Diffraction is the ability of sound to bend around corners, reflection allows sound to reflect off of surfaces, and transmission allows sound to penetrate through obstacles, losing a portion of its energy as a result.

Our geometry is comprised of game obstacles, which are represented as non-intersecting polygons. The geometry can be simplified if required, to reduce the computations of the sound system. We define sound source as an entity that produces a sound signal at a certain point in our 2D environment. There are no limitations on where a sound source can be positioned, as our model does not require game obstacles to be solid. For the sake of clarity, we call these sound sources "primary sound sources," as we later introduce other types of sources as well. In the figures of this paper, the green lines represent the obstacles, the red dot represents the sound source, and its maximum range of coverage is displayed as a circle.

In our model, sound primarily moves around the level geometry using diffraction. We do not use reflection to explore the environment, rather, reflection is used to simulate echos and delays in sound propagation. We use transmission as a complement to diffraction, to cover areas that are not properly accessible for the diffraction model. In this section, we explain our techniques of modelling each of these behaviors of sound.

Diffraction

Our idea to let sound reach around corners is that since the primary source cannot directly reach the area occluded by an obstacle, we can create a virtual sound source at a vertex (an intermediate point), that has vision over both the primary source and the occluded area. This method can be considered an approximation of the Huygens–Fresnel's diffraction model (Wikipedia contributors 2021). Figure 1 shows two diffraction sources and their regions of coverage.

Model Computation. To model diffraction, the first step is to shoot rays from the location of the primary sound source to the vertices of the geometry. Each of these vertices is a possible location for a diffraction source, which plays the same sound as the primary source, but with a lower amplitude. More specifically, we can compute the amplitude using the sound propagation equation (2). Another distinct feature of a diffraction source is that it does not have full radial coverage.

A diffraction source will be created at a vertex as long as all the following conditions are met.

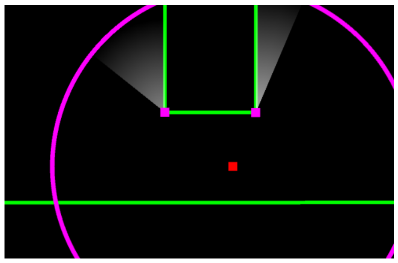
Figure 1: Diffraction sources are created by shooting rays from the source toward the vertices of the obstacles. The purple dots are diffraction sources.

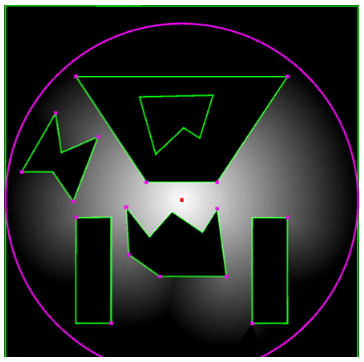
Figure 2: The coverage of a sound model with the diffraction property. Purple dots represent diffraction sources.

- The amplitude of the potential diffraction source would be nonzero.
- The ray would not intersect any obstacles on its way to the target vertex.
- The ray would not penetrate through the obstacle if extended by a small length.

Once we have the first-order diffraction sources, we use them to create higher-order sources to cover all reachable regions in the environment. Figure 2 shows the coverage of a sound source with diffraction enabled.

Sound Computation. Now that we have a diffraction model, we can use it to compute sound for a particular listener. We break down the process into two steps: computing the sound created by each individual diffraction source, and combining the results. The first step is done by applying (2) for each diffraction source. To combine the computed amplitudes, we use the maximum operator, as not only does it create a smooth sound heatmap, but it also results in a final amplitude that is at least as high as the amplitude of all individual diffraction paths to the listener. The output of the diffraction model is hence computed as follows:

$$A_l = \max\{\max\{A_d - \mathit{pdf} \times d_d,\ 0\} : d \in \mathit{dlist}\} \qquad (3)$$

Where *dlist* is the list of all computed diffraction sources, A_d is the amplitude of a diffraction source, d_d is the distance between source d and the listener, and *pdf* is the sound propagation decay factor.

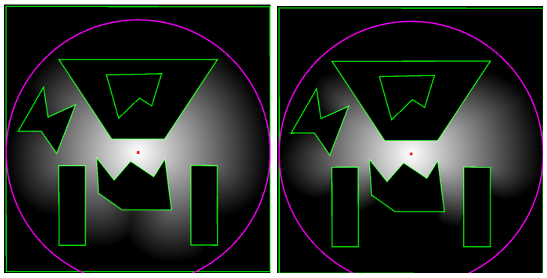
Figure 3: Left: normal diffraction model. Right: decaying diffraction model.

Optimization Techniques. Computing the second-order diffraction sources can be time consuming, but there are ways to speed-up the process. We will mention two such methods below.

Depending on the geometry of the game level, our diffraction sources can cause sound to loop in circles. Since we eventually end up taking the maximum amplitude as our diffraction output, circling around is a completely unnecessary computation. We can prove that only a maximum of one diffraction source is required at each vertex of the geometry, and that source is the one with the highest amplitude regardless of its orientation; therefore, by simply remembering the amplitude of a diffraction source on a vertex, we can decide whether a new source should be created on that point or can we just discard it and prevent any other recursive calls that would otherwise follow. To maximize the pruning, at each vertex, we need to place the highest amplitude source before any other source. This motivates the use of a greedy heuristic when we are creating the second-order diffraction sources. Specifically, at any point, we prioritize generating the children of the diffraction source that has the highest amplitude.

Another optimization method is to cache a table that speeds-up the process of deciding whether a higher-order diffraction source can be created at a certain target vertex, from a certain source vertex. Since both the source and the target are fixed vertices of the geometry, we can easily precompute information such as the angle between the two points, or whether the interconnecting ray would intersect an obstacle.

Variations. To control the shape and the feel of the diffraction model, we can modify parts of the model computation. Figure 3 displays a diffraction variation in which sound loses more energy the more it bends. Later on, in figure 10, we have examples of changing (2) to achieve a more realistic sound field.

Reflection

In our model, reflection is used to simulate echos. We use a slightly different sound propagation model to support sound delay.

$$A_d = \max\{(A_s - \mathit{pdf} \times d) \times R,\ 0\}$$
$$\tau_d = \tau_s + \frac{d}{c} \tag{4}$$

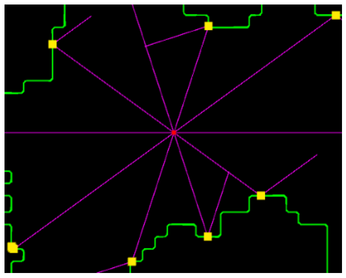
Figure 4: Reflection rays bouncing around the environment and creating the yellow reflection sources.

We basically use the same equation for the amplitude, but we have also added an equation for the delay, in which τ_d is the delay at the destination, τ_s is the delay at the source, d is the distance between the the two positions, and c is the speed of sound in our environment. The constant R controls the portion of sound to be reflected once it hits an obstacle. A smaller R may result in earlier algorithm termination.

Motivation. There are two types of reflection: specular and diffuse. Specular reflection is when a ray hits a mirror-like surface and ideally reflects with a definite angle. On the other hand, diffuse reflection happens when a ray hits a rough surface and is scattered in a random direction. In practice, most sound reflections become diffuse after the first few bounces (Savioja and Svensson 2015). Therefore, we do not rely on reflection to produce the main output of our model; instead, our sparse reflection model produces echoes of the primary sound, which are created by using a small number of rays with a sequence of specular bounces, followed by a diffuse reflection. Properly modelling the sound field using reflection is not cost-efficient, considering its minimal contribution to the player experience in video games, as we noted in our initial experiments. The advantage of our method is its simplicity to compute, as it normally does not need many rays or many bounces. It also does not require a specific listener for model computation, which is useful for real-time environments combining immobile sound sources with dynamic listeners.

Model Computation. To model reflection, we fire a number of rays from the primary sound source, in different directions. The rays will specularly bounce around the environment, and each time they hit an obstacle, a reflection source is created at the hit point. A reflection source has a certain amplitude, delay, and a specular reflection direction associated with it. The first two are calculated using (4), and the last one is the direction of the ray after it would specularly reflect at the hit point. Figure 4 shows how the rays travel through the geometry and create reflection sources.

Sound Computation. Similar to what we did for diffraction, we have a two-step method to compute sound for a listener. In the first step, we compute the effect of each individual reflection source using the following equations.

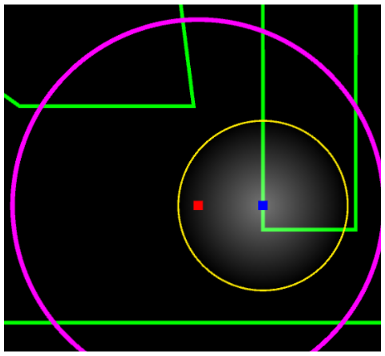
Figure 5: A transmission source is created on the edge of an obstacle. The blue dot is a transmission source and the yellow circle represents its coverage.

$$A = \max\{A_s - pdf \times d,\ 0\} \times (\vec{dir}_s \cdot \vec{dir})^{\alpha}$$
$$\tau = (\tau_s + \frac{d}{c}) \times rdf \tag{5}$$

Where A_s, τ_s, and $\vec{dir}_s$ are the amplitude, delay, and the specular reflection direction of the reflection source, $\vec{dir}$ is the direction from the source to the listener, *rdf* is the reflection delay factor, and α is a constant that adjusts how much emphasis is put on the last ray being specular or diffuse.

Using these equations on all reflection sources results in a list of (amplitude, delay) pairs. In other words, the listener receives the same sound as the primary source, but as a number of delayed signals with lower amplitudes. Once we have the pairs, we can decide on a combination technique to produce the final output of the reflection model. This could be a function of all pairs, or it could be as simple as grabbing only the first few pairs and discarding the rest.

Transmission

Transmission enables sound to go through obstacles. Our transmission model works similarly to the diffraction model; that is, it tries to create intermediate sources to extend sound coverage. Figure 5 shows a simple example of a transmission source, created on the edge of an obstacle.

Model Computation. Similarly to the reflection model, we fire rays in all directions around the sound source. As each ray hits an obstacle, it creates a transmission source at the hit location. The amplitude of this source is computed using the following equation.

$$A_d = \max\{(A_s - pdf \times d) \times T,\ 0\} \tag{6}$$

The constant T controls what portion of the sound should transmit through the obstacle. A smaller value may result in earlier algorithm termination.

The ray will then continue moving in the same direction to create other transmission sources. The amplitude of the new ones can be computed using the same equation, but this time, the source would be the most recent transmission source created along the ray. Figure 6 shows the coverage of the transmission model.

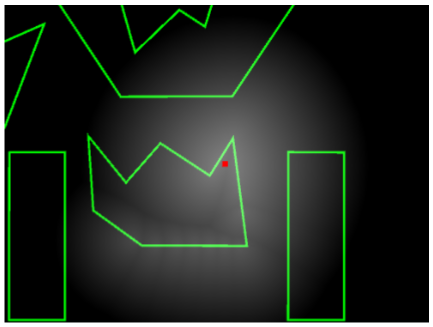
Figure 6: The Output of the Transmission Model

Sound Computation. Sound computation is identical for both diffraction and transmission, which results in a single equation combining both models.

$$A_l = \max\{\max\{A_d - pdf \times d_d,\ 0\} : d \in dlist \cup tlist\} \tag{7}$$

Where *dlist* and *tlist* are the list of diffraction sources and transmission sources, respectively.

Experiments and Model Evaluation

In this section, we evaluate our model from two aspects: performance and quality.

Performance Evaluation

We define four criteria to evaluate the performance of our sound model: number of rays fired, number of secondary sources created, model computation time, and sound computation time. Performance data is collected for two environments with different geometry complexities from the `movingai.com` maps, as shown in figures 7 and 8 (Sturtevant 2012). Tables 1 and 2 show the measurements for each of the diffraction, reflection, and transmission behaviors, separately. The computation times are averages of 50 attempts, with a maximum variation of 23 percent.

Concluding from the tables, the complexity of our model is directly related to the geometry. Both the processing time and the required memory are tied to the shape of the environment. Continuing, we will attribute the time complexity to the number of rays fired, and the memory complexity to the number of secondary sources created.

For diffraction, the number of sources required is at most equal to the number of vertices in the environment, since we only require a maximum of one diffraction source at a vertex. The time complexity of diffraction depends on how many vertices around the primary sound source should be processed. The number of rays fired may increase exponential to the number of vertices in the geometry; however, in practice, we can simplify our geometry and have quick computations.

For reflection and transmission, both the computation and the memory complexities depend on the number of rays we decide to shoot, and their hits and bounces in the environment; therefore, we can achieve different levels of complexity by adjusting the ray count.

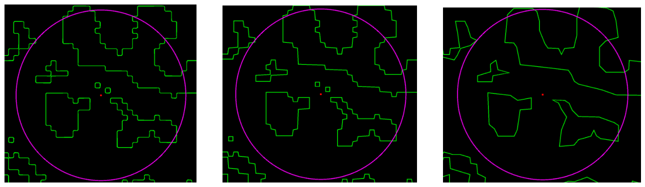

Figure 7: Performance test geometries. On the left, we have a complex map with 264 vertices inside the range circle. On the middle, we have simplified the geometry and the circle now contains 133 vertices. On the right, we have further simplified the environment and the vertex count is reduced to 67.

Property		Rays	Sources	MCT (ms)	SCT (ms)
Diff	S0	574	76	7.84	0.32
	S1	204	43	2.25	0.16
	S2	56	19	0.54	0.12
Ref	S0	40	22	0.10	0.03
	S1	37	19	0.10	0.03
	S2	34	16	0.09	0.02
Trans	S0	36	61	0.19	0.18
	S1	36	61	0.19	0.18
	S2	36	46	0.15	0.10

Table 1: Performance evaluation of different sound properties in the test environments from Figure 7, which are denoted by S0, S1, and S2, determining their simplification level. The last two columns stand for Model Computation Time and Sound Computation Time, respectively.

It should be mentioned that our model computation does not consider a specific listener position. If we aimed to compute sound for a certain listener, computation times would be much lower, especially for modeling diffraction. In that case, our problem would become similar to a search problem in which we only try to find a single answer rather than all of them.

Quality Evaluation

In this section, we begin by inspecting our model in terms of sound continuity and coverage, followed by a comparison to a more precise model for an accuracy evaluation.

We require our model to not have noticeable sound discontinuity. To evaluate the smoothness of the model, we compute the sound heatmap and pass it to an edge detector. The threshold for the detector is proportional to the distance between two points, based on (2). For example, a pair of points that are two units away from each other should have amplitudes that are at most $2 \times pdf$ different from one another, where pdf is the propagation decay factor of the environment. Figure 9 shows the output of the edge detector, which has only detected sound discontinuity at the edges of the obstacle.

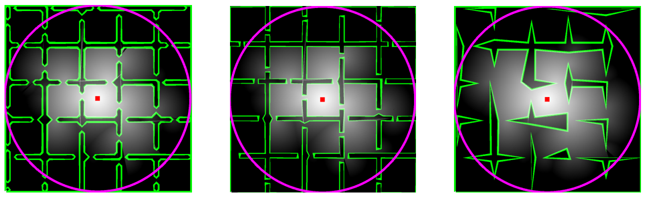

Figure 8: Performance test geometries. Similar to figure 7, different geometry simplification levels were applied, resulting in 235, 133, and 66 vertices in the range circle, respectively.

Property		Rays	Sources	MCT (ms)	SCT (ms)
Diff	S0	354	65	3.68	0.08
	S1	204	54	1.52	0.06
	S2	76	32	0.53	0.03
Ref	S0	25	15	0.08	0.02
	S1	24	14	0.07	0.02
	S2	23	13	0.07	0.02
Trans	S0	36	81	0.21	0.03
	S1	36	78	0.21	0.03
	S2	36	73	0.20	0.02

Table 2: Performance Data for Figure 8

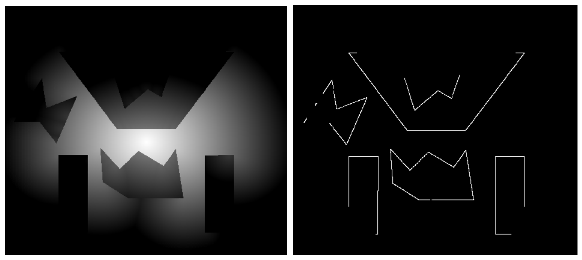

Figure 9: Sound continuity. Left: a heatmap of the output of our sound model. Right: the discontinuities in the heatmap.

Continuing with the idea of heatmaps, we can determine the area that a sound model covers. Table 3 shows the percentage of the range circle that is covered by each sound model, in the same scenario as figure 9. There is a large gap between a model with diffraction and a simple visibility polygon, which infers the necessity of diffraction in sound simulation. Transmission plays an important role as well. An interesting property of transmission is that we do not need that many transmission rays to get the most out of the model. In this case, 100 rays is almost as good as 10000, which results in cheap computation for effective coverage.

Initial informal testing indicates our approach produces a realistic and natural soundscape. For additional validation

Sound Model	Transmission Rays	Coverage
Visibility Polygon	-	17.70
Diffraction	-	49.38
Diffraction + Transmission	5	60.90
Diffraction + Transmission	10	62.06
Diffraction + Transmission	100	65.08
Diffraction + Transmission	1000	65.23
Diffraction + Transmission	10000	65.62

Table 3: Coverage Percentage of Different Sound Models

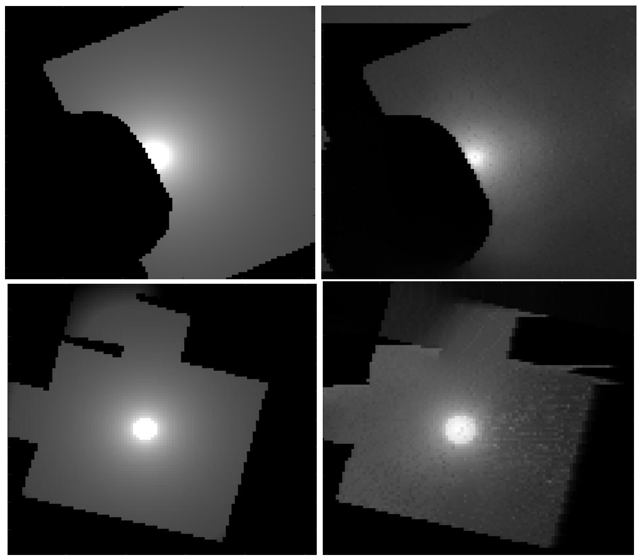

Figure 10: Comparison of our model with Project Acoustics'. On the left we have the output heatmaps of our model, and on the right we have the corresponding heatmaps produced by Project Acoustics.

we measure the accuracy of our sound propagation model by comparing its output to the output of Project Acoustics (Microsoft 2019). After hours of precomputation on the cloud, this acoustics engine creates a fairly accurate wave-based sound model that supports interactive experiences. To perform the comparison, we try to first visualize the models. Since Project Acoustics simulates sound for a 3D environment, we only visualize it for a single height level in the geometry, which corresponds to the height of a normal human character inside the environment. The visualization is a heatmap representing the volume of the output sound at each sampled point.

Figure 10 shows two cases for the comparison. We have modified our model propagation equation to resemble the custom audio curve that Project Acoustics uses for its output. Note that there are some inaccuracies in Project Acoustics' heatmap as a result of noisy sampling and not having a straightforward method to extract the volume levels for the 3D environment. Our model is able to maintain the general shape of the sound output compared to the accurate wave-based model, although we do lose some fine-grain detail, likely due to the lack of con/destructive interference. We believe the inaccuracies are justified by the very low resources required for the design, and by the perception that most video games do not require a great level of detail for sound simulation, even if they can still benefit from a model that addresses basic sound behaviors such as diffraction.

Conclusion and Future Work

We presented a geometry-based sound model for 2D interactive real-time video games. Our model is able to simulate diffraction, reflection, and transmission, using a small number of ray casts, which keeps the computation times low and supports dynamic geometries. We evaluated the model based on its performance and quality, and came to the conclusion that it can provide a cheap method to achieve relatively reasonable accuracy, which provides extra information to the player and increases the player experience.

To further evaluate the model, we are planning to perform a human study to collect data on the usefulness of the model to players. Our goal is to test whether the model provides information about the geometry solely based on sound propagation. This information could be about the shape of the geometry, or the surface material of the walls and obstacles. We are also aiming to find the sound accuracy levels that indicate whether a sound model is perceived as beneficial to a video game, or is it overkill and is using resources that other components of the game could better utilize. If interested in our work, you can download an early version of the user study software in the following link: https://github.com/TinSlam/SoundPropagationDemo

Acknowledgements

This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) grant RGPIN-2019-05213.

References

Allen, J. B.; and Berkley, D. A. 1979. Image method for efficiently simulating small-room acoustics. *The Journal of the Acoustical Society of America* 65(4): 943–950. doi:10.1121/1.382599.

Boev, S. 2015. Sound Propagation in Hitman. https://www.gdcvault.com/play/1022774/Sound-Propagation-in. [Online; accessed 2-August-2021].

Botteldooren, D. 1995. Finite-difference time-domain simulation of low-frequency room acoustic problems. *The Journal of the Acoustical Society of America* 98(6): 3302–3308. doi:10.1121/1.413817.

Brown, E.; and Cairns, P. 2004. A Grounded Investigation of Game Immersion. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '04, 1297–1300. New York, NY, USA: Association for Computing Machinery. ISBN 1581137036. doi:10.1145/985921.986048.

Cowan, B.; and Kapralos, B. 2015. Interactive rate acoustical occlusion/diffraction modeling for 2D virtual environments amp; games. In *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, 1–6. doi:10.1109/IISA.2015.7388078.

Dion, L.-P. 2017. Game Design Deep Dive: Dynamic audio in destructible levels in Rainbow Six: Siege. https://www.gamasutra.com/view/news/288565/Game_Design_Deep_Dive_Dynamic_audio_in_destructible_levels_in_Rainbow_Six_Siege.php.

Foale, C.; and Vamplew, P. 2007. Portal-Based Sound Propagation for First-Person Computer Games. In *Proceedings of the 4th Australasian Conference on Interactive Entertainment*, IE '07. Melbourne, AUS: RMIT University. ISBN 9781921166877.

Funkhouser, T.; Carlbom, I.; Elko, G.; Pingali, G.; Sondhi, M.; and West, J. 1998. A Beam Tracing Approach to Acoustic Modeling for Interactive Virtual Environments. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, 21–32. New York, NY, USA: Association for Computing Machinery. ISBN 0897919998. doi:10.1145/280814.280818.

Funkhouser, T.; Tsingos, N.; Carlbom, I.; Elko, G.; Sondhi, M.; West, J. E.; Pingali, G.; Min, P.; and Ngan, A. 2004. A beam tracing method for interactive architectural acoustics. *The Journal of the Acoustical Society of America* 115(2): 739–756. doi:10.1121/1.1641020.

Krokstad, A.; Strom, S.; and Sørsdal, S. 1968. Calculating the acoustical room response by the use of a ray tracing technique. *Journal of Sound and Vibration* 8(1): 118–125. ISSN 0022-460X. doi:10.1016/0022-460X(68)90198-3.

Kulowski, A. 1985. Algorithmic Representation of the Ray Tracing Technique. *Applied Acoustics - APPL ACOUST* 18: 449–469. doi:10.1016/0003-682X(85)90024-6.

Mahjoob, M.; and Malakooti, S. 2008. Acoustic simulation of building spaces by ray-tracing method: Prediction vs. experimental results. *23rd International Conference on Noise and Vibration Engineering 2008, ISMA 2008* 4: 2303–2312.

Microsoft. 2019. Project Acoustics. https://docs.microsoft.com/en-us/gaming/acoustics/what-is-acoustics.

Savioja, L. 1999. *Modeling Techniques for Virtual Acoustics*. Ph.D. thesis, Helsinki University of Technology.

Savioja, L.; and Svensson, U. P. 2015. Overview of geometrical room acoustic modeling techniques. *The Journal of the Acoustical Society of America* 138(2): 708–730. doi:10.1121/1.4926438.

Strikwerda, J. C. 2004. *Finite Difference Schemes and Partial Differential Equations, Second Edition*. Society for Industrial and Applied Mathematics. doi:10.1137/1.9780898717938.

Sturtevant, N. 2012. Benchmarks for Grid-Based Pathfinding. *Transactions on Computational Intelligence and AI in Games* 4(2): 144 – 148. URL http://web.cs.du.edu/~sturtevant/papers/benchmarks.pdf.

Takala, T.; and Hahn, J. 1992. Sound Rendering. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '92, 211–220. New York, NY, USA: Association for Computing Machinery. ISBN 0897914791. doi:10.1145/133994.134063.

Taylor, M.; Chandak, A.; Mo, Q.; Lauterbach, C.; Schissler, C.; and Manocha, D. 2012. Guided Multiview Ray Tracing for Fast Auralization. *IEEE Transactions on Visualization and Computer Graphics* 18(11): 1797–1810. doi:10.1109/TVCG.2012.27.

Tsingos, N.; Funkhouser, T.; Ngan, A.; and Carlbom, I. 2001. Modeling Acoustics in Virtual Environments Using the Uniform Theory of Diffraction. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, 545–552. New York, NY, USA: Association for Computing Machinery. ISBN 158113374X. doi:10.1145/383259.383323.

Wikipedia contributors. 2021. Huygens–Fresnel principle — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Huygens%E2%80%93Fresnel_principle&oldid=1035947717. [Online; accessed 2-August-2021].

Zamith, M.; Passos, E.; Brandão, D.; Montenegro, A.; Clua, E.; Kischinhevsky, M.; and Leal-Toledo, R. C. 2010. Sound Wave Propagation Applied in Games. In *2010 Brazilian Symposium on Games and Digital Entertainment*, 211–219. doi:10.1109/SBGAMES.2010.29.