

# **Real-Time Acoustics Modeling in Games**

Joachim Bengtsson

## **Abstract**

*Abstract* This paper evaluates algorithms for modeling the acoustic properties of sounds in real time in games, and in particular evaluates J. Borish' "Extension of the image model to arbitrary polyhedra" [1] through implementation inside the RMS game engine. **[REST IS TBD.]**

## 1. Introduction

It's common for today's games to use the geometry surrounding the listener (that is, the player) to calculate occlusion and some dampening, but they still often use more traditional methods for applying echo and reverb, through DSP effects instead of real physical simulation; mapping regions in space to presets of DSP settings.

As gaming hardware and commonplace PCs grow ever faster, however, and get multiple execution units (such as in the Intel Core Duo/Quad), it is becoming more and more realistic to do a more sophisticated modeling of the acoustics of the player's environment, simulating the environment in real-time without "cheating" with DSP.

Moreover, creating a "real", realistically-simulated sound environment has been a dream for me for many years; I first envisioned it maybe seven years ago, although I could hardly code then.

This essay will go through primarily the Image-Source Model of acoustics modeling, but also talk about some other possible modeling methods and compare them to the Image-Source Model. It'll see what other games use. The essay will go on to discuss the architecture and implementation of a game sound engine that uses the Image-Source Model in real-time, and its subjective quality and objective performance. Finally, it will conclude whether this is good enough, and list some possible enhancements to the engine.

I will try to answer the following questions:

- What methods are available [(apart from [Borish] and ray tracing)] for DSP-free acoustic modeling?
- What methods of acoustic modeling are used in games today?
- How can the image model be implemented inside the RMS game engine?
- Will the image model perform within realistic performance requirements (that is, be able to run concurrently with the rest of the game engine while still having a smooth, fluid gaming experience)?

I expect to find that the algorithm in [Borish] to suit the task performance-wise and quality-wise to work in the sound engine in RMS.

## Hypothesis

With proper limits, the Image-Source Model can model the acoustic environment in real-time in a first person shooter game.

## Variables

Variable	Bounds
Input	<ul style="list-style-type: none"><li>▪ Simultaneous sounds</li><li>▪ Orders of reflection</li><li>▪ Environment detail</li></ul>
Output	<ul style="list-style-type: none"><li>▪ Detail</li><li>▪ Performance</li><li>▪ Subjective performance</li></ul>
Environment/ Boundary	<ul style="list-style-type: none"><li>• Effect of other components in the engine is ignored, but kept as constant as possible</li><li>• Architecture of the engine might have a great impact on performance, but at least it's not changing.</li></ul>

*Table 1.1: Input, output and boundary variables for this report.*

## Methodology

In order to find an answer to the above questions, I plan to

- Define the task of 'Acoustic modeling'
- Outline what methods of acoustic modeling are available, and their feasibility for real-time application
- Implement the Image Model, which I believe is the most feasible model, in the game engine RMS <sup>1</sup>
- From this, it should be possible to evaluate if
  - the model gives a good-sounding result
  - the model works in real-time without having to be limited to the point it gives no benefit to the sound model. [Dålig formulering]

## Time Plan

v37	Kick-off
v38	
v39	
v40	
v41	
v42	
v43	
v44	Written about methods of acoustic modeling
v45	
v46	<b>control seminar</b>
v47	Implementation of 2D reference model of image modeler
v48	
v49	
v50	Implementation of 3D image modeler
v51	Written about engine impl, done performance measurments
v52	
v1	Written conclusion, about enhancements, researched other games
v2	<b>Seminar</b>

## 2. Acoustic Modeling?

Acoustics is the science of sound and its mechanical waves. Many games work well without any explicit modeling by only playing the **direct sound**, matching a position in the game world to relative volume levels in the user's speakers. However, when the world you are trying to model gets more complex, there are a number of things you likely want to model. Below is a number of these.

**Occlusion** Is there a wall between me and the sound source? Then it probably shouldn't be heard.

**Reflection** Sound bounces off of surfaces, and may thus reach the listener many times at different strengths and at different times and even different pitch. Reflection causes **echo**, **early reverberation** and **late reverberation**. The latter two are considered separately because they are commonly modeled separately [Savioja, p37].

**Diffraction** Diffraction can cause sound to bend past edges, thus

allowing you to hear sound through doors or past objects that occlude the source from the listener.

**Attenuation** The strength of the sound decreases with distance, from air absorption and absorption caused by the reflecting surfaces [Savioja, p35-36]

**Auralization** Modeling the listener as two point 'microphones' in the virtual world yields a rather unconvincing result. A common solution is to have many points (i.e. surround sound systems with 5 or 7 speakers). Another solution is to model how the outer ear, head and torso affect the sound with a *Head-Related Transfer Function*, or HRTF.

[? More?]

This paper will cover occlusion and reflection, basic attenuation and basic surround auralization.

### 3. Methods of Acoustic Modeling

There are a number of ways to compute an acoustic model of a room. However, I will concentrate on the following three.

- Wave based modeling
- Ray traced modeling
- Image Source-modeling (which is a form of ray tracing as well)

Each is given an overview and its viability in games is analyzed below.

#### Wave Based Modeling

Modeling sound as waves is of course the most realistic modeling technique, as sound is a form of wave. This involves using the wave equation, which is a partial differential equation. Since a real time application such as a game involves large amounts of complex geometry, the equation must be solved numerically. There are a number of available techniques such as FEM<sup>2</sup>, BEM<sup>3</sup> and FDTD<sup>4</sup>. All are rather complex, although Savioja succeeds in modeling a 500 poly room and many other acoustical properties on three 300-600MHz SGI computers using the digital waveguide mesh (a form of FDTD) in real time, so it's obviously possible and feasible to use this modeling technique for real time applications, especially with today's hardware.

#### Ray Traced Modeling

Regular ray tracing is conceptually simple but computationally expensive. In the most basic implementation, rays are fired from the sound source either in random directions, or uniformly over a sphere. Whenever a ray hits the listener, the sound, direction and strength is recorded, and either played like that or further preprocessed.

[\[Read a paper on the subject and find more algo details and pitfalls\]](#)

Ray tracing only gives an approximation of the acoustic model of the room, since the pattern and number of rays shot from the sound source isn't likely to perfectly model how the sound actually emanates.

#### Viability for real-time applications

Compared to optical ray tracing—which is generally regarded as not

worth doing in real time—acoustic ray tracing seems doable, if only because specular reflections on rather large surfaces need to be considered.

### Image Modeling

Ray tracing and the image-source model are both ray based methods (in contrast to the wave based method in 3.1). However, while the ray tracing described in 3.2 ‘mirrors’ the rays by reflecting them on surfaces in rooms, the image model mirrors the rooms themselves.

Say that we have a rectangular room with a single sound source (such as the sound of a foot step)



Figure 3.1: A room with a sound source

This room can also be represented as a number of mirror images. If we now have a listener in this room, this listener can't just hear the sound from that one original sound source, but also the sound from all the mirror sound sources.

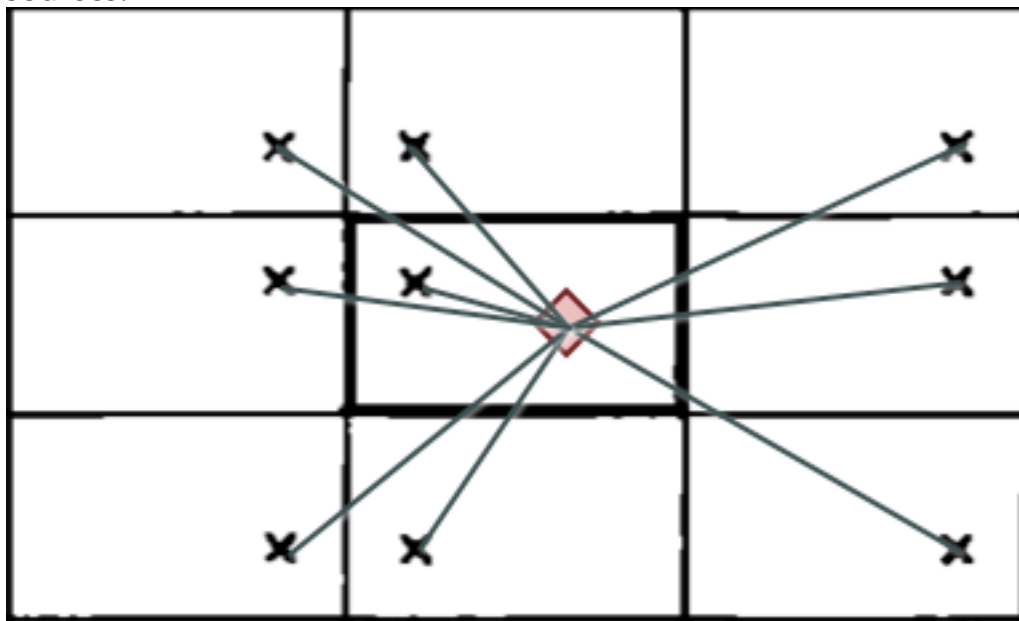


Figure 3.2: A room with mirror rooms

Each of these mirror sound sources (henceforth “virtual voice”) represent a reflection in a wall. The bent red line is the path of the sound when it is reflected once in the northern wall and then received by the listener. The green line has the exact same length and direction as the red line when it hits the listener; the virtual voice in the room being mirrored over the northern

wall correctly models a reflection in said wall.

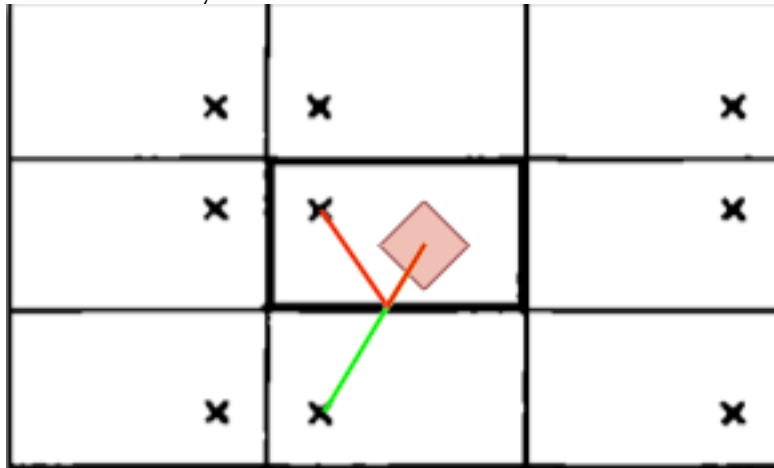


Figure 3.3: Real and modeled sound path

This can be extended indefinitely until the reflected path is so long that the sound can no longer be heard, and the model can be deemed complete.

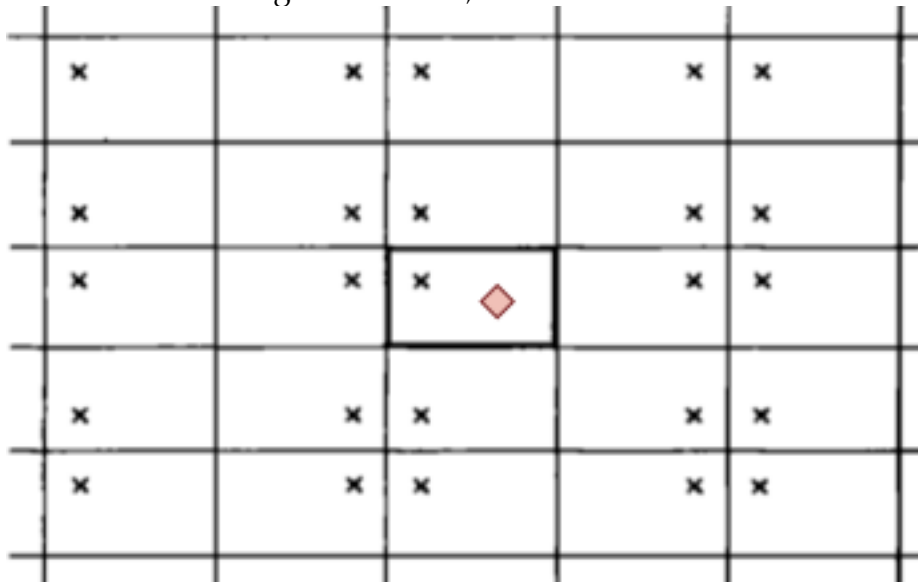


Figure 3.4: Extending the model

Jeffrey Borish describes in his '82 paper how this idea can be extended to arbitrary polyhedra, such as the triangles that all make up the geometry in a game.

The image model also has the advantage over ray tracing that the calculation deriving all the virtual voices for a sound source only have to be calculated once, and all sound reflections will continue to sound correctly, even if the listener rotates or moves around, as long as the sound source itself doesn't move. Table 4.1 lists in detail the conditions under which the model needs to be recomputed when using image modeling.

## Extending this to arbitrary polyhedra

$$[\mathbf{r} = \mathbf{s} + 2d\mathbf{n}]$$

### Virtual voice validity: false image points and visibility

Each created virtual voice must fulfill three criteria, according to [Borish, p1828, 1829]:

- It must be valid: the image point must fall within the same room
- It must be within a vanishing distance (voices reflected many times are too far away to be heard)
- It must be visible: there must not lie geometry between the virtual voice and the listener. However, it is still kept, as reflections generated from this reflection may be visible, until all child reflections are found.

### Viability for real-time applications

If an image modeler would be allowed to work until completion, it would yield a complete model of the acoustic space. However, it would be very expensive, as it would have to generate an exponentially growing number of room models. This generally involves calculating up to eight levels of reflections [Borish, p1830]. Thus, it will have to be cut short, giving an incomplete model which lacks late reverberation.

For the image model to be viable in real time applications like games, it would need to be capped to early reverberation, and combined with other methods, such as computing the late reverberations using ray tracing [Savioja, p27].

### Methods Used in Games Today

[[Måste hitta spel att undersöka. Half-Life 2 är en kandidat eftersom nästan hela spelmotorn är släppt. Men utöver det?]]

### Fmod

???

#### **4. Studying the Image Model and its Implementation in RMS**

The core of the sound engine in RMS is an ordinary ordinary game sound engine. It models a game world of omnidirectional point sources that may play, loop, be pitched, have position, velocity and acceleration; then maps these sources to volume levels for each user speaker; and generates the corresponding sound streams. The core also needs to take the speed of sound into consideration, delaying sounds that are far away, in order for echoes or reverbs to be even possible to model (see below). Also, in order for there to be a smooth transition between delays when this distance changes, the core must also adjust the pitch of the sound accordingly (i e, implement the doppler effect).

In addition, the core has an interface to modeling module. The scope of the modeling taking place is just to take the shape of the geometry of the environment into consideration in order to calculate:

- Occlusion; visibility of each sound source (whether it should be heard and isn't blocked by geometry)
- Reflections (by which one sound source may be heard multiple times coming from many different directions)

There are many other acoustical attributes that can be modeled (See chapter 2), but these are beyond the scope of this paper and implementation.

## Architecture

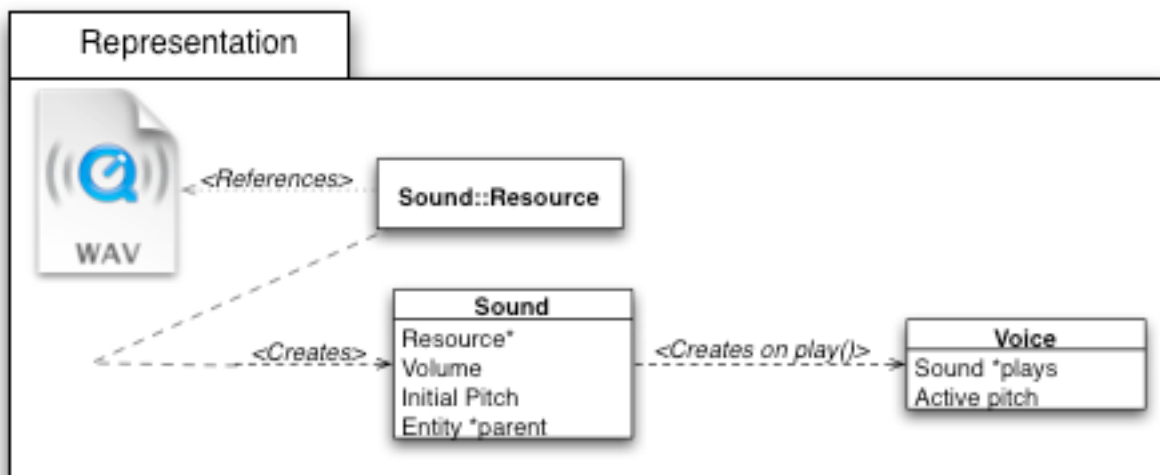


Figure 4.1: The representation package

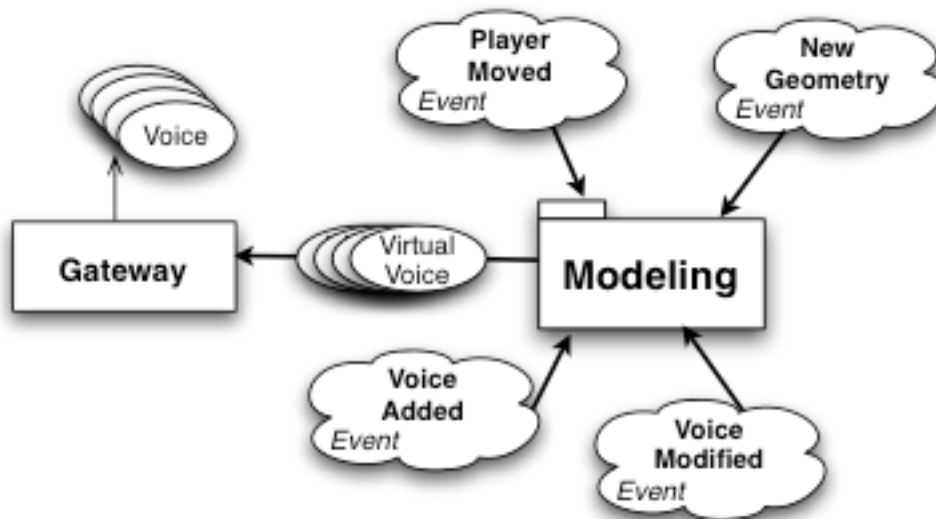


Figure 4.2: The Gateway and Modeling packages

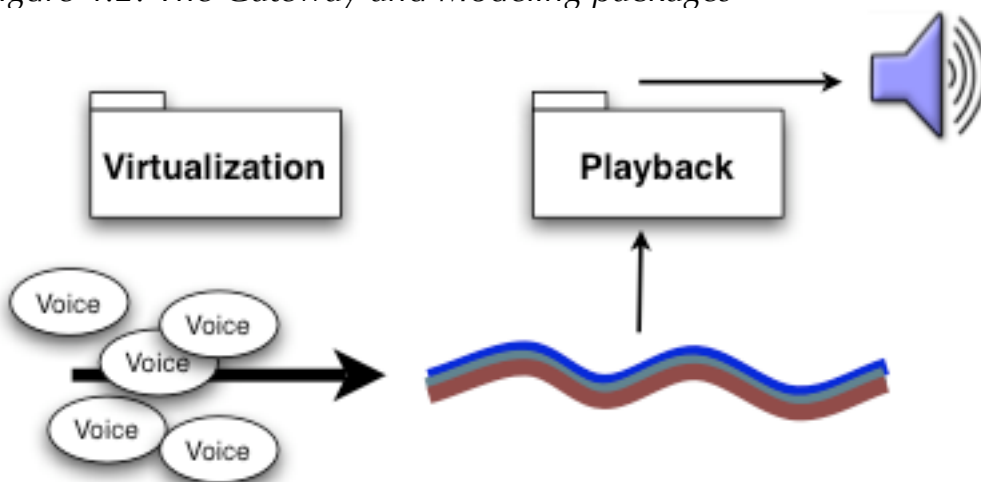


Figure 4.3: The Virtualization and Playback packages

The RMS sound engine is divided into four packages and one coordinator

(the Gateway).

## **Sound Engine - Generation & Auralization**

The generation and auralization packages are what I previously described as the “core of the sound engine”. These behave as in most other sound engines, and are contained in the packages in fig. 4.1 and 4.3.

The Representation package (Fig. 4.1) represents

(a) resources on disk

(b) Sounds; a preset representing a resource, combined with attributes such as pitch and volume, and bindings to a game world entity, giving Sounds the ability to move with objects, have a velocity, etc.

(c) Voices; currently playing Sounds, that can be paused, pitched, etc.

The Gateway class (fig. 4.2) contains the current model of the aural game world.

The package Virtualization (fig. 4.3) asks the Gateway for this model — i.e., all the Voices currently playing — and flattens it into sound streams, one for each speaker. While doing this, it also applies some acoustic physics to simulate the doppler effect (although this functionality would be better suited in Modeling, it ends up in Virtualization because of the way the auralization is implemented [Kan jag inte bara fixa det??]).

The Playback package is the hardware abstraction layer, responsible for outputting the streams into their corresponding speaker.

## **Acoustics Modeling**

The Modeling package (fig. 4.2), however, is of my own design. It behaves as an agent in an independent working thread, building a representation of the 3d environment and its geometry, and tells the gateway of *virtual voices* as they are found. These virtual voices are just ‘symbolic links’ to real voices, and represent reflections (See chapter 3.3).

The Modeling package listens for four categories of events that may change or invalidate the current model (Adaption from [Savioja, p40]).

Event	Effect
<b>Player moved</b>	Invalidates visibility calculations. Modeling must regenerate information about whether a voice can be heard from the current position with the current geometry.
<b>New geometry</b>	Signals removal, addition or modification of surroundings (usually in response to a change in frustrum culling). This class of events invalidates visibility, but may also invalidate virtual voices dependent on geometry that was removed or modified; and require the generation of new virtual voices for new geometry
<b>Voice added</b>	Requires that a new set of virtual voices are created to model reflections for this voice
<b>Voice modified</b>	May mean that a voice was removed: all its virtual voices should be removed; or a voice was moved: all its virtual voices should be recalculated for the new position

*Table 4.1: Events in the RMS sound engine that require action by the Modeling package*

## Implementation

[Generate virtual voices level-by-level, or depth-first tree traversal?

Level by level: Can be incremental and let the engine continue running pretty fast, after only a level or two.

Depth-first: Tremendous decrease in memory requirements. However, will probably sound very weird if played before the model is done.

Börja breadth-first i två nivåer, gå sedan depth-first. Delaya uppspelande i rummet tills de där två första nivåerna är klara?

]

## Subjective Opinions about the Quality of the Output

### Performance Measurements

#### Latency

#### Processing time

## **5. Conclusion**

### **Analysis**

**What do my numbers say?**

**Bad performance inherent to the IM or my implementation?**

### **Possible enhancements**

#### **Sound generation**

#### **Late reverberation**

As stated in the introduction to chapter 4, late reverberation is []

Through ray tracing parameters in a pre-compilation stage.

### **Binaural reproduction**

HRTF

#### **Adherence to room surface materials (material reflection filters)**

Filters

#### **Transfer through solids**

### **Wrap-up**

## References

- [Borish] J. Borish, Extension of the image model to arbitrary polyhedra, Acoustical Society of America Journal Vol. 75, 1984.
- [Savioja] L. Savioja, Modeling Techniques for Virtual Acoustics, <http://www.tml.tkk.fi/~las/publications/thesis/>, 1999
- [Krokstad et al] A. Krokstad, S. Strom, S. Sorsdal, Calculating the acoustical room response by the use of a ray tracing technique, J. Sound Vib., 8(1):118-125, 1968

## Appendix A: Glossary

**RMS Engine** The RMS engine is the game engine used in Friendly Stapler's "Rymdvarelses Mot Soldater" (third year group project in BTH's Game Programming's class of 2004, in which I work). See [friendlystapler.se](http://friendlystapler.se).

---

<sup>1</sup> See <http://www.friendlystapler.se/> for a detailed description of this engine.

<sup>2</sup> Finite Element Method

<sup>3</sup> Boundary Element Method

<sup>4</sup> Finite-Difference Time-Domain method