Sound Rendering

Tapio Takala* and James Hahn†

*Helsinki University of Technology
02150 Espoo, Finland

†The George Washington University
Washington, DC 20052

Abstract

We present a general methodology to produce synchronized soundtracks for animations. A sound world is modeled by associating a characteristic sound for each object in a scene. These sounds can be generated from a behavioral or physically-based simulation. Collision sounds can be computed from vibrational response of elastic bodies to the collision impulse. Alternatively, stereotypic recorded sound effects can be associated with each interaction of objects. Sounds may also be generated procedurally. The sound world is described with a sound event file, and is rendered in two passes. First the propagation paths from 3D objects to each microphone are analyzed and used to calculate sound transformations according to the acoustic environment. These effects are convolutions, encoded into two essential parameters, delay and attenuation of each sound. Time-dependency of these two parameters is represented with keyframes, thus being completely independent of the original 3D animation script. In the second pass the sounds associated with objects are instantiated, modulated by interpolated key parameters, and summed up to the final soundtrack. The advantage of a modular architecture is that the same methods can be used for all types of animations, keyframed, physically-based and behavioral. We also discuss the differences of sound and light, and the remarkable similarities in their rendering processes.

CR Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Animation.

Additional keywords: audio, multimedia, soundtrack, physically-based modeling, virtual reality.

1 Introduction

In traditional hand-drawn animation, sound effects and their synchronization to the motion are of utmost importance. Often the whole story is first composed as a musical bar sheet, on which the movements are designed based on key events [17]. Sounds are usually dubbed onto the action after the motions are generated but this is a tedious manual process and requires expertise to synchronize correctly. In computer animation, similar techniques have been used in key-framing and rotoscoping. However, most of the computer animations produced have emphasized the motion itself and have added sound during post-production. The major reason is that there does not yet exist a general framework for sound synchronization. This is especially true in behavioral or physically-based motion control where the lack of absolute control have made the motion-to-sound synchronization difficult. A notable exception from the mainstream is the film "More Bells and Whistles" [9]. There, both music and animation were designed as a single score, which served as a synchronizing controller for both MIDI-driven music synthesizers and animated synthetic actors.

Video games and multimedia installations [22] have also relied on synchronized sound. Sound as another dimension of interaction has been utilized in user interfaces to symbolize objects and to draw attention to alerts [6, 2]. In scientific visualization, sound can be used to characterize objects' attributes and to emphasize timing of events [10]. Simulations of sound wave propagation have been done to analyze room acoustics or dummy head models [21], and special hardware built to perform these effects in real time [5], but to our knowledge none of these has been combined with animated graphics except for visualizing the acoustical parameters [15].

1.1 The Nature of Sound

As waves, both light and sound behave similarly in many ways. Both propagate as wavefronts, spreading in all directions according to the Huygens' principle. They reflect and refract whenever there are discontinuities in the medium, and obstacle edges cause diffraction. However, their different propagation speeds and wavelengths cause essential practical differences. Light travels instantaneously without noticeable delay, and its wave nature only appears when it interacts with structures of microscopic size. On the other hand, the speed of sound easily causes delays perceived as echo or reverberation. There is considerable diffraction, too, because the wavelength is of human proportions – the sound propagates around the edges of objects. For these reasons, sound propagation cannot be accurately traced using simple ray optics.

The human sensory organs for light and sound are sensitive to different characteristics. The eye has good spatial resolution, whereas the ear integrates signals from all directions. Light is sensed as a color spectrum integrated over a period of time, and not as individual waveforms. Sound instead is perceived in time rather than as a stationary spectrum in frequency domain.

1.2 Sound Signals and Convolutions

Sound is a one-dimensional signal that can be represented as an intensity regularly sampled in time. Out of possible representations this appears to be the most general and suitable for our purposes. The concepts of pitch and note length are akin to traditional music but are unable to describe speech or noise. A spectral distribution (Fourier transform) with phase and amplitude of each component is theoretically complete, but is intuitively difficult for other than stationary waveforms, and is not at all suitable for time-dependent effects like echo. Similar arguments apply to other integral representations, like Laplace transform for example.

As an object generates sound in a three-dimensional environment, it acts like a light source in image rendering. Its signal is propagated in all directions, reflected and refracted by other objects and participating media, and finally caught by a receiver. The received signal is an integral of signals via multiple simultaneous paths from the emitter to the receiver. To compute this integral, each possible path can be independently traced through the environment, its effects on the sound signal calculated, and the results composed as a convolution of the original sound:

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

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

The convolution weighting function $w(t,\tau)$ corresponds to the paths of different lengths causing different delays and attenuations for the same signal. Generally it is time-dependent, continuous and semi-infinite. For computational purposes it has to be discretized into a finite number of samples. The script format we are going to present is a possible way to do this, emphasizing the role of moving objects as the cause of time-dependencies in convolution.

1.3 Outline of the Paper

In this paper we present a structured methodology for generation, representation, and synchronization of sounds in computer animation. In section 2, we will outline the overall system structure as a pipeline with sound signals flowing through it. This is followed by a detailed description of each module of the system. First, (section 3) we describe how natural sounds may be generated by physical phenomena and how characteristic sounds are attached to geometric objects in three-dimensional space (section 4). Next, we show how the modulation of signals due to their propagation in an environment can be represented as convolutions (section 5). The final process is the rendering of key-framed sounds by resampling the original signals (section 6). Examples of various effects are also presented.

2 Sound Rendering Pipeline

The basic idea in our approach is that sounds are treated as one-dimensional data objects associated with geometric objects of a three-dimensional world. Each sound is a time-dependent signal. Much like a graphical texture map, it is located on a geometric object, but extends in time instead of spatial dimensions. It may be bounded or semi-infinite. In the latter case it can be either a repeated waveform or a fractal-like continuously changing noise pattern.

We call *sound rendering* the process of forming the composite soundtrack from component sound objects. This is due to its methodological similarity to image rendering and texture mapping and its close connections to the same geometric data. To keep the process manageable, we have divided it into a pipeline of processes, as shown by the overall system architecture in figure 1. First process (I) is the generation of the prototype sounds that are characteristic of each object and its interaction with others. These can be either based on modal analysis of elastic body vibrations, or they can be user-defined (recorded or synthesized). In the next process (II), the prototype sounds are instantiated and attached to moving three-dimensional objects, based on the same physical or behavioral simulation that controls motion. Following that, (III) the modulatory effects of the three-dimensional environment are computed as transformations from sounds on the objects to sounds at a microphone. These time-dependent sound transformations are represented independently of the original object sounds, and are finally (IV) applied to the sounds in the last process. Thus just a reference of prototype sound goes through processes II and III, and the samples are actually used only in process IV.

The sounds produced by objects may be endogenous like barking, yelling or talking. Such prototypes can be digitally recorded, analogous to using digitized photographic images as textures. Alternatively they may be synthesized by behavioral models, like the buzz of a bee determined by the way it flies. Other, more physically-based sounds emanate from vibrations of bodies, caused by interactive

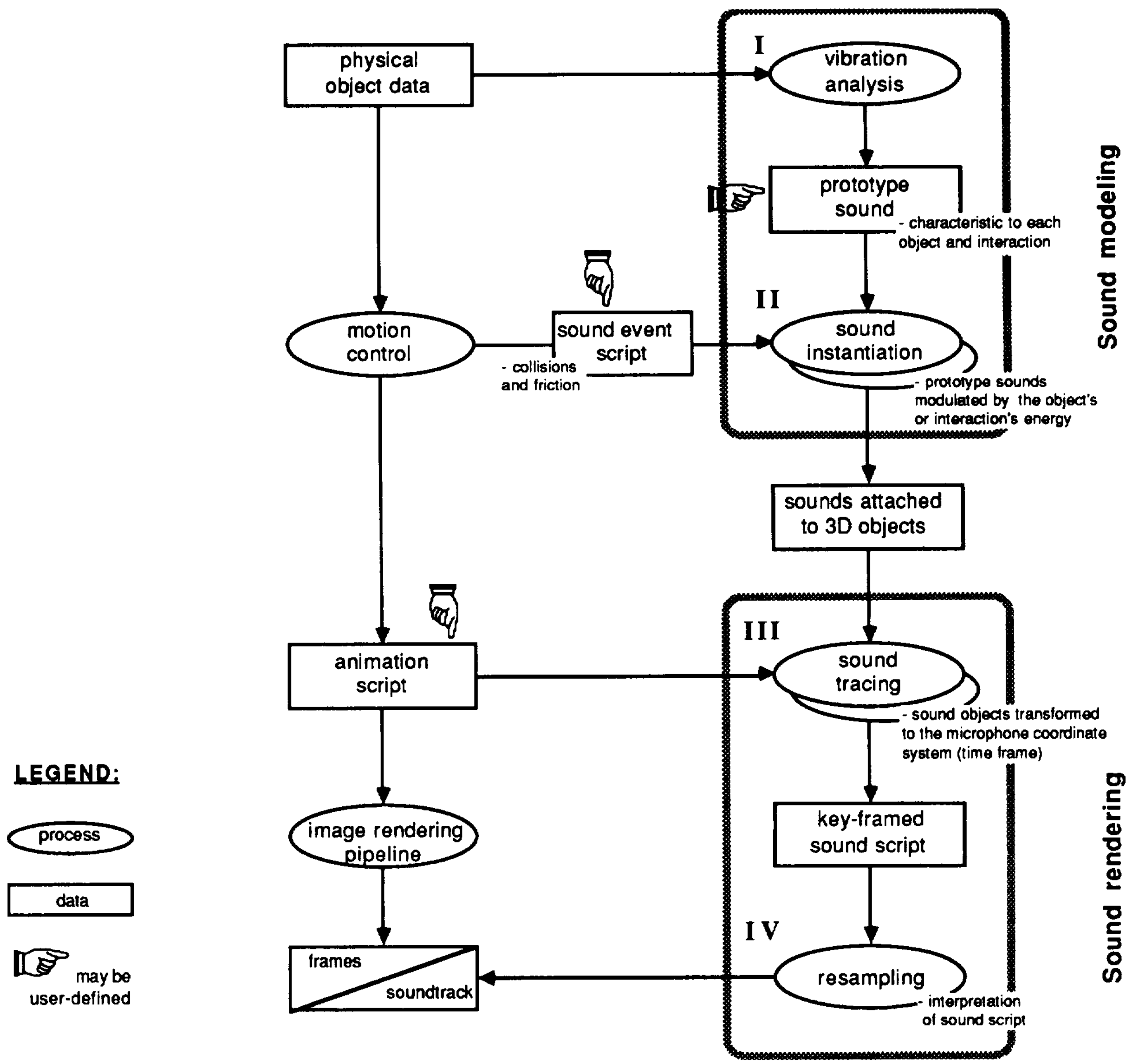


Figure 1: Parallel pipelines for image and sound rendering.

forces between objects and their environment. Examples of these are collisions, friction, and turbulent air flow around an object. Spectral composition may be used for these sounds, as has been used for textures [12].

In our system, we separate a sound and its association to objects. The characteristic potential sound is represented as a sampled signal or procedure, whereas its instantiation is described in a *sound script*. The script describes when and how a characteristic sound is actually generated, determined by certain events in the animation, like physically calculated collisions. Attachment of prototype sounds to objects located in space-time is comparable to the geometric mapping of textures to object's surfaces.

For an animation, the sounds are recorded by virtual microphones usually attached to a virtual camera. A sound may undergo various modulations during its course from a geometric object to the microphone, through the simulated environment. It is attenuated due to the dispersion of energy in space, and is delayed by the traveling time along a sound path. The emission of sound from an object may vary by direction, and it may be reflected, refracted or obscured by other objects, much as light waves are. The sound's spectrum (sound color or timbre) may be modified in these interactions, as well as due to the participating medium. Finally, the intensity of a sound heard by a microphone may vary due to its directional sensitivity. These processes correspond to the mapping of textures from geometric objects to the image plane. This is why we use the terms *object space* and *image space* for the time-frames of sounds as well as for the corresponding geometric coordinate systems.

Since sound signals are additive, all the effects described above are computed in a resampling process independently for each emitted sound, and summed up (mixed) to make a complete soundtrack. In stereo, this will be done separately for each channel.

We have implemented an experimental sound renderer as a collection of software modules that can be pipelined together. Our implementation is written in C language and currently runs on a Macintosh IIci. For this equipment the sound intensities are represented with only 8-bit sample resolution, causing noticeable digitization noise. For higher quality sounds, we have used AIFF files with 16-bit resolution, from which the sounds are reproduced with Silicon Graphics Indigo. As an alternative, we have also considered a MIDI-based interface to a high-quality music synthesizer, with object sounds selected from its library and modulated in real time. However, it has appeared rather difficult to find suitable MIDI-driven components to produce all the effects needed.

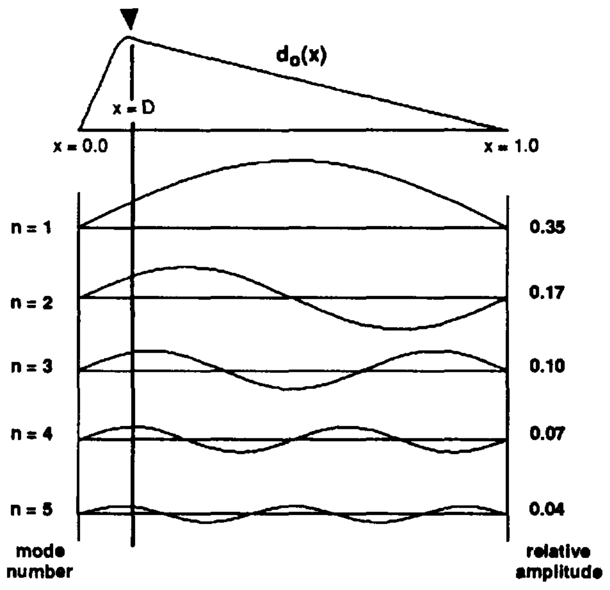


Figure 2: Vibration modes of a guitar string.

3 Sound Synthesis

Sound can be synthesized from physical principles in the same way as physically-based motion control generates motion [1, 16]. However, this is a much more difficult problem, because many material factors involved are poorly known. In this sense it is analogous to determining color of a material based on its physical properties [8]. Some heuristics can be used to simplify the process for certain classes of sounds.

3.1 Vibration of Bodies

Many sounds are generated when objects vibrate after being struck. The process is difficult to model for arbitrary objects without using complex finite-element models [14]. One way to attack this problem is to calculate the natural vibration modes of a body, and to approximate a complex vibration as a weighted sum of them [11]. Though not strictly correct for transient vibrations, this approach can be developed to give subjectively quite acceptable results. Crucial in it is the ability to calculate the appropriate weight for each mode, based on the body's deformation at a collision. This is generally a complex problem. In the following we demonstrate it with a simple example.

In thin bars or strings of musical instruments, the natural harmonics are standing waves with displacements along the string given as $d_n(x) = w_n \sin(n\pi x)$. The vibration modes are orthogonal, so assuming the initial rest position of the string as $d_0(x)$, the weighting factor of each mode is the integral

$$w_n = \int_0^1 d_0(x) \cdot d_n(x)\,dx$$

For example, we assumed the initial displacement to be triangular, as when stretching a guitar string with a pick (figure 2). Then

$$d_0(x) = \frac{x}{D} \text{ for } x \le D, \quad d_0(x) = \frac{1-x}{1-D} \text{ for } x \ge D$$

and the weights will be

$$w_n(D) = \left(\frac{1}{D} + \frac{1}{1-D}\right)\left(\frac{1}{n\pi}\right)^2 \sin(n\pi D) \ .$$

The sum of the weighted components is multiplied by an exponential decay function, with damping factor proportional to the frequency. This gives a waveform corresponding to the displacement of string at the point where it was struck, which can be used as a prototype collision sound for the string. Each stroke of the string will then generate an instance of that sound, with amplitude proportional to the magnitude of the collision impulse.

3.2 Friction and Turbulence Noise

Sounds generated by two objects rubbing against each other is a poorly understood process caused by microscopic surface features of the materials of the two objects. We have chosen to represent the process as a combination of two factors. First, the surface features cause both of the objects to vibrate in the same way as a phonograph needle vibrates when dragged in the groove of a record disk. The waveform of the sound generated is similar to the shape of surface imperfections of the objects. The so-called 1/f-noise [19, 20] could be used to model this phenomenon for rough surfaces. Another type of frictional noise we have simulated is generated by an object which alternates between sliding and sticking on a surface. This kind of interaction happens, for example, between a violin string and the bow. The resulting sound signal is approximated as a sawtooth waveform. Variation in the coefficients of friction can be modeled by adding some randomness to the wavelength, introducing noise to the sound.

Objects that disturb the air by causing turbulence also generate sounds. Modeling the frequency components of this phenomenon is difficult and outside the scope of this paper. We simply represent this as white noise modulated by the velocity of the object through the air and by the amount of air that it disturbs (a purely geometric factor with experimental values).

3.3 Other Procedural Sounds

Other subjectively interesting sounds can be generated with procedures that are not physically-based but model the behavior of an object. For example, we have synthesized the sound of an ambulance siren as a single tone frequency-modulated by a slow sine wave (figures 4 and 7).

The insect-like sound for behaviorally animated bees (figure 9), was generated as a constant wave-form with a noisy but narrowly tuned wave-length. Each bee of a swarm has a distinct base frequency that slowly drifts around a preselected mean frequency, either randomly or according to the bee's velocity.

4 Attaching Sounds to Objects

In hand-drawn animation, all important events, like the bounce of a hit in figure 3, are located at frame times. These key frames can be used to synchronize sound effects. For example, we can mark the starting times of each recorded sound effect and use a sequencer and mixer to put them all together.

In addition to the starting time, we can also specify the sound's amplitude any time along its duration. Between these key values the amplitude can be interpolated (lower part of figure 3).

What we have described are *sound threads*. These are sequences of key values extending over the sound's lifetime for each sound event. They can be put in an interface file as a sound script, described in detail in section 6.

The key events of a script can be either defined by the animator, like in the example above, or automatically computed by a behavioral or physically-based motion control. For example, we have used the impulse after a collision calculated by a physically-based motion control for rigid bodies [7] to define the script (figure 8 shows a frame from the animation). For each object Involved in a collision, a sound thread is started at the time of collision with an amplitude proportional to the impulse. With this method we can automatically produce accurately synchronized sound effects for complex situations that would be very difficult to achieve by other means.

Some sounds, like the turbulent wind noise, have infinite extent. They are represented in the script as threads extending throughout the duration of an animation scene.

5 Sound Propagation

In the following, we overview various effects of the acoustical environment and demonstrate how they can be incorporated into the system. The purpose is not to make detailed modeling of acoustics, but to show that each effect can be encoded as a transformation in the sound rendering script.

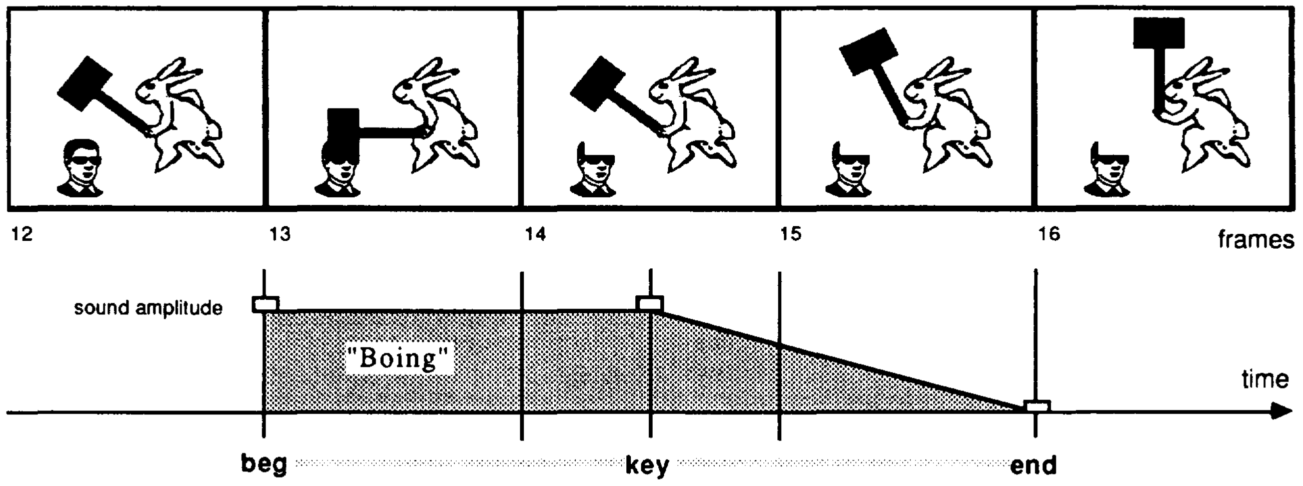


Figure 3: Keyframed sound thread for a hand-drawn animation.

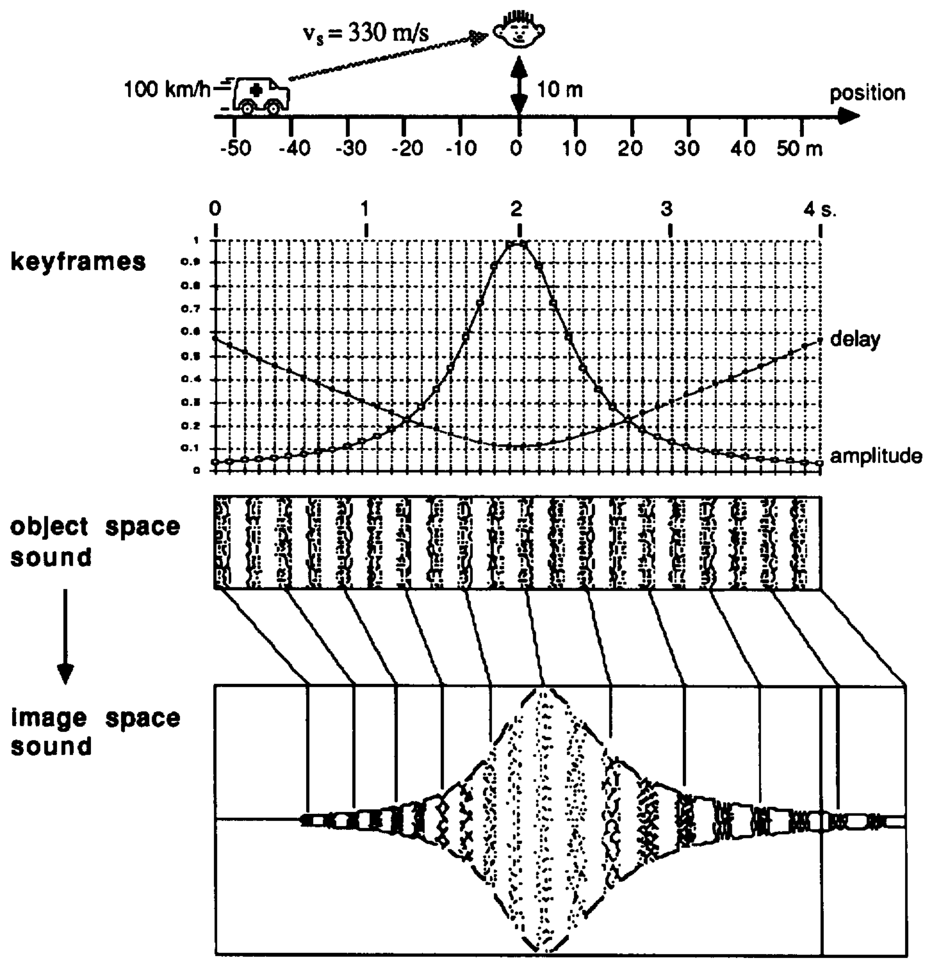


Figure 4: Key-framed values of amplitude and delay for a moving sound source, and the corresponding mapping of sound signal from object to image space.

5.1 Effects of Distance and Direction

The simplest acoustic effect is due to the dispersion of sound waves into the environment. Assuming no other attenuation, the intensity decays proportionally to the square of the distance travelled, and the signal is delayed by a time proportional to the same distance. Denoting sound velocity by v_S, we have for amplitude and delay

$$A(d) = A_0 \cdot \left(\frac{d_0}{d}\right)^2 , \quad \tau(d) = \frac{d}{v_S} ,$$

where d is the distance traveled, and A_0 is the sound intensity at a certain reference distance d_0. The effect of delay should be taken into account if the scene dimensions are of macroscopic magnitude, comparable to the sound speed. A delay of less than 50 ms is already clearly noticeable, corresponding to a distance of 15 m in atmosphere with a sound velocity of approximately 330 m/s.

Figure 4 illustrates sound animation of an ambulance passing a standing listener. The image space sound envelope shows the amplitude variation due to the 1:5 ratio in minimum and maximum distance between source and receiver of sound. Driving speed of 100 km/h gives a noticeable Doppler effect, with a deviation in pitch roughly corresponding to two halftones up and down from the car's original sound. When the car moves from the left field of view to the right, it gives a stereophonic illusion of movement.

The emission of sound may be directional. However, usually some sound is emitted in all directions. There seldom are as abrupt edges in its directional distribution as there are in directed light sources with flaps. Thus a spherical harmonics approximation [13] is appropriate.

Similarly, the microphone's directional sensitivity can be described with a spherical harmonic function. A representation sufficient in practice is given by the formula

$$S(\mu) = C + (1 + \cos \mu)^k$$

where μ is the angle between the microphone's principal axis and the incoming sound wave. Value k=0 defines an omnidirectional shape, and k=1 produces the typical cardioid sensitivity shape (figure 5). Lateral stereophonic hearing, due to differences in directional sensitivity of left and right ears, can be easily simulated with two microphones attached to opposite directions on a camera. With the sensitivity parameter k=1, the sum of left and right channels will be omnidirectional. Both sensitivities can be multiplied by a forward directed shape function, if rear attenuation is desired.

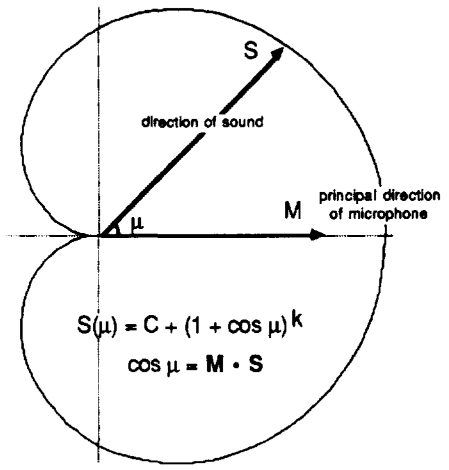


Figure 5: Microphone sensitivity shape (with k=1, C=0).

The total modulation of sound intensity is now the product of all directional functions in addition to the attenuation by distance.

$$A(d) = A_0 \cdot \left(\frac{d_0}{d}\right)^2 \cdot S(\sigma) \cdot S(\mu)$$

where σ and μ are the angles between the sound ray and the principal directions of sound source and microphone, respectively.

5.4 Sound Tracing in an Acoustic Environment

A common effect in acoustic environments is reverberation. It is caused by reflective objects in the scene acting as secondary sound sources. Algorithmically, this means making multiple instances of sound threads, with different delay and amplitude corresponding to each path through the environment. Generally this transformation would be a convolution of the signal with a continuous weighting function. In practical calculations, the weighting function is discretized to a finite number of samples. Reverberation is thus modeled with multiple echoes. In our approach each echo generates a separate thread for the sound renderer.

Because the sound wavelength is comparable to the geometric features of the objects, reflections are usually very diffuse. Specular reflection is soft, and refraction can in most cases be neglected. Due to diffraction, sound waves can propagate around a corner. Thus an obstacle occluding visibility does not completely cut off a direct sound, but only gives a smooth damping.

Based on these arguments and the fact that the purpose of sound in animation is rather illustrative than exact acoustical analysis, we use a heuristically simplified sound tracing model to calculate the effects of interfering objects. The basic idea is to find all major paths of sound from a source to the microphone. Each of them is represented as a separate sound thread with delay and attenuation corresponding to the path length and reflectivity coefficients.

As in radiosity calculation, each reflecting surface is considered a secondary diffuse point source receiving energy proportional to its area and the cosine of the angle between its normal and sound source direction. To calculate the amount of re-radiated sound we use the same reflection formula as is traditionally used for shading, consisting of a diffuse radiation in all directions, and a Phong-like specular term. The ambient sound in an environment is not modeled as a separate term for each object, but as a global background sound at each microphone.

Due to diffraction, the shadowing effect of objects obscuring a sound path (the form factor) is more approximate than with radiosity. In this paper we do not attempt to make a detailed geometric analysis, but use a rule of thumb. Attenuation is approximated with a form factor proportional to the amount of occlusion.

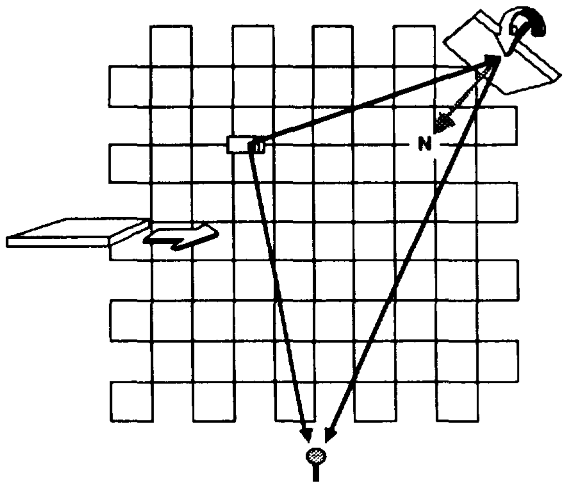


Figure 6: Principle of sound tracing.

A frame from an animation to demonstrate these effects is shown in figure 6. A standing car is producing continuous sound, which is received through two paths, directly and reflected by a wall. As the wall rotates, the echo is strongest when it is in an ideal mirroring position, and disappears when the tangent plane of the wall separates the car and the listener. As another wall moves by, it obscures visibility of the car first and then the mirror, damping each sound for a period.

6 Representation and Processing of Sound Scripts

The sound script defines how a prototype sound signal will be instantiated, and how it is transformed by the acoustic environment. This process will be best understood by an analogy with the mapping of a texture from texture to image space.

In a sound script all acoustic effects of the three-dimensional world are encoded in a geometry-free form. An advantage of making this transformation a separate process is that all the complex effects can be precomputed and combined in the same way as a sequence of geometric transformations can be concatenated into a single transformation matrix. Then each object sound can be rendered and checked separately, before mixing (overlaying) all sounds together. Another benefit is that this part of the sound renderer can be used for any key-framed animation, without necessarily having a three-dimensional model.

In this section we first describe how a sound script is used to describe transformations of sounds from prototypes to the soundtrack. Then the resampling algorithms interpreting the script and performing the transformation for each sound are discussed.

6.1 Representing Sound Transformations by Key-Framing

For describing a sound signal we have two independent coordinates, time and intensity, each of which can be transformed. For intensity, only scaling (amplitude modulation) is usually acceptable, since any non-linearity is perceived as distortion. Simple translation in time defines when a sound begins or how much it is delayed, whereas modifying the time scale causes change in pitch (frequency modulation). An arbitrary monotonic mapping – excluding reversal of time, which would be rather unnatural – can be approximated by interpolating specified key values of sound *attenuation* and *delay* (corresponding to the amplitude and phase angle of stationary signals).

Our script format for sound transformations consists of threads of values for attenuation and delay, one thread for each instantiated object sound. Each line of a script file contains the following information:

- keyword (start, in-between key, or end of a thread)
- ID (number) of a thread
- time stamp of the key
- attenuation and delay (for each stereo channel).

The file may be sorted by thread ID and or time stamp, depending on the resampling algorithm that calculates the soundtrack.

Key values may be specified at any points of time. In between the values are interpolated. Since the ear's resolution of intensity is not very high, we have found that linear interpolation is usually sufficient for amplitude values. However, the ear is very sensitive to abrupt changes in pitch. Thus, in order to keep the delay's rate of change (pitch level) continuous, either higher order interpolation or dense spacing of keys is needed. For simplicity, we have used the latter solution, specifying keys at each frame of an animation.

6.2 Resampling Algorithm

Just as in texture mapping, a sound transformation can be computed with an algorithm doing resampling either in sound object space or sound image space.

With an object space algorithm, each sample of a prototype sound is multiplied by the corresponding interpolated amplitude, and is written to a sound image buffer translated in time according to the corresponding interpolated delay (figure 4). A major problem with this approach is that gaps may be left in image space if the signal is expanded.

An image space algorithm determines the value for each image buffer position by sampling in object space at a point determined by the inverse transformation. The inverse mapping cannot be directly computed in our case, because the mapping of time (the delay) is defined in object space. We solve this problem by mapping the key positions from object space to image space, and interpolating the mapping parameters linearly. The interpolated delay gives the inverse mapping for each sample.

The same aliasing problems arise in sound transformations and sampling, just as they do in texture mapping, and could be solved similarly [4].

7 Conclusions

Sound rendering is a novel approach allowing the motion to define the sound, as opposed to the traditional way of starting with a soundtrack. This allows all types of motion control schemes to synchronize sound. With a physically-based motion control the sound events can be automatically produced, even if the objects' prototype sounds are not physically defined. With keyframed motion, the sound parameters can be keyframed as well. If the motion is three-dimensional, physically-based simulation of sound propagation can determine many acoustical parameters in a natural way.

In this work we have defined a modular sonic extension of the image rendering pipeline. Analogies have been drawn between sound and texture. Both are transformed from object to image space using similar algorithms with similar aliasing problems. We have simulated acoustical effects by sound tracing, analogous to forward ray-tracing and radiosity. With this analogy, the sound transformations correspond to shaders in a shade tree during its traversal from a source to a camera [3, 18]. We have applied the technique to generate sounds for a complex animation using keyframing, behavioral, and physically-based motion controls that illustrates its generality and flexibility.

One area we are currently working on is more rigorous physically-based modeling of sound-causing phenomena and sound propagation in acoustical environments. In this paper we have only given hints on how to approach the problem heuristically.

8 Acknowledgements

This work has been made possible in part by Helsinki University of Technology, who provided support for Tapio Takala's sabbatical to the George Washington University.

This work was also partly supported by funding from NASA (NCC 5-43) and NRL (NAVY N00014-91-K-203).

We would like to thank the students in the Computer Graphics and User Interface Group at George Washington University for their assistance. Special thanks go to Larry Gritz, Daria Bergen, and Rudy Darken for providing the image from "Graphic Violence".

We would also like to thank the reviewers for their enthusiastic comments and many useful suggestions.

References

[1] Barzel, R, A.Barr. Modeling with Dynamic Constraints. Proc. SIGGRAPH'88, ACM Computer Graphics, Vol.22, No.3, pp.178-188.

[2] Blattner, Meera, D.Sumikawa, R.Greenberg. Earcons and Icons: Their Structure and Common Design Principles. Human-Computer Interaction, Vol.4, No.1, 1989, pp.11-44. Also reprinted in: E.Glinert (ed.), Visual Programming Environments – Applications and Issues, IEEE Computer Society Press 1990, pp.582-606.

[3] Cook, Robert. Shade Trees. Proc. SIGGRAPH'84, ACM Computer Graphics, Vol.18, No.3, pp.195-206.

[4] Feibush, E.A., M.Levoy, R.Cook. Synthetic Texturing Using Digital Filters. Proc. SIGGRAPH'80, ACM Computer Graphics, Vol.14, No.3, pp.294-301.

[5] Foster, S, E.Wenzel. Virtual Acoustic Environments: TheConvolvotron. [multimedia installation] SIGGRAPH'91 Virtual reality and hypermedia show (August 1991).

[6] Gaver, William. The SonicFinder: An Interface That Uses Auditory Icons. Human-Computer Interaction, Vol.4, No.1, 1989, pp.67-94. Also reprinted in: E.Glinert (ed.), Visual Programming Environments – Applications and Issues, IEEE Computer Society Press 1990, pp.561-581.

[7] Hahn, James. Realistic Animation of Rigid Bodies. Proc. SIGGRAPH'88, ACM Computer Graphics, Vol.22, No.3, pp.299-308

[8] He, Xiao, K.Torrance, F.Sillion, D.Greenberg. A Comprehensive Physical Model for Light Reflection. Proc. SIGGRAPH'91, ACM Computer Graphics, Vol.25, No.3, pp.175-186.

[9] Lytle, Wayne. More Bells and Whistles. [video] in SIGGRAPH'90 film show. Also described in Computer, Vol.24, No.7, July 1991, p.4 and cover.

[10] NCSA Visualization Group and CERL Sound Group. Using Sound to Extract Meaning from Complex Data. [video] SIGGRAPH'91 Screening Room, Visualization & Technical series (August 1991).

[11] Pentland, Alex, J.Williams. Good Vibrations: Modal Dynamics for Graphics and Animation. Proc. SIGGRAPH'89, ACM Computer Graphics, Vol.23, No.3, pp.215-222.

[12] Perlin, K. An Image Synthesizer. Proc. SIGGRAPH'85, ACM Computer Graphics, Vol.19, No.3, pp.287-296.

[13] Sillion, François., J.Arvo, S.Westin, D.Greenberg. A Global Illumination Solution for General Reflectance Distributions. Proc. SIGGRAPH'91, ACM Computer Graphics, Vol.25, No.3, pp.187-196.

[14] Smith, J.W. Vibration of Structures - Applications in Civil Engineering Design. Chapman & Hall 1988

[15] Stettner, Adam, D.Greenberg. Computer Graphics Visualization for Acoustic Simulation. Proc. SIGGRAPH'89, ACM Computer Graphics, Vol.23, No.3, pp.195-206.

[16] Terzopoulos, Dimitri, J.Platt , A.Barr, K.Fleischer. Elastically Deformable Models. Proc. SIGGRAPH'87, ACM Computer Graphics, Vol.21, No.3, pp.205-214.

[17] Thomas, Frank, O.Johnston. Disney Animation – The Illusion of Life, chapter 11. Abbeville Press 1981.

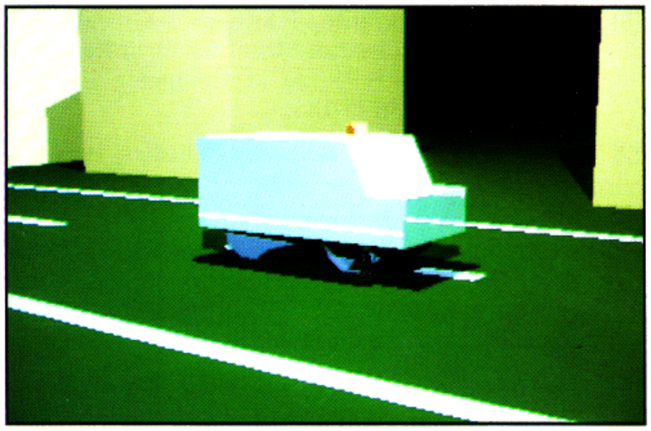
[18] Upstill, Steve. The RenderMan Companion. Pixar/Addison-Wesley, 1989.

[19] Voss, Richard and J.Clarke. "1/f noise" in Music: Music from 1/f Noise. J. Acoust. Soc. Am. 63 (1), January 1978, pp.258-263.

[20] Voss, Richard. Fractals in Nature: Characterization, Measurement, and Simulation. SIGGRAPH'87 Course Notes.

[21] Wightman, Frederick, D.Kistler. Headphone Simulation of Free-Field Listening. I: Stimulus Synthesis. J. Acoust. Soc. Am. 85 (2), February 1989, pp.858-867.

[22] Wyshynski Susan, The Vivid Group. The Mandala VR System. [multimedia installation] in SIGGRAPH'91 virtual reality and hypermedia show (August 1991).


Figures 7: Frame from a key-framed animation.

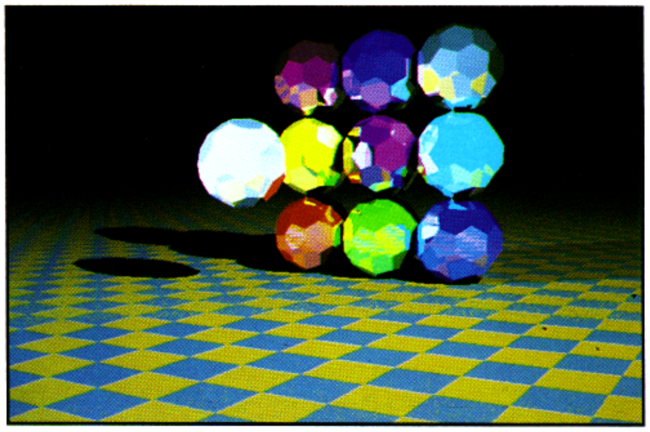


Figure 8: Frame from a physically based animation.

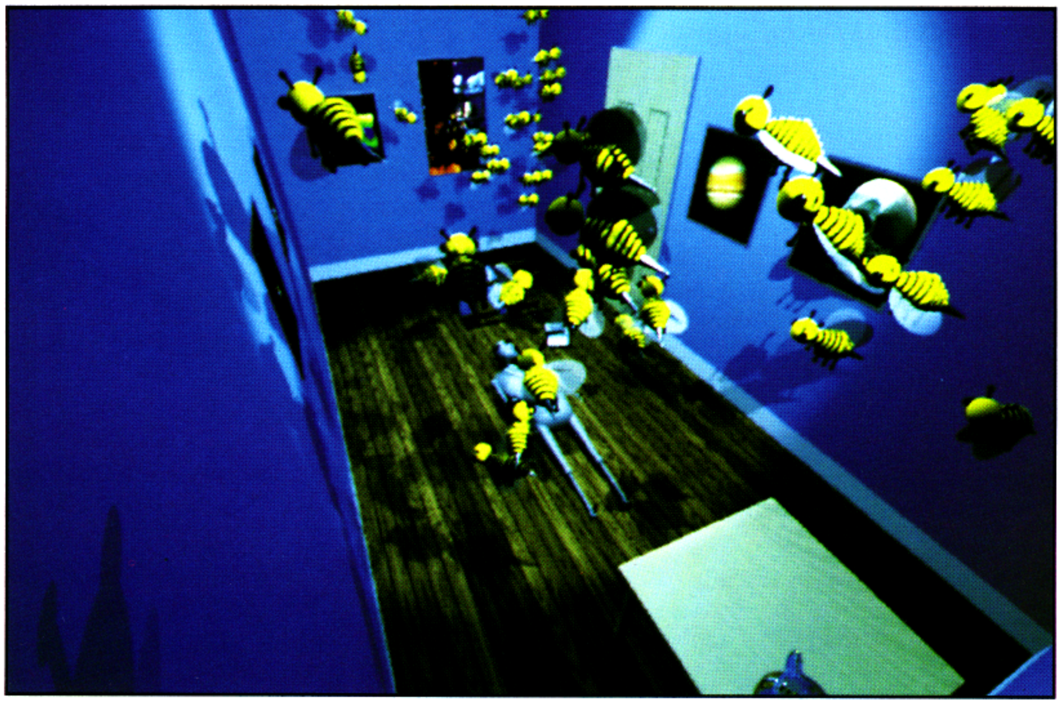


Figure 9: Frame from a behavioral animation.