SYNTHESIZING AUDITORY ICONS

William W. Gaver

Rank Xerox Cambridge EuroPARC
61 Regent Street, Cambridge CB2 1AB, U.K.
gaver@europarc.xerox.com

ABSTRACT

Auditory icons add valuable functionality to computer interfaces, particularly when they are parameterized to convey dimensional information. They are difficult to create and manipulate, however, because they usually rely on digital sampling techniques. This paper suggests that new synthesis algorithms, controlled along dimensions of events rather than those of the sounds themselves, may solve this problem. Several algorithms, developed from research on auditory event perception, are described in enough detail here to permit their implementation. They produce a variety of impact, bouncing, breaking, scraping, and machine sounds. By controlling them with attributes of relevant computer events, a wide range of parameterized auditory icons may be created.

KEYWORDS

interface techniques, multimedia, auditory interfaces, sound

INTRODUCTION

Over the last several years, I have been developing a strategy for creating *auditory icons*, everyday sounds mapped to computer events by analogy with everyday sound-producing events [6, 8]. Auditory icons are like sound-effects for computers: Objects make sounds as they are selected, dragged, bumped against one another, opened, activated, and thrown away. But they are not designed merely to provide entertainment; rather they convey information about events in computer systems, allowing us to listen to computers as we do to the everyday world.

A number of systems have been created which illustrate the potential for auditory icons to convey useful information about computer events. In particular, these systems suggest that sound is well suited for providing information:

- about previous and possible interactions,
- indicating ongoing processes and modes,
- useful for navigation, and
- to support collaboration.

For instance, the SonicFinder used sound to supplement a single-threaded, single-user graphical interface [7]. Although most of the sounds indicated simple user-initiated actions, the system demonstrated that sound could be incorporated in the interface in natural and useful ways.

More powerful functions for sound were demonstrated in the ARKola system [9]. This used the SharedARK environment [12] to create a simulation of a manufacturing plant which participants ran with or without auditory icons. Our observations suggested that sounds changed both the way participants perceived the plant and the way they worked together. These two examples, along with several others, indicate that auditory icons can both complement and supplement more traditional graphical cues in interface design.

PARAMETERIZED ICONS

Auditory icons not only reflect categories of events and objects as visual icons do, but are *parameterized* to reflect their relevant dimensions as well. That is, if a file is large, it sounds large. If it is dragged over a new surface, we hear that new surface. And if an ongoing process starts running more quickly, it sounds quicker.

The possibility of parameterizing icons, whether auditory or visual, has largely been neglected in interface design [though see 4]. But parameterized icons can serve as more than mere labels for their referents, providing rich sources of information about relevant dimensions such as size, age, or speed as well. Parameterization allows single objects or events to be assessed along a number of dimensions. In addition, it creates families of icons that retain perceptual similarity while allowing comparison among members [c.f. 1]. In general, parameterized icons allow a great deal of information to be conveyed perceptually rather than symbolically.

Creating Parameterized Auditory Icons

Unfortunately, it is difficult to parameterize auditory icons because it is difficult to control a virtual source of a sound along relevant dimensions. Standard synthesis techniques have been developed for creating music, and thus afford changes of a sound's pitch, loudness, duration and so forth. But they do not make it easy to change a sound from indicating a large wooden object, for instance, to one specifying a small metal one. It is easy to create a wide variety of beeps and hums using standard synthesis techniques, but difficult to create and manipulate sounds along dimensions that specify events in the world.

Because of the limitations of standard synthesis techniques, interfaces using auditory icons have relied on digital sampling in their implementations. Desired sounds are captured by recording them on a computer, shaped by a designer, then played back and manipulated under the control of the interface (see Appendix A for a brief introduction to sampling and synthesis). This enables the use of much more complex and realistic sounds than can be created by readily available synthesis algorithms. However, there are several drawbacks of sampling that limit its utility as a technique for creating and using auditory icons:

- It is difficult to capture an actual event that sounds like what is desired, because sounds are invariably coloured by the technologies used to record them.
- Shaping recorded sounds along dimensions relevant for auditory icons is difficult because available software is designed for making music.
- Real-time modification of sounds on playback is even more limited.
- The amount of memory needed for complex auditory interfaces is often prohibitive (on the order of 10K bytes per second of sound).

These factors constrain the possibilities for designing auditory icons, and make their creation difficult and time-consuming.

In this paper, I suggest an alternative in the form of a new type of synthesis algorithm developed as a result of basic research on auditory event perception [6, 8], and describe several examples in sufficient detail to allow readers to implement and explore them. These algorithms allow sounds to be specified in terms of their sources rather than their acoustic attributes. They promise to overcome both the limitations of traditional synthesis algorithms and of sampling by allowing parameterized auditory icons to be specified along dimensions of virtual source events.

ACOUSTIC INFORMATION FOR EVENTS

Creating algorithms that allow synthesis of virtual events implies an understanding of the acoustic information for event attributes – how sounds indicate the material or size of an object, for instance. Such attributes often have very complex effects on sounds, effects that must be described as functions of frequencies and amplitudes over time that describe the *partials*, or frequency components, that make up a sound. If these functions are understood, source attributes can be specified directly, instead of via separate controls over partial frequencies, amplitudes, and durations. But how can we determine what these functions are?

Analysis and Synthesis of Events

One approach to this problem is suggested by the analysis and synthesis methods [11] used by computer musicians to capture the relevant properties of traditional instrument sounds (Figure 1). This approach involves recording sounds that vary along dimensions of interest and analyzing their acoustic structure using Fourier analysis or similar techniques. Hypotheses about acoustic information suggested by the analysis can be tested by synthesizing sounds based on simplified versions of the data. For instance, if one supposes that the temporal features of a sound indicates the event that caused it, but that its frequency makeup is irrelevant, one might use the amplitude contour from the original sound to modify a noise burst. The hypothesis can then be assessed simply by listening to the result.

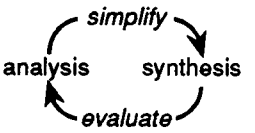

Figure 1: Traditional analysis and synthesis.

In practice, however, it is often difficult to identify the acoustic information for events in the mass of data produced by acoustic analyses. Thus it is useful to supplement them with analyses of the mechanical physics of the event itself (see Figure 2). Studying the physics of sound-producing events is useful both in suggesting perceptible source attributes and in indicating the acoustic information for these attributes. Acoustic analyses help both in checking the adequacy of physical models and in evaluating particular parameters. Finally, the resulting models can provide the basis for synthesis algorithms that allow sounds to be specified in terms of sources attributes.

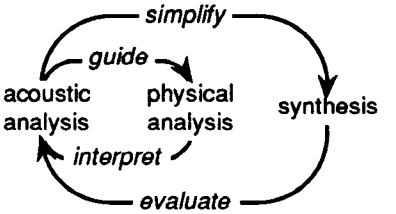

Figure 2. Analyzing and synthesizing events requires physical as well as acoustic analyses.

In the following sections, I discuss several case studies of events that have been studied in this manner and describe the synthesis algorithms that have resulted. The algorithms have been chosen for their utility in creating auditory icons, and are described in order of their complexity. I start with the sounds made by mechanical impacts, which involve a simple interaction of objects. Next I describe how more complex bouncing, breaking, and spilling sounds can be produced by specifying the temporal patterning of a series of impacts. A third algorithm distinguishes objects from the interactions that cause them to make sound, allowing the same virtual object to be hit and then scraped. Finally, I describe an algorithm for producing machine-like sounds, showing that high-level attributes of complex events may be synthesized directly.

IMPACT SOUNDS

Many of the sounds we hear in the everyday world involve one solid impacting against another. Tapping on an object, placing it against another, letting it fall – all involve impact sounds. In the interface, impact sounds are useful in the design of a variety of auditory icons that indicate events such as selecting a file, moving it over another, or attaching one object to another.

Several studies have explored the perceptible attributes of impact events and the acoustic information about them. In this section, I briefly review these studies, then show how the information they provide can be used to create a synthesis algorithm that allows impact sounds to be specified along dimensions of the virtual source.

Mallet Hardness, Material, and Size

Freed [5] studied people's perception of the hardness of mallets used to strike objects. He recorded the sounds made by hitting cooking pans with mallets of various hardnesses, asked people to judge hardness from the sounds, and used a model of the peripheral auditory system to analyze the acoustic correlates of their judgements. He found that the ratio of high to low frequency energy in the sounds and its change over time served as the most powerful predictors of subjects' hardness judgements. To a good approximation, then, mallet hardness is conveyed by the relative presence of high and low frequency energy.

I studied the acoustic information available for the length and material of struck wood and metal bars and people's abilities to perceive these attributes [6]. I recorded and analyzed the sounds made by wood and metal bars of several different lengths, and developed a model of the physics of impacts that combined analytical solutions to the wave equation for transverse vibrations in a bar [e.g., 10] with empirical measurements of damping and resonance amplitudes. This model was used both to aid interpretation of the acoustic analyses and to synthesize new tokens.

The material of the bars made several effects on the sounds they made. Perhaps most important, materials have characteristic frequency-dependent damping functions: the sounds made by vibrating wood decay quickly, with low-frequency partials lasting longer than high ones, while the sounds made by vibrating metal decay slowly, with high-frequency partials lasting longer than low ones. In addition, metal sounds have partials with well-defined frequency peaks, while wooden sound partials are smeared over frequency space. These results accord with Wildes and Richards' [14] physical analyses of the audible effects of the internal friction characterizing different materials, which show that internal friction determines both the damping and definition of frequency peaks.

Changing the length of a bar, on the other hand, simply changes the frequencies of the sound it produces when struck, so that short bars make high-pitched sounds and long bars make low ones. However, the effects of length may interact with the effects of material. For instance, frequencies change monotonically with length, but the frequency of the partial with the highest amplitude depends on material and thus may change nonlinearly with length [6]. These nonlinearities – and the perceptual confusion they cause – may be avoided by simplifying the model so that partial amplitudes do not depend on material.

A Synthesis Algorithm for Impact Sounds

These results may be captured in a synthesis model that uses frequency and amplitude functions to constrain a formula for describing exponentially decaying sounds. This formula describes a complex wave created by adding together a number of sine waves with independent initial amplitudes and exponential decay rates:

$$G(t) = \Sigma_n \Phi_n e^{-\delta_n t} \cos\omega_n t \quad (1)$$

where G(t) describes the waveform over time, Φ_n is the initial amplitude, δ_n the damping constant, and ω_n the frequency of partial n.

This formula has two properties that make it a useful foundation for synthesizing auditory icons. First, its components map well to event attributes. Second, it can be made computationally efficient using trigonometric identities.

Mapping Synthesis Parameters to Source Attributes

By constraining the values used in this formula, useful parameters can be defined which correspond well to the attributes of impact sounds discussed above. The formula involves three basic components: the initial amplitudes of the partials Φ_n, their damping $e^{-\delta_n t}$, and their frequencies $\cos\omega_n t$. These can be set separately for each partial. However, these three components also correspond to information for mallet hardness and impact force, material, and size and shape respectively (see Table 1). Thus it is more useful to define patterns of behaviour over the partials for each component.

Table 1: Mapping Parameters to Events

Term	Effect	Event Attribute
Φ_n	initial amplitudes	mallet hardness; force or proximity
$e^{-\delta_n t}$	damping	material
$\cos\omega_n t$	partial frequencies	size; configuration

For example, the partial frequencies ω_n can be constrained to patterns typical of various object configurations. The sounds made by struck or plucked strings, for example, are harmonic, so that $\omega_n = n\omega_1$. The sounds made by solid plates, in contrast, are inharmonic and can be approximated by random frequency shifts made to a harmonic pattern. The sounds made by solid bars can be approximated by the formula $\omega_n = (2n + 1)^2/9$. Finally, the sounds made by rectangular resonators are given by the formula $\omega_{(p,q,r)} = c/2\ \sqrt{(p^2/l^2 + q^2/w^2 + r^2/h^2)}$, where c is the velocity of sound, l, w, and h are the length, width and height of the box respectively, and p, q, and r are indexed from 0 [14]. An algorithm based on Formula 1, then, can be constrained so that one of these patterns is used to control the partial frequencies ω_n. In addition, ω_1 can be specified such that $\omega_1 \propto 1/\text{size}$ to reflect the size of the object (this affects all the other partial frequencies).

The initial amplitude of the partials, Φ_n, can be controlled by a single parameter corresponding to mallet hardness. Recalling that Freed's [5] results identified the ratio of high to low-frequency energy as a predictor of perceived mallet hardness, we might maintain a linear relationship among the partial's initial amplitudes, and use the slope from Φ_1 to control perceived hardness. Thus $\Phi_n = \Phi_1 + h(\omega_n - \omega_1)$, where h is the slope – note that h should often be negative, so that higher partials have less amplitude than low ones; thus a useful range of amplitude slopes might range from about -0.001 to 0.001. Φ_1 (and thus all the amplitudes) may also be changed to indicate impact force or proximity.

Finally, the damping constants for each partial (δ_n) can be controlled by a parameter corresponding to material. A useful heuristic is to set $\delta_n = \omega_n \delta_0$, with δ_0 ranging between about .001 for metal and about .5 for a highly-damped material like plastic. This means that high harmonics will die out relatively quickly for highly damped materials and last longer for less damped materials (e.g., metal, which has low damping, tends to ring; wood, which is highly damped, tends to thunk). This strategy is suggested both by Wildes and Richards [14], and by my own research [6].

In sum, Formula 1 can be controlled by parameters that make effects corresponding to attributes of impact events. Controlling overall frequency corresponds to the object's size, while the pattern of partial frequencies corresponds to its configuration. The overall initial amplitude corresponds to the force or proximity of the impact, while the pattern of partial amplitudes corresponds to mallet hardness. Finally, the degree of damping corresponds well to the virtual object's material. By controlling these five parameters, then, a wide range of sounds can be created which vary over several useful dimensions.

An Efficient Algorithm for Synthesis

Formula 1 is useful in allowing parameters to be defined in terms of source events. It is also attractive because it can be implemented in a computationally efficient way.

An efficient implementation of this formula relies on Euler's relationship $e^{i\omega t} = \cos\omega t + i\sin\omega t$ to rewrite Formula 1 as:

$$\begin{aligned} S_n &= \mathrm{Re}[(a_n + ib_n)(p + iq)] \\ &= \mathrm{Re}[(a_n p - b_n q) + i(b_n p + a_n q)] \end{aligned} \quad (2)$$

where S_n is the nth sample, a_0 is the initial amplitude, $b_0 = 0$, $i = \sqrt{-1}$, $p = e^{-\delta t}\cos\omega t$ and $q = e^{-\delta t}\sin\omega t$. (A full derivation is available upon request.)

Samples can thus be generated by calculating p and q, setting a and b to the initial amplitude and 0, and applying equation 2. The output sample is the real part of the result, and a and b are updated to the real and imaginary parts respectively (see pseudocode in Figure 3). Computationally expensive sine and cosines need only be calculated once, and only four multiplications, one addition and one subtraction are needed for each partial for a given sample. The efficiency of this implementation allows fairly complex impact sounds to be generated in realtime on many computers.

```
p = cos(freq * 1/samplerate) * power(e, -1 * damping.rate
    * 1/samplerate);
q = sin(freq * 1/samplerate) * power(e, -1 * damping.rate
    * 1/samplerate);
a = initial.amplitude;
b = 0;

repeat for duration.in.secs / samplerate:
    real.part = a * p - b * q;
    imaginary.part = b * p + a * q;
    a = real.part;
    b = imaginary.part;
    output = real.part;
end repeat;
```

Figure 3. Pseudocode for efficient generation of a exponentially-decaying cosine wave (equation 5).

BREAKING, BOUNCING, AND SPILLING

The impact algorithm can serve as a fundamental element in algorithms used to synthesize more complex sounds. For instance, an early example of analysis and synthesis of sound-producing events is Warren and Verbrugge's [13] study of breaking and bouncing sounds. In this study, they used acoustic analyses and a qualitative physical analysis to examine the auditory patterns that characterize these events, and verified their results by testing subjects on synthetic sounds.

Consider the mechanics of a bottle bouncing on a surface (Figure 4A). Each time the bottle hits the surface, it makes an impact sound that depends on its shape, size, and material (as discussed above). Energy is dissipated with each bounce so that, in general, the time between bounces and the force of each impact becomes less. Thus bouncing sounds should be characterized by a repetitive series of impact sounds with decreasing period and amplitude.

When a bottle breaks, on the other hand, it separates into several pieces of various sizes and shapes (Figure 4B). Thus a breaking sound should be characterized by an initial impact sound followed by several different, overlapping bouncing sounds, each with its own spectrum and period.

Acoustic analyses of bouncing and breaking sounds confirm this informal physical analysis. In addition,

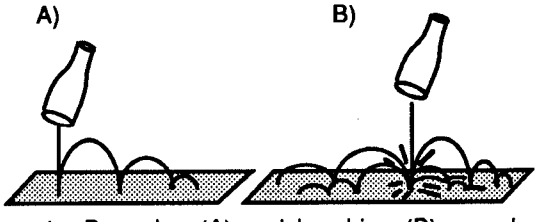

Figure 4. Bouncing (A) and breaking (B) sounds are characterized by the temporal patterning of a series of impacts [After Warren and Verbrugge, 13].

Warren and Verbrugge [13] found that people were able to distinguish tokens of bouncing and breaking sounds that were constructed by using these rules to splice tapes of impact sounds together.

Synthesized Breaking, Bouncing and Spilling

To create bouncing sounds, then, we need simply imbed the impact algorithm in another that calls it at exponentially decaying intervals. To create breaking sounds, the bouncing algorithm is imbedded in another algorithm that calls it with parameters specifying sources of different sizes at times corresponding to several exponentially decaying time series.

Several new event parameters become relevant for these algorithms: The initial height of the virtual object is indicated by the time between the first and second bounce, its elasticity by the percentage difference of delays between bounces, and the severity of breaking by the number of pieces produced. In addition, the asymmetry of the perceived object can be varied by adding randomness to the overall temporal pattern.

It becomes clear upon listening to sounds synthesized using this algorithm that although Warren and Verbrugge [13] claimed that information for breaking and bouncing depends only on temporal patterning, the perceived event depends on the virtual materials involved as well. For instance, if impacts specifying wooden objects are produced in a temporal pattern typical of breaking, we are liable to hear spilling rather than breaking. Similarly, if each of several virtual objects has different material properties, we again hear several spilling objects rather than breaking.

In sum, the impact algorithm described above can be used not only to generate the sounds made by mallets of different hardnesses striking virtual objects of a wide variety of shapes, sizes, and materials, but can also serve as the basis for more complex bouncing, breaking, and spilling sounds. As such, it serves as a research tool that allows the space of such sounds to be explored. Moreover, it provides an efficient method for generating families of related auditory icons. For instance, parameterized impact, bouncing, breaking and spilling sounds might be used to differentiate and provide details about the results of actions involving icons, windows, containers, and so forth.

FROM IMPACTS TO SCRAPING

The sounds made by impacts and patterns of impacts are generally useful for creating auditory icons, but it is desirable to have access to sounds made by a wider range of events. In particular, it would be useful to generate the sounds made by the same object being interacted with in different ways. Using such algorithms, auditory interface designers might map a particular file to a virtual object, and then hit, bounce, or scrape it depending on the relevant computer interaction.

In order to create such algorithms, it is necessary to separate the specification of a virtual object from that of the interaction that causes it to produce sound. This turns out to be possible because objects tend to vibrate only at certain invariant resonant frequencies. For example, the spectrogram in Figure 5 shows the sound made by a piece of glass being hit and then scraped across a rough surface. Note that despite the different temporal patterns of the sounds, the resonant modes of each are the same. These modes specify the object, then, interaction determines the temporal pattern and amount of energy introduced to each.

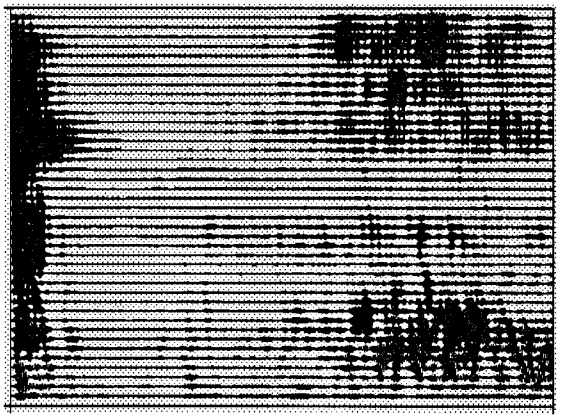

Figure 5. Spectrogram of a piece of glass being hit and then scraped: The resonant frequencies remain invariant over different interactions.

Because the effects of interactions and objects are distinct, each can be modelled separately. The resonant modes of a virtual object may be modelled as a bank of filters that allow energy to pass at particular frequencies. Interactions, then, can be specified by the pattern of energy passed through the filter bank.

Modelling Objects as Filter Banks

A simple formula for a bank of one-pole filters is:

$$y_n = \Sigma_m \Phi_m (c1_m x_n + c2_m y_{n-1} - c3_m y_{n-2}) \qquad (3)$$

where Φ_m is an amplitude scalar for partial m, y_n is the nth output, x_n is the nth input, and:

$$c1_m = (1 - c3_m)\,[(1 - c2_m{}^2)/4c3_m]^{.5}$$

$$c2_m = (4c3_m \cos 2\pi f_m)/(c3_m + 1)$$

$$c3_m = e^{-2\pi b_m};$$

where f_m is the frequency and b_m the bandwidth of partial m.

The parameters used to control the impact algorithm can also be used to control this sort of filter bank. However, manipulating filter bandwidths to control damping actually provides more information for material than do simple manipulations of sine wave damping. Bandwidth b is proportional to damping: the narrower the resonance peak of the filter the longer the resonant response to excitation. This correlation between damping and the smearing of partials in frequency space corresponds well to the characteristics of sounds made by materials such as wood or metal [6, 14].

Simulating Interactions with Input Waveforms

A virtual object can be defined by the characteristics of the filter bank described above. The waveform passed through the filter bank, then, models the interaction that causes the object to sound. In this section, I describe two sorts of input waveforms that I have explored. The first models impact forces, the second scraping.

When objects are struck, the input forces are characterized by short impulses such as those shown in Figure 6. The energy of such impulses is spread out over many frequencies: the pulse width reflects low frequency energy, while its angularity reflects high frequency components. This corresponds to Freed's characterization of mallet hardness [5]. Hard mallets introduce force suddenly to an object, deforming it quickly, and thus introduce a relatively high proportion of high frequency energy to the resonant object. Soft mallets, in contrast, deform as they hit the object, introducing energy relatively slowly, and thus the corresponding impulses are characterized by a high proportion of low frequency energy. Shaping the impulses used to excite a filter bank, then, is a physically realistic way to control perceived mallet hardness.

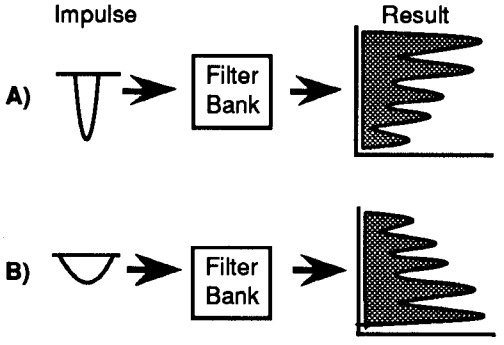

Figure 6. Sample impulse waveforms characterizing different impacts (see text).

When an object is scraped, force is applied more continuously. Scraping has been relatively unexplored in terms of its physical or perceptual attributes. However, an informal physical analysis suggests that the pattern of force on an object generated as it is scraped across a surface can be approximated by band-limited noise, where the center frequency of the noise corresponds to dragging speed, and the bandwidth to the roughness of the texture (see Figure 7). Although these parameters are only approximate, being less well motivated physically or psychologically than those used to model impacts, experience shows that a wide variety of realistic scraping noises can be produced using these heuristics.

In sum, the filter-based algorithm described in this section is based on a physically plausible model of sound producing events. By separating the parts of the model that specify the object from those specifying the interaction, a wide range of virtual sound-producing events can be simulated. The model can create any of the impact sounds that the algorithm described in the last section can.

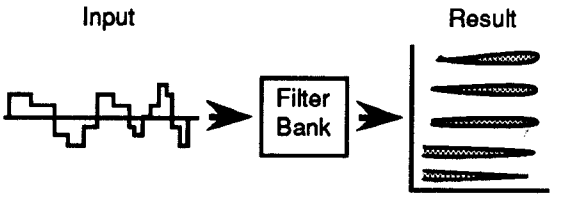

Figure 7. A sample force waveform characterizing a scrape with increasing speed.

In addition, it can also be used to create a variety of scraping sounds (and, potentially, any other sound involving solid objects).

The ability to generate the sounds of the same object being caused to sound by different interactions offered by this algorithm has great potential for the creation of auditory interfaces. It allows the design of parameterized auditory icons in which the same interface object (e.g., a file) might make sounds indicating a variety of events (e.g., selecting, dragging, opening).

MACHINE SOUNDS

Just as complex interactions such as scraping can be modelled by a few summarizing parameters, so might still more complex events be captured succinctly by high-level descriptions. For instance, another class of sounds useful for auditory icons are those made by small machines. Sampled machine sounds were used effectively in the ARKola simulation [9], indicating ongoing processes that were not visible on the screen. More generally, they might be used to indicate background processes such as printing or compiling in more traditional multiprocessing systems.

A detailed account of the mechanical physics of machinery seems prohibitively difficult. But just as the scraping waveforms described above model the overall parameters of a complex force rather than each of the contributing details, so an approximate model of machines might capture some of the high-level characteristics of the sounds they produce. In particular, three aspects of machine sounds seem relevant for modelling: First, the overall size of the machine is likely to be reflected in the frequencies of sounds it produces; second, most machines involve a number of rotating parts that can be expected to produce repetitive contributions to the overall sound; and third, the work done by the machine can be expected to affect the complexity of the sound.

FM Synthesis of Machine Sounds

I have been exploring an efficient algorithm for creating a variety of machine-like sounds that capture these properties. The basic strategy is to synthesize a sound using complex tones that vary in a repetitive way, indicating cyclical motion. The rate at which the virtual machine is working, then, can be indicated by repetition speed, the size of the virtual machine by the base frequency, and the amount of work by the bandwidth of the sounds (see Figure 8).

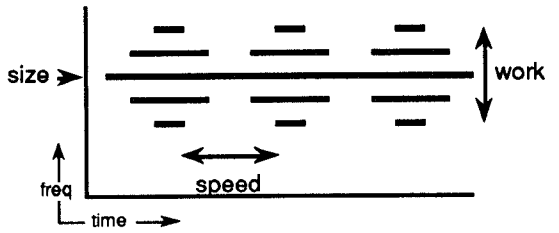

Figure 8. Machine sounds can be characterized by a complex wave that varies repetitively over time.

This class of sound may be synthesized efficiently using Frequency Modulation (FM) synthesis [2]. FM synthesis involves modulating the frequency of a carrier wave with the output of a modulating wave. This produces a complex tone with a number of frequency components spaced equally around the carrier wave and separated from one another by the modulating frequency. The number of components (and thus the bandwidth of the sound) depends on the amplitude of the modulating wave (see Figure 9). Thus machine sounds can be created simply by associating the carrier frequency with the size of the virtual machine, setting the maximum amplitude of the modulating wave to the amount of work done by the virtual machine, and modulating the amplitude of the modulator according to the speed of the virtual machine (see pseudocode in Figure 10).

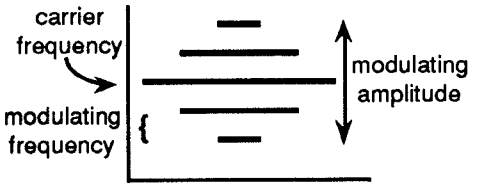

Figure 9. FM synthesis, which allows simple control over complex sounds, is useful for generating machine sounds.

The resulting sounds are pitched humming noises that pulse at the speed of the virtual machine. When "work" is low, the throbbing is subtle; when it is high, it becomes quite pronounced. Moreover, the quality of the sounds can be varied by changing the ratio of modulating to carrier frequency: when the two are an integral multiple of one another, the resulting sound is harmonic, when they are not, the sound is inharmonic or noisy. Because the physical and acoustic analyses underlying this algorithm are far more approximate than those used for the various impact and scraping algorithms described above, the sounds it produces are far less realistic. Nonetheless, they seem to capture some of the features of machine sounds well enough to use as indicators of ongoing processes in multiprocessing systems.

```
mod.wave.amp = work * sin(speed * time/samprate);
mod.wave.sample = mod.wave.amp * sin(mod.freq *
    time/samprate);

output = amp * sin((size + mod.wave.sample) *
    time/samprate);
```

Figure 10: Pseudocode for generating machine-like sounds characterized by speed, size, and work.

CONCLUSIONS

The algorithms described here allow the synthesis of a variety of everyday sounds specified in terms of attributes and dimensions of the events that cause them. Because they are based on a combination of acoustic and physical analyses and use relatively sophisticated synthesis techniques, they capture a great degree of the richness and complexity of their naturally produced counterparts. Because they are specialized for the classes of event they are to simulate, they are easy to implement, efficient, and can generate sounds in realtime on many computers. Finally, because they have been designed with potential applications in mind, the events they simulate are those useful for auditory icons.

These algorithms vary in their physical accuracy. Some are based on quantitative physical analyses, while others are based on more qualitative, informal descriptions of events. Moreover, even the quantitative analyses are only approximate. For instance, the physics of a struck bar of wood is much more complex than implied by the simple account given here. Insofar as these algorithms are approximate, the sounds they produce will differ from those made by real events.

Nonetheless, these algorithms do produce quite realistic sounds. Listeners comment that they have the impression of hearing an actual event rather than a synthesized sound. Insofar as the sounds do differ from those made by real events, they may be considered as "cartoon-sounds," sounds which capture the relevant features of their sources just as visual caricatures (or graphical computer icons) capture those of theirs. For the purposes of simulating sound-producing events, then, these algorithms are adequate. For the purpose of creating auditory icons, they show great potential, combining flexibility, intuitive controls, efficiency, and relevance.

I have specified these algorithms here in sufficient detail that readers may implement and explore them, in the hope that they will spur further research on parameterized auditory icons. These algorithms open many possibilities for the design of rich auditory interfaces. Impact and scraping sounds can be used to increase the tangibility of graphical objects in direct manipulation interfaces. Bouncing, breaking, and spilling sounds can be used to indicate events in virtual reality systems. Machine sounds might allow us to hear a remote printer as our job reaches the queue, and characteristics of the sound might tell us how fast the job is printing or how much time it will take. In sum, using these algorithms we can design interfaces that we can listen to the way we do to the everyday world.

ACKNOWLEDGEMENTS

I am grateful to Dave Woodhouse and Roy Patterson for help with the physical and acoustical analyses presented here, and to Don Norman, Anne Schlottmann, William Mace, Allan Maclean, Wendy Mackay, and Michael Turvey for valuable discussions about this research.

REFERENCES

1. Blattner, M., Sumikawa, D., & Greenberg, R. Earcons and icons: Their structure and common design principles. *Human-Computer Interaction*, 4 (1989).
2. Chowning, J. The synthesis of complex audio spectra by means of frequency modulation. *Journal of the Audio Engineering Society*, 21 (1973): 526-534.
3. Dodge, C., & Jerse, T. *Computer music: Synthesis, composition, and performance*. New York: Schirmer Books, 1985.
4. Draper, S., Waite, K., & Gray, P. Alternative bases for comprehensibility and competition for expression in an icon generation tool. *Proceedings of Interact'90* (Cambridge, UK. 27 - 31 August, 1990). Amsterdam: North Holland..
5. Freed, D. Auditory correlates of perceived mallet hardness for a set of recorded percussive sound events. *Journal of the Acoustical Society of America*, 87 (1990): 311-322.
6. Gaver, W. Everyday listening and auditory icons. Unpublished doctoral Dissertation, University of California, San Diego, 1988.
7. Gaver, W. The SonicFinder: An interface that uses auditory icons. *Human-Computer Interaction*, 4 (1989): 67-94.
8. Gaver, W. What in the world do we hear? An ecological approach to auditory source perception. *Ecological Psychology*, 5 (1993): 1-29.
9. Gaver, W., Smith, R. B., & O'Shea, T. Effective sounds in complex systems: The ARKola simulation. *Proceedings of CHI 1991* (New Orleans, April 28 - May 2, 1991) ACM, New York.
10. Lamb, H. *The dynamical theory of sound*. 2nd ed. New York, Dover, 1960.
11. Risset, J., & Wessel, D. Exploration of timbre by analysis and synthesis. In D. Deutsch (Ed.), *The psychology of music*. New York: Academic Press, 1982.
12. Smith, R. A prototype futuristic technology for distance education. *Proceedings of the NATO Advanced Workshop on New Directions in Educational Technology*. (Nov. 10 - 13, 1988, Cranfield, UK.)
13. Warren, W. & Verbrugge, R. Auditory perception of breaking and bouncing events: A case study in ecological acoustics. *Journal of Experimental Psychology: Human Perception and Performance*, 10 (1984): 704 - 712.
14. Wildes, R., & Richards, W. Recovering material properties from sound. Richards, W. (ed.), *Natural computation*. Cambridge, MA: MIT Press, 1988.

APPENDIX A: DIGITAL SAMPLING AND SYNTHESIS

Digital sampling and synthesis both rely on the fact that analog sound waves may be described by a stream of numbers (or *samples)* that represent the amplitude of the wave over time (see Figure 11). These samples, in turn, can be used to control a loudspeaker, recreating the sound.

As Figure 11 suggests, the accuracy with which a waveform may be represented depends on the sampling rate: waves at frequencies over 1/2 the sampling rate (the *Nyquist frequency*) cannot be specified unambiguously. Similarly, the number of bits per sample determines the different levels of amplitude that may be captured.

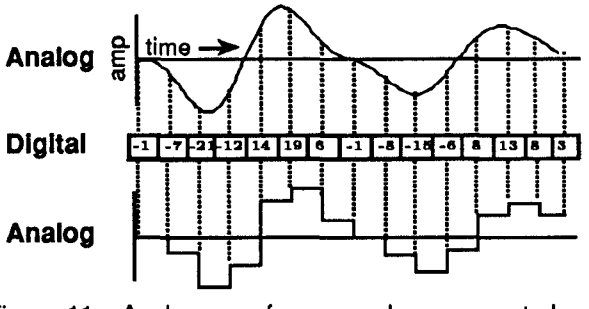

Figure 11. Analog waveforms can be represented as samples indicating the wave's amplitude over time.

Digital sampling involves recording a sound, digitizing the waveform, and then recreating the waveform using the (possibly manipulated) samples.

Synthesis, in contrast, involves generating sample values algorithmically. Many synthesis algorithms rely on Fourier's theorem that complex waves may be analyzed – and thus synthesized – as the sum of many (possibly varying) sine waves. In any case, synthesis algorithms are merely methods for creating streams of numbers that describe sounds.

There are several software packages that facilitate synthesis (e.g., the Csound software from the MIT Media Lab, or Cmusic from UCSD, both of which can be obtained without charge for educational and research purposes). The algorithms described in this paper may be explored using such packages. However, for use in interfaces, these algorithms should be implemented as functions to be called by other software; the samples they produce can simply be sent to the onboard digital-to-analog converter just as sampled sounds are.

Clearly this is only a brief introduction to the basic concepts of digital sampling and synthesis. More detailed introductions to these topics may be found in several textbooks [e.g., 3].