Ad-hoc synthesis of auditory icons

Stéphane Conversy

Laboratoire de Recherche en Informatique – CNRS URA 410
Bâtiment 490 - Université de Paris-Sud
91405 Orsay Cedex, France
conversy@lri.fr
www-ihm.lri.fr/~conversy

Abstract

This article introduces ad-hoc synthesis, an approach to designing auditory icons and synthesis algorithms that emphasizes the perception of the sounds by users instead of the analysis of actual sources and sound. We describe two substractive synthesis algorithms for generating and controlling wind and wave sounds in real-time by means of high-level parameters. Even though these sounds are not audiorealistic, they convey information in a non-intrusive way and therefore are suitable for monitoring background activities. These sounds capture the main invariants of the sounds they imitate, enabling users to recognize and understand them easily. We then push the approach further by showing how an auditory illusion, i.e. a sound that does not exist in the real world, can be used to convey the notion of speed in a natural and non-intrusive way.

1 Introduction

Sound can be used in user interfaces for several kinds of information. For example, sound can be used to provide immediate feedback of the users' actions, e.g. clicking a button, to inform the users of the state of computer processes, e.g. mail arrival, or to help users' awareness of each other in a cooperative environment [4]. The work presented in this article addresses the use of sound for monitoring background activities [6], i.e. to help users to be aware of and recognize the current state and the state changes of applications, devices or other users. Sound is an adequate medium for such monitoring for two main reasons. First, we can focus our attention on a particular sound and be aware of several other sounds simultaneously. Second, continuous sounds can fade into the background when we do not attend to them and come to the foreground when they change or when we decide to listen to them. Continuous notification of the current state of a process allows prevention as opposed to cure: when a sound changes, it may be a harbinger of an impending issue that we can address before it happens. For example, knowing that the printer is about to run out of paper when we start a print job is more useful than being notified that the tray is empty when the job is about to be printed. However using continuous sounds presents the obvious problem of information overload. Therefore the challenge for an interface designer is to provide continuous information with sounds without disturbing the user.

We have experimented with wind and wave sounds for monitoring continuous activities. In the real world, the sound of the wind makes us aware that it might rain, and the sound of the waves tells us whether we can go swimming or surfing. We are used to these sounds because they are the manifestation of natural elements, and we can pick-up the information they convey instantly. Finally, sounds of natural elements are less likely to disturb the user if they are played at a low level.

In order to add environmental sounds to an interface, we could have used sampled sounds. However we wanted to control the sounds in real-time with high-level parameters, so we decided to design synthesis algorithms that produce sounds perceptually similar to those of the real events. We found that simple algorithms could be used to generate convincing and informative sounds. Based on this perceptual approach, we also experimented with auditory illusions and used Sheppard-Risset tones [17] to convey the notion of speed.

In the next section we present the two main trends in the design of synthesis algorithms. In the following section we explain why neither of them is really suitable for environmental sounds. We then argue that a more abstract approach can help in designing perceptually meaningful auditory icons. In the last two sections, we describe the algorithms for synthesizing wind and wave sounds and the Sheppard-Risset tones, and sample applications that use them.

2 Related work

There is a large body of work on sound analysis and synthesis in the musical domain. The goal of scientists and musicians is to exactly reproduce the sound of an instrument. We can distinguish two main approaches: synthesis based on *source* analysis and synthesis based on *sound* analysis.

Source analysis consists of understanding the physics of the source and stimulation in order to design a physical model that reproduces their behaviour. This type of analysis usually results in complex algorithms that cannot synthesize the sounds in real-time on most machines [18].

Sound analysis consists of recording instruments and compute their spectrogram. By varying the parameters of the performance, the analyst can gain an understanding of the relationships between the spectrum and the performance. Usually a short-window fast Fourier transform (FFT) is used to create the spectrogram. The resulting set of coefficients can be manipulated and transformed back to samples with an inverse FFT [8].

The source and sound analysis models are not exclusive. Usually one approach is used to improve the model built with the other, e.g. checking the spectrum of a physical model under various conditions to validate it or understanding some of the physics to discover relationships between frequency patterns. These methods are particularly suitable for musically interesting sounds. Researchers know *which* physical parameters make an instrument sound differently, and their task is to discover *how* these parameters affect the sound. Moreover, they have full control over the sources and sounds they study, e.g. the sound of a violin playing a C with a particular gesture. The situation is quite different with non-musical sounds, as we will see below.

In the auditory display domain, researchers use both trends to synthesize everyday sounds. For example, Van Den Doel and Pai analyze the influence of the shape of an object to synthesize impact sounds [20]. Miner and Caudell [14] use a sound analysis approach based on a wavelet transform instead of an FFT. Their goal is to create realistic sounds to be included in virtual reality systems. By manipulating the coefficients of the wavelet transform to shift frequencies, they were able to give the sensation of varying size and stream of motor and river sounds.

Gaver [10] uses a mix of these methods to design impact and scrape sounds. Gaver argues that in order to understand a sound, one must analyze the source as well. This results in a better understanding of the relationship between the spectrogram and the source and helps in developing a better algorithm in an iterative process. Moreover, the design of the algorithm helps understand the physics of the sound.

3 An alternative for auditory icon design

Source and sound analysis have some drawbacks when applied to natural sounds. We propose to use an ad-hoc approach, more suitable for this kind of sounds.

3.1 Problems with source and sound analysis

Source and sound analysis only work when the various dimensions of the source and the sound can be explored systematically. For example, a collection of impact sounds can be analyzed by varying the size, shape and material of the object. It is difficult to analyze complex sources and interactions like wind and waves, because it is very difficult to collect an organized set of sounds. Moreover, the physics of these sounds is probably too complex to help design efficient synthesis algorithms. Similarly, sound analysis is difficult because it is almost impossible to reproduce the sounds under controlled conditions. If we had a set of sounds for one type of event, varying step by step along its perceptual dimensions, a sound analysis could help extract the relationships between perceptual dimensions and FFT coefficients.

Unlike impact sounds, which resemble each other so much that we can almost talk about "the" sound of an impact, it is impossible to choose an example of "the" sound of the wind or "the" sound of waves. Many different sounds can be recognized as wind or waves and from each such sound we can extract the strength of the wind or the size of the waves. When people imitate the sound of a wave, they make a sound based on their idea of the sound. Such imitations are effective because they capture the perceptually significant aspects of the sound. Therefore, since we want to convey the notion of wind strength rather than the *sound* of the wind itself, it is more useful to analyze the way people perceive a wind sound than to analyze the sound or the sources themselves.

Ballas conducted experiments about the recognition of short auditory icons [2], arguing that there are two types of cognitive expectancy: expectancy about a source for a particular sound, and expectancy about a sound for a particular source. The experiments showed that listeners are better at recognizing a sound when it matches their mental model of that sound. In other words, one will recognize (associate a cause to) a sound better if this sound resembles (has the same invariants as) the sound he expects when primed with the cause. For an auditory icon designer, this results in a shift in what to analyze in a sound. The designer should focus on the specificities of the sound and reproduce them, or even emphasize them.

Gaver [10] took this approach to design a machine sound. He pointed out that source analysis is useless because the complexity of a machine does not allow us to understand its underlying physics. Thus, he listed a set of "high-level perceptual characteristics" for such a sound, and designed an algorithm controllable along these dimensions. For example, a mechanism is usually cyclic, thus it should produce a cyclic sound. Gaver used a frequency modulation (FM) algorithm, which produces a cartoon-like sound. Gaver’s impact sounds were designed by a source and sound analysis which resulted in "audiorealistic" sounds. In contrast, the FM algorithm that generates the machine sound captures the features of what we perceive as a machine sound but it cannot be associated with a real source. The important point is that the users can recognize the sound and pick-up the information it conveys. Since we are addressing interaction with a computer system and not realistic environments, quality is not as important as meaning.

3.2 Ad-hoc synthesis

Synthesis algorithms based on source and sound analysis aim at defining a model that produces as many invariants [6, 12, 13] of the sound as possible, hoping that humans will perceive them and extract the information they convey. Gaver’s machine sound algorithm on the other hand is based on the invariants that we perceive the most and that we use to describe sounds verbally or imitate them. Sound and source analysis prove difficult and sometimes useless to synthesize sounds that catch perceptually important invariants. In order to design such sounds, we have investigated what we call "ad-hoc synthesis".

Ad-hoc synthesis was developed by the pioneers in electronic music [19]. The most well-known is substractive synthesis, i.e. shaping a waveform with dynamic filters. FM synthesis [5], used by Gaver for the machine sound is another example. With these algorithms musicians abandoned the idea of reproducing exact acoustic sounds, and yet, many sounds are perceptually recognizable even though they have an electronic "touch". Moreover, with today’s computer, they are computable and controllable in real-time. We believe that designers of auditory icons should investigate these techniques even though musicians failed at reproducing audiorealistic sounds with them. Unlike musicians, we are not interested in sounds that have the spectrograms of the actual events, but in sounds that have the invariants of the actual events.

4 Synthesizing wind and waves

In order to create a sound with ad-hoc synthesis, the designer must answer the following two questions:

• How a sound is better related to an event than another?

• Which attributes do we want listeners to pick up?

In other words, the designer must find the invariants that allow listeners to recognize a sound, and the invariants that convey information. The two sets are not necessarily disjoint. For example, we can recognize a machine sound because it is cyclic and the frequency of the cycle informs us of the speed of the machine.

In this section, we first describe invariants of the sounds of wind and wave and the algorithms that synthesize and control them in real-time.

4.1 Wind and wave auditory icons

Wind is associated with a blowing sound gliding up and down continuously, according to a rate that depends on its *strength*. A wave sound ressembles a "shhh" evolving over time in both amplitude and brightness. It can be divided into three parts: the wave breaks (high volume), it rumbles (lower volume, low pitch) until it reaches the beach (more noise and higher pitch). High-level parameters for a wave sound are the *size* which controls the duration and overall amplitude of the sound, the *shape* which describes whether the wave spreads widely on the beach or breaks onto itself, and the *beach*, which describes the type of beach onto which the wave breaks, e.g. sand or rocks.

We can imitate both sounds with algorithms based on dynamic filtering of a white noise. The filter is an IIR (infinite impulse response) filter, whose main effect is to emphasize a set of frequencies. Its parameters are the center frequency it emphasizes and the bandwitdth of the main lobe. An equation for such a filter is [15]:

$$y(n) = G[x(n) - Rx(n-2)] + b_1y(n-1) + b_2y(n-2) \qquad (1)$$

where

$R = e(-\pi B/S)$

$G = 1- R$

$b_1 = 2R\cos(2\pi f_c/S)$

$b_2 = -R^2$

`x(n-t)` represents an input sample delayed by `t` samples

`y(n-t)` represents an output sample delayed by `t` samples

B is the bandwith of the filter, F its center frequency and S the sampling rate.

To create a time-varying filter, we only need to recompute the parameters of eq.1 when B or F changes. Both B and F change according to multi-segments time envelopes controlled by high-level attributes, as described in the following sections.

4.2 Synthesizing the wind sound

For the wind sound, we chose to vary only the center frequency. As the frequency follows a segment, it makes the sound gliding up if the segment is upward and down if the segment is downward. The strength parameter controls the length of each segment, as well as the range of its end points. The greater the strength, the shorter the segments, and the higher the ordinates of the end points (Fig 1).

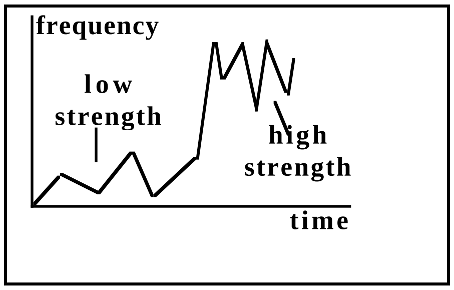

Figure 1: change in filter frequency over time

	wind strength = 1	**wind strength = 100**
Min duration	2 s	0.2 s
Max duration	3 s	0.5 s
Min frequency	100 Hz	700 Hz
Max frequency	200 Hz	900 Hz

Table 1: value domains for the wind sound.

The values given in table 1 give the most realistic results. Min and max duration correspond to the length of a segment. For example, for a wind of strength 1, the duration of the next segment should be selected randomly in the range 2 to 3 seconds. For a very strong wind (strength = 100), the duration should be between 0.2 and 0.3 seconds. For an intermediate wind, linear interpolation can be used. The pseudo-code to compute wind samples is in the appendix.

4.3 Synthesizing the wave sound

The algorithm for synthesizing the wave sound is similar to that of the wind sound, except that more envelopes are involved: a five-segment envelope for the volume, and two four-segment envelopes for the bandwidth and the center frequency of the filter (Fig 2). The y axis represents a percentage of the length of the sound. Intermediate values are interpolated linearly between the coordinates of the control points. The s value in the bandwitdh envelope depends linearly on the shape of the wave, with a range of 600 to 8000. The greater s is, the wider the wave seems to break. The beach parameter acts as a gate, retaining the random value defining the noise for few samples. It makes the sound noisy, as if the wave breaks on rocks instead of sand. The pseudo-code for the generation of wave sound is in the appendix.

Both the wind and wave sounds have been incorporated in the ENO audio server [4]. By using a filtered noise, slow attacks and random values, we avoid high-pitched, high-volume recurrent sounds, making them suitable for continuous play at low volume without annoyance.

4.4 Applications

So far, we have described how to create auditory icons based on the perception of a sound rather than the physical characteristics of the sound. Our approach addresses the issues of recognizing the sound and extracting the parameters of the cause. When a designer wants to incorporate auditory icons in an interface, he must adress the issue of mappings: the mapping of the cause of the sound to a system artifact, and the mapping of the parameters of the cause of the sound to the parameters of the artifact.

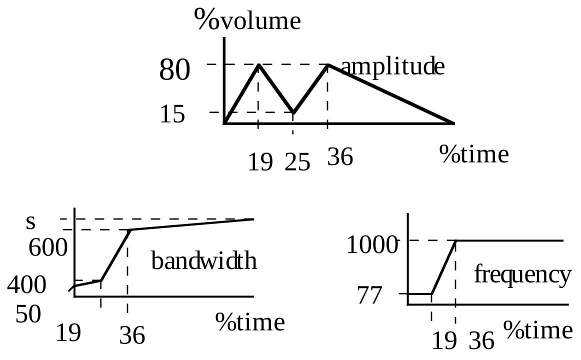

Figure 2: Envelopes for the wave sound

In the Sonic Finder [11], Gaver uses an iconic mapping for the scrape and impact sounds, which should not require any learning from the users. When using genre sounds to monitor file sharing activity, Cohen [7] uses a metaphorical mapping. Similarly, Mynatt, Back and Want [16] use beach sounds to monitor the amount of email. These metaphorical mappings require a learning phase from the user, but experiments showed that when explained, the mappings were well remembered by the users.

Wind and wave sounds correspond to natural phenomena that have no obvious mapping with computing concepts, unlike, e.g., folders in the desktop metaphor. In order to take advantage of the users' knowledge of these sounds, the processes being monitored with these sounds should have a behavior that matches as closely as possible that of the natural events. Since the natural phenomena evolve slowly, using these sounds to reflect a fast-changing activity would confuse the user. Furthermore, perceiving a perturbation in the environment should lead the user to infer that something is going wrong. Thus, using the wind sound to monitor a network flow is inappropriate, since a strong wind would mean a high throughput, which means that the network is performing well. Instead, the strength of the wind should be bound for example to the number of lost packets.

Our current application for these sounds is a process monitor for a network of workstations. We are experimenting with various mappings for the sounds and their parameters. The main idea is that the sound should be noticed when something goes wrong, e.g. overloaded network or workstation, broken hardware. This requires the use of additional sounds for events (as opposed to continuous processes) and a spatial layout of the sounds to help monitor several of them at once. A second application is the monitoring of printing jobs. Parameters like place in the job queue, amount of paper, etc. are mapped to wave parameters, enabling the users to be aware of the state of the process.

While the two applications above involve metaphorical mappings, we could use iconic mapping for activities that depend on weather conditions, e.g. Air Traffic Control or a real-time surf-oriented weather report system.

5 Perceiving high-level attributes in abstract sounds

Since we are more interested in the users' perception of the sound than the sound's physical characteristics, we can use ad-hoc synthesis to define auditory icons that are based on sounds that do not exist in the real world. When monitoring an activity it is often important to know how fast it is progressing. Therefore it would be useful to have a sound that naturally conveys the notion of speed. Sheppard-Risset tones [17] have precisely this characteristic: like M.C. Escher's infinite staircase, they are perceived as going upward or downward forever. In addition the speed of the upward or downward motion can be changed, and the sound can be made «steady».

We have designed an auditory icon that uses Sheppard-Risset tones [3]. Its parameters are the motion direction, the motion speed, and its average pitch. Varying the pitch parameter gives the impression of a size, though there is no real source for such a sound. In the same way, the motion and its speed parameter have no counterparts in the real-world, and yet we are able to associate them to the notion of « motion », « speed » and « size » for the first parameter. These sounds are abstract, because there is no sources in the actual world that can produce them, but we can associate high-level parameters with them, as if they catch invariants from real sounds. This fact shows that sounds need not be natural, as long as they convey useful information through easily recognizable parameters.

5.1 Synthesizing the motion sound

Sheppard-Risset tones are made of a set of partials whose amplitude is bounded by a bell-shaped curve (Fig 3). Over time, the frequency of each partial is shifted upward, while its amplitude is adjusted to fit the bell curve. New partials enter the lower end of the bell curve while partials reaching the other end of the curve disappear. As a result, the overall frequency of the sound is constant, but is perceived as going up continuously. Descending sounds are created similarly by shifting the partials downward.

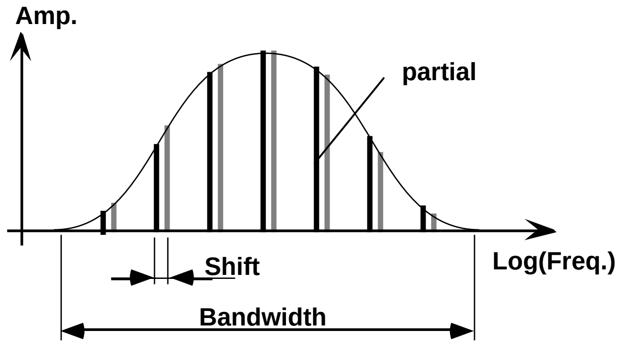

Figure 3: a semi-logarithmic plot of the spectrum of a Sheppard-Risset tone. Over time, the partials' frequencies are shifted upwards.

Sheppard-Risset tones are controlled by 4 parameters :

• Base frequency: the center frequency of the bell curve;

• Bandwidth: the ratio between the highest and lowest frequencies of the envelope curve;

• Speed: the relative shift in frequency per unit of time. The sound is steady when the speed is zero, it moves upward when the speed is positive and downward when the speed is negative;

• Density: the ratio between the frequencies of successive partials. For a given bandwidth, the density determines the number of partials.

We determined empirically the bounds of the parameters necessary to maintain the auditory illusion. Six partials and a density around 1.2 give satisfying results. More partials actually make the sound richer and more intrusive. With these values, the optimal bandwidth is defined by a ratio of 3 (1 1/2 octave). We also found that the bell curve can be approximated with a triangle envelope with no perceivable loss in sound quality. This allows for a more efficient synthesis algorithm.

5.2 Applications

We have used the motion sound to complement the visual display of a progress bar. Progress bars act as a notification mechanism when the system is engaged in a long operation, such as copying a large file, and the progress of the operation (percent-done) is known. The problem with visual progress bars is that they need to be looked at to know the progress of the operation. Since the operation is long, the user is likely to engage in other tasks.

In our system, the speed conveyed by the sound reflects the speed of the progression. Therefore, the sound conveys the instantaneous speed at which the operation progresses rather than the current "percent-done". This way, it is very easy to know when the operation stops progressing: the sound stops moving. This technique also works when the percent-done is not known but the rate of progression can be monitored. For example, when downloading a document, the Netscape World-Wide Web browser displays the current throughput of the connection, even if the size of the document is not known. The throughput can be mapped to the speed of the motion sound, giving an accurate and non-intrusive way to monitor the progress of the operation.

6 Conclusion and future work

Synthesis of auditory icons is usually done using a combination of source and sound analysis. These analyses are not always possible or meanigful, e.g. for environmental sounds. We have introduced in this article ad-hoc synthesis, an approach that emphasizes the perception of the sounds by users instead of the analysis of actual sources and sound. We have described two substractive synthesis algorithms for generating and controlling wind and wave sounds in real-time by means of high-level parameters. These sounds capture the main invariants of the sounds they imitate, enabling users to recognize and understand them easily. We then pushed the approach further by looking for an «abstract» sound that conveys the notion of speed. This led us to use an auditory illusion called Sheppard-Risset tones.

If synthesized sounds are to be meaningful for a large number of users, we must make sure that they match the way the users imagine these sounds. Thus, we can wonder whether the idea of a sound is shared among a large majority, or in other words whether invariants are perceived equally by humans. This raises the question of how people describe sounds. Ballas has conducted related experiments in which listeners were expecting the sound of a particular event [2]. Thus, the experiment partly measured the fit between a real sound and the invariants listeners were paying attention to. Maybe designers could use the set of sounds listeners found adequate for the expected event as a basis for synthesis algorithms. Another solution could be to ask professionnal audio designers to describe verbally the invariants of a particuler sound, or even to design sounds with real-time and control constraints in mind (not like movie or tv-show) [1].

We plan to investigate synthesis techniques such as FM or additive synthesis to design a larger number of sounds. When we have enough sounds, we will conduct experiments to test and validate their usefulness in real applications.

Acknowledgments

Thanks to Michel Beaudouin-Lafon and Paul Janecek for their useful comments on this paper.

Appendix

Pseudo-code for the wind auditory icon

```
proc next_time(integer   strength) : integer
do
 dmin = (0.2 - 2) * strength + 2
 dmax = (0.5 - 3) * strength + 3
 return dmin + (dmax-dmin) * rand()
end

proc next_freq(integer   strength) : integer
do
 fmin = (700 - 100) * strength + 100
 fmax = (900 - 200) * strength + 200
 return fmin + (fmax-fmin) * rand()
end

proc next_sample(integer   time): float
do
 if(time == ntime) do
        ptime = ntime
        ntime = next_time(wind_strength)
        pfreq = nfreq
        nfreq = next_freq(wind_strength)
 endif
 f = interpolate_frequency(ptime,ntime,pfreq,nfreq)
 compute_filter_coef(filter, B, f)
 return sample_through_filter(filter, rand())
end
```

rand() is a function returning a random value between 0 and 1. It is used to select the values for frequency and duration (next_time and next_freq) and to synthesize a white noise (call to sample_through_filter).

pseudo-code for the wave auditory icon

```
proc get_interpolated_amplitude(time t, time total_time) : float
do
  ratio =100 * t / total_time
  if(ratio<0.19)
        result =  (80-15)/(19) * ratio
  elsif(ratio <25)
        result =  (15-80)/(25-19) * ratio - (15-80)/(25-19) * 19 + 80
  ...
  return result / 100
end

proc get_interpolated_amplitude(time t, time total_time, int maxshape)
...
proc get_interpolated_frequency(time t, time total_time)
....

proc next_wave_sample(time t, time total_time, int max_shape, float rand) : float
do
  amp = get_interpolated_amplitude(t, total_time) // return a value between 0 and 1
  band = get_interpolated_bandwidth(t, total_time, max_shape)
  freq = get_interpolated_frequency(t, total_time)
  compute_filter_coef(filter, band, freq)
  return amp * sample_through_filter(filter, rand)
end

proc compute_wave(size, shape, beach )
do
  length = 2 + 3 * (size /100) // duration between 2 and 5 sec.
  s = 600 + 7400 * (shape / 100 )
  integer i = 0
  float random_value
  while(i <> length)
        if (i % beach ==0) // the more is beach, the noiser the sound
              random_value = rand()
        next_wave_sample(i, length, s, random_value)
  end
end
```

References

1. Back, M. Micro-Narratives in Sound Design : Context, Character, and Caricature in Waveform Manipulation. In *Proc. International Conference on Auditory Display, ICAD'96*, 1996.
2. Ballas, J. A. Delivery of Information Through Sound. In *Proc. International Conference on Auditory Display, ICAD'92*, pages 79-94, 1994.
3. Beaudouin-Lafon, M., and Conversy, S. Auditory Illusions for Audio Feedback. In *Companion Proceedings, ACM Human Factors in Computing Systems, CHI'96*, Vancouver (Canada), pages 299-300, April 1996.
4. Beaudouin-Lafon, M., and Gaver, W. W. ENO: Synthesizing Structured Sound Spaces. In *Proc. Symposium on User Interface Software Technology, UIST'94*, ACM, pages 49-57, 1994.

5. Chowning, J. M. The Synthesis of Complex Audio Spectra by Means of Frequency Modulation. In *Journal of Audio Engineering Society* 21, pages 526-534, 1973, reprinted in *Computer Music Journal* 1:2, pages 46-54, 1977.
6. Cohen, J. Monitoring Background Activities. In *Proc.International Conference on Auditory Display, ICAD'92*, pages 499-531, 1992.
7. Cohen, J. « Kirk here: » Using Genre Sounds To Monitor Background Activites. In *Adjunct Proceedings, ACM Human Factors in Computing Systems, INTERCHI'93*, Amsterdam (The Netherlands), pages 63-64, 1993.
8. Freed, A., Rodet, X. and Depalle, P., Synthesis and Control of Hundreds of Sinusoidal Partials on a Desktop Computer without Custom Hardware. *In proc. International Conference on Signal Processing Applications and Technology, ICSPAT'93*. DSP Associates, Boston, MA, 1993.
9. Gaver, W. W. What In The World Do We Hear ? An Ecological Approach To Auditory Event Perception. In *Ecological Psychology,* 5(1), pages 1-29, 1993.
10. Gaver, W. W. How Do We Hear In The World ? Explorations In Ecological Acoustics. In *Ecological Psychology,* 5(4), pages 285-313, 1993.
11. Gaver, W. W. Auditory Interfaces. In *Handbook of Human-Computer Interaction, 2nd edition*. Amsterdam, The Netherlands: Elsevier Scienc., 1997.
12. Gibson, J. J. The Senses Considered as Perceptual Systems. *Boston : Houghton Mifflin*, 1966.
13. Gibson, J. J. The Ecological Approach to Visual Perception. *Lawrence Erlbaum Associates*, 1986 (originally published in 1979).
14. Miner, N. E. and Caudell, T.P. Using Wavelets to Synthesize Stochastic-based Sounds for Immersive Virtual Environments. In *Proc.International Conference on Auditory Display, ICAD'97*, 1997.
15. Moore, F. R. Elements of Computer Music.*Prentice Hall*, 1990.
16. Mynatt, E. D., Back, M., Want, R. Designing Audio Aura. In *Proc. Human Factors in Computing Systems, CHI'98*, pages 566-573, ACM, 1998.
17. Risset, J.C. Paradoxes de Hauteur. *IRCAM Report no. 10*, IRCAM, 1977.
18. Roads, C. Initiation à la Synthèse Sonore par Modèles Physiques. In *les cahiers de l'IRCAM (2)*, pages 145-172, 1993.
19. Roads, C. and Strawn, J. Foundations of Computer Music. *The MIT Press*, 1985.
20. Van Den Doel, K. and Pai, D.K. Synthesis of Shape Dependent Sounds with Physical Modeling. In *Proc.International Conference on Auditory Display, ICAD'97*, 1997.