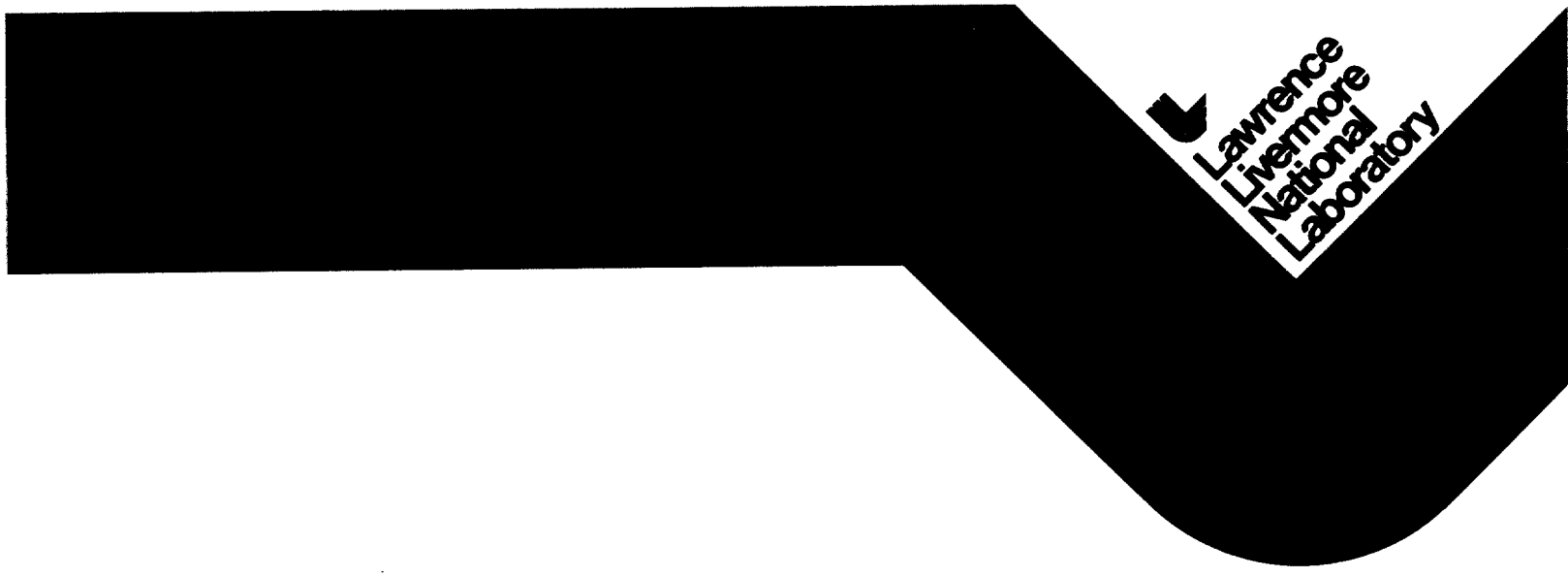
UCRL-LR-128454

Presentation of Dynamically Overlapping Auditory Messages in User Interfaces

Albert Louis Papp III

PhD Thesis

September 1997



Work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-ENG-48.

UCRL-LR-128454
Distribution Category UC-705

Presentation of Dynamically Overlapping Auditory Messages in User Interfaces

Albert Louis Papp III

Doctor of Philosophy
Thesis

Manuscript date: September 1997

LAWRENCE LIVERMORE NATIONAL LABORATORY
University of California • Livermore, California • 94551

Presentation of Dynamically Overlapping Auditory Messages in User Interfaces

BY

ALBERT LOUIS PAPP III

B.A. (Rutgers College, Rutgers University) 1991
M.S. (University of California, Davis) 1995

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Committee in Charge

1997

Abstract

This dissertation describes a methodology and example implementation for the dynamic regulation of temporally overlapping auditory messages in computer-user interfaces. The regulation mechanism exists to schedule numerous overlapping auditory messages in such a way that each individual message remains perceptually distinct from all others. The method is based on the research conducted in the area of *auditory scene analysis*. While numerous applications have been engineered to present the user with temporally overlapped auditory output, they have generally been designed without any structured method of controlling the perceptual aspects of the sound.

The method of scheduling temporally overlapping sounds has been extended to function in an environment where numerous applications can present sound independently of each other. The *Centralized Audio Presentation System* is a global regulation mechanism that controls all audio output requests made from all currently running applications. The notion of *multimodal objects* is explored in this system as well. Each audio request that represents a particular message can include numerous auditory representations, such as musical motives and voice. The Presentation System scheduling algorithm selects the best representation according to the current global auditory system state, and presents it to the user within the request constraints of priority and maximum acceptable latency.

The perceptual conflicts between temporally overlapping audio messages are examined in depth through the *Computational Auditory Scene Synthesizer*. At the heart of this system is a heuristic-based auditory scene synthesis scheduling method. Different schedules of overlapped sounds are evaluated and assigned penalty scores. High scores represent presentations that include perceptual conflicts between overlapping sounds. Low scores indicate fewer and less serious conflicts. A user study was conducted to validate that the perceptual difficulties predicted by the set of heuristic algorithms actually existed in test subjects.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This dissertation investigates the creation and regulation of auditory user interfaces that use dynamic, temporally overlapping, non-speech auditory messages. One way people normally perceive their environments is through the auditory channel. This task requires the perception and analysis of many different temporally overlapping sounds constantly arriving at the ears. The ability of the human perceptual system to group sound into discrete, coherent, and recognizable units can be used effectively in auditory user interfaces. Through the intelligent creation and regulation of auditory interfaces, the user can be presented a very rich and informative computer environment without reliance upon the visual channel.

1.1 Auditory User Interface Design

The usage of non-speech audio has become a common means of communication in computer-user interfaces. Although non-speech sound has been generally used for alerting the user to some particular event of interest, other auditory user interfaces have communicated a wide variety of information. The types of information represented include, but are certainly not limited to, seismic data [Hay94], execution behavior of parallel algorithms [JF94], mathematical equations [SE96], background computer network activity [Coh94], and interactions in virtual environments [Beg94a]. Even visual user interface content has been translated into combined speech and non-speech audio output [Myn94a] [Ram96]. Approximately forty such interfaces are described in the 1992, 1994, and 1996 Proceedings of the International Conference on Auditory Display [Kra94a] [KS94] [FK96].

Interest in auditory-based and auditory-enhanced user interfaces has been motivated by a number of factors, including the following:

1. The ear is very sensitive to air pressure changes. We receive a great deal of information about our surrounding environment through the auditory channel. It should be possible to take advantage of this fact to present information to the user through the computer interface.

2. There are situations in which the visual channel is not available for use. Examples include the following circumstances:

   - Computer operation in a dark or otherwise low-vision environment.
   - Computer usage by vision-impaired users.
   - When the eyes are busy looking away from the computer monitor in order to oversee some other task, such as the operation of a motor vehicle.
   - When no visual display is available, such as with very small hand-held or ubiquitous computing devices.

3. The time-varying nature of sound makes it a natural medium for the presentation of time-varying data.

4. Sound is a natural interaction modality used by people. There exists great potential to simplify user interfaces through the use of well designed auditory messages.

1.2 Temporally Overlapped Auditory Messages

In some instances it is advantageous to temporally overlap auditory messages to increase the density and speed of the presented information or to explore temporal relations between different data. This approach has been tried in a number of auditory user interface designs. The main problem is that each sound must be carefully designed so that it can be clearly segregated from a number of other concurrently played sounds. No auditory user interface design has incorporated a control mechanism that regulates the overlapped auditory output in such an interface. Brewster did examine how non-speech audio could be effectively used in the interface, but did not consider the issues of temporally overlapped audio. His emphasis was on temporally non-overlapped auditory messages [Bre94].

This dissertation provides a methodology for controlling the output of auditory user interfaces that employ temporally overlapping non-speech auditory messages. The perceptual problems encountered in this type of interface are unique and have not been investigated in the context of user interface design. This fact has not, however, stopped researchers from creating new and novel computer interfaces that

use audio in the described manner. Unfortunately, there is very little experimental data on human performance in using such interfaces.

1.3 Auditory Grouping Models

The human perceptual system must constantly perform an analysis of the frequency components that reach the ears. This analysis must group together components arising from the same source both at a given time and over time. At a given time, the components must be assigned to their respective originating sources, and over time, each set of components arising from the same source must be grouped into the same perceptual unit. This process was formalized by Albert Bregman who, in 1990, published the book entitled *Auditory Scene Analysis* [Bre90]. The model presented in this book has formed a basis for a number of computational approaches to sound analysis, including the research conducted in this thesis. The term *auditory stream segregation* refers to the process by which the perceptual system segregates sounds into individual perceptual units that are distinct from one another and that arise from different sources.

The process of auditory scene analysis is of particular interest in the field of auditory computer-user interfaces because it governs how the user will perceive the auditory output from the interface. However, to date, there have not been any auditory interface design methodologies that attempt to exploit the research done in perceptual psychology as it relates to audition. The lack of sound usage in everyday computer applications may be attributable to the lack of auditory user interface design research that addresses auditory perception.

Computational models of auditory scene analysis have been created for purposes such as identifying sound sources and predicting the most likely interpretation of the auditory scene. Some computational models attempt to separate and isolate the different auditory sources which comprise some real-world recorded audio segment in the same way as the human perceptual system does. A number of different approaches have been taken to solve this problem, including "data-driven" sequential processing methods [Bro92] [Coo93] and "prediction-based" methods [Ell96]. With these types of models the user can do such things as remove footsteps (and other unwanted sounds) from a vocal recording by isolating the voice as a separate sound source from the footsteps. The voice can then be recreated without the presence of the footsteps in the recording.

Other models operate only on sequences of pure tones and attempt to predict how the auditory system will organize and group the tones into higher level perceptual structures [Wil89] [BM91] [Van77b].

1.4 Research Goals

The broadest goal of this dissertation is to demonstrate that the presentation of dynamic temporally overlapping auditory messages can be controlled in such a way that the user can segregate each individual sound source. Furthermore, only a relatively small number of heuristics is needed to construct compound sound presentations containing the qualities that allow the easy stream segregation of each individual source. Each heuristic describes a condition that leads to a perceptual conflict between two sound sources. Through the identification of the possible conflicts, and through the development of a methodology to prevent such problems, auditory user interface designers can benefit through simplified sound design requirements and perceptually clear dynamic presentation of auditory messages to users. The process of constructing auditory output that consists of a number of temporally overlapped yet easily discernible sound sources will be named *Computational Auditory Scene Synthesis.*

A number of specific topics have been investigated in the pursuit of the general thesis. Briefly, these ideas include the following:

- **Multimodal Objects:** If user interface objects are defined to have more than one representation, then the presentation of a particular object can be made using any one of the given forms. If numerous auditory representations are available, the most easily perceived form can be dynamically selected at the time of presentation in the context of other currently playing auditory messages. For example, a message indicating pulse rate could have three auditory representations: a voice form in which the rate is spoken periodically, a sound effect form that emits a series of sound pulses that correspond to actual pulses, and a tonal form in which the pitch varies according to the current pulse rate.

- **Modification of Auditory Messages:** A particular auditory message can be dynamically altered in a well-defined manner to improve its predicted stream segregation from other currently playing auditory messages. An example is the change in pitch of a spoken voice so it is perceived more clearly in the presence of musical tones.

- **Useful Computational Auditory Scene Synthesis Heuristics:** A set of heuristics is needed that sufficiently describes the perceptual conflicts that might arise between particular sounds that overlap in a specific way relative to each other.

- **Creation of Dynamic Auditory Messages:** Since sounds are analyzed at run-time, new sounds can be created during the execution of a program and presented to the user. This type of sound will normally be created in response to some dynamically changing data value that is measured or computed during the program execution. Each dynamically created sound can be analyzed and presented to the user using the same techniques as statically created sounds.

- **Scheduling of Auditory Messages:** It is presumed that auditory messages must be temporally overlapped because of time constraints on each individual message. The possible scheduling combinations of a group of sounds within a given time interval grows exponentially as the number of sounds increases. However, a number of different schedules must be examined in order to find a "good" solution. A good solution is one that allows the listener to easily discriminate each individual sound source in a temporally overlapped presentation of many auditory messages. A fast and efficient search technique is required to find a good solution in a "reasonable" amount of time.

The method for presentation of dynamically overlapping audio for user interfaces has evolved through the implementation of three systems and a user performance study. These systems demonstrate that auditory user interfaces can be regulated through an intelligent server, and that the user's perception of the resulting auditory output is improved over an unregulated auditory output system.

Flinn and Booth proposed a hypothetical audio server that presents multiple audio sources in a perceptually distinct manner, but they were interested in considering the impact of various qualities of sound (timbre, pitch, etc.) in isolation from each other [FB95]. This approach could be problematic because the change of one qualitative sound property has a profound impact on the perception of the other qualitative properties. This problem is also evident in the multi-dimensional data sonification projects described in Chapter 3 that map sound attributes (such as timbre and pitch) to different time varying attributes of the data.

The cognitive process of assigning meaning to a given sound is an important issue but is beyond the scope of this research. Elizabeth Mynatt has conducted studies that examine people's interpretations of naturally occurring sounds when perceived out of context [Myn94b].

1.5 Thesis Contents

This dissertation is divided into seven chapters. The first chapter is this introduction.

Chapter 2 briefly summarizes the relevant research conducted pertaining to the perception of temporally overlapped sounds. The majority of the work comes from two disciplines: perceptual psychology and music perception. The field of perceptual psychology has supplied numerous controlled experiments that have greatly contributed to Bregman's formulation of auditory scene analysis. Many observations have been made in music perception that directly pertain to the perception of temporally overlapping sounds. Music perception, and specifically *counterpoint*, describe how certain compositional techniques control the listener's perception of sequences of multiple note sequences that overlap each other. At times, many notes played in unison can create the auditory sense of a single, full sound. Different concurrent melody lines can also be designed to remain quite distinct from each other. Much of this work is used in the composition and arrangement of music. With such knowledge, composers and arrangers can exercise some control over the perceptual effects created by the performance of music.

Chapter 3 describes a selection of auditory user interfaces that have been created and are somehow relevant to this thesis. Generally, sound is used to monitor some number of time-varying data inputs, or as an indicator of relevant events that are occurring in the particular system.

In Chapter 4, two implemented systems are described. These systems were programmed by the author to investigate the issues involved in dynamic temporally overlapping auditory user interfaces. The first implementation is the *Auditory Map System* in which the user receives auditory feedback about selected regions of a map. The second implementation is the *Centralized Audio Presentation System.* This system runs as a server that receives "audio requests" in real-time from other running applications. The sounds are presented by the server in a way such that the perceptual conflicts between overlapped sounds are minimized. Within well-defined parameters, the Presentation System alters certain auditory and temporal characteristics of each sound request. It schedules each sound according to priority, latency, and the current global auditory state. Each scheduled sound (that is optionally modified) is played within its latency constraint.

Chapter 5 describes the architecture and implementation of a more sophisticated sound presentation tool named the *Comprehensive Auditory Scene Synthesizer.* It presents a number of inputted sounds to the user within a given time interval. Different temporal layouts of the sounds within the time interval are assigned dif-

ferent "perceptual scores," where a lower score indicates a clearer presentation of the sounds to the user. The problem then is to find the best layout possible in a reasonable amount of time. The heuristics used in the Comprehensive Auditory Scene Synthesizer are more complex than in the Centralized Audio Presentation System. Both the heuristics and the search methods are described.

In Chapter 6, the results of a user study are presented. The perception scores computed by the Comprehensive Auditory Scene Synthesizer were compared against user performance in the identification of distinct sounds in a set of temporally overlapped sound sources. Also, a prediction function is described, which approximates average user performance based upon the data collected.

Finally, in Chapter 7, the results of this research are summarized. Also outlined are some of the many interesting areas to explore as a result of this work.

Chapter 2

An Overview of Auditory Scene Analysis and Music Perception

In the design of any auditory user interface, the human task of perception must be investigated. The consideration of perceptual issues is even more critical when numerous sounds occur simultaneously. The interactions between temporally overlapping sounds can greatly affect the listener's perception of the sounds. One example of this is masking. Masking occurs when a louder sound perceptually hides a quieter sound. However, the effect of masking can often be negated through the use of the various perceptual cues described in this chapter. Many more such interactions can take place, depending upon the properties of the sounds used and their relative temporal positions. The problem becomes further compounded as the number of concurrent sounds increases beyond two.

Two fields of study that have considered the problems in the perception of overlapping sounds are perceptual psychology and music perception. These areas have approached the problems encountered with sound perception differently. Mostly within the last 50 years, perceptual psychology has contributed experimental results that provide insight into the mechanisms that regulate sound perception. These experiments have been done generally using very simple sounds such as tones and noise bursts. Music perception studies have explored ways to control the perception of sound (specifically, music) through compositional techniques, and have arrived at conclusions based upon observations and methodologies used in music composition for hundreds of years. The methods are evident in musical forms such as *fugue*, and more generally, *counterpoint*. The domain of sounds used in music has generally consisted of the sounds generated by musical instruments. Music composers and theoreticians have amassed a large amount of knowledge related to presentation of different instrument timbres, orchestration, arrangement, and spatial effects of sounds.

The information presented in this chapter is intended as a brief introduction to the areas relevant to this research. The disciplines of sound related perceptual psychology and music perception are each large research areas by themselves. Bregman has written a comprehensive starting point into the investigation of these areas [Bre90].

2.1 Terminology

Auditory scene analysis is a term first used by Albert Bregman [Bre90, p. 3]. It is the process by which the human perceptual system can translate the raw auditory data that reaches the ears into a coherent description of the listener's environment. For example, an auditory scene might consist of a dog barking, a bird chirping, a refrigerator humming, and a person speaking. The frequencies that comprise each individual source are jumbled together upon reaching the ear, yet through our perceptual systems we are able to recognize four distinct auditory sources. Some definitions are necessary to further describe and discuss this phenomenon.

A sound can be described as a wave. If that wave is periodic, or if it is non-periodic but of a limited duration, it can be represented as the sum of a number of sine waves, each with a particular frequency and amplitude. In Figure 2.1, the periodic wave on the bottom is formed by adding the three sinusoidal waves above it. Each sine wave is a *frequency component* of the original wave. The method of finding the frequency components contained in an arbitrarily complex wave is called *Fourier analysis*, named after the French mathematician who derived the technique. The process of recomposing a complex wave from its frequency components is called *Fourier synthesis*. The representation of a wave as amplitude over time is called the *time domain representation* of the wave. A wave represented as amplitude over frequency is called the *frequency domain representation*, or *spectrum* of the wave. Figure 2.2 shows the frequency domain representation of the wave on the bottom of Figure 2.1. The heights of each spike in Figure 2.2 correspond to the amplitudes of each component sine wave. In this case, the amplitudes of each component sine wave are equal.

Pitch is a quality that lets the listener judge the "highness" or "lowness" of a sound. In music, a note is described as having particular pitch. The frequency of a periodic wave is the most significant contributor to the sensation of pitch. A *tone* is a sound with a pitch. Tones can have different degrees of perceived pitch, and are roughly classified into two categories: *harmonic tones* and *inharmonic tones*. A tone that can be described as having a very strong, definite pitch is called a *harmonic tone*. Harmonic tones contain a number of frequency components that are all integral multiples of a common low frequency. For example, a sound com-

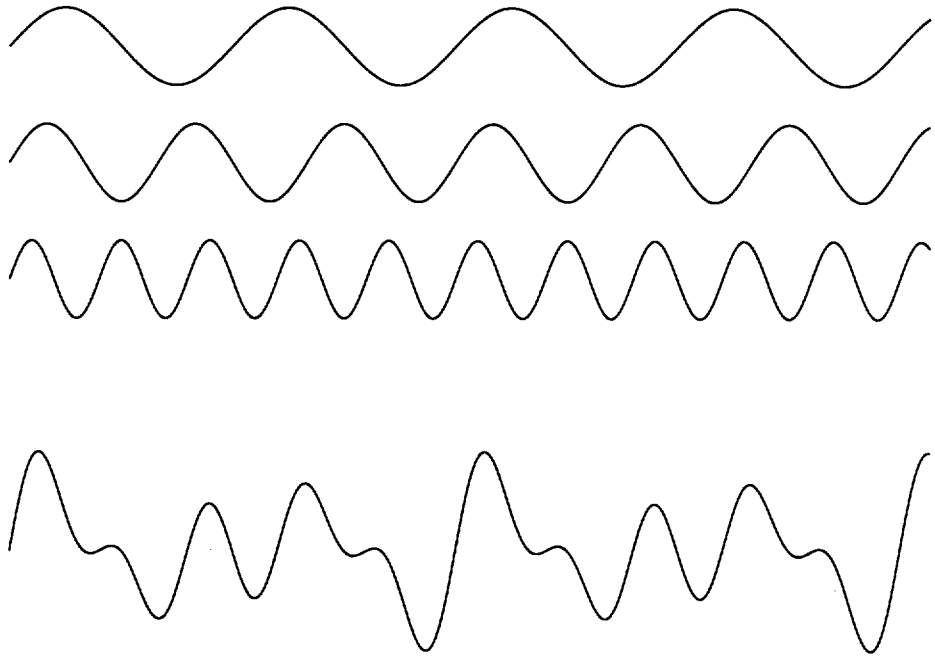


Figure 2.1: The addition of three simple sine waves into a complex wave.

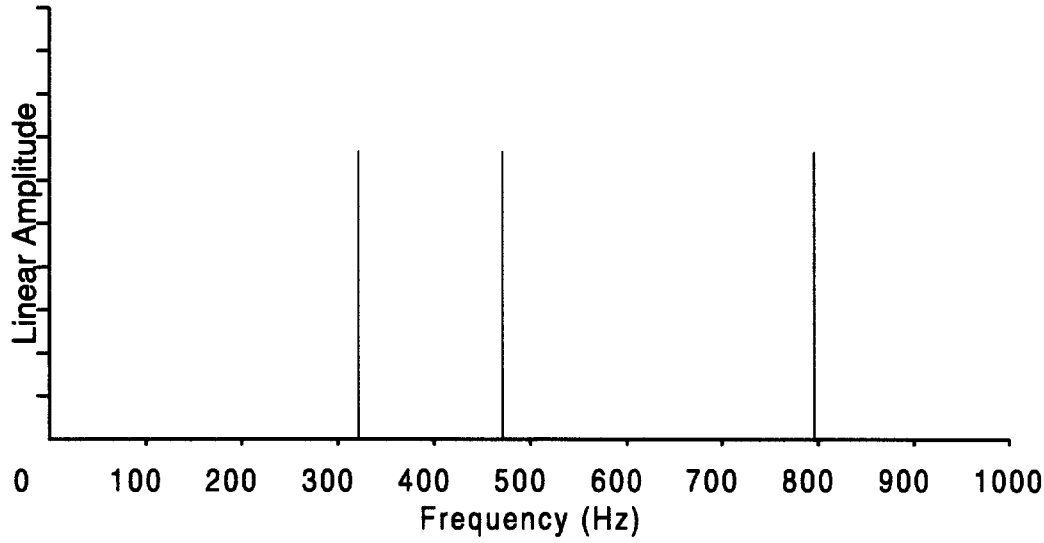


Figure 2.2: The frequency domain representation of the three frequency components forming the complex wave in Figure 2.1.

prised of frequency components 100Hz, 200Hz, 400Hz, and 900Hz, is a harmonic tone since each frequency component is an integral multiple of 100Hz. The low frequency (in this case, 100Hz) is the *fundamental frequency*, or f_0. Each frequency component that is an integral multiple of the fundamental frequency is called a *harmonic*. The fundamental frequency (or f_0) is the lowest harmonic, also known as the 1^{st} harmonic. The 2^{nd} harmonic (or f_1) is the frequency component that is twice the frequency of the 1^{st} harmonic. In general, the n^{th} harmonic is the harmonic that has the frequency $n * f_0$. The n^{th} harmonic is also called f_{n-1}. The set of harmonics for a fundamental frequency is called the *harmonic series* of that fundamental frequency. In a harmonic tone, not all of the harmonics for the fundamental frequency need to be present. In fact, the fundamental frequency does not even need to be present. In a tone containing the frequency components of 100Hz, 200Hz, 400Hz, and 900Hz, the fundamental frequency is present (100Hz) as are the 2^{nd}, 4^{th}, and 9^{th} harmonics.

Some tones may not be perceived as having a very strong pitch attribute. These tones are *inharmonic tones*, and they have frequency components that are not all multiples of some fundamental frequency. For example, a tone with frequency components of 100Hz, 250Hz, 479Hz, and 792Hz, is an inharmonic tone since there is no fundamental frequency of which each component is an integral multiple.[1] The frequency components therefore cannot be called harmonics. They are instead referred to as *partials*. A partial is a general term for a frequency component of a tone that may or may not be harmonically related to a fundamental frequency. Inharmonic tones are generally machine-created for the purposes of experimental perceptual studies. They generally do not occur in natural environmentally produced sounds. Musical instruments generally only produce harmonic tones in relation to the length of a tube or the length of a resonating string.[2] A *complex tone* is a tone comprised of partials. It may be a harmonic tone or an inharmonic tone. The partials therefore may or may not be harmonically related to the fundamental frequency.

The amplitudes of the frequency components of a sound can vary over the time in which the sound exists. Consider the example of a trumpet note. A trumpet note has a strong sense of pitch, and therefore is a harmonic tone. Since the note is a harmonic tone, it makes sense to talk about the harmonics that comprise it. Each harmonic increases in amplitude from 0 (silence), and eventually returns to 0. Over the lifetime of the note, each harmonic will vary in amplitude to some degree. The spectrogram representation of a trumpet note is shown in Figure 2.3.

[1] Technically, each of the frequency components is an integral multiple of 1Hz, but since humans cannot hear sounds at such a low frequency, this case is ignored.

[2] Certain instruments, such as snare drums, produce sounds that do not contain as strong a sense of pitch as most other instruments.

It shows how each harmonic varies in amplitude in a series of equal time steps through the duration over which the sound is audible.

2.2 Auditory Stream Segregation

Although the trumpet spectrogram is clearly comprised of many different frequency components, the listener hears only a single sound, recognizable as a trumpet note. The process by which the perceptual system groups related frequency components into a single basic unit is called *auditory stream fusion.* A trained listener may be able to overcome the effect of auditory stream fusion and actually hear some of the individual frequency components, but only under controlled conditions and with full concentration dedicated to the task.

Generally, in the process of perceiving the environment, the human auditory system must reconstruct discrete sound events from an extremely large number of frequency components constantly arriving at the ears. Auditory scene analysis is the process by which a perceptual description of the surrounding environment is formed through the grouping together the frequency components that arise from the same sound source. This process of segregating from the auditory scene a related group the frequency components arising from a particular sound source is referred to as *auditory stream segregation.* The environment may consist of numerous sounds, including harmonic tones, speech, birds chirping, and wind noise. Each of these sound events becomes an unique and recognizable object in the environment, even though a very eclectic set of frequency components from all these sounds is the raw data that arrives at the ears. From this low level information, the perceptual system creates sound source entities that have attributes such as volume, movement, and timbre. This process is outlined in Figure 2.4. All of the frequency components continually arising through time from a particular sound source must be assigned to the same perceptual unit known as an *auditory stream.*

It appears that this process occurs through a "voting" mechanism of sorts, where numerous properties of the frequency components contribute varying amounts to the final decision of how the auditory scene is to be perceptually organized. An example of one such property is spatial location. The frequency components perceived as originating from the same spatial location tend to be grouped together into a singular auditory event. This property and a number of others are described throughout this chapter.

Principles from Gestalt psychology have been used to explain the auditory streaming process as well. Just as in vision, sounds form auditory "gestalts." Figure 2.5

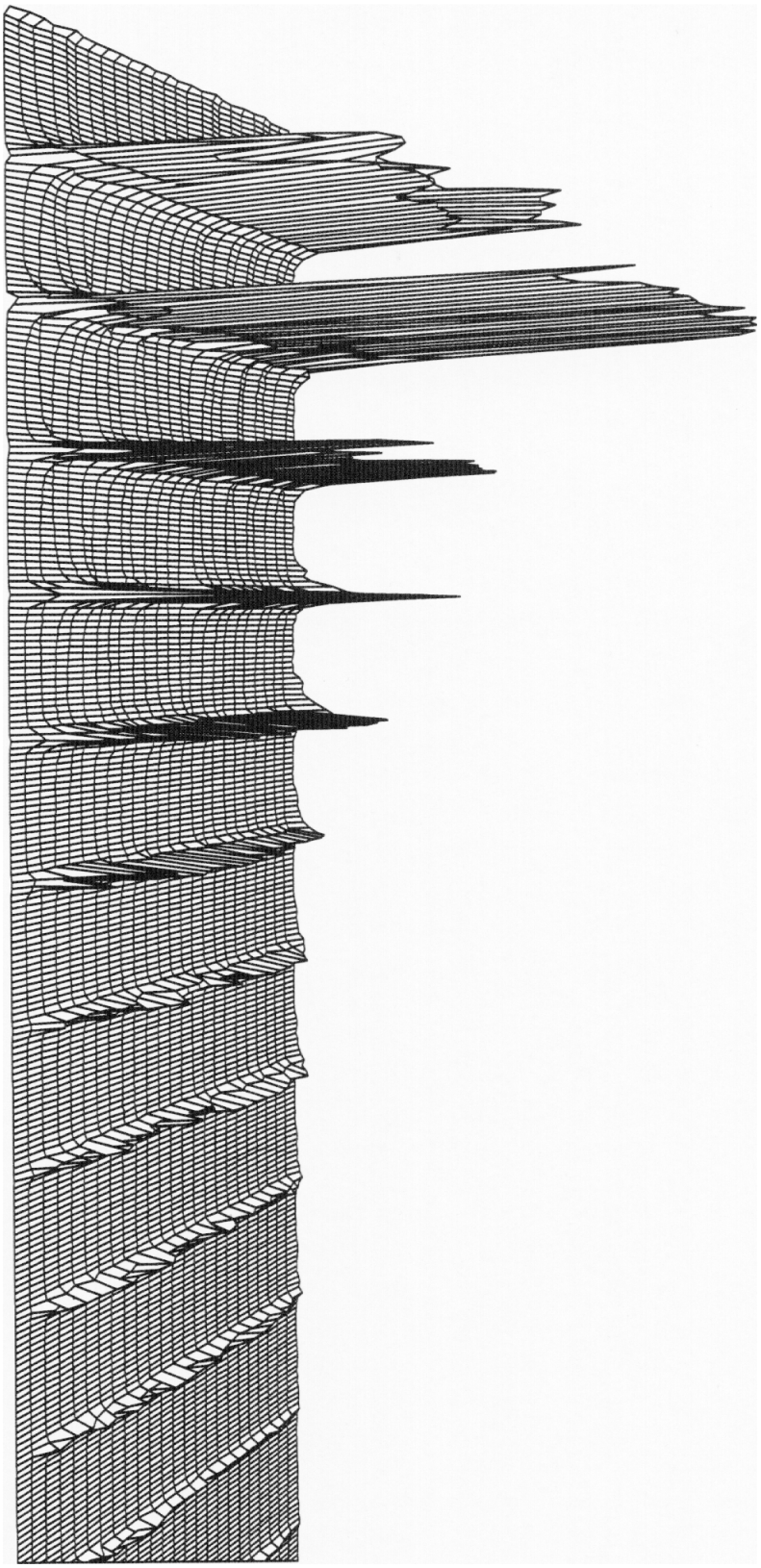


Figure 2.3: The partial spectrogram of a trumpet note. The note is F above concert A, where concert A is defined as 440Hz. The first 11 harmonics are shown. Frequency is on the x-axis (left to right). Time is on the y-axis (into the page). Amplitude is on the z-axis (height). Frequency components are integral multiples of the fundamental frequency because this is a harmonic tone.

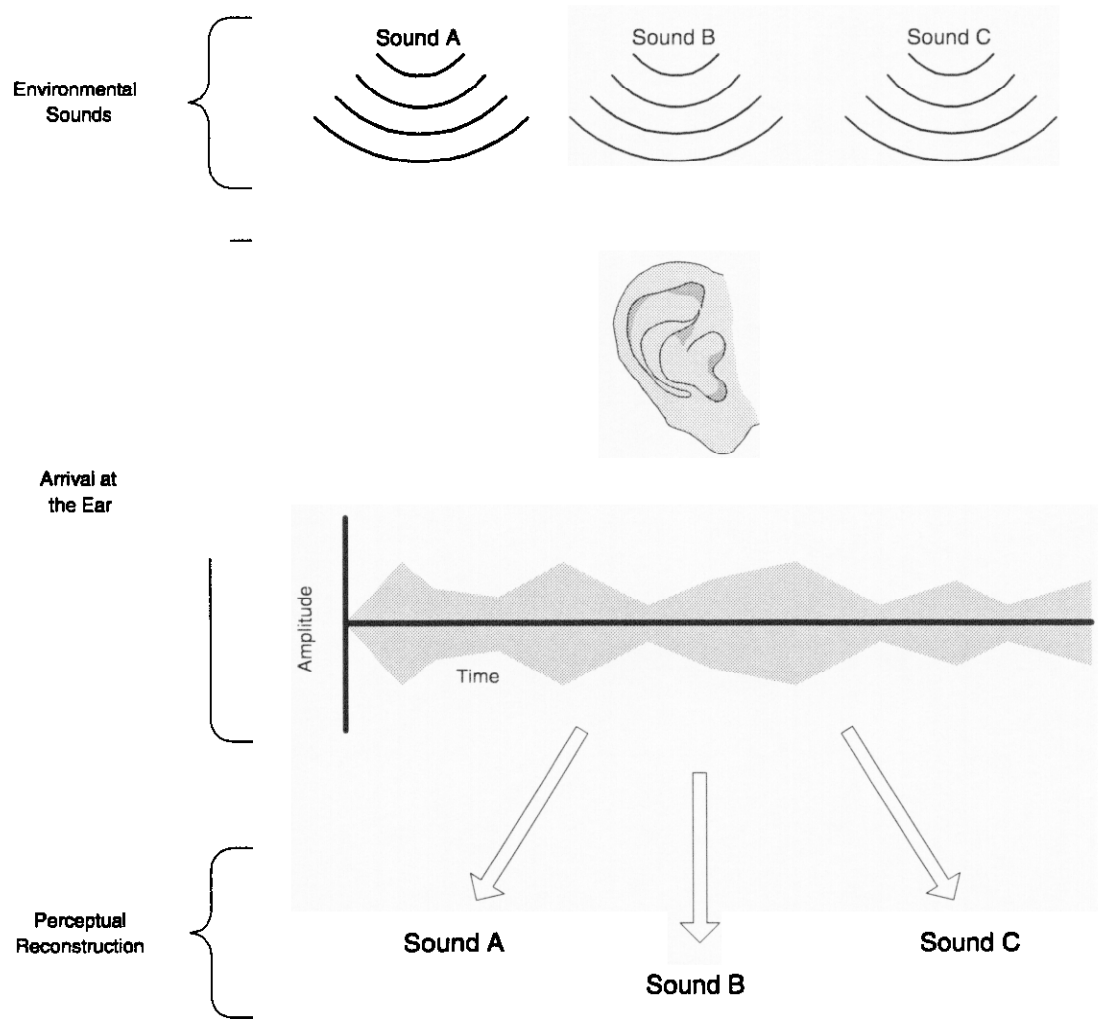


Figure 2.4: The process of auditory scene analysis: the ear receives frequency components from numerous sources. When measured at the ear, all of these components form one "compound" sound. Somehow, the perceptual system must reconstruct from this compound sound all of the original sound sources along with positional and distance estimates for each contributing sound source.

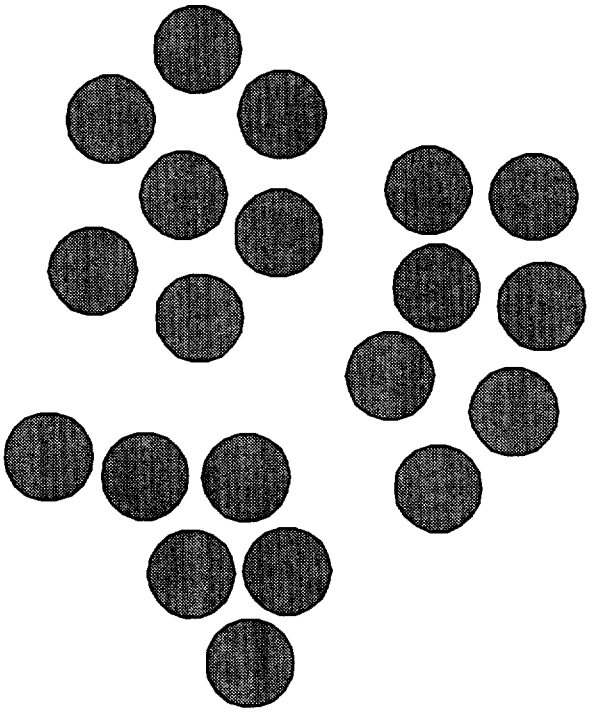


Figure 2.5: The circles in the figure tend to be visually organized into three groups according to the Gestalt principle of proximity. Items appearing spatially clustered together are perceptually grouped.

shows an example of a visual gestalt based upon the principle of proximity. A related auditory gestalt, *frequency proximity*, will be described in Section 2.2.1.2. An example of sound groupings based on frequency proximity is illustrated in Figure 2.8. Williams has written a concise description of the Gestalt psychology principles and their relation to auditory streaming [Wil94]. A number of the Gestalt principles that relate to the perceptual organization of sound are described later in this chapter.

In his formulation of auditory scene analysis, Bregman categorized auditory streaming cues into two rough categories: sequential and simultaneous [Bre90, Ch. 2–3]. Sequential integration is the method by which the perceptual system groups sound components over time into an auditory stream. Simultaneous component integration is the process of grouping frequency components (all present at some given time) into larger perceptual units recognizable as atomic sound sources. A variety of perceptual cues appears to guide these processes. Each cue, regardless of its type, provides some guideline to the perceptual system to assist in recreation of the sound scene from the low level auditory data.

All of the following auditory streaming cues are known to be inherent to the human auditory perceptual system — none needs to be learned or practiced. These cues are basic to human experience of auditory scene analysis. Each has been identified through the process of scientific experimentation. Furthermore, no one cue acts independently of the others. Since there are many cues that are contending simultaneously (sometimes with conflicting information), the perceptual system must weigh the significance of each and appropriately apply or ignore that information.

2.2.1 Sequential Integration Cues

The first cues described are the sequential cues. They guide the perceptual system in determining which frequency components arising over time are from the same sound source, and hence should be assigned to the same auditory stream. The cues discussed in this section are provided as an introductory sampling to this body of research — there is evidence for many other cues beyond those discussed here.

2.2.1.1 The Spatial Location Cue

One of the strongest cues for sequential integration is the spatial location cue. It makes sense that frequencies arising from the same spatial location are all parts of the same complex sound. The spatial cue, among others, helps the listener determine, for example, that a sequence of chirps emanating from the same spatial location originates from the same bird. If the perceived location of a series of bird chirps alternated between the left and right sides of the listener, this would be a strong indication that more than one bird (or more generally, more than one sound source) was present.

Although spatial location is a strong stream segregation cue, it can be overcome by other cues. A good example of this is an experiment conducted by Deutsch, known as the "scale illusion" [Deu82]. The note pattern in Figure 2.6A was presented to test subjects. The most common perception was that of Figure 2.6B. The notes were perceptually grouped according to frequency proximity (described in the following section) rather than spatial location.

2.2.1.2 The Frequency Proximity Cue

There is considerable evidence in support of the assertion that sequential tones stream together based upon similarity or proximity in frequency. In a classic audi-

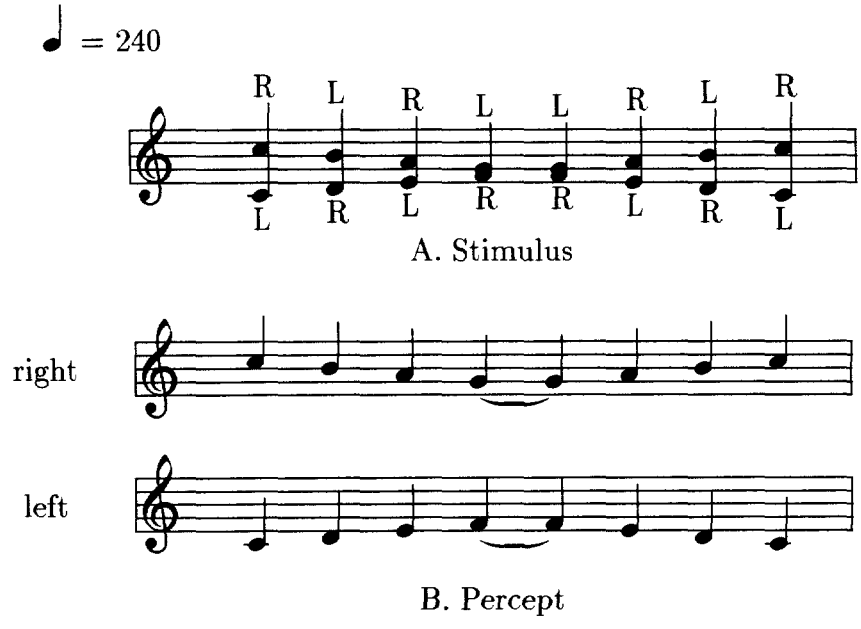


Figure 2.6: (A) The scale illusion stimulus that was presented repeatedly to test subjects. The notes of an ascending scale were presented alternately to the left and right ears, as were the notes of a concurrently played descending scale. (B) The most common perception of the stimulus, grouped by right and left ear. Notes were grouped by frequency proximity rather than by spatial location (Adapted from [Deu82, p. 102]).

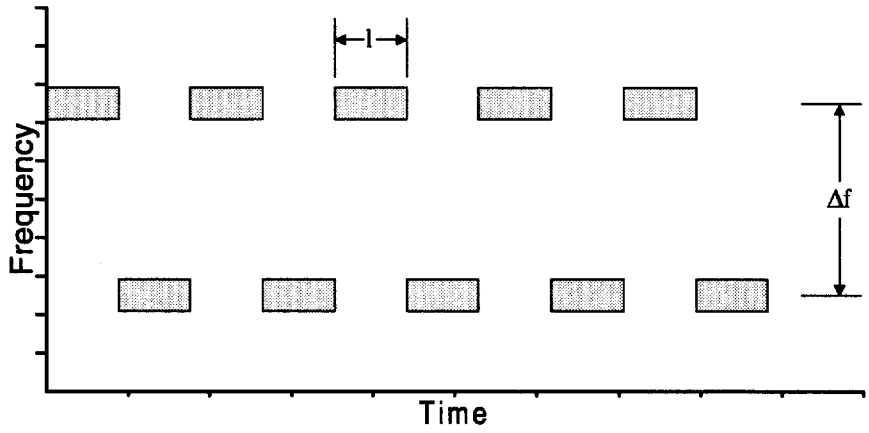


Figure 2.7: A sequence of tones used to test for frequency proximity. Depending upon l and Δf, the sequence will be perceived as either one or two auditory streams.

tory streaming experiment, van Noorden played two tones of alternating frequencies in sequence, as in Figure 2.7 [Van77a]. Through the variance of the speed and frequency differences, he measured the threshold of where the perceptual system groups the notes into either one or two streams. When the frequency difference between the tones (Δf) was great enough, and the length of each tone (l) was short enough, the sequence was heard as two separate streams. There was a loss of temporal relation between the streams, indicating that the two streams were perceptually distinct from each other. Generally, when l is larger (longer tones than 100 ms.), the notes are heard as a single stream, even when Δf is relatively large (up to one octave of separation). As l becomes smaller (shorter tones than 100 ms.), Δf must be relatively small (5 semitones or less) in order to hear the sequence as one stream.

Further frequency proximity experiments were carried out by Bregman using the tone patterns shown in Figure 2.8 [Bre78]. Tones A and B were always the same frequency, and the frequencies of tones X and Y were varied. The relative temporal positions of the four tones were always kept constant. Depending upon the frequencies of X and Y, A and B would be grouped into the same stream (as in the left half of the figure) or into different streams (as in the right half of the figure). The different perceptions are caused by the "competition" factor of multiple tones to group with other tones similar in frequency to themselves.

Bregman found that in the absence of other streaming cues, a tone B following another tone A in time will fuse into the stream of A provided there is no better candidate available to group with A. A better candidate would be another tone

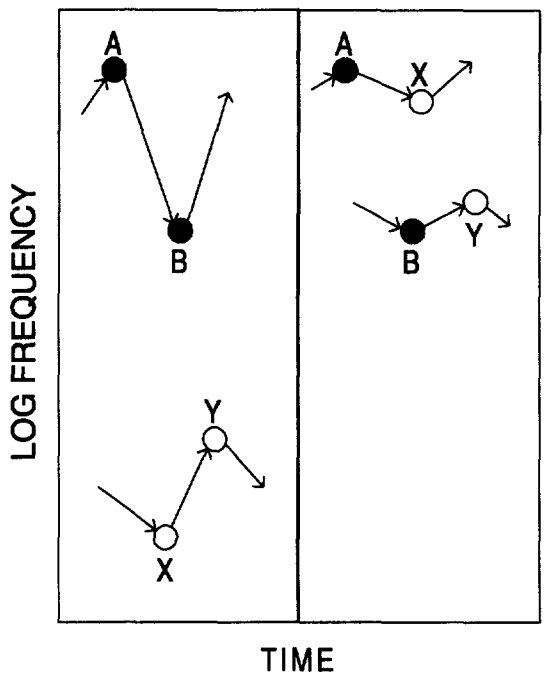


Figure 2.8: Competition of frequency proximity on stream segregation (From [Bre78]). (Left) In this repeated pattern, tones A and B form one auditory stream, and tones X and Y form another. The arrows indicate the tones that stream together. (Right) In this repeated pattern, tones A and B are unchanged. However, the frequencies of tones X and Y are changed. Now, because of the new proximal relationships of the tones, A and X form an auditory stream, as do B and Y.

that is closer in frequency to A than B is to A. Even if the frequency difference between A and B was large, B would still stream with A as long as there were no other tones in competition with B that would make better streaming candidates at that given time.

2.2.1.3 Rhythm Patterns and Streaming

There is evidence that regular rhythmic patterns of sounds allow the perceptual system to "predict" the next future sound that will become part of an auditory stream [JKW81]. However, the mechanisms that govern this prediction are unclear. The process may be based on lower-level auditory streaming processes (perhaps the periodic nature of certain nervous system activities) as noted by Jones, Kidd, and Wetzel [JKW81], or may be schema-based according to the particular listener's experiences, as noted by Bregman [Bre90, p. 442].

Other experiments on rhythm perception have shown that when tone sequences such as in Figure 2.7 divide into separate streams, the relative rhythmic relation between them is at least partially lost. This is an instance of a general streaming rule: when sounds form unique auditory streams, each stream tends not to perceptually interact with any other stream.

If this rule was always true, the perception of a music performance would be much different than it actually is. For instance, in a musical duet, the two instrument parts frequently form two distinct auditory streams for the listener. However, there are times when the streams blend together, creating, for example, a consonant musical interval. If no interaction between the streams was possible, the listener would not have the experience of hearing the harmonic relation between the two notes. So the stream segregation between the instruments may be thought of as dynamic, varying in amount according to the currently changing auditory situation. The fusing of the auditory streams can occur as a result of the presence of numerous common note onsets and offsets in the instrument rhythms.[3] However, when the rhythms of the instrument parts are syncopated in relation to each other (i.e., the onsets of the different instruments' notes do not coincide), segregation is more likely. See Figure 2.9. Bregman discussed the concept of *hierarchical grouping of sounds* to explain how a duet can be perceived both as two streams and as a single sound source [Bre90, pp. 203–205]. Other relations present between overlapping sounds can cause fusion or segregation as well. They are the subject of the following section.

[3]The effect of common note onsets and offsets on auditory stream segregation is discussed in Section 2.2.2.2.

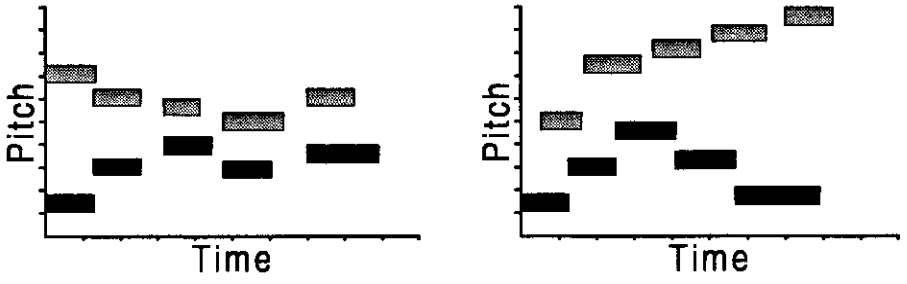


Figure 2.9: The effect of syncopation on auditory stream segregation. Two different instruments play a sequence of notes. (Left) The (non-syncopated) notes of the different instruments tend to begin at the same time. (Right) The notes of the two instruments are syncopated in relation to each other because they do not begin at the same time. These syncopated notes tend to encourage auditory stream segregation between the two instrument parts.

2.2.2 Simultaneous Integration Cues

The remaining cues described are used by the perceptual system to group simultaneous frequency components into perceptually discrete higher level auditory objects. The ear is constantly deluged with frequencies ranging throughout the audible spectrum. The components at any given time originate from numerous sources, and the perceptual system must group the related components back together. The cues described in this section assist in that task.

2.2.2.1 The Spatial Localization Cue

Spatial position is a strong cue for not only sequential integration, but for simultaneous integration as well. This is logical, because components arising from the same source at a given time must arise from the same physical location. The components of the auditory scene that emanate from a common spatial location therefore tend to fuse into the same stream. Binaural audition is required for strong sound localization. Using only monaural audition, other cues can help to distinguish between streams originating from different locations in space, but the stream segregation will generally not be as strong.

Begault has shown that in situations where many sounds must be presented to a listener concurrently, such as in an aeronautical cockpit or a traffic collision avoidance system, three-dimensional spatial separation of each sound source improves the perceptibility of each source markedly [Beg93] [Beg94b]. Wenzel has outlined

the performance advantages to be expected by adding spatial sound display to the presentation of multiple sound streams [Wen94].

2.2.2.2 Common Grouping Mechanisms

The ear is very sensitive to timing and small timing variations of sounds. The timing of related auditory events is important. It assists in grouping related components of a sound into a single source. A basic law in auditory grouping mechanisms is that acoustic frequency components that change in similar ways are related and tend to fuse into a common stream. This rule comes from the Gestalt *common fate* theory of perception.

Bregman observed,

> It is exceedingly improbable that unrelated sounds will just happen to go on or off at exactly the same time. Therefore, synchrony is an excellent indication that acoustic components must have arisen out of the same sonic event. [Bre90, p. 261].

The perceptual grouping of components based upon common starting time is the principle of *common onset.* Similarly, the grouping of acoustic components that end simultaneously is the principle of *common offset.* These particular grouping cues are effective only when the onset or offset times of a set of frequency components are relatively similar to each other. This can happen even if the components arise from numerous sound sources. In this case, other cues are needed to perceptually segregate the multiple sounds from each other.

Other common changes that occur over time include different forms of modulation. These include modulations in frequency and amplitude. As long as a particular modulation occurs over all related components of a source at the same rate and in phase, it tends to fuse those components. However, it was shown that if some subset of frequency components is modulated differently from the others, that set of components tends to fuse into a different source from those components that do not have the changed modulation [Moo82, p. 193].

In 1979 John Chowning discovered that the partials of his computer-synthesized singing voice would not fuse into a unified stream until he introduced a small amount of frequency modulation to each component comprising the synthesized voice. All real instruments and human voices display this property of *micromodulation* [Cho80]. Micromodulation is a small frequency modulation in a sound, typically ranging from 1% for a clarinet to 20% for a singer's vibrato [Bre90, p. 253].

It is a ratio modulation, meaning that each partial from a common sound source modulates by the same multiplicative constant. All partials, therefore, remain harmonic throughout the sound. It was shown by McAdams that these modulations help fuse the partials into a common stream. In his experiments, trained listeners were able to discern the first five to seven individual harmonics of a sustained, unmodulated sound. However, with micromodulation added, the subjects could no longer do this, but rather heard only one fused pitch [McA84].

2.2.2.3 Harmonic Relations of Complex Tones

The partials of complex tones have a strong tendency to stream together if they are all multiples of the same common fundamental frequency. In reference to harmonically related frequency components, Rasch and Plomp stated that

> We become familiar with the complex tones of speech signals (both of our own speech and of other speakers) from an early age. It would not be efficient to perceive them all separately. [RP82, p. 7]

They determined that the fusion of harmonics occurs because of the Gestalt principles of common fate. While common fate of each harmonic helps to fuse the tone, it does not explain why inharmonically related partials do not also fuse as well into one tone. After all, the partials of an inharmonic tone can possess the same Gestalt properties as a harmonic tone. It is not clear exactly why the harmonic relation of partials is such a strong cue for the fusion of those partials, but it may be related to the necessity of the auditory system to approximate the pitch of naturally resonating objects. Generally, objects resonate at many frequencies simultaneously, all of which are harmonically related.

If the relative amplitudes of a series of harmonics are such that the lower partials mask the higher ones, then the fusion is even stronger. Masking occurs when a louder sound conceals the presence of a softer sound. It is well known that lower frequencies mask higher ones better than higher frequencies mask lower ones. Therefore, a set of partials tends to become more strongly fused into one tone when the lower partials have a greater amplitude than the higher partials. This is normally the case with the harmonic tones of acoustic instruments — they usually contain a pattern of partials that have decreased amplitude with increased harmonic number for a given time interval. Figure 2.3 illustrates this general trend for a trumpet.

It has been shown that the perception of a pitch is tied to the auditory system's ability to find a regular spacing of harmonics from a given source [Bre90, pp. 232–235]. If some harmonics are missing, they become "filled in" by the perceptual

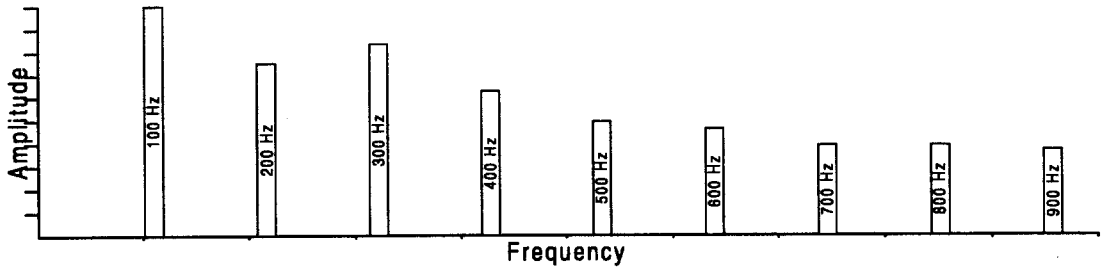
system. In other words, the timbre of a sound is controlled, at least partially, by the harmonic frequencies which are present in the sound through time; however, the perceived pitch is controlled, at least partially, by a combination of the present and the implicit frequencies which together construct a regular harmonic series. An interesting case of pitch perception occurs when the fundamental frequency is not present. Assuming that there are at least two consecutive harmonics in the spectrum, the perceived pitch is that of the missing fundamental, since the period of the wave is the same regardless of the presence of the fundamental frequency. Essentially, the frequency of the perceived pitch for a group of harmonically related partials will be that of the smallest frequency interval in its spectrum. The mystery of the missing fundamental is shown in Figure 2.10.

2.2.2.4 Common Harmonics of Multiple Complex Tones

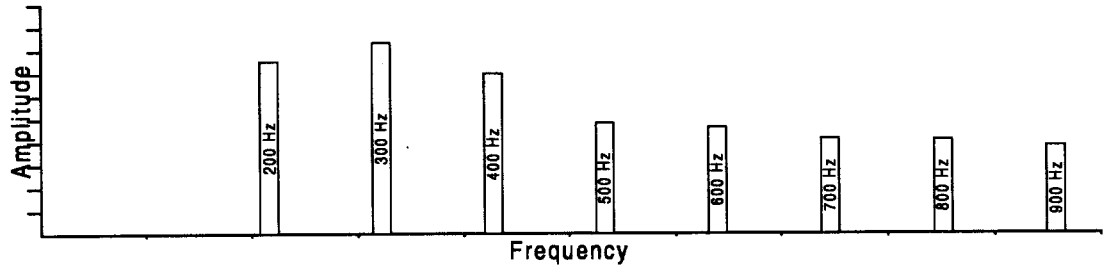
Interesting perceptual interactions can take place when more than one complex harmonic tone is playing simultaneously. In some cases, two tones may fuse into one. In other cases they do not. The fusion of two tones into one is likely if both tones are built upon the same fundamental frequency. The tones then share the same harmonic series. It has been shown that pitches remain more distinct when the two fundamentals have as few common harmonics as possible [Ras78]. This is supported by the observation that two unique notes are most difficult to distinguish when they are at exactly an octave apart [BW82]. In this situation, the harmonics of the higher note all coincide with harmonics of the lower note. Even if every coincident partial has zero amplitude in the lower note (as is the case with a tone containing only odd numbered harmonics), the result will still be a fusion of the two tones into one rich sounding tone. In Figure 2.11, examples of complex tones and their overlapping harmonics are presented. In the cases where many harmonics are common between multiple tones, segregation of the tones can only occur if there are other strong cues present (such as spatial localization and various common grouping cues) to indicate that the tones belong in different perceptual streams.

2.3 Music Perception Issues

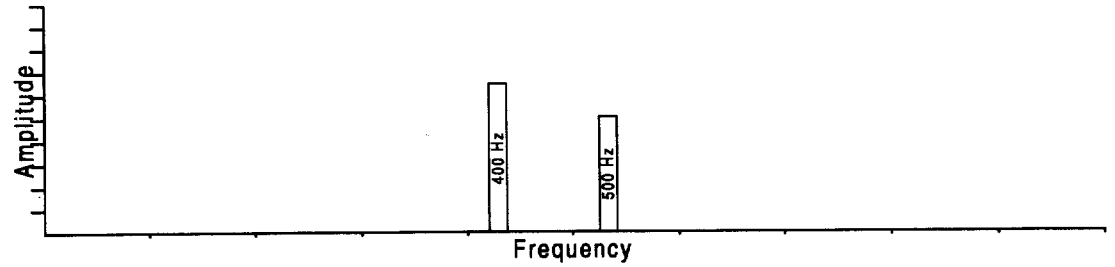
It is probably not a coincidence that many of the guiding principles of music perception seem to be related to some corresponding experimental results from perceptual psychology. Music perception is based upon the human ability to form auditory



This sound, consisting of nine frequency components, is perceived as a single 100 Hz. tone. This tone is rich in harmonics and will have a "full" sound.



This sound is also perceived as a 100 Hz. tone. It has a "hollower" timbre. This is an example of the mystery of the "missing fundamental." The period of the wave form is the same as that of a 100 Hz wave.



This sound may be perceived as a 100 Hz. tone as well. It has a very "thin" timbre. In this case the fundamental is missing as well as next two harmonics. Higher harmonics are not present either. In the absence of additional auditory stream fusion cues (i.e., common frequency or amplitude modulations), the frequency components may actually be heard as two individual tones.

Figure 2.10: Pitch perception for harmonic sounds. The auditory system perceives the tone corresponding to the smallest frequency interval in the set of harmonics.

Tone 1	100Hz	200Hz	300Hz	400Hz	500Hz	600Hz	700Hz	800Hz
Tone 2		200Hz		400Hz		600Hz		800Hz

Example 1: Tone 2 is one octave higher than tone 1. All partials in tone 2 are also in tone 1. Tendency for fusion of the two sources into one source is strong, in the absence of stronger conflicting cues.

Tone 1	100Hz		300Hz		500Hz		700Hz	
Tone 2		200Hz		400Hz		600Hz		800Hz

Example 2: Tone 2 is one octave higher than tone 1. There are no common partials between the two sources. Both sources still tend to fuse into one source, because both tones are composed of harmonics which have a common fundamental. In this case, that common fundamental is 100Hz.

Tone 1	100Hz	200Hz	300Hz	400Hz	500Hz	600Hz	700Hz	800Hz
Tone 2		180Hz		360Hz		540Hz		720Hz	...

Example 3: Tone 2 is between one and two octaves higher than tone 1. Segregation between the tones is strong since all harmonics in tone 2 are unique from tone 1, and no harmonics between the two sources share a common fundamental, as in examples 1 and 2.

Figure 2.11: Coincident harmonics in two concurrently sounded tones.

streams and to resolve the ambiguities that arise from the presentation of numerous temporally overlapping sound sources. Many of the details and intricacies of music perception are beyond the scope of this discussion. However, there are certain elements of music perception that are important to the computational auditory scene synthesis method to be described in Chapter 5. Those topics are described briefly.

Many of the sounds used in this research are complex tones and sequences of complex tones. These tone sequences form motives, or small musical passages. The timbres of the tone sequences are easily recognized as familiar musical instruments. Music perception research is potentially quite useful in the presentation of numerous temporally overlapping motives, where each motive must remain perceptually distinct from all others. Through adjustment of the relative temporal positions of the motives to be played, it may be possible to create a sound presentation that contains strong stream segregation properties for each individual motive. The process of temporal adjustment assumes, of course, that there is no requisite temporal or harmonic relation between any of the motives.

In the following discussion, the terms "horizontal" and "vertical" refer to the relation of notes when visualized on the musical staff. Horizontal notes follow each other in time and they appear horizontally next to each other, although they can

be moved up or down in pitch. Vertical notes are those which are sounded concurrently in time. They appear on the staff to be on top of one another in a vertical formation.

2.3.1 Counterpoint

The musical form of *counterpoint* is of particular interest to this research. Counterpoint is a method, perfected by Johann Sebastian Bach in the early 18^{th} century, of creating a composition that has two or more concurrent yet distinct melody lines. Composers have found, without the benefit of psychoacoustic or perceptual psychology research, methods to encourage the perceptual system to segregate each melody line into a different auditory stream.

It is well known that melody lines tend to be more cohesive when there are no large pitch intervals, or horizontal jumps, between consecutive notes. To segregate two concurrent melodies, one melody line will rarely cross the other in the "vertical" component of the composition. In other words, if the two melodies were visualized as lines composed of segments connecting adjoining notes, those two lines would have few, if any, intersections. This creates the effect of a "higher" melody and a "lower" melody. The pitch similarity of consecutive notes and absence of pitch crossings are properties related to the auditory streaming principle of frequency proximity. It is not a surprising observation then, that different parts of musical arrangements (such as for a string quartet) are often separated by pitch intervals spanning more than one octave.

The ability to segregate each instrument into a unique pitch range or to overlap pitches when desired gives the composer flexibility to control the amount of stream segregation between the different instruments. At some points in the composition, harmonic relations between different instruments playing in a similar pitch range will tend to form unified chords. At other times, each instrument may play in a unique pitch range to encourage the perception of each part as a separate entity. The control over stream segregation is further enhanced by the use of rhythm and syncopation between parts. Perceptual psychology has shown that common changes in frequency components tend to group them into the same stream. This has been known and used in music composition for hundreds of years. Different instrument parts that are to fuse into the same stream will tend to have many similar properties such as common onsets and offsets, and common crescendo and decrescendo. Parts that should be segregated do not have these properties relative to each other. These tools allow the composer to create expressive relations between parts.

2.3.2 Familiarity of Motives

It seems reasonable to expect that a familiar motive would tend to stand out in the presence of other motives. A cognitive process could essentially "predict" the next notes and, if actually present, assign them to the same auditory stream. There is evidence that this happens. Meyer contended that styles in music are complex systems of probability relationships [Mey57, pp. 54–56]. For example, in the literature on music theory, there are observed relations between consecutive notes and chords in music. Walter Piston observed that root progressions in music follow a number of probabilistic rules, a few of which are listed here [Pis41]: [4]

> I is followed by IV or V, sometimes by VI, less often II or III.
>
> II is followed by V, sometimes VI, less often I, III, or IV.
>
> III is followed by VI, sometimes IV, less often II or V.[5]

The probabilities of certain roots following others exist because the listener has an *expectation* based upon a broad range of musical relations he or she has been exposed to. The expectations of musical relationships are not the same for all people of a culture, and certainly vary between cultures. Eastern music is based upon a tonal system different than that of Western music. As a result, different styles of harmonic relations and motives exist. The probabilities are different as are the expectations of the listener. Even within the work of a single composer, certain musical styles are presupposed, but may not be present in a particular composition. The expectation of each individual work may require the understanding of other expectations from others' compositions. Meyer noted that

> Thus, although the full cadence and diatonic melodic motion are not prevalent in the style of Wagner, for example, Wagner's style nevertheless presupposes these as basic norms. It seems to this writer that in stylistic study and analysis there is no substitute for a sensitive response to style. This can be achieved only through practice in listening and better still in performance. [Mey57, p. 56]

The realization of expectations and predictability of motives may then be the process of listening to a large enough sampling of a particular musical style to gain the ability to discern the deviations from the "norm" in the structure of the motive.

[4] The Roman numerals are labels for the ascending notes of the tonal system or scale used. Each represented note is the root of a chord built upon it.

[5] Partial listing of table taken from [Mey57, p. 54]

Dowling performed an experiment in which two familiar melodies were interleaved in the same pitch range [Dow73]. Subjects could not recognize either upon listening. After being told the name of one melody, they could hear the melody after listening to the example three or four more times. This experiment demonstrates that familiar motives can be distinguished from the auditory scene, even when there are no other helpful cues to assist in the perceptual formation of that motive. However, this cognitive process requires concentration and information about what might be heard.

2.3.3 Timbre

It is very difficult to define exactly what timbre is. It is often categorized as "that which allows one to differentiate between two instruments playing the same note at the same volume." The research conducted on creating multi-dimensional timbre spaces would indicate timbre to be some function of physical properties [Wes79] [Gre75]. The dimensions of the timbre spaces are attributes such as brightness and rate of attack. However, the dimensions identified in these studies were fit to the limited timbre inputs used in the analysis. It is problematic to take a new timbre and fit it into the timbre spaces designed from other timbres. Other approaches have been taken to describe timbre. Slawson examined the spectral location and movement of formants to characterize timbres [Sla81]. Others have proposed that the physical cause of a sound (i.e., striking or scraping) allows categorization of timbres. For example, an object struck in different ways can greatly affect the spectral content of the sound, but it is still recognized as the same timbre but with a different characteristic of striking force [Bre90, pp. 483–485].

The qualities of instrument timbres influence how they are used in a composition. Composers author scores with the knowledge of how the timbres interact to cause different effects such as *ensemble timbres*, or new timbres created from the layering of other timbres. There is a tendency for similar timbres to group together perceptually. This principle allows one to recognize a violin section in an orchestra without necessarily being able to hear any individual performer's contribution. Different timbres are often segregated into different streams as well. The section of violins tends to be very perceptually distinct from the section of trumpets. The two instruments have different sets of frequency components that also vary in different ways. The common grouping mechanisms allow the listener to organize these into separate streams.

2.3.4 Spatial Separation

Bregman described the work of Henry Brant, related to music composition and performance with respect to space [Bra67] [Bre90, pp. 500–502]. Brant observed that to further enforce the perceptual separation between different sources, spatial effects are used. He found that two different instruments frequently blend both in rhythm and timbre if spatially close; however the listener can distinctly hear the rhythm and timbre differences when the same instruments are spatially separated. Furthermore, the listener receives a better sense of the distinctness of different parts of the composition when the sources are further apart in space, even when the pitches of different sections overlap. There is a tradeoff for music perception in using the large spatial separation between sections: the spatial separation will cause different sections to segregate so strongly that any rhythmic or tonal relation between the sections (such as a chord in which each note is contributed by a different instrument section) will be much less pronounced, if detected at all.

2.4 Conclusion

The human perceptual system uses numerous techniques to resolve the auditory scene analysis problem. These techniques include detection of common changes in frequency components, frequency proximity, and spatial location cues. Sometimes the information received through these cues is ambiguous or contradictory. The perceptual system must resolve these situations somehow, and that resolution may occur through a type of "voting" mechanism in which each cue has some number of votes. Section 4.3 will discuss this point further.

There was little formal study of the processes that control auditory stream segregation until the 20^{th} century. However, the underlying perceptual processes were clearly understood by music composers since the Renaissance period. This is evident in music styles such as the fugue, a particular type of imitative polyphony [KK80, pp. 38, 185]. In the early 18^{th} century, Johann Sebastian Bach and many other composers used counterpoint, a type of nonimitative polyphony in which numerous melodic lines were concurrently played. In these polyphony styles, the composer had to control the stream segregation of each overlapped melody line. To increase the segregation between melody lines, each line would frequently be scored in a unique pitch range, consecutive notes would have small pitch jumps, and pitch crossings between concurrent melody lines were minimized. These methods (as well as others used) have all been shown through psychological experimentation to increase stream segregation.

This very brief introduction to auditory scene analysis and music perception serves as a foundation for the computational methods to be discussed in the following chapters. This chapter is not intended to be an exhaustive overview of these areas of research. Rather, the topics are presented because each has direct significance to the design of the comprehensive auditory scene synthesis method described in Chapter 5. The material presented in this chapter forms a foundation for the heuristic-based control of auditory stream segregation between asynchronously presented temporally overlapping sounds in computer-user interfaces.

Chapter 3

Non-Speech Sound for User Interfaces

Sound has been integrated into user interfaces primarily in two ways: as a tool to represent a set of multi-dimensional data, and as an event marker or auditory cue. When sound is used to represent data, different attributes of sound are commonly mapped to different dimensions of data. When sound is used as an event marker, the qualities of the sound convey information about the particular event and possibly what the system has done in response to it. Also of interest in this case is the structure of the sounds and how they effectively form a sonic "language."

This chapter examines the ways in which sounds have been structured for user interfaces, and describes a number of auditory interfaces used for a variety of applications. Each auditory interface is briefly examined in relation to auditory scene analysis.

3.1 Data Representation in Sound

The efforts in data sonification have been directed towards the translation of temporally varying multi-dimensional data into sound. For example, the meteorological data of temperature, air pressure, humidity, wind speed, and wind direction, all measured at the same location in one minute intervals for a year could comprise a five-dimensional data set that one might wish to examine. Each of the five variables can be mapped to a different sound "attribute," such as pitch, brightness, loudness, timbre, and spatial location. Pitch, for example, could be mapped to temperature — higher pitches could represent higher temperatures, and lower pitches

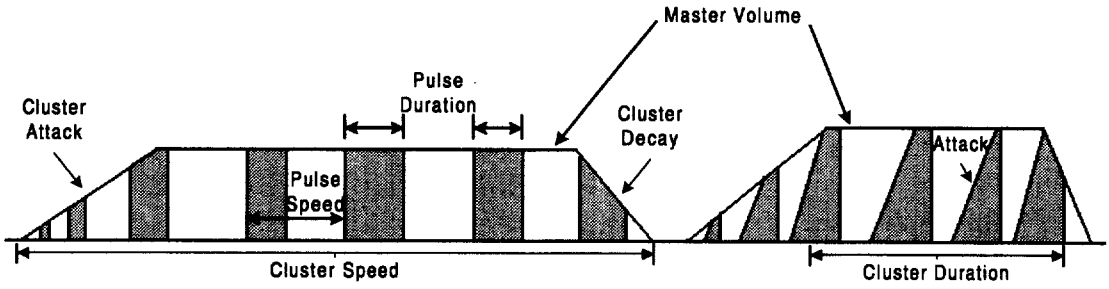


Figure 3.1: Loudness Nesting Parameters (Taken from [Kra94b, p. 193]).

could represent lower temperatures. Each sound parameter continuously varies according to the data set as the sound continues through time. Upon listening to the sound created from the data, perhaps a better understanding of the complex relationships of weather indicators could be attained. For example, through the nature of the resulting sound, it could be determined that air pressure and wind speed correlate dependent upon the humidity level. These types of relationships may become evident in the translation of the physical measurements to auditory output.

Kramer used a technique he called "parameter nesting" to create many parameters from only a single sonic parameter such as amplitude or frequency [Kra94b]. Parameter nesting is the alteration of the same basic parameter on different relative time scales simultaneously. For example, Kramer has defined five parameters based upon what he called "loudness nesting." A continuous sound periodically has its amplitude reduced to zero for a short duration. This creates the effect of sound "pulses." The resulting *pulse speed* is the first parameter that a dimension of data can be mapped to. The *duration* of a set of pulses at maximum amplitude is another parameter. The *envelope* of each pulse (specifically, the attack and decay rates) provides one, or possibly two, additional parameters. One can put an amplitude envelope over groups of pulses (referred to as *clusters*) to create the parameter of *cluster speed.* The *master volume* of all sound pulses is the final parameter. In Figure 3.1, a sound signal is shown that illustrates the relation of each parameter on its loudness.

3.1.1 Example Data Sonification Experiments

An example of auditory data representation is the work done by Mansur, Blattner and Joy, in which points on an x-y graph were translated into sound with time on the x-axis and pitch on the y-axis [Man84] [MBJ85]. This simple data sonification project translated the values of a continuous function into a stream of audio in

which the pitch varied according to the value of the function. Time was mapped to the x-axis, so the listener heard the values of the function changing as the x value increased. In this example, only a one-dimensional function was translated into sound.

Yeung believed that as many as twenty dimensions could be used for the encoding of data into sonic form [Yeu80]. In his experiments, however, he used only seven-dimensional data sets that represented the amounts of seven metals in a number of obsidian samples taken from the San Francisco, California area. The task was to classify each sample into one of four categories, dependent upon the metals contained in it. The amount of each metal was mapped to a different sound parameter. Since this set of data was not time-varying, a tone was generated from each set of data values and then was repeated in sequence some n number of times, followed by a rest period. The parameters used in the data sonification included two simultaneous tones with pitches, loudness, damping, spatial direction, duration/repetition for each tone (governed by n), and the length of rest period between presentations of the tone sequence. Yeung found that subjects of his experiment were able to classify samples correctly over 90% of the time after one training period, and over 98% of the time after two training periods.

Scaletti and Craig created a set of tools to aid in the production of data sonification [SC91]. Among the data sonifications they created with their tools was one that translated into audio the impact of forest fires on Yellowstone National Park from 1690 to 1990. In this sonification, an auditory *histogram* was used. Each age category of a forest was represented by a particular tone at a unique frequency. The amplitude of that tone was controlled by how much forest area was present in the age range. Fires were represented with noise, and the size of the area on fire controlled the amplitude of the noise. The sonification was presented concurrently with a visualization, but certain data elements were only represented in sound. The listener heard a number of simultaneous tones, each varying in amplitude as the respective age category grew and shrank. It was noted that the overall age growth of the forest was very easy to hear during the fire suppression period in the park between 1872 and 1972.

3.1.2 Sound Attributes

There is no agreed upon set of sound attributes to use as independent parameters of data sonifications. However, most multi-dimensional data sonification projects use variance in the same physical sound properties as parameters. The characteristics are frequency (the major contributor to perceived pitch), amplitude (directly related to perceived loudness), and rhythm (or duration) of tones in a tone sequence.

Researchers have tried using many other parameters in addition to these common ones. Smith, Grinstein, and Pickett included attack rate, decay rate, and depth of frequency modulation to make their data audible [SGP91]. Lunney and Morrison used attack, decay and wave form shape to present analytical chemistry data represented by sound to visually impaired students [LM90]. Rabenhorst, Farrell, Jameson, Linton, and Mandelman used de-tuning (the sounding of two notes that are very close together in frequency), and stereo balance (relative volume of the stereo channels) in their experiments [RFJ+90]. Wenzel separated sound sources into the three spatial dimensions by modeling the pinna of the ear [Wen92]. Spatial effects which give the illusion of three dimensions were also created by Ludwig, Pincever, and Cohen [LPC90].

3.1.3 Perceptual Issues in Data Sonification

In her dissertation, Bly noted the difficulties in using volume as a reliable parameter due to its dependence upon the physical properties of both frequency and intensity (the Fletcher/Munson effect) [Bly82] [FM33]. Similar difficulties were reported by Lunney and Morrison, among others [LM90]. This is an instance of an incongruity between a perceptual sound parameter (volume) and physical sound parameters (intensity and frequency). Similar problems also arise with physical and perceived spatial location, and with frequency/amplitude and perceived pitch. In many cases, the perception of a physical attribute changes non-linearly and even unpredictably dependent upon the how the physical attributes change. For example, a common auditory perceptual problem is spatial localization of an object directly in front or directly behind the listener, both positions being equally distant from the head. These two positions are often indistinguishable until either the listener can move his or her head, or until the sound source moves off-center in space. For data sonifications that use spatial location as a sonic attribute, the *perceived* spatial location is an accurate attribute, unlike the physical spatial location which can lead to misinterpretations of the represented parameter.

There has been very little work conducted in controlling the perceptual attributes of data sonifications. This shortcoming is noted in the literature and some solutions have been proposed. Yeung noted that there would be a loss of perceptual independence between sonic parameters as the parameters increased towards as many as 20 [Yeu80]. Kramer argued that interactions between different sound parameters in a data sonification would be confusing to the listener [Kra94b]. He went on to state that using a unique auditory stream for each data dimension could be problematic if the streams became perceptually fused or if the listener's attention was shifted from one stream to another, possibly less important, stream.

Smith noted that one major problem with data sonification has been the arbitrary mapping of data to sound parameters, without regard for the perceptual characteristics of the sound parameters [Smi90].

A computer model has been created by Barrass that begins to address at least a few perceptual issues important to data sonification [Bar94]. The model uses the parameters of pitch, brightness, and timbre attributes, and uses non-linear scaling of each axis (from the physical characteristics of each parameter) in an attempt to make a linear perceptual axis. The problem explored was to control the stream segregation of a sequence of tones by varying the pitch and brightness of a subset of the tones. At a certain threshold, the tones with the changed attributes form a separate stream from the other tones. Through this method, the perceptual limits of identifying a single stream are found, and the axis for the given parameter can be scaled accordingly to the usable range. The model is currently limited to creating data sonifications that use pitch and brightness sonic attributes. The interaction between the parameters has not yet been accounted for in the creation of auditory data representations. Apparently this leads to changes in the perceived volume depending upon the pitch and brightness of a given note. Other perceptual attributes, such as rhythm formation and grouping of simultaneous sounds according to harmonic content and common fate, have not been considered.

There is much research yet to be done in data sonification, especially with regard to perceptual issues. This is evident by the lack of user evaluation in comparing sonifications to visualizations of the same data.

3.2 The Structure of Auditory Messages

There is a distinct difference between the uses of sound for auditory data "display"[1] and the uses of sound for the presentation of discrete auditory messages. The efforts in auditory data display have generally focused on the data-driven mapping of data points (usually a data point in an n-dimensional space) to audio output. The use of sound for auditory messages has focused on a more abstract mapping of messages to sound. Issues such as syntax, semantics, and the structuring of an abstract sonic "language" have been considered important.

The structure of auditory messages has been studied by Gaver [Gav86] [Gav93], Blattner [BSG89] [BGK92], and Sumikawa [Sum85]. Gaver used "real-world" sounds, called *auditory icons*, to convey information about user interface objects.

[1]The term "display" is used metaphorically in this context. Auditory display refers to the translation of information into sound.

Each object produced noise, such as hitting and scraping, dependent upon the user's interaction with the object. Gaver's auditory icons are imitations of familiar sounds that are presumably familiar to the user. Conversely, Blattner, Sumikawa, and Greenberg used *abstract earcons*, which are sequences of tones used as a basis for building messages. Earcons are designed to be abstract — there is no inherent intuitive mapping between the sound of a particular tone sequence and its defined meaning. This property is similar to words in written English, which do not generally bear intuitive resemblance to what they signify.[2] Conversely, auditory icons are familiar sounds that should have an intuitive meaning. Dependent upon the context in which an auditory icon is sounded, it is intended that the listener will understand its meaning.

3.2.1 A Description of Earcons

Blattner, Sumikawa, and Greenberg defined two different types of earcons: *abstract earcons* and *representational earcons* [BSG89]. Representational earcons are real-world sounds that represent what they sound like. An association is made between the sound and the object from which the sound originated. For example, the sounds of a bird would represent a bird, and the sound of a clock tick would represent a ticking clock.

Abstract earcons are note sequences created using musical rules. The meaning of an abstract earcon is arbitrary — earcons form a musical "language" much as words of a language do. In the terminology of earcons, a short sequence of tones is called a *motive* [BSG89]. This word comes from the musical definition of a short passage that is re-used and developed throughout a musical composition. In the construction of earcons a motive is used as a building block for larger groupings. Each single motive has a number of musical parameters. They are as follows:

- **Rhythm:** The note durations of an earcon form a particular rhythm. This rhythm can be changed by changing the tempo or by changing particular note durations.
- **Pitch:** Each note in an earcon has a particular pitch. The pitch attribute can be changed through transposition of the motive or through the pitch change of particular notes. Generally, the pitches of each note of an earcon should fall within an octave range.

[2]Many languages, including English, do have some words that, when spoken, bear resemblance to the word's meaning. This is the basis for the word construction method of onomatopoeia. Examples in the English language include the words "moo" and "buzz."

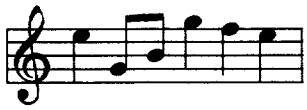


Figure 3.2: An example of a simple earcon.

- **Timbre:** An earcon is sounded using a particular timbre. This attribute is altered by either changing the timbre entirely (for example, from a saxophone to a violin) or by changing the existing timbre (such as increasing the brightness).
- **Dynamics:** The dynamics parameter of an earcon refers to its volume. This parameter can be changed for all the notes of an earcon, or altered dynamically to produce effects such as crescendo and decrescendo.
- **Register:** A given earcon can be played in any number of different registers. The change of register can be thought of as a transposition of the entire earcon one or more octaves higher or lower in pitch.

These parameters can be easily manipulated to create somewhat different sounding, but sonically related, earcons built upon the same basic motive.

Simple earcons, created from one motive, are the building blocks for the compound earcon forms. Simple earcons contain a few notes (usually less than eight) and are relatively short in duration. The notes all share a common timbre and register. Each note contributes to the rhythm and could have a pitch or could be "pitchless" (represented by a "click"). The earcon has a particular dynamics parameter. A simple earcon is depicted in Figure 3.2.

Blattner, Sumikawa, and Greenberg outlined three ways of creating a compound earcon from the simple earcon form [BSG89]. The techniques are *combination*, *transformation*, and *inheritance*.

3.2.1.1 Combination of Simple Earcons

Combination of earcons is a technique in which a new compound earcon is formed by a sequence of two or more simple earcons. As an example, let A be a simple earcon representing a computer file. Let B be a simple earcon representing "deleted." A new earcon C can be formed by concatenating A and B together. The meaning of C is then "file deleted." In Figure 3.3 the combined earcon "computer malfunction" is formed from two simple earcons.

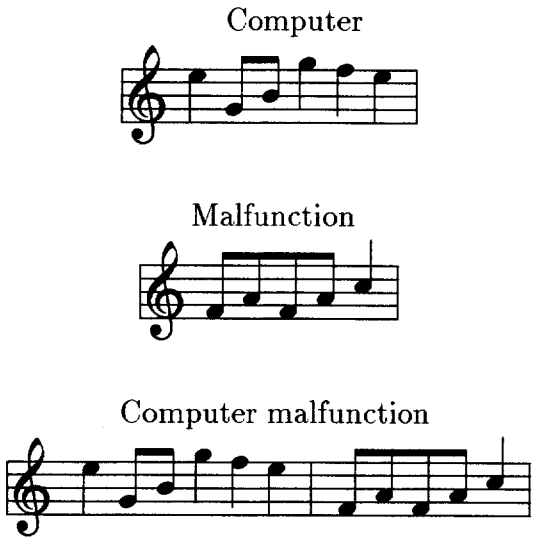


Figure 3.3: An example of a combined earcon.

3.2.1.2 Transformation of Simple Earcons

The second compound earcon form is created through a transformation. A transformation of a given earcon is a change in one or more of that earcon's parameters. For example, if D is the simple earcon representing "large," then D' can be a transformation of D that represents "very large." D' sounds similar to D, except that one or more parameters of D is changed. D' could be sounded in a different register or with a different rhythm. The degree of largeness could, for example, be mapped to the tempo of an earcon, so that largeness could be represented as a continuum based upon the rhythm parameter of the earcon.

Another transformation style is taken from the work of Schoenberg [Sch51]. He proposed that a transformed set of tones might be recognized as similar if the descending intervals became ascending and vice-versa (inversion), or if the notes were presented in the reverse order (retrogression), or both (retrograde-inversion). The ability to associate earcons transformed in this way with the original is debatable, as discussed by Deutsch [Deu82, pp. 283–284]. An example of this type of transformation is illustrated in Figure 3.4.

3.2.1.3 Inheritance of Simple Earcons

The third compound form is inheritance. A hierarchy of earcons can be formed in many ways, but the particular method described by Blattner, Sumikawa, and

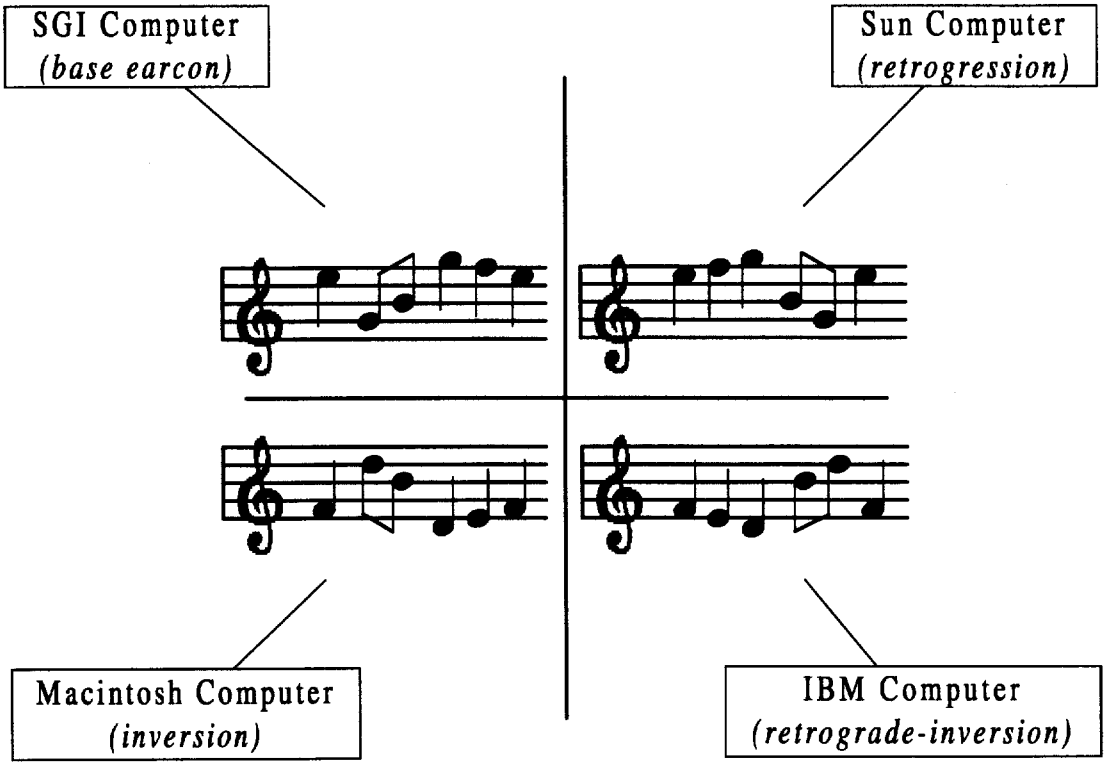


Figure 3.4: An example of a transformed set of earcons, motivated by the work of Schoenberg. Each earcon has a different, but related, meaning. In this case, each earcon represents a different type of computer.

Greenberg is as follows: A simple family motive is an unpitched earcon with only a rhythm attribute. This is the root of the hierarchical structure. The next level of earcons in the hierarchy is formed by adding the attribute of pitch to the base earcon. These earcons use only a simple sine wave to produce pitch. The presentation of these earcons is made by first playing the root (unpitched) earcon followed by the pitched earcon. The actual meanings of the second level in the hierarchy should specialize upon the meaning of the root earcon. The third level of the hierarchy introduces a specific timbre to the earcon at the previous level. The presentation of earcons at this level consist of three distinct components: the unpitched earcon followed by the pitched earcon followed by the pitched earcon with the timbre. The meaning of an earcon at the third level should specialize further upon its parent earcon. The presentation of any hierarchical earcon can be made without the parent earcons. This shorter form is for the "expert" who is familiar enough with the particular earcons to clearly distinguish each parameter. Other earcon hierarchies can be constructed using different parameter orderings.

In Figure 3.5, a hierarchy of earcons is shown. The root earcon is unpitched and means "printer." The earcon shown at the next level represents "printer error" and has the added parameter of a particular set of note pitches. This earcon is further specialized into three types of printer errors: out of paper, out of toner, and paper jam. Each of these earcons is identical to the parent earcon but with the added attribute of a unique pitch.

Throughout the remainder of this text, if an earcon reference does not specify either abstract or representational, it is assumed to be abstract.

3.2.2 A Description of Auditory Icons

Auditory icons, developed by Gaver, are everyday sounds meant to convey information by analogy in the user interface [Gav86]. Auditory icons are familiar real-world sounds, such as bumping, scraping, and breaking. Their meaning in the user interface is supposed to be intuitive; no training period should be necessary in working with these sounds. Auditory icons are different from representational earcons because auditory icons have a metaphorical meaning associated with them — they do not represent the actual objects that they sound like. The sounds of auditory icons usually describe the interaction of "objects" in the user interface, and the sounds created through their manipulation. A scraping sound, for example, is heard when a window is "dragged" to a new location. A selection operation gives feedback that the object had been "struck." The dropping of a file into the Macintosh trash can produces a "crash."

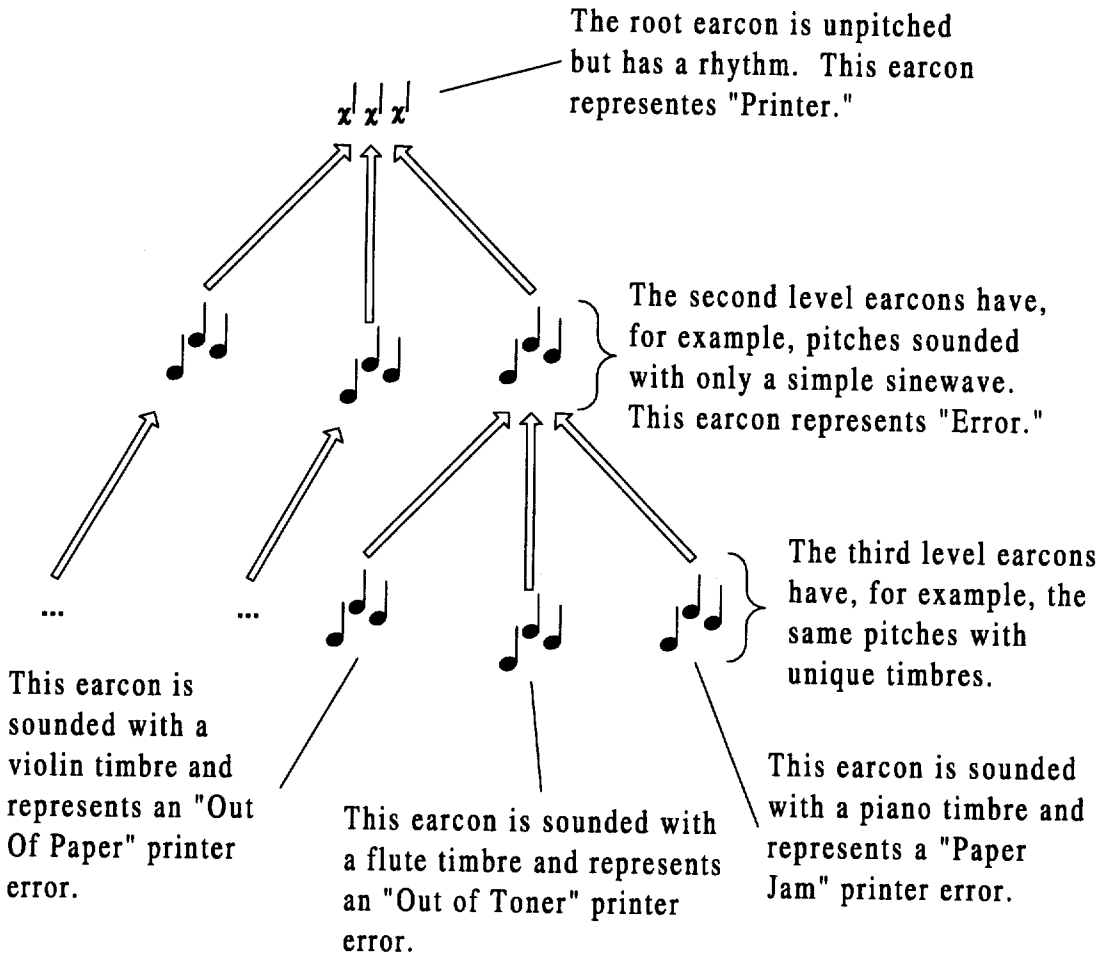


Figure 3.5: A set of hierarchically related earcons.

An auditory icon can be parameterized to some extent. For example, the frequency of the playback can be altered, as can some of the "material" interactions such as scraping and hitting. In the SonicFinder (an audio enhancement to the Macintosh user interface) Gaver used material metaphors such as wood and metal to differentiate objects like files and folders [Gav89]. The hitting sound of a selection operation sounded like a physical interaction with the selected object's material. The scraping sound of a dragged window varied in frequency with the size of the window, so small windows produced a higher pitched scrape than large windows.

3.2.3 A Comparison of Earcons and Auditory Icons

The most common objection to the use of earcons in the user interface is that the user must learn the meaning of each abstract sound. There is no intuitive mapping between the sound and its semantic meaning. Auditory icons presumably require no training since the sounds are intuitively obvious. This objection to earcons actually turns out to be a large benefit in many cases. Abstract earcons do not have to correspond to the objects they represent, so objects that either make no sound or make an unpleasant sound still can be represented by earcons without further explanation. There is no concern that a real-world sound might be misinterpreted. Auditory icons often can be recognized quickly. However, the recognition must be as intended by the creator of the sounds. This is not easy to guarantee, since the context of the user interface is not known as sounds are played. Mynatt observed that choosing the right sounds to use in the interface is an art with many pitfalls, and success is dependent on the skilled and gifted designer [Myn94b]. She also noted that sounds of different objects sometimes sound the same. For instance, the sounds of typewriters, computers, and cash registers are confused, so perhaps they should not be used together in the user interface.

Another potential problem in the use of auditory icons is that sometimes the user makes unexpected associations between sounds and objects. This problem does not arise with earcons because all of the sounds are by definition abstract. This problem is illustrated in the *ShareMon* system implemented by Jonathan Cohen [Coh94]. ShareMon used auditory icons to notify the user of background computer activity such as entry into a computer network. The auditory icons used included footsteps at various speeds, a knock sound, a door slam, and an "ahem" sound as if uttered by a person trying to get the attention of another. One of the interesting usability problems in Cohen's implementation, as determined by his tests on users, was the realism of his sampled sounds. It was not always clear that the "ahem" auditory icon (denoting that a network connection to the user's hard disk is still active) was produced by the computer and not by a person out of the direct

sight of the user. Similar difficulties were noted with the footsteps. Such problems could possibly be remedied with a careful redesign of slightly less realistic sounds. Even though the auditory icons were relatively easy to identify, the user still had to identify the relation of the sounds to the messages they were delivering.

Another problem was determined from user studies of ShareMon. The sounds used were not intuitive to the users even when they knew that file sharing was being monitored through the use of auditory icons. Almost no users could figure out that footsteps indicated the percentage of CPU utilization being used by network file sharing, or that the "ahem" sound meant that a user was connected to the file sharing system but was inactive. This type of problem arises because many objects and concepts do not have any intuitive representation in sound. The more robust and abstract language formed by earcons can represent these ideas, but the user must learn a possibly large set of earcon definitions.

Another aspect to the design of auditory user interfaces is the determination of the types of sounds considered acceptable to the user. Several experiments have shown that earcons are preferred over other types of auditory messages and can be used successfully. Jones and Furner compared earcons, auditory icons, and synthesized speech. Their results showed that subjects preferred the sounds of earcons but were better able to associate auditory icons to commands [JF89]. Brewster, Wright, and Edwards found earcons to be an effective form of auditory communication in the desktop of a computer interface [BWE93]. They recommended six basic changes to the earcon form to make them more easily recognizable by users. These changes were as follows:

1. Use synthesized musical timbres (as opposed to simple, machine-generated waveforms such as sine and square waves).
2. Pitch changes are most effective when used with rhythm changes.
3. Changes in register should be several octaves.
4. Rhythm changes must be as different as possible.
5. Intensity levels must be kept close.
6. Successive earcons should have a temporal gap between them.

The tests run by Brewster, Wright, and Edwards had a very short training period associated with them. The earcons were heard only once before the test. In spite of this, the subjects could use them effectively.

3.3 Example Auditory Interface Systems

A number of auditory interfaces have been created using sounds including earcons and auditory icons. The interfaces described in this section all have a similar quality — the presentation of numerous dynamically changing, temporally overlapping, auditory messages. It is this style of auditory user interface that can benefit the most from the research described in this dissertation.

All auditory interfaces known to the author, (with the exception of those designed by the author, to be described in Chapters 4 and 5) rely on playing auditory messages at certain times without regard to what other auditory messages may be playing concurrently. While the designers of each system attempted to make the auditory interfaces usable, it would be impossible, in general, to account for all possible combinations of sound overlaps (and hence the perceptual problems that might arise as a consequence).

In a discussion concerning auditory user interfaces with temporally overlapping sounds, Gaver, Smith, and O'Shea have written,

> But the design of auditory cues for such systems is difficult because it is desirable to present a number of cues simultaneously, and some will be continuous (e.g., to indicate processing rates) while others will be discrete. Care is needed to design an ecology of sounds to work together so that each sound may be heard and understood. We have little experience with designing sounds for such environments and little user testing has been performed on such systems to date. [GSO91, p. 85].

At best, unregulated dynamically created auditory messages will perceptually interface with each other occasionally. At worst, the entire sound presentation will be an auditory "smearing" of each individual source, and of no informational value to the user. In a dynamic environment, there is no way to be sure of the clarity of the auditory environment unless some form of the global "regulation" is applied on all outputted sounds.

The research presented in Chapters 4 and 5 formalizes the problem of auditory stream segregation in auditory user interfaces, and describes the methods used to regulate temporally overlapping auditory output so that each sound may be heard and understood.

At this point, a few implementations that employ overlapping auditory messages are described. Their descriptions serve to clarify the types of auditory interface designs that this research can benefit, and the perceptual problems that can arise.

3.3.1 The Varèse System for Satellite Monitoring

Albers created an interface to monitor six subsystems of a satellite [Alb94]. He used a different auditory icon to represent each subsystem. There were three states each subsystem could exist in: normal, warning, and critical. Three different, but related auditory icon variations were used to indicate the state. For example, the power subsystem used the auditory icon of an automobile engine. The normal state was indicated a normal running engine sound; the warning state was indicated by a sputtering engine sound; and the critical state was indicated by a dying engine sound. The auditory icons were presented with a certain repetition speed that changed as a satellite subsystem approached a state change. For example, the repetition speed of the normal engine sound would increase as the power subsystem got closer to the warning state.

The auditory output of the Varèse System consisted of the six repeated sounds. Presumably, the sounds used were chosen so that they encouraged stream segregation when presented concurrently. There was no indication that any more sophisticated methods were used to ensure that the user could clearly hear each of the six auditory streams formed by the satellite subsystem data.

3.3.2 Sonifying the Body Electric

Fitch and Kramer created an interesting implementation that has attempted to incorporate the positive elements of auditory data representation, earcons, and auditory icons [FK94]. It is a monitoring system similar to what might be used in a hospital to track a patient's heart rate, blood pressure, body temperature, etc. Some sounds used were abstract; others had the qualities of representational earcons and auditory icons, such as the "thudding" of the heart rate indicator. This helped minimize the training time of the system. The sonic parameters of the heart rate were modified to indicate other conditions such as carbon dioxide level. It was found that the detection of each variable was very easy when visual cues were introduced along with the auditory cues. However, auditory cues alone made detection of medical problems much quicker than just the visual cues alone.

This was a very easy to understand simulation because the sonic parameters used seemed to remain perceptually independent of one another and easy to hear out. The success of this system must have been due, at least in part, to the careful design of the audio output. The design made good use of frequency proximity within auditory streams and minimized pitch crossings between streams. The sound design task may have been feasible because the sounds produced by the interface

were quite constrained. Most of the possible auditory output from this program could be anticipated, and therefore analyzed for perceptual clarity. This is an advantage when designing sounds for a highly constrained environment.

Listeners were not confused by the "auditory icon" qualities of the sound, because, for instance, the "thud" representing a heart beat was a natural mapping, and therefore, an appropriate use of an auditory icon. High pitched earcon-like tones were used for the more abstract measurements that had no easily associated sound. This is a good use of an abstract sound since no intuitive mapping was possible. The use of a familiar sounding auditory icon would only have served to create a false or misleading association.

3.3.3 The ARKola Bottling Plant Simulation

Another interesting application was the ARKola simulation by Gaver, Smith, and O'Shea [GSO91]. Auditory icons were used to represent the various steps involved in the process of bottling. There was a visual representation of the plant as well, but the user could see only a small portion of the entire plant at any given time. However, the sounds produced by the machines in the plant could be heard regardless of where the view was set.

The auditory icons used included a whooshing sound for the heater and a clanking sound for the bottle dispenser. These auditory icons were relatively short and repeated at a rate proportional to the operating speed of the particular machine. The absence of a sound indicated that a machine was not running and must be attended to. Other auditory icons were used to indicate problems. A splash sound informed users that liquid was spilling, and a crashing sound indicated that bottles were breaking. These sounds would direct the user's attention to the problems causing the undesirable actions, regardless of where the user had set the view.

A difficulty noted by the authors was that alarms were sometimes not sufficiently discernible and went unattended. Also, it was difficult and time consuming to design the sounds in the user interface such that they would not perceptually mask each other when played concurrently. Each sound had to be crafted by hand so that it would be audible in the presence of the other sounds used in the bottling plant simulation. This method does not scale well to systems that use numerous different sounds.

3.4 Conclusion

All of the example auditory display and auditory user interface systems described do nothing to regulate the perceptual elements of the auditory presentation. In the case of the manipulation of a single auditory stream, the difficulty has been in the identification of sound dimensions that do not perceptually conflict. In the case of multiple overlapping auditory streams, the sounds used must be very carefully designed so as not to interfere with each other upon concurrent playback. This puts a severe constraint on the design of the auditory user interface. If any two sounds might temporally overlap, each sound used must be designed not to conflict with any other sound. This becomes quickly unmanageable as the number of sounds used increases. Considerations for overlaps of more than two sounds (without a dynamic method for sound scheduling that considers perceptual issues) become even more complex that only severely constrained special purpose systems can be designed with any success.

At least for auditory interfaces with multiple data streams, it would be useful to have the flexibility to use any desired sound, and to let the computer interface control the exact presentation of the sounds used. If the computer could somehow regulate the presentation of sounds with respect to the the perceptual factors involved, there would be a number of benefits to both the designer and to the user.

- The designer could concentrate efforts on making good, informative sounds instead of on trial-and-error methods to make a set of sounds perceptually distinct from each other under all overlapped playback conditions.
- When a new sound needs to be added to an existing auditory user interface, the designer does not have to recheck and possibly redesign the other sounds of the interface.
- Sounds from multiple applications can be used together in a global auditory environment, if all sounds are regulated through one central control mechanism. The user can run a number of sound producing applications without the worry of destroying the perception of individual sounds from each application.
- Sounds that are dynamically created at run-time can be used. The auditory interface is no longer constrained to only "canned" sounds that are modified slightly according to a well-defined set of parameters.

Through Chapters 4 and 5, a method for the presentation of dynamic, temporally overlapping auditory messages is developed. Two separate auditory interface con-

trol mechanisms have been implemented. They are described in detail, as are the applications written to demonstrate them.

Chapter 4

Towards Regulation of Sound Perceptibility in User Interfaces

In an attempt to generalize both the problems and solutions of maintaining auditory clarity in a dynamic interface, two project implementations were extremely instructive. In the first project, named the *Auditory Map System* [BPG94], the user was presented a visual representation of a map. Upon a user selection, numerous sounds were played describing the selected map region. Each sound represented a feature in a user-selected area of the map. It was evident that problems existed with the perception of overlapping sounds in the Auditory Map System. This was the motivation to design and implement the second system described, named the *Centralized Audio Presentation System* [PB94] [PB96]. This system was the first attempt to regulate, in real-time, the overlapping auditory output of numerous concurrently running computer applications. Finally, a brief motivation and description of a more sophisticated auditory interface regulation mechanism, the *Computational Auditory Scene Synthesizer*, is given. This system is thoroughly described and analyzed in Chapter 5.

4.1 The Auditory Map System

The Auditory Map project was created as an example of decluttering a two-dimensional visual display through the representation of map attributes in sound [BPG94]. Maps typically have many attributes that are of interest, but cannot be concurrently displayed in graphics. Beyond some point, visual perception is overwhelmed and no more visual information can be effectively displayed. For example, placing elevation contour lines plus other physical land markings visually on

a street map would make the job of city street navigation much more difficult. However, if that same graphically displayed street map had elevations and physical landmarks represented in a sonic form, one could navigate as usual using the visual representations of the street and still gain the elevation and landmark information concurrently through the auditory representations of the other features.

The Auditory Map was an initial attempt at creating a useful environment in which to study the perceptual issues of temporally overlapping auditory messages. This system would help evaluate the ability of the perceptual system to segregate overlapping sound sources with very little auditory stream analysis and control.

4.1.1 Auditory Map Interactions

A graphical representation of a facility containing buildings, roads, and parking lots, was presented to the user. Other map objects were represented only in sound. These included access restriction levels, computer installations, underground power lines, and underground structures. The user could select an area of the map with the mouse, and then hear a representation of what was in the selected region. There were two implemented methods of selection:

- **Point Selection:** In this method, the mouse was used to select a spot on the map. The auditory message of the map object containing that point location was played.

- **Area Selection:** The user could move and resize a circle on the screen. All of the corresponding auditory messages of the objects within the selection circle would be played concurrently.

The example Auditory Map shown in Figure 4.1 shows the area selection tool.

The user could choose between two listening modes: concurrent or sequential. Concurrent sounds are sounds that play together, either simultaneously or in some way temporally overlapped. In the Auditory Map implementation, if more than four sounds were to be presented in concurrent mode, then four sounds were initially presented to the user — as each sound ended another began, until all sounds were played.

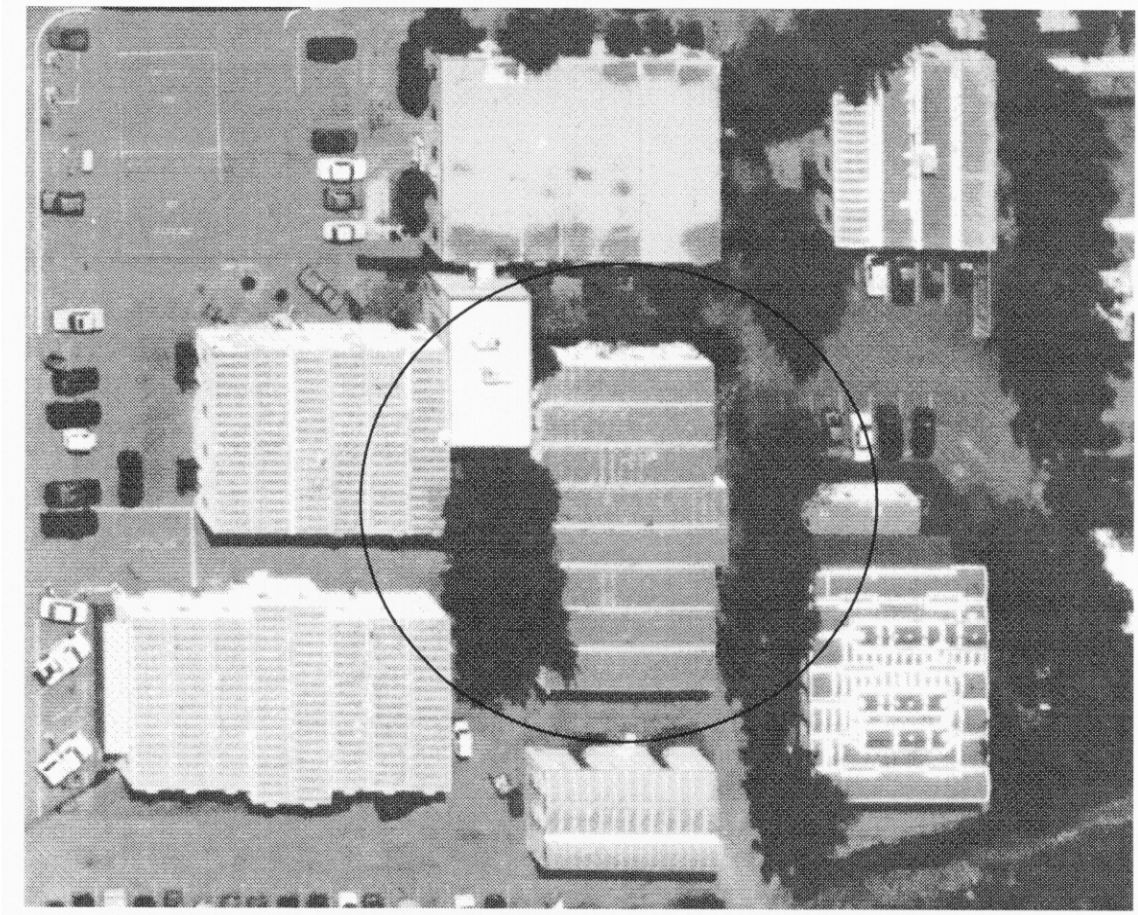


Figure 4.1: An example Auditory Map showing the area selection tool.

4.1.2 Map Object Representations

The map objects that had auditory representations included administrative buildings, access restriction levels, computer installations, underground power lines, and underground structures. Each had three auditory representations: a long earcon form, a short earcon form, and a voice form. The user selected which was to be used at any given time.

The long earcon representations were organized as follows: Each type of object had a base earcon that possessed a unique timbre. Three types of objects had secondary, more specific, information to be conveyed. In these cases, the base earcon was compounded in some way to relate the further information. In the case of access restrictions, a tom-tom drum timbre conveyed that a restriction was present, and the speed and pitch of the drum hits differentiated between the access restriction levels of confidential, secret, and top secret. Higher access restrictions were indicated by faster, higher pitched tom-tom drum hits. Figure 4.2 shows the access restriction earcons used.

The presence of administrative buildings was indicated with a two note earcon played in a saxophone timbre. To further specify the department of the administrative building, the base earcon was combined with a physics, engineering, or computations earcon, also sounded in a saxophone timbre. The earcons used are shown in Figure 4.3.

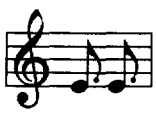
Earcons representing computers were sounded with a flute timbre and consisted of four notes. Four different computer types were represented. Each of the four types was related using the musical transformations of *retrogression* and *inversion* as defined by Arnold Schoenberg [Sch51]. This transformation is a conceptual "reflection" of notes over the x-axis (*retrogression*), the y-axis (*inversion*), or both axes (*retrograde-inversion*), such that each translation occupied a different "quadrant" of the x-y graph. An example of this transformation is shown in Figure 3.4.

The short earcon form of each object was designed to present rapid summary information over a large area selection. For each object in a selection, a single note was sounded in the timbre of that object. The large combination of notes in different timbres indicated overview information such as where the concentrations of computers were, or where the most (or least) access restrictions were present. The short earcon form did not encode any of the specific secondary information present in the full earcon form, but it did present a fast summary of the selected map region, and was generally more appropriate for auditory display of large map selections.

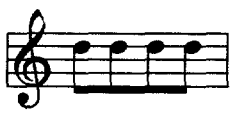
Voice was the final auditory output selection. It was most useful for very specific information that is difficult to represent in non-speech forms. Its application in

Tom-Tom 

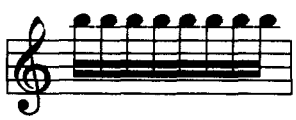
The base earcon means "Access Restriction"

Tom-Tom 

This transformed earcon means "Confidential."

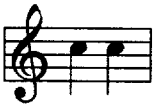
Tom-Tom 

This transformed earcon means "Secret."

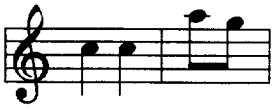
Tom-Tom 

This transformed earcon means "Top Secret."

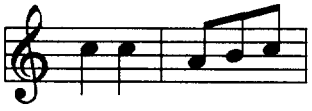
Figure 4.2: The access restriction earcons used in the Auditory Map.

Saxophone 

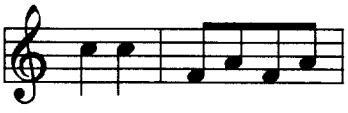
The base earcon means "Administrative Building"

Saxophone 

The "Physics" earcon is combined with the base earcon to indicate an administrative building for the physics group.

Saxophone 

The "Engineering" earcon is combined with the base earcon to indicate an administrative building for the engineering group.

Saxophone 

The "Computations" earcon is combined with the base earcon to indicate an administrative building for the computations group.

Figure 4.3: The administrative building earcons used in the Auditory Map.

the auditory maps was to give details, such as building number information, that were not encoded into the earcon forms.

4.1.3 Analysis

The dynamic, temporally overlapping output created by the Auditory Map in concurrent mode was not regulated in any way. This was the project's largest shortcoming. All of the selected audio objects played concurrently in four channels, and as one sound ended, another began. New sounds were played without examination of the global auditory state. Consequently, careful earcon design was critical if each earcon was to be clearly segregated from others playing concurrently. This is the same design problem that was evident in the systems described in Chapter 3. For example, each earcon was designed to play in a relatively small pitch range to encourage stream segregation based upon frequency proximity. As will be described in this section, certain perceptual conflicts between overlapping earcons cannot be avoided through the design of earcons, and can only be addressed through a centralized auditory output control mechanism that monitors and controls the global auditory system state. Such a system is developed in Section 4.2.

Perhaps the most problematic aspect of the Auditory Map sound presentation was that numerous earcons of the same family and with the same timbre would be played simultaneously. This situation is particularly troublesome when the notes of the overlapped earcons fall into the same pitch ranges. This occurred whenever an earcon was overlapped with another earcon of the same family. The stream segregation cue of frequency proximity would work to confound, instead of assist, the listener in the perception of two or more concurrent auditory messages. Further difficulties could arise from all of the common fate cues, such as common onset, common offset, and common pitch changes.

An interesting benefit did emerge from the perceptual difficulties of segregating each individual source. When using the short earcon form for large area selections in concurrent playback mode, the resulting auditory output was a large combination of sounds in which each base auditory component was a short, single note played in a particular timbre. The resulting sound was reminiscent of the noise that might be produced by an orchestra during a warming-up period when each participant is playing notes unrelated to everyone else's. The sounds formed a dynamically changing chaos of timbres that varied according to the content of the area selection. It was easy to recognize map areas that had mostly administrative buildings because the saxophones dominated the mix of timbres. An area dense with computer equipment, for example, was sonically dense with flutes.

Even within the eclectic auditory free-for-all of large area selections with short earcon representations, the tom-tom earcons segregated well from the other sounds, even if only one such sound was presented. The particular access level was actually encoded into the short earcon sound through the use of pitch. A short earcon tom-tom sound with a higher pitch represented a higher access level. The result of the clear segregation for tom-tom drum sounds was that map areas could be quickly surveyed for general security level, while at the same time, summaries of building types and computer equipment could be made. Certain relationships in the map then became evident — for example, the map areas dense with computer equipment usually required higher security clearance.

Through use of the Auditory Map System, it became evident that timbre discrimination was a very important element in the perceptual segregation of overlapping earcons. The timbre of each earcon family had to be carefully picked so that it would stand out from the others during concurrent playback. The earliest versions of the Auditory Map used simple computer generated waveforms such as sine waves, square waves, and sawtooth waves, which had no envelopes, harmonics, or modulations added to them. These sounds were very difficult to differentiate, especially when sounded concurrently. The later versions of the Auditory Map used sampled instrument timbres, which were varied in sample playback rate to achieve different pitches. This technique does have the drawback of limited potential timbre manipulation. However, it does allow for the use of more complex timbres that sound familiar to people. Recognition was much improved using this technique over the original waveforms. This result is supported by the *habit*, or *familiarity* principle of auditory stream segregation. This principle states that a sound which is familiar will tend to form a stream in the same manner in which it has done in the past. In Western culture, the sampled instrument waveforms of flutes, saxophones, and tom-tom drums are quite common and readily identifiable. Most people are not accustomed to listening to simple synthetic waveforms such as square and sawtooth waves that contain no envelope or amplitude/frequency modulation.

The Auditory Map project had certain weaknesses that caused perceptual difficulties in some cases. Three of the most important problems are listed below.

- Rhythm issues were not addressed. Upon a particular map selection, the earcons of selected map objects were asynchronously played with a maximum of four earcons sounded at a time. The design of the earcons could not alleviate problems arising from rhythm conflicts between perceptually unrelated auditory messages because there was no way to predict how the sounds might overlap as they were played. This meant that under certain conditions, the note onsets of concurrently presented earcons temporally coincided. This introduced the perceptual grouping effect of common onset when it was unde-

sirable. The perceptual system could become "tricked" into grouping sounds which should remain perceptually unrelated. A similar effect (common offset) happens when multiple sources end in unison. These phenomena undoubtedly tend to counter the desired stream segregation. Also, rhythm patterns between multiple earcons could form new structures under certain circumstances. For example, if the same earcon was sounded twice with one instance starting momentarily after the other, the second might be perceived as an echo and so both instances of the earcon would be mistaken as belonging to the same stream. Or, the notes of two earcons with similar timbres might become interleaved and encourage the sequential integration of the notes from both earcons thus forming one new, but undesired, auditory message.

- Sound localization issues were not addressed. In fact, the entire auditory presentation was monaural. It is known that sound localization is one of the stronger stream segregation cues. Since all sound sources originated from the same loudspeaker without any spatial processing, the localization cue actually tended to encourage grouping of all auditory messages into the same perceptual unit. Other stream segregation cues had to be used to counter this effect.

- There was no method for dynamic timbre discrimination analysis. This meant that the timbres of earcons had to be carefully hand picked for the particular application to ensure that they could be individually perceived when up to four different timbres were playing concurrently. This problem has been shared by some of the other implementations described in Section 3.3.

  The inclusion of an additional family of earcons into the Auditory Map required a great deal of work. The new earcons had to be designed to minimize conflict with any other earcons that may be sounded concurrently. Of course, the potential for interference could not be eliminated since there was no dynamic control mechanism to regulate the auditory output in any way. A large part of task of adding a new family of earcons was the creation and incorporation of a new timbre into the system. This new timbre would have to remain perceptually distinct from all the other timbres currently in use since the new earcon may be playing concurrently with any other earcon in the system. The trial-and-error procedure for earcon creation was very time consuming and generally produced mediocre results.

Through informal user testing, it was found that people could sometimes perceive the different concurrently playing earcons, and also make accurate summary judgments about map areas using the short earcon form. With only a few minutes of training, users could generally remember the mappings of sounds to objects, but

it is unclear how that memory would persist over time. Users encountered particular difficulties with remembering the mapping between the related compound earcons with flute timbres and the computer types. However, it was clear that the earcons with the flute timbres represented some type of computer. The difficulty in remembering specific computer types is likely caused by the similarity of each earcon in the set, and the lack of a logical progression between sound and meaning. This progression is present in the tom-tom earcons which represent access level. Faster pounding indicates higher access level.

Some of the perceptual problems described in this section were solved by implementation of the Auditory Map System through a dynamic auditory output control mechanism named the *Centralized Audio Presentation System.* A thorough description of this system is the topic of the following section. The results of the second Auditory Map implementation are described in Section 4.2.4.

4.2 The Centralized Audio Presentation System

The perceptual difficulties encountered with the Auditory Map System, along with the desire to allow many different applications to use auditory messages concurrently, led to the design of the *Centralized Audio Presentation System* [PB94] [PB96]. This system consisted of an audio server that monitored and regulated the dynamic global auditory state of the computer. Decisions affecting auditory output could be made based in the context of the current global auditory state. The lack of this functionality is what led to much of the perception problems between overlapping auditory messages in the Auditory Map project.

4.2.1 Presentation System Overview

User interfaces that support concurrently running applications have little, if any, audio management. Typically, some number of audio channels exists as an operating system resource, and applications request exclusive use for some of those audio channels. The operating system either grants or denies the requests. Therefore, in environments where multiple programs output sound, each individual program has no overall context of the auditory system state with the possible exception of how many audio channels the operating system has allocated for other applications.

Research has been conducted in creating audio servers to aid in the resource management issue of audio presentation [Aro91] [RK92]. However, concurrent sound presentations can potentially lead to numerous problems in the perception of the

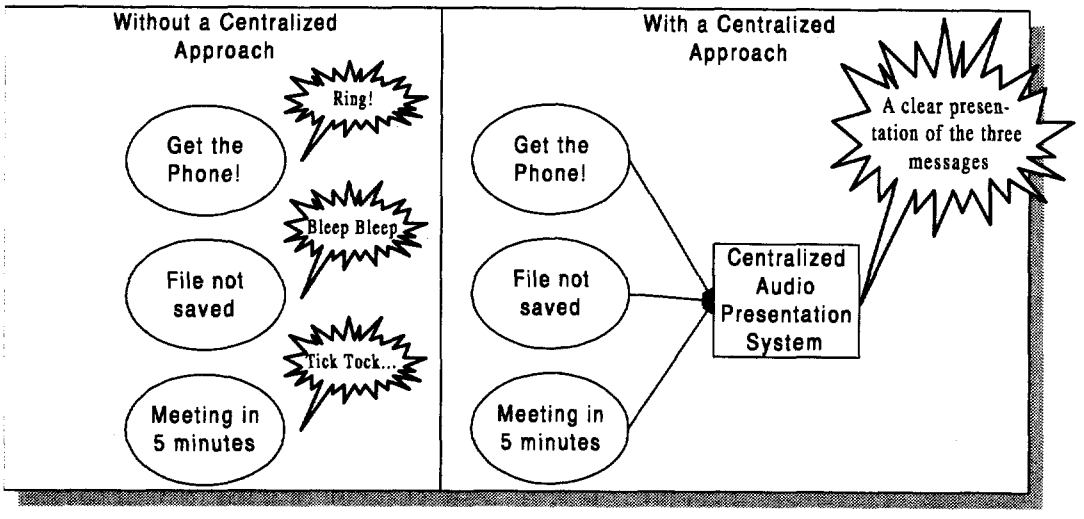


Figure 4.4: A comparison between audio environments without and with a server.

various overlapping sounds. This issue has not been addressed with audio servers. Programs typically play audio without regard for the overall auditory environment, and this can lead to perceptual unintelligibility of various auditory messages. These perceptual problems can be addressed through the continual examination of the global auditory state of the computer system. If an audio presentation can be altered intelligently, the clarity of each individual auditory message can be maintained in the presence of others. The Centralized Audio Presentation System is designed to do this. The structure of an audio environment with and without a centralized audio server is shown in Figure 4.4.

The Presentation System receives descriptive request messages that contain information about system activities and program states, as specified by the user or application programmer. The auditory output of the set of running programs and the overall auditory system state is controlled by the Presentation System. It chooses how the information is to be represented in sound, within the constraints of the request. A request message may encode numerous alternate auditory representations, any one of which may be chosen depending on the auditory context to which it will be introduced. Glinert and Blattner have discussed the concept of dynamic representations of information in different forms, or *multimodal objects* [GB93]. Figure 4.5 illustrates the multiple representations of given messages that might be received by the Presentation System.

There does not appear to be any other implemented system that alters the auditory output of applications within constraints defined by the programmer and the end user, with the purpose of regulating the perceptual characteristics of tempo-

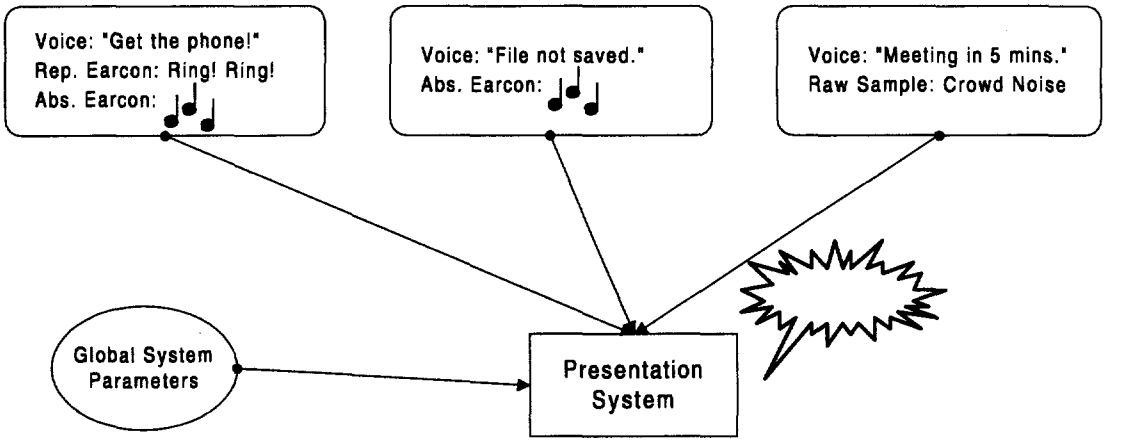


Figure 4.5: Multiple representations of messages sent to the Centralized Audio Presentation System. The messages are, "Answer the phone," "File not saved," and "Meeting in 5 minutes."

rally overlapping sounds. Constraint-based multimedia authoring systems have a different goal: they create a multimedia presentation from time-line based requirements [All83]. There is no analysis of possible auditory misperception, and there are no alternate representations of auditory messages that may be used. The *Selectors* system abstracts the representation away from the action of choosing an item from a list in a visual user interface [JNZM93]. Glinert's and Blattner's *multimodal objects* system further abstracts away the representations of all user interface objects, but the implementation is only in the beginning stages [GB93]. Finally, Flinn and Booth describe a system that would dynamically alter the auditory output of programs in accordance to certain psychoacoustic rules, but to date no implementation exists [FB95].

In order to give the Presentation System as much flexibility as possible, applications can send multiple auditory representations in a request to the server. Depending upon the information received and the current auditory state of the system, the most appropriate auditory representation for the data will be used (and possibly modified in some way) in its presentation. By using this method, each auditory source will most likely be perceptually distinct and the developer will be alleviated of the tedious task of designing sounds that do not conflict with the auditory output of the current and other running applications. This job is impossible in a multitasking, dynamic environment, since the designer of sounds cannot predict what sounds from other applications might be concurrently playing at any given time. With the Presentation System, the user is required to learn more than one representation for the same semantic, but receives the benefit of a perceptually clearer presentation of audio.

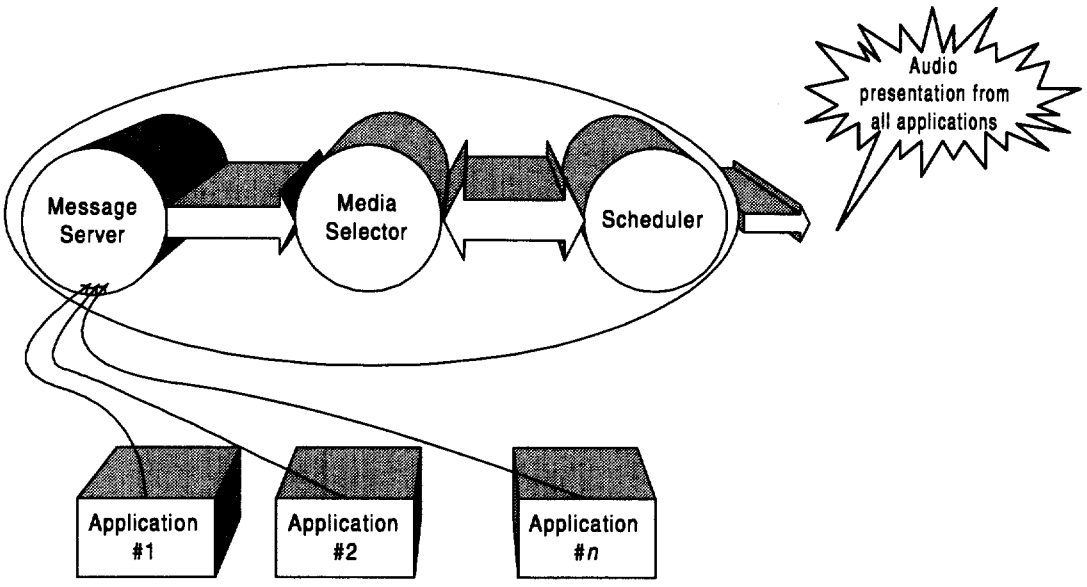


Figure 4.6: The system organization of the Centralized Audio Presentation System.

4.2.2 Presentation System Design

The Centralized Audio Presentation System has been implemented on a Silicon Graphics Indigo computer, using the C programming language. No sound hardware beyond that supplied with the computer system was necessary.

The Presentation System is composed of three distinct components: the descriptive request message server, the media selector, and the scheduler. Figure 4.6 illustrates the overall system organization. A message passing paradigm serves as the underlying model for communication between applications and the Presentation System. The arrows indicate the flow of messages. The content of these descriptive request messages and each of the parts of the Presentation System are described in the following sections.

4.2.2.1 Presentation System Descriptive Request Messages

Whenever an application is to represent some information in sound, it sends a request message to the Presentation System. This request includes one or more auditory representations of the information to be displayed, as well as other information that guides the Presentation System in making its decisions concerning *how*

and *when* the auditory message is to be presented. This allows the application programmer to concentrate on the application itself rather than the details of its auditory presentation. Furthermore, it guarantees the user a better overall auditory presentation since presentation decisions are made in the context of the current global auditory state of the system. Additionally, the user can set certain parameters affecting the Presentation System decisions. All of these factors are considered in the decision making process.

Descriptive Request Message Representations

Each request sent to the Presentation System can have a number of audio representations associated with it. Upon presentation, certain forms can be somewhat modified in order to enhance the perceptibility of the given auditory message. Each type is briefly described, along with the allowable modifications, if any.

- **Abstract Earcons:** Abstract earcons are the short, distinctive musical patterns that were discussed in Section 3.2.1. Abstract earcons are particularly good for representing abstract notions in the user interface for which no intuitive mapping between sound and meaning exists. The Presentation System has an extensible set of predefined earcons that makes up a basic vocabulary and can alert the user to system state information and certain common messages from applications.

  In the Presentation System, earcons are defined in a slightly more constrained way than normal. Each earcon is specified within a particular register, and no notes of the earcon can have pitches that fall outside of that register. The register parameter is defined to have no semantic relation to the meaning of the earcon. The Presentation System can dynamically modify the register when presenting an abstract earcon. The modification of register is done in such a way so that no concurrently playing earcons are ever in the same register. With these constraints, it can be guaranteed that there are no pitch crossings between overlapping earcons, and each concurrently playing earcon only sounds in its unique pitch band. This encourages perceptual grouping according to frequency proximity.

  The user must know that the register of an earcon might be different each time it plays. With this constraint, earcons can be presented simultaneously in a reasonably distinct manner.

- **Representational Earcons:** Representational earcons are familiar real-world sounds that represent the objects that they sound like. Their advantages and disadvantages are discussed in Section 3.2.1. The Presentation System can accommodate any representational earcons that the application programmer creates.

There are no modifications allowed for these sounds, because, in general, there is no guarantee that a modified representational earcon will retain the salient features necessary for the user to properly identify it. Gaver has done some research in altering real-world sounds according to physical parameters [Gav94]. This approach would work equally well with representational earcons, and would be a useful addition.

- **Synthesized Speech:** Speech is, under certain circumstances, the most reliable auditory means to communicate very specific information, such as exact numeric values. However, speech requires a higher cognitive load than any of the previously mentioned auditory constructs, and it is a relatively slow medium in which to communicate. This can potentially be problematic in a busy auditory environment. However, when conditions are amiable, speech can be a very natural and precise communications medium.

  In order to improve perceptual segregation, the Presentation System could dynamically alter certain qualities of the speech. Speech synthesizers usually allow changes in voice pitch and speed, at a minimum. The user would have to realize that the only informational content of the spoken message was the actual words. A faster spoken message does not imply a higher or lower priority than a more slowly spoken message. No modifications to the voice were made in the implementation, due to the relatively poor quality of the freely available speech synthesizers at that time. The implementation of speech pitch modification would be an important addition to this work, since it would potentially minimize pitch overlaps and crossings between speech and non-speech messages. The functionality of changing the spoken message speed would assist in allowing more speech representations to be used per unit time, and in preventing a backlog of unfilled requests from forming.

- **Parameterized Music:** Parameterized music is useful for the monitoring of certain persistent application variables. The music is continuous, and the human ear is very sensitive to small changes in tempo, volume, and pitch [Sum85]. The implementation limits music to monophonic, single note sequences. The user is responsible for learning the semantic mapping of application variables to music parameters. The functionality of this representation could be greatly enhanced through polyphonic music and stylistic parameters in a future version of the Presentation System.

- **Raw Audio Data:** Sometimes the application programmer desires certain sound effects, or wishes to convey some form of information not supported by the Presentation System. In this case, a raw audio representation can be included which contains no semantic information. Therefore, the representation of the message cannot be altered in way.

As an example, consider the sound of a coin falling onto a table. If the recorded sound of this event was modified, (for instance, in pitch or in spectral content) it may no longer be identified as a coin falling. Gaver's work leads to the possibility of modifying such a sound based on the physical interactions between materials that cause sound [Gav94]. If this auditory message had a representational earcon form, and if it had parameters defined specifically for the physical interactions that produce the sound, then perhaps some modification could be done; the qualities such as pitch and spectral content would be modified, but the qualities of the sound that allows one to recognize what it represents would remain.

Descriptive Request Message Parameters

Each descriptive request message contains a number of parameters that give preference information to aid the Presentation System in the choice of which encoded representation to use. The parameters also set constraints on how and when the selected representation is to be presented. The value of each request parameter is fixed by the application programmer during the design of an auditory user interface. The parameters are as follows:

- **Priority (*Low* through *High*):** This parameter indicates the perceptual importance of the request. A high value indicates that it is very important that the auditory message is clearly perceived over perhaps many other concurrently presented auditory messages. A low value means that the message is not of critical perceptual importance. The Presentation System must choose the representation of the request and the exact time of its presentation accordingly. Requests with high priorities are scheduled so that few messages overlap and are presented with an increased volume to avoid possible masking from other concurrently playing auditory messages. If necessary, lower priority auditory messages are preempted to help emphasize the higher priority message.

- **Latency (*Zero* though *Infinity*):** The latency value helps the Presentation System scheduler make temporal decisions. Auditory requests that, for example, need to coincide with visual elements should have a very low value for latency. Requests that are not so time critical can have higher values. This allows for more flexibility in the scheduling of numerous queued messages with higher latency values. A latency value of infinity means that the auditory request is delayed until no other auditory request (with a value other than infinity) is waiting to be sounded.

- **Preemption Rules (*No Preemption, Preempt & Reschedule, Preempt & Continue,* or *Preempt & Terminate*):** This parameter encodes the semantics of what should happen if the auditory message associated with this request is selected for preemption after it has starting playing. Some requests are encoded to disallow preemption. The requests that allow preemption must declare the action to take upon preemption. The auditory message can be terminated or continued from the point of preemption at a later time, or the request associated with the auditory message can be sent back to the media selector for reconsideration.

- **Encoded Auditory Form Preferences:** This parameter specifies to the Presentation System exactly which forms of audio are available for this particular descriptive request message, and how desirable (from the application programmer's point of view) each is to use. Many different auditory forms can be encoded into the same request, each with its own preference value ranging from 0 to 1. As an example, a request message could be defined to represent an alarm notification. It could have the following representations and preference values:

Voice Form — "Alarm Notification"	0.5
Rep. Earcon Form — (Alarm Sound)	0.8
Abs. Earcon Form — (Four note sequence)	0.2

4.2.2.2 Global System Parameters

Global system parameters do not have direct bearing on any particular application. Rather, they control certain aspects of the overall interface, and are maintained by the end user or the end user's environment. The following list describes the global parameters used in the Centralized Audio Presentation System.

- **Familiarity Level (*Low* through *High*):** The Presentation System can alter the representation of an auditory request dependent upon the user's familiarity with the system. A lower familiarity level causes the presentation system to represent auditory messages sequentially where possible, to avoid excessive overlap at the risk of queuing too many requests before they can be processed. A higher value indicates that the Presentation System can more liberally overlap auditory messages in order to avoid a backlog of unscheduled requests. This parameter also changes the interpretation of the actual descriptive request message parameters. For the novice (familiarity level set to low), a high priority message would be emphasized even more than for the

advanced user. Someone who is familiar with the interface would not need the assistance of an over-inflated priority level since that user would be more familiar with the auditory characteristics of a high priority message.

- **User Preferences (for media selection):** The application programmer sets the application preference parameters in each message request. The user can, in turn, define user preference parameters for each supported auditory form to influence the auditory output characteristics more to his or her liking.

  When the Presentation System chooses which representation should be used, user preferences are part of the computation for the most appropriate choice. For example, some users might prefer to hear as little voice representation as possible, and thus could indicate higher preference values for all of the other forms of auditory output.

  The user preferences are represented by a set of values between 0 and 1. One value exists for each auditory representation used. A low value indicates that the user prefers not to hear that type of representation as much as if a higher value were set. For example, let P be a user preference value. Then, if $P_{Abs.\ Earcon} = 0.8$ and $P_{Voice} = 0.4$, this would indicate that the user's preference for abstract earcons was in general twice the preference for voice messages.

- **Repeated Representation Penalty Function:** Some auditory forms are more suitable to be played simultaneously than others. This parameter contains a penalty function for each auditory representation. A candidate representation is penalized according to how many other auditory messages are currently sounding with the same type of representation. So, for example, if a voice is already playing, a new candidate voice representation would have the penalty value $R_{Voice}(1)$, where R_{Voice} is the penalty function for repeated voice representations, and 1 is the number of voice representations currently playing.

  The voice penalty function returns the greatest values because voice messages require a considerable cognitive load to understand and overlapped voice messages become very hard to perceive. The earcon penalty function returns very low values for $R_{Rep.\ Earcon}(1)$ and $R_{Rep.\ Earcon}(2)$, and increasingly higher values as the function's input value increases.

- **Ambient Noise Level (*Low* through *High*):** Dependent on this parameter, the Presentation System alters the master volume of all auditory output to make sure the sounds are audible over the current background noise. The ambient noise level is sampled periodically through a microphone and the parameter is automatically set accordingly.

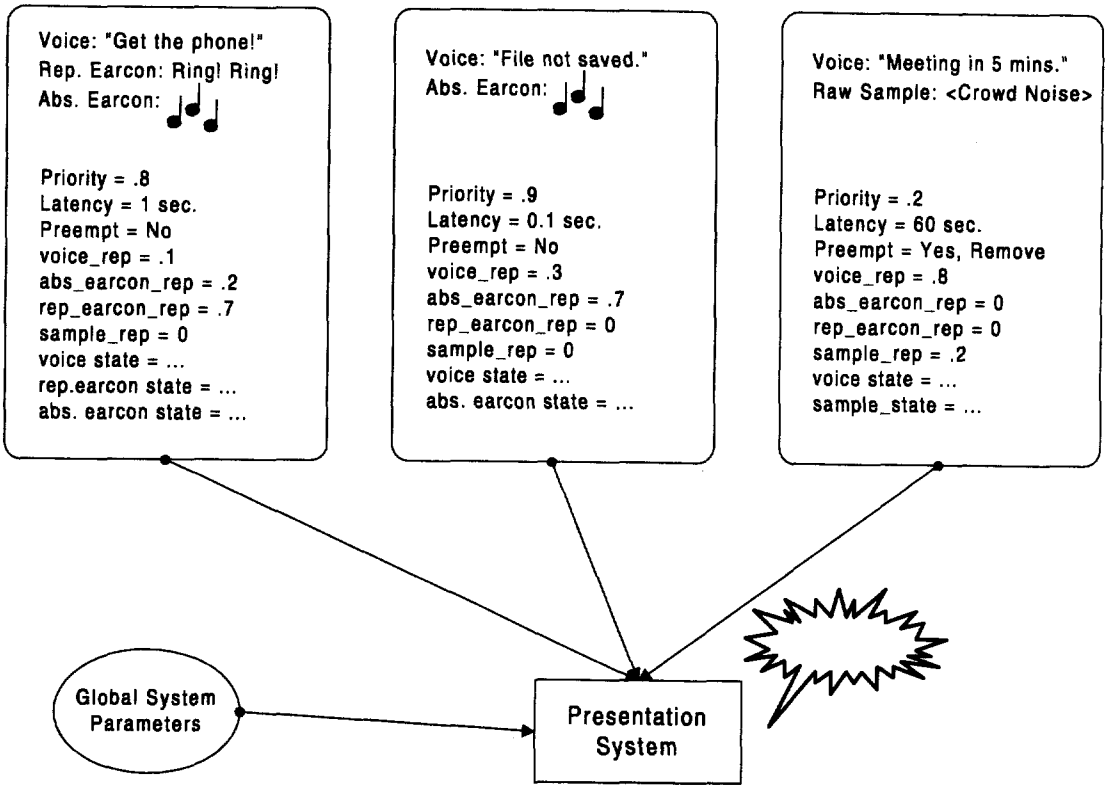


Figure 4.7: An example of three messages for which the Presentation System must choose an output format and then schedule.

Figure 4.7 updates the example in Figure 4.5. Now each request contains not only the alternate representations for the message, but also the request parameters that guide the Presentation System in making its media and scheduling decisions.

4.2.2.3 The Audio Message Server

The audio message server receives all auditory event requests from running applications. The dynamic variables describing global system information are read by the server and added to the requests before the Media Selector examines them. Each received request is stored in a request pool until it is examined by the media selector. The processing order of the requests is governed by the *priority* and *latency* fields of the pending requests. As requests age in the waiting pool, they are given an inflated priority value to avoid starvation. The priority increases as a polynomial function of the request's age.

4.2.2.4 The Media Selector

The media selector chooses which auditory representation of a given request is to be used. Each of the auditory representations works best to communicate specific types of information. However, the decision of which auditory form to use must take into account what audio is currently being sounded and what auditory forms are present for a particular request. After the media selector makes its decision as to how the information is to be represented, the actual audio data is scheduled and played. The work done by the media selector can be trivial, such as in the case that the incoming message is raw audio data. Since the media form is already predefined, the message is left unaltered and is immediately considered for scheduling according to its *priority* and *latency* parameters. The media selector might also have to do more work, such as in the case of a request that contains many representations. Based upon the application's preferences (encoded in the request), the user's preferences, values in the request fields (i.e., priority and latency), and the current auditory state, a representation must be elected.

This election process occurs as follows: A value v_i is computed for each representation i present in the request. The set of currently playing auditory forms affects the computed value of v_i, so the values will be different according to the global auditory state. The smallest v_i is the chosen auditory form for the given circumstance at the current time.

The values v_i are computed from the weighted equation

$$v_i = w_1(A_i) + w_2(U_i) + w_3(F(i)) + w_4(R(n_i))$$

where w_1 is the weight for the application preference, A_i is the application preference for representation i, w_2 is the weight for the user preference, U_i is the user preference for representation i, w_3 is the weight for familiarity level, $F(i)$ is the adjustment value for familiarity with respect to representation i, w_4 is the weight for repeated representation penalties, n_i is the current number of playing auditory messages using representation i, and $R(n)$ is the repeated representation penalty value for n. Each of the four weighting values are themselves user-adjustable.

After media selection, the auditory form can then be possibly altered in specific ways according to the type of representation. The allowable alterations are described in Section 4.2.2.1.

4.2.2.5 The Scheduler

The scheduler displays the chosen auditory form of each request taking into account the *latency* and *priority* parameters. The priority value included in a request indicates how important the clear perception of that request is. A high priority request must be conveyed precisely and clearly to the user within the latency requirement. The presentation of a low priority request might become masked, or possibly preempted, by higher priority requests. If an audio message becomes delayed in the scheduler for too long (currently 0.05 seconds), the corresponding original request is sent back to the media selector for re-evaluation since it is possible that the global auditory state of the system has significantly changed and the original representation may no longer be a suitable choice. At this point the *latency* parameter is reset to a smaller value to reflect the period of time that the request has been delayed.

As messages age in the scheduler, their priorities are temporarily increased by a polynomial function to avoid possible starvation. This priority increment does not affect the volume of the auditory message upon playback, nor does it affect the possible preemption decision of other auditory messages by the scheduler. The inflated priority does ensure that the request is played back before its latency period expires.

The incremented priority is a polynomial function of the age of the request:

$$p_n = p_i + an^2$$

where n is the age of a request, p_n is the priority of a request at age n, p_i is the initial priority of the request, and a is a system-wide constant. One behavior that follows from the dynamic aged priority scheme is that requests with longer maximum latency times can attain higher incremented priorities than equivalent requests with shorter maximum latency times. This is not necessarily a good result. Perhaps it would be better to have a maximum attainable priority parameter for each request, or even the ability for a request to have its own specific aged priority function. Then, the priority of each request could be altered as a function of age as is fit for its need.

In the selection of which auditory messages to play, both pending and waiting messages are considered. The following procedure occurs at a regular time step interval, currently set at 0.05 seconds. Each candidate message is assigned a score as a function of its *latency* and *priority* parameters. This includes messages currently playing. The scores of currently playing messages are further increased by a constant value. This additional score ensures that currently playing messages will generally have higher scores than new messages. The four highest scoring messages are selected for output in the current time step. However, there is a caveat

to this procedure. If an already playing auditory message is not selected for the next time step, then its preemption rules must be considered. If the message is encoded to be preempted and rescheduled, it is removed and then the original request is placed back in the pool of waiting requests, with all parameters set back to their original values. If the message is encoded to be preempted and continued, then upon preemption, a new request is created in the pool, containing only one representation — the remaining output of the original auditory message. If the message is encoded to be preempted and terminated, then the preempted message is not rescheduled in any way. Finally, if preemption is completely disallowed, then that message is scheduled to be played in the next time step, and the forth highest selected message is delayed. This effectively forces non-preemptable messages to be scheduled at every time step until they are completed. As such, the use of non-preemptable messages is discouraged in general, and should not be used for requests that contain long auditory representations.

4.2.3 Example Application: Maze Navigator

The maze navigator is one of a number of applications written to take advantage of the Centralized Audio Presentation System. The maze navigator outputs auditory requests to direct the user through a maze. No visual information is available whatsoever. In addition, the navigator keeps the user constantly notified of the straight-line distance to the exit, as well as speed and heading information.

The application has a number of descriptive request messages it sends to the Presentation System at appropriate times. Each message is queued, scheduled, and played by the Presentation System according to the parameters of the message. For example, the navigator application outputs very different audio with a user bias for voice than with a user bias for the non-speech representations. It also outputs different audio dependent upon concurrently scheduled auditory messages from other applications running simultaneously.

The speed, heading, and distance messages are presented constantly to keep the user regularly updated. Upon approaching a junction, the appropriate directional messages (North, South, East, and/or West) are played to alert the user of the directional choices. After a short time, the suggested directional choice is played (Go North, Go South, Go East, or Go West). Table 4.1 outlines which messages contain which auditory representations. Few messages contain a representational earcon because the messages are relatively abstract.

The distance message has only one representation: it is a quiet tone that varies in pitch with the straight-line distance to the goal. It is implemented using the

parameterized music representation supported in the Presentation System. The speed message is a raw sample representation that is repeatedly sent by the application at a frequency proportional to the user's speed. The earcon forms of the directional messages are each short motives that are played with a specific timbre. Table 4.2 summarizes the various timbres used in the messages.

Many overlapped auditory messages are presented to the user concurrently, yet the informal user results are positive. There is a good sense of speed and distance to the goal, and, with a little practice, the abstract earcons work to indicate directional information. The voice representations of the directional messages were very easy for users to understand without any training.

Occasionally, low priority messages like to speed indicator were preempted and not re-scheduled at all. This caused no problem in practice — the regular speed indicating beeping noise would momentarily drop out of the auditory scene only to reappear a moment later. This pause did not seem to affect the user's sense of speed in the maze, since the disruption occurred infrequently and at irregular intervals.

4.2.4 Example Application: Auditory Map

The Auditory Map project was re-implemented using the Centralized Audio Presentation System. Because Presentation System requests are at a high abstraction level, the redesign was simple and removed all of the sound playback details from the Auditory Map source code, resulting in a much simpler and more compact program. In its initial implementation, the Auditory Map played earcons only in the register defined by the earcon. The semantics of the Presentation System are that the register of an earcon is meaningless. This allows the register to be altered by the Presentation System if it will improve the perceptual attributes of the auditory output. For the new Auditory Map implementation, concurrent earcons would never play in the same register. This was a large improvement over the original implementation, since the frequency proximity cue could now assist in the perceptual segregation of each earcon. Furthermore, pitch crossings between concurrently played earcons were completely eliminated since each was sounded in a unique register.

The new Auditory Map implementation did take advantage of the ability to have multiple representations for each map object. A new mode was available to the user: mixed representation mode. When in this mode, audio requests would be encoded with the earcon representation of each object, as well as the voice representation. This allowed the Presentation System some flexibility in creating the auditory representation of the user's selection. Generally, a voice representation would be present as well as up to three earcons. Each earcon played in a unique

Message	*Abs Earcon*	*Rep Earcon*	*Voice*	*Raw Samp*	*Other*
North at Int	X		X	X	
South at Int	X		X	X	
East at Int	X		X	X	
West at Int	X		X	X	
Go North	X		X	X	
Go South	X		X	X	
Go East	X		X	X	
Go West	X		X	X	
Heading			X	X	
Speed				X	
Dist to Goal					X
Intersection	X	X			

Table 4.1: Representations used in the navigator application.

Message	*Instrument Timbre*
North at Int.	Trumpet
South at Int.	Tom Drum
East at Int.	Acoustic Guitar
West at Int.	Electric Keyboard
Go North	Trumpet
Go South	Tom Drum
Go East	Acoustic Guitar
Go West	Electric Keyboard
Intersection	Saxophone

Table 4.2: Instrument timbres associated with the abstract earcon representations of various messages in the navigator application.

pitch range. Two spoken voices would not overlap each other unless the user had specified a very large voice preference setting.

4.2.5 Centralized Audio Presentation System Analysis

The Centralized Audio Presentation System solved many of the original perceptual problems concerning presentation of temporally overlapping auditory messages in the Auditory Map project. In general, abstract earcon discrimination in the second version of the Auditory Map[1] was much improved over the initial implementation. There are a number of reasons for this. The concept of distanced fundamentals was implemented by ensuring that all concurrently sounding earcons played in a different register. This improves separation of audio sources in many cases, but not in all. Frequency proximity was enforced within each earcon through two rules:

- All notes of a given earcon were defined within one register of pitch range.
- Each earcon was dynamically played in a unique non-overlapping register from any other concurrently played earcons.

The biggest perceptual problems arose from common fate cues. Similar timbres between overlapping earcons caused perceptual difficulty in clearly perceiving each as a separate auditory message. Other common fate cues worked to fuse together streams that was supposed to remain distinct. For example, if the common onsets of successive notes in two earcons fell close to each other, discrimination become more difficult.

With the addition of allowable latency periods the Presentation System selection algorithm could stall the presentation of certain auditory messages until a clearer auditory state emerged. This contrasts the method in the original Auditory Map of presenting all pending auditory messages without regulation. In the original implementation, the messages were played in four concurrent streams, and no attempt at improving the perceptual qualities of the auditory output was made.

As in the original Auditory Map, the principle of familiarity was helpful for the segregation of earcons in the Presentation System. This is because the non-speech audio representations were built from either sampled instrument timbres or directly from a digital recording, so users could recognize most sounds immediately as familiar instruments.

[1]This refers to the Auditory Map version that used the Centralized Audio Presentation System

As previously mentioned, each earcon played in a unique register. If no free register was currently available, then another auditory form of the request would be used if available. In the case where no other form was present in the request, the Presentation System scheduler would either stall playing the request until one of the other auditory events ended, or it would preempt a currently playing earcon to free up resources for the new one. The decision between these two scenarios is based upon the relative priority and latency parameters of all of the currently playing requests as well as the pending requests.

One weakness in the Presentation System scheduling analysis method is that although messages play without overlapping fundamentals, they may contain overlapping harmonic content. The worst case of this is when two note sources have fundamentals that are nHz and $2n$Hz. The fundamentals are in different octaves, but the even harmonics of the lower pitched note coincide with all harmonics of the higher pitched note. The next worst case occurs when two note sources have fundamentals of nHz and $3n$Hz. In this case, every third harmonic of the lower note coincides with each harmonic of the higher note. It would be of value to have a more sophisticated method of ensuring that as few harmonics as possible were in common between concurrently presented sounds at any given time.

As with the original Auditory Map project, timbre discrimination could become problematic. The Presentation System would schedule speech representations when it could, or serialize the earcons it was requested to play, but in many circumstances the most appropriate action was to overlap non-speech auditory messages. In these cases, no analysis was done to help segregate each audio source, other than the streaming that would occur from playing different timbres in different pitch ranges. Furthermore, there was no analysis done during combined speech and non-speech audio events. This combination does not appear to be as troublesome as overlapped earcons, probably due to the rich and dynamic time-varying spectrum of speech audio that assists the auditory system in stream segregation. However, it certainly merits analysis in a busy acoustic environment.

Rhythm issues were ignored as well. The Presentation System would attempt to minimize overlap of audio events by delaying auditory messages where possible, but in many cases the overlap was unavoidable. There was no analysis of the overlaid rhythm patterns and how they might work to fuse overlapped audio sources into one stream. This problem appears to be secondary to the timbre problem, but it warrants further examination and possible regulation in a dynamic environment.

A relatively simple solution to aid in the discrimination of both timbre and rhythm between concurrent auditory messages is to localize each individual message in a three-dimensional virtual acoustic environment. Spatial position is one of the strongest cues for auditory stream segregation, and it has been used successfully

for applications in which a user must monitor a busy auditory environment such as an aeronautical cockpit simulation [Beg94b]. Sound localization in a virtual acoustic space was not implemented, but would be a very desirable future enhancement.

4.3 Comprehensive Auditory Scene Synthesizer

The Centralized Audio Presentation System had one general shortcoming that lead to perceptual problems between concurrently sounded auditory messages: there was no significantly intensive analysis performed to uncover potential perceptual problems between different temporal schedules of auditory messages. The choice of playing an auditory message of delaying it was made based upon the current auditory forms that were playing. It would be more accurate to analyze the actual sounds being produced and determine if a given sound could be added to the current auditory scene while still maintaining perceptual segregation of each sound.

Since the Presentation System was processing and serving requests in real-time, analysis was limited to some relatively simple heuristics such as limiting the temporal overlap of speech, and playing concurrently sounded earcons in non-overlapping pitch ranges. There were still a number of other perceptual cues that could arise between concurrent sounds in the auditory presentation, that would act to fuse together auditory streams that should remain perceptually segregated. Such cues include common onsets of notes, common offsets of notes, common note change patterns, timbre similarities, rhythm conflicts, and common harmonic content between concurrent sounds. More sophisticated techniques are necessary to detect and control these problems. It was questionable as to whether a more computationally intensive analysis could be done in real-time. So the next logical step in the exploration of controlling auditory output was to attempt to perform the analysis outside of a real-time environment to see if a good set of perceptual rules could be designed to further improve segregation between different auditory messages. For the domain of abstract earcons, these ideas were explored in the *Comprehensive Auditory Scene Synthesizer*. The temporal scheduling analysis could then be integrated into the Centralized Audio Presentation System at a later date, assuming that a reasonably efficient analysis technique was designed and implemented.

The number of ways to arrange a set of earcons within a given time interval grows exponentially with the number of earcons. For example, assume that the smallest addressable block of an earcon is 0.1 seconds. For an earcon of length two seconds (or 20 blocks), there are 31 valid temporal locations in which it can be placed in a five second (or 50 block) interval. To schedule three earcons, each 20 blocks long, in the same time interval, there would be 31*31*31 possible ways, or 29,791 differ-

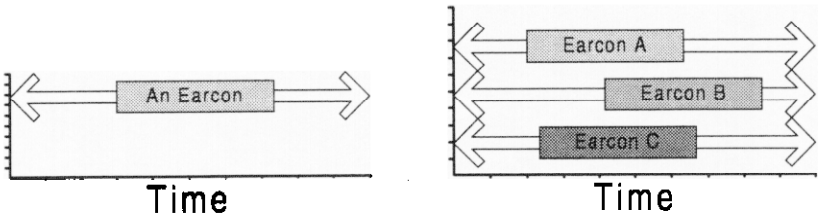


Figure 4.8: An earcon can be scheduled at many different offsets in a time interval. The scheduling possibilities grow exponentially with the addition of new earcons. If one earcon can be scheduled in 31 positions, two can be scheduled in 31*31 positions, and three can be scheduled in 31*31*31 positions.

ent possible schedules. See Figure 4.8. Depending on the particular schedule for the earcons, a user will have more difficulty in perceptually formulating and recognizing that there are three distinct and unique sources.

However, if it is possible to algorithmically predict when perceptual problems arise, a computer can analyze different layouts and select the most desirable temporal positions for the sounds. This sound presentation will contain many of the perceptual streaming cues that encourage the human auditory system to segregate each earcon into a separate auditory object in the presentation.

Bregman discussed the methods by which he believes the human auditory system analyzes the auditory scene. He started out by defining the term *heuristic*.

> The computer modeling approach has contributed an important idea that will be used in the coming chapters. This is the notion of a heuristic. The idea was involved in the process of designing computer programs to solve difficult problems for which no mathematical solution was known. The approach taken by the designers was to employ heuristics, which are defined as procedures that are not guaranteed to solve the problem, but are likely to lead to a good solution. An example would be the use of heuristic tests by computer chess programs to determine whether a proposed move would lead to a good position (e.g., to test whether the move would result in the computer controlling the center of the board or whether the move would lead to an exchange of pieces that favored the computer). Each move is evaluated by a number of such heuristics. No one of them can guarantee success, but if there are a large number, each with some basis in the structure of the game of chess, a move that satisfies most of them will probably be a good one. Furthermore, if each of the heuristic evaluation processes

> has a chance to vote for or against the move, the program will be less likely to be tricked than it would be if it based its move on only one or two criteria, no matter how good they were. [Bre90, pp. 32–33]

He continued then, to equate this heuristic evaluation process to how the perceptual system weighs a number of auditory scene heuristics.

> I believe that the perceptual systems work in similar ways. Having evolved in a world of mixtures, humans have developed heuristic mechanisms capable of decomposing them. Because the conditions under which decomposition must be done are extremely variable, no single method is guaranteed to succeed. Therefore a number of heuristic criteria must be used to decide how to group the acoustic evidence. These criteria are allowed to combine their effects in a process very much like voting. No one factor will necessarily vote correctly, but if there are many of them, competing with or reinforcing one another, the right description of the input should generally emerge. If they all vote in the same way, the resulting percept is stable and unambiguous. When they are faced with artificial signals, set up in the laboratory, in which one heuristic is made to vote for integration and another for segregation, the resulting experiences can be unstable and ambiguous. [Bre90, p. 33]

This method by which the perceptual system evaluates raw auditory data is the basis for the design of the Comprehensive Auditory Scene Synthesizer. A number of heuristics evaluate a possible candidate (or schedule for the sounds in an auditory scene), and its overall perceptual clarity is computed by a weighted equation (a "voting" technique where each voter has possibly more or less than exactly one vote). The synthesized auditory scene with the lowest score is the winner for the particular problem.

Each heuristic used in the implementation corresponds to some rule from either auditory streaming theory or music perception. A well synthesized auditory scene should allow the human perceptual system to easily segregate and perceive each of the sources of which it is comprised. It is intended that this model be used in general purpose user interfaces where numerous applications require some form of audio resources. The specific problem that the implementation addresses is how to more clearly present a given set of sound sources (specifically, earcons) within some time interval t. The problem is particularly of interest when the sum of the lengths of the sounds to be presented is much greater than t. Previous efforts in *computational auditory scene analysis* have been to solve the inverse problem: to decompose a sound that includes many separate sources into its component sources, much as the human auditory system does [Bre95] [CB92].

In the following chapter, a detailed description of the Comprehensive Auditory Scene Synthesizer is presented.

Chapter 5

The Comprehensive Auditory Scene Synthesizer: Architecture and Implementation

It is clear from the implementations of the Auditory Map System and the Centralized Audio Presentation System, that the problems associated with auditory stream perception in the user interface are complex and must be considered in the design of any auditory user interface that contains temporally overlapping audio. For this problem to be addressed successfully, a method is needed that takes into account the processes by which the perceptual system resolves sounds containing multiple auditory sources into their corresponding streams. With this information, an auditory presentation can be designed from singular auditory components, that encourages perceptual fusion and stream segregation of each individual source. The presentation will also minimize the situations that cause perceptual conflicts between sources. This chapter describes the design and implementation of the *Comprehensive Auditory Scene Synthesizer.* It is a computer application that analyzes a number of sounds and stores information about the predicted magnitude of perceptual conflicts between the different sounds. This analysis is performed for all possible discretized temporal overlaps between all pairs of sounds. The implementation is fully functional only for earcons. The application also creates a representation of sequential ambiguities that arise from playing one particular sound after another. The computed information is used to temporally arrange all of the inputted sounds within a given time interval. This problem is of interest when the combined length of the inputted sounds is much greater than the specified time interval. In this case, some degree of temporal overlap must occur between sounds.

The Comprehensive Auditory Scene Synthesizer was implemented in the C++ programming language using the Microsoft Visual C++ Developer Studio $v4.1$, running under the Microsoft Windows NT $v4.0$ and Windows 95 operating systems. The source code is object-oriented and designed to be extensible. All of the data structures and algorithms are portable to any platform with a C++ compiler, but the visual user interface elements and file I/O routines requires the Microsoft Foundation Class Library. This class library is available for platforms other than Microsoft Windows. The actual routines that output audio use the Microsoft DirectX 2 API, which is currently available only for Windows 95 and Windows NT 4.0. However, the audio routines are relatively simple and can be readily ported to another platform supporting 16-bit, 44.1kHz audio output.

5.1 Terminology

The following is a list of terms and concepts used throughout this chapter.

- A *Sound Source* s is an object representing a sound. In the current implementation, the only sound sources that can be fully analyzed are earcons. However, the implementation supports the creation and limited analysis of other sound source types. These will be further described later in the chapter. The length, in blocks, of a sound source is l_s.
- An *Earcon*, for the purposes of this implementation, is defined to be a sequence of notes, each with differing pitch and duration. Each note of a given earcon has the same timbre, and there are no rests present between the notes. An earcon contains only one note at any given time. There is no polyphony allowed within an earcon.
- A *Sound Presentation* SP is defined as a set S of n sound sources and a time interval t consisting of b blocks. The b blocks are numbered from 0 to $b-1$. Note that $\forall s \in S,\ l_s \leq b$.
- A *Node* n contains a one-to-one mapping of each element s in S to an integer value $o(s)$ between 0 and $b-1$. The integer values represent the starting offsets for each sound source in S within t. For example, if $\forall s \in S,\ o(s) = 0$, then that would mean all sound sources in the given sound presentation are scheduled to start playing simultaneously at offset 0. If, however, for two sound sources A and B, $o(A) = 3$ and $o(B) = 12$, that means that source A begins playing at offset 3 and source B begins playing at offset 12. If sound source A was longer than 9 blocks, there would be some temporal overlap between A and B.

A *Node* n also contains a score. It represents a computed metric of the perceptual clarity for the given arrangement of sound sources s when played with their offsets $o(s)$ relative to the beginning of time interval t. The score of a given node is a relative measure compared to other nodes that share the same t and have a different $o(s)$ mapping from the same S. The method by which the node score is computed will be examined thoroughly later in this chapter.

- The *Search Space SS* of a sound presentation *SP* is the set of all possible unique nodes for a given set of sound sources S and a time interval t. One can visualize a simple search space as follows: for each sound source $s \in S$, define an unique axis a_s in a discrete n-dimensional space.[1] Each s in S is associated with its unique axis. Let the sound source for axis a be referred to as s_a. The values along a given axis a_s correspond to the valid offsets (in blocks) that the associated sound source s_a can have and still be contained totally within t (which is of length b blocks). So, the values along axis a_s are $0, \ldots, b - l_s - 1$.

  The description of a simple example will help for clarity. In this example, two sound sources are to be scheduled. Sound source A has $l_A = 7$ and sound source B has $l_B = 5$. The sound presentation *SP* consists of the set $\{A, B\}$ and $t = 1.0$ and $b = 11$. The search space $SS(SP)$ has two dimensions (the x- and y-axes). Let A be assigned to the x-axis, and B to the y-axis. The indices along the x-axis range from $0, \ldots, b - l_A - 1$, or $0, \ldots, 4$. Similarly, the indices along the y-axis range from $0, \ldots, 6$. Every point in the two-dimensional search space corresponds to a unique node sharing S and t. Therefore, each node in the search space corresponds to a unique schedule of A and B. Figure 5.1 shows this graphically. Each of the circles represents a unique schedule of the two sounds within the time interval t. For example, the point $(4, 2)$ represents sound source A playing from block 4 (and ending after block $4 + 7 = 11$)[2] and sound source B playing from block 2 (and ending after block $2 + 5 = 7$).

  The score of each node in this two-dimensional search space is not represented in Figure 5.1. Figure 5.2 shows an example of two earcons, arranged on an x-y graph. Each node score is represented by the z-component at a given (x, y) point. On the resulting surface, the global minimum node score (o_x, o_y) represents the offsets of the earcons in the sound presentation that yield the predicted most perceptible schedule for the two sounds. In this example, $o_x = 0$ and $o_y = 7$. Similarly, the global maximum node score represents the predicted least perceptible schedule of the two sounds.

[1] Recall that n is the number of sound sources in S.

[2] Remember, block numbering starts at 0. Block 4 is the 5^{th} block!

- A *Heuristic Search Algorithm* is an algorithm that remembers the best n states as it searches through the search space. If exploration of one of these states does not appear promising any longer, the algorithm can switch to the next most promising state it stored previously. Heuristic search algorithms use different sizes of n and different criteria for determining when a given path through the search space no longer appears as promising as an alternate state. The number of nodes in a search space for non-trivial problems makes it impossible to exhaustively evaluate every node score. A heuristic search technique is one approach to finding a good solution quickly while minimizing the number of node evaluations.

Throughout this chapter, a number of earcons and search spaces are referenced. Earcons appear with labels such as [TP3]. The definition of each earcon can be found in Appendix A. Search spaces are referenced with labels such as [SS4]. Their definitions can be found in Appendix B.

5.2 The Architecture of the Comprehensive Auditory Scene Synthesizer

In this section, the organization of the data structures used in the architecture is described. Also discussed are the various alternate representations of each sound source. The availability of the sound data in multiple forms simplifies the implementation of the algorithms that operate on the data. Each individual algorithm uses the most appropriate representation for solving its particular problem. These structures and representations are organized into a class hierarchy that makes it easy to extend the architecture to work on new audio data types and representations.

5.2.1 *SoundSource* Methods

The class structure of sound sources is illustrated in Figure 5.3. The base class of any sound in the system is *SoundSource*. The *SoundSource* class is a *virtual* class. This means the class cannot be directly instantiated. Rather, the *SoundSource* class is a repository of methods that will operate on any of the subclasses of *SoundSource*. All *SoundSource* derived classes inherit the following functionality:

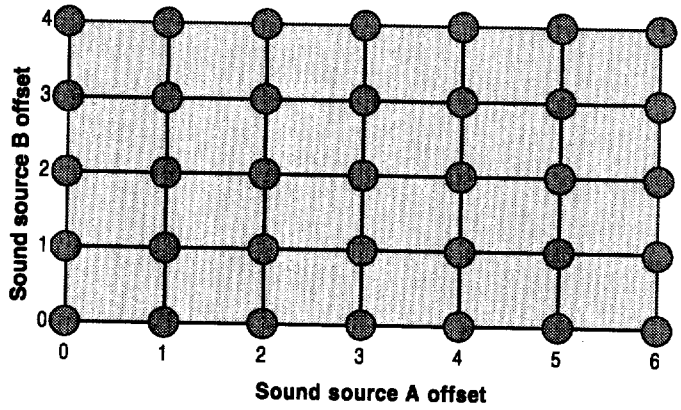


Figure 5.1: The search space for two sound sources A and B.

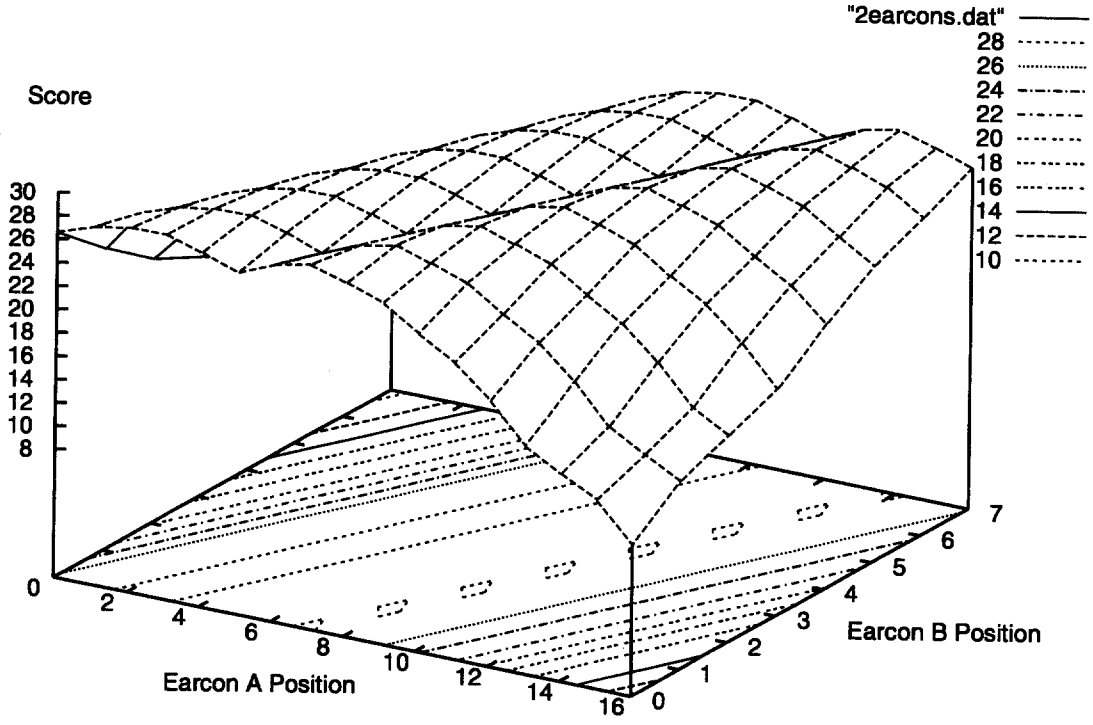


Figure 5.2: A 2-earcon search space with the value of each node plotted on the z-axis.

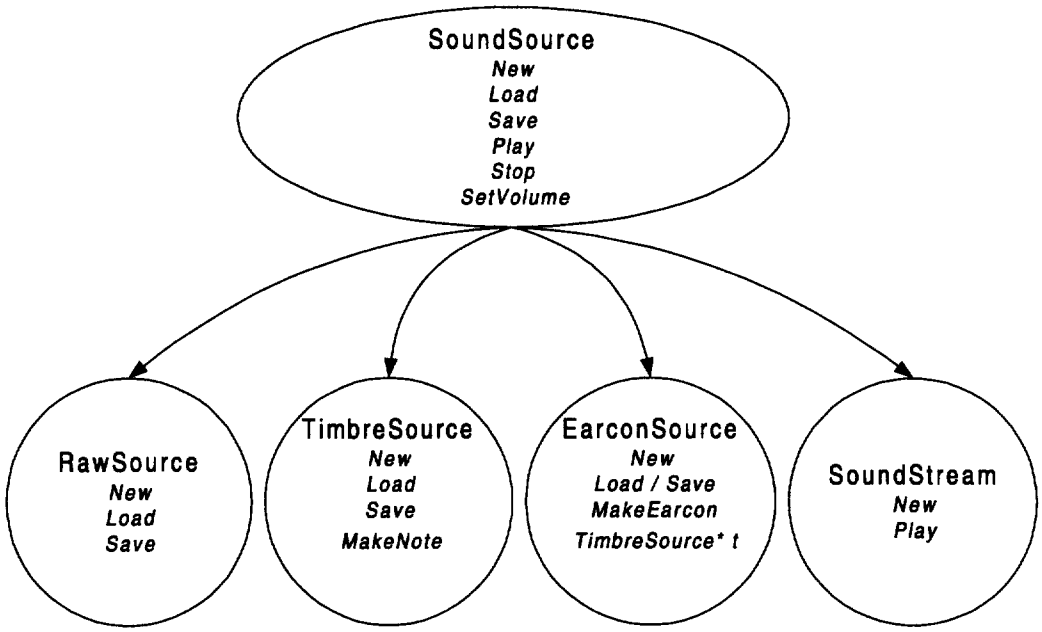


Figure 5.3: The SoundSource class hierarchy and selected methods.

- *New* – Set up and allocate necessary items to hold a sound.
- *Load* – Read a file from the disk and store the contents in the *SoundSource* object.
- *Save* – Write a file to the disk containing the *SoundSource*.
- *Play* – Play the *SoundSource* object.
- *Stop* – Abort the playback of a sound immediately.
- *SetVolume* – Change the loudness of a *SoundSource* object.

The first three methods, *New*, *Load*, and *Save*, are *pure virtual functions*. This means that there is no implementation of the methods in the *SoundSource* class. Each subclass must define these three methods to use them. The methods *Play*, *Stop*, and *SetVolume* are implemented in the *SoundSource* class and can be redefined by a subclass if necessary.

The *RawSource* class is a specific type of *SoundSource*. It contains the methods necessary to operate on a digital recording of sound. It will load and save monaural samples. The *TimbreSource* class is a *SoundSource* that represents an instrument timbre. The class contains the inherited methods of *SoundSource*, plus an

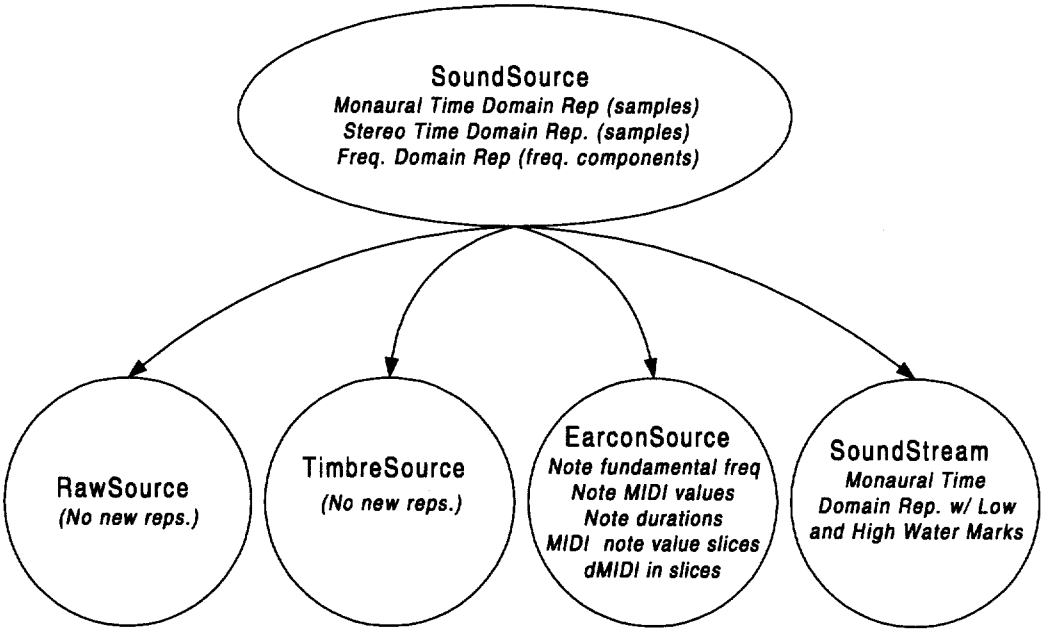


Figure 5.4: *SoundSource* derived classes and their sound representations.

additional method *MakeNote* that will create a note of a certain pitch and duration using its timbre. The *EarconSource* class contains the inherited methods plus a new method *MakeEarcon* to initialize the object to a particular earcon specified by a sequence of <Pitch, Duration> 2-tuples. The *EarconSource* also contains a pointer to a *TimbreSource* object that represents the timbre of the *EarconSource*. Finally, the *SoundStream* class allows the creation of a continuing sound that has no predefined ending point. *SoundStream* redefines the *New* method, and currently has no capabilities of loading and saving.

5.2.2 *SoundSource* Data Representations

The Comprehensive Auditory Scene Synthesizer stores a number of different representations of a sound depending upon the sound type. The multiple representations are useful because algorithms can operate on the representation that is most natural for the type of operation being done. The representations held in each class are outlined in Figure 5.4.

The *SoundSource* class holds representations common to all subclasses in the hierarchy. There are three representations stored in a *SoundSource* object.

- **Monaural Time Domain Representation:** The monaural time domain representation is an array of samples that represent the sound source. All sounds have a sample rate of 44.1kHz, and each sample has 16 bits of resolution. In some cases, the monaural time domain representation is created from an actual digital recording. In other cases, the representation is computed from a parameterized synthesis algorithm.

- **Stereo Time Domain Representation:** The stereo time domain representation is derived from the monaural time domain representation. It is an array of sampled values, only each value is repeated. The array is exactly double the size of the array in the monaural time domain representation. The reason for storing this representation is to allow processing for possible stereo effects in the presentation of any sound. These effects could assist in the localization of a given sound source.

- **Time-Stepped Frequency Domain Representation:** This representation is also computed from the monaural time domain representation. The time domain samples are first stored as a series of blocks, and then a Fast Fourier Transform (FFT) is performed on each block. This representation, stored as a list of arrays, contains information on the frequency content of any sound source at any given time block. Figure 5.5 shows a visualization of this representation for the earcon [TP3].

Additional representations are used by the subclasses of *SoundSource*. The *SoundStream* class redefines the monaural time domain representation so that there is a "high water mark" and a "low water mark" for the buffer that holds the actual samples. The number of samples in the *SoundStream* should always fall between these two values. Since the samples are constantly being played (and as such, "drained" from the object's buffer) a function to maintain the correct level must periodically fill the buffer up to the high water mark with the appropriate samples. The high water mark should not be so high as to create a long latency if the sound must suddenly change. (The old, "stale" samples must be played before the new, "fresh" samples are played. The new samples reflect the change in the sound.) The high water mark should also not be so low as to require too much monitoring and buffer refilling. This causes too much overhead in its upkeep. Values around 0.1 seconds for the high water mark seem to work reasonably well. The low water mark should not be so low that the possible overhead of refilling the buffer at the low water mark will result in the buffer completely emptying before refilling. This results in an audible break in the flow of the audio stream. The low water mark must also not be so high that it causes excessive latency when new, "fresh" samples enter into the buffer.

Figure 5.5: A visualization of the time stepped representation of an earcon in the Comprehensive Auditory Scene Synthesizer.

The *EarconSource* class uses the inherited representations of the *SoundSource* class, as well as a number of new representations. The first new representation is the <Pitch, Duration> 2-tuple list. An earcon has a particular timbre associated with it, and is a sequence of notes. So, an earcon can be represented as

Timbre T
<Pitch, Duration>, <Pitch, Duration>,...,<Pitch, Duration>

where each <Pitch, Duration> 2-tuple represents a note in the earcon. This representation is very compact and encodes the important information about an earcon into a form that can be quickly and efficiently examined. Note that the pitch values stored are MIDI note values. These values range from 0 to 127 and represent the pitches for 128 different notes of the chromatic scale. The MIDI note values represent pitch linearly — to increase a value by an octave, one adds 12 to the value. (There are 12 semitones in an octave).

Another representation used is very similar to the <Pitch, Duration> 2-tuples, only the Pitch is replaced by the fundamental frequency, f_0, of the note. The $<f_0$, Duration> 2-tuple representation is more natural to operate upon for certain algorithms. Specifically, it is very easy to compute the harmonic series of a pitch from its fundamental frequency since the series consists of the integral multiples of f_0. Frequency values grow exponentially as pitch grows linearly — to increase the pitch of a note by an octave, one doubles the fundamental frequency value.

The next representation stored for earcons is the Pitch-per-Block format. The <Pitch, Duration> 2-tuple representation is used to create a new representation in which the pitch value of the note present in each block is stored in an array. This representation allows algorithms to very quickly reference what pitch is present in any block of an earcon. For the cases where two notes are present in a block (this occurs during a transition between two notes), the pitch that is present longer is stored. In Figure 5.6, the first four notes of [TP3] are actually the same length. Due to the rounding off of notes into blocks, the lengths of the notes in this representation become 2, 3, 3, and 2 blocks. The accuracy of this representation can be improved by using smaller blocks or longer notes.

The last representation for earcons is the Change-in-Pitch-per-Block form. This representation stores, on a block by block basis, the change in the pitch (measured in MIDI note values). Many of the values will be 0, since a note usually spans a number of blocks, and has no change across that range. Figure 5.7 shows the representation for the earcon [TP3].

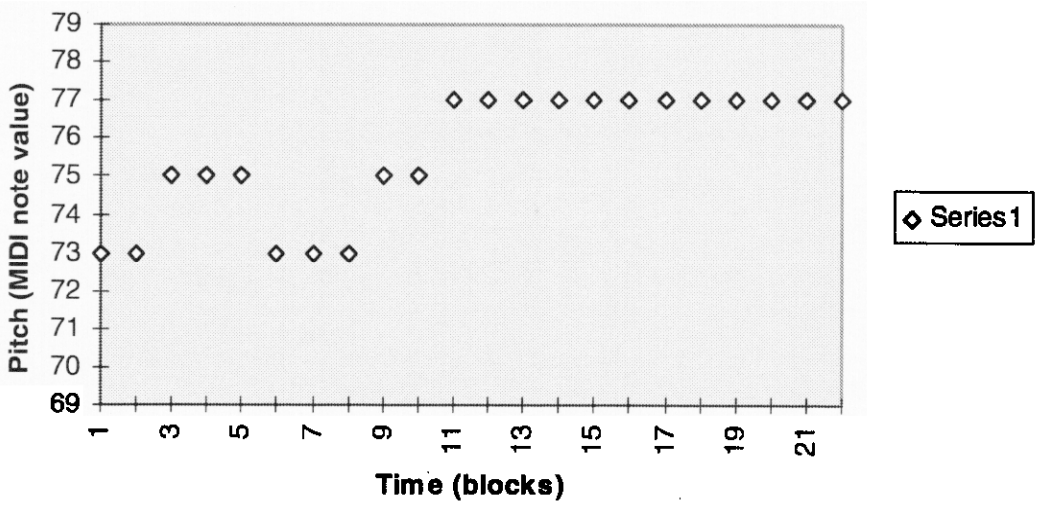


Figure 5.6: An example of the Pitch-per-Block representation for [TP3].

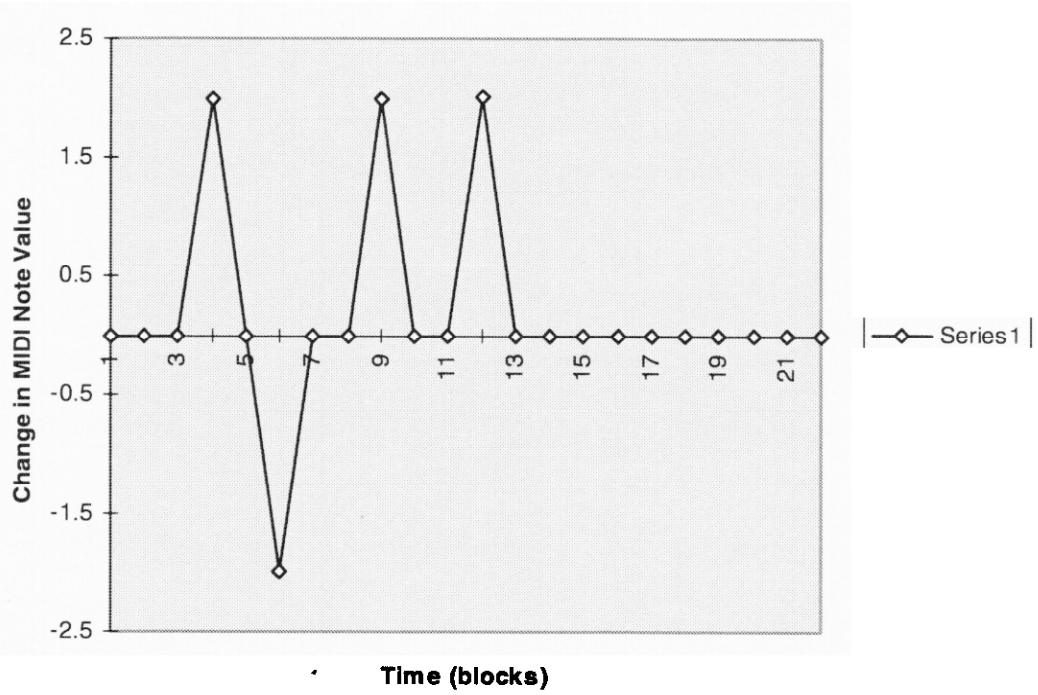


Figure 5.7: An example of the Change-in-Pitch-per-Block earcon representation for [TP3].

5.2.3 Auditory Data Block Size

During the building of sound source representations, all sounds are broken down into a series of blocks from their time domain representation. A block is a section of the sound that is of a given length. It is represented as a sequence of samples in the time domain. All blocks in the actual implementation are 4096 samples long. Since all sounds are created at a sampling rate of 44.1kHz, a block represents $4096/44100 \approx 0.093$ seconds of a sound. There are a number of tradeoffs in choosing a block size. Let s be the size of a block, and b be the number of blocks in a sound source. As s decreases, b increases (and vice-versa). Many of the data structures for the analysis of n sounds increase in size by $O(b^n)$ where $n \geq 1$. Therefore, making s smaller will increase the size of data structures and the time to operate on them at a polynomial rate. The number of possible schedules for a sound presentation grows exponentially with n, and as such, larger values of n rapidly increase the search space from which a solution must be found. The growth rates of search spaces are discussed further in Section 5.4. The entire search space cannot be examined in reasonable time except in the trivial cases, but nonetheless, a smaller search space allows a greater percentage of that space to be examined, and could yield a better result.

Finally, block size affects the resolution of frequencies within the block that can be examined. Nyquist's Theorem states that a signal can represent frequencies up to one half of the sampling rate. A block contains 4096 samples created at a sample rate of 44.1kHz. It can represent frequencies from 0Hz To 22.05Hz. However, when the block is analyzed by itself, it can contain only 2048 unique frequencies, since there are only 4096 samples to consider. These frequencies are linearly spaced from 0Hz to 22.05kHz. All frequencies within a band of 22050/2048, or approximately 10.77Hz, will be represented in the block as the same frequency. This loss of resolution can become important when there are frequency components close together in a sound and if a decision should be based upon those close frequency components. If the frequencies are grouped together in the same frequency band, the decision cannot even be considered. By increasing the block size s, the loss of resolution is lessened, thus yielding more precise frequency component information in the frequency domain. This would minimize any decision errors by the heuristics that use the frequency information. A decrease in s would yield a greater loss in resolution, and hence increase the possibility of bad decisions by heuristics. Table 5.1 summaries these tradeoffs.

Small Block Size	Large Block Size
Bigger Tables and Data Structures	Smaller Tables and Data Structures
Poorer Frequency Resolutions	Better Frequency Resolutions
More Sound Presentation Layouts	Less Sound Presentation Layouts
Larger Solution Space to Search	Smaller Solution Space to Search
More Flexibility for Presentation	Less Flexibility for Presentation
Longer Computation Time	Shorter Computation Time

Table 5.1: The tradeoffs involved with choosing a block size for the auditory data representation.

5.2.4 Exclusion Tables

At the heart of the design for concurrent sound analysis is the *exclusion table.* Each sound source in a sound presentation has its own exclusion table. The table stores the all of the penalty values for all possible overlaps of itself with any other sound in the presentation. Table 5.2 shows the exclusion tables for a group of three earcon sources in a sound presentation. The number of columns in each exclusion table is governed by the length of the earcon of the table (the values of L_A, L_B, and L_C). For example, if a table's earcon is six blocks in length, then there are six ways to place any other earcon such that the other earcon does not start before the table's earcon, and so that there is at least one block of overlap between the two earcons. Note that a row for the table's earcon is included in its own table. The values for that row represent the penalty scores when the earcon is played concurrently with itself at different relative offsets.

The values that fill the exclusion tables come from the algorithms described later in this chapter. A weighted sum of the penalty values from each algorithm is stored as an exclusion table entry. A detailed description of this weighted sum is in Section 5.3.8. Typically, there is a trend of descending scores as the offset increases. This is because there is less of an overlap between the two sources, and therefore, less chance of penalties associated with the overlap. Of course, if there is no overlap between two earcons, the overlap penalty is zero and it is not stored in the exclusion tables.

The exclusion tables are dynamically created upon the exploration of a search space. They are designed such that new sounds can be added very easily. This is important so that possible future versions of the Comprehensive Auditory Scene Synthesizer running in real time can register new, previously unanalyzed, sounds quickly. When a new sound is registered, the exclusion table for each current sound

	Offset 0	Offset 1	Offset 2	Offset 3	Offset 4	Offset 5...	Offset L_A
Earcon A	15.1	14.3	12.9	11.8	12.7	10.0	0.4
Earcon B	3.5	2.4	1.0	0.3	1.6	0.8	0.3
Earcon C	1.6	4.9	3.5	3.1	2.9	2.3	1.5

Exclusion Table for Earcon A (Earcon A always starts at Offset 0)

	Offset 0	Offset 1	Offset 2	Offset 3	Offset 4	Offset 5...	Offset L_B
Earcon A	3.1	2.3	3.9	1.8	1.4	0.9	0.3
Earcon B	12.5	11.4	11.3	10.3	8.7	7.8	1.3
Earcon C	5.9	5.6	4.5	2.9	2.4	2.1	0.8

Exclusion Table for Earcon B (Earcon B always starts at Offset 0)

	Offset 0	Offset 1	Offset 2	Offset 3	Offset 4	Offset 5...	Offset L_C
Earcon A	4.2	3.5	4.4	2.8	2.0	2.3	0.4
Earcon B	3.5	2.4	1.0	0.3	1.6	0.8	0.3
Earcon C	20.6	18.4	17.8	15.1	16.1	15.3	1.5

Exclusion Table for Earcon C (Earcon C always starts at Offset 0)

Table 5.2: An example of exclusion tables for a sound presentation of three earcons.

grows by one row. This row represents the penalty values for the temporally overlapped presentation of the new sound source with the table's sound source. Working from the example in Table 5.2, when a new earcon D is introduced into the system, each of the three exclusion tables gets another row for "Earcon D." The system computes the penalty values for this row and saves the values in the table. Additionally, another table is dynamically created for the new earcon D. Continuing with the current example, the new table has four rows. The penalty values are then computed and saved in the table. The resulting exclusion tables are represented in Table 5.3.

5.3 The Presentation Evaluation Method

A sound presentation contains a collection of sound sources and a time interval in which all the sound sources must be scheduled. To find a good schedule, the search space of the sound presentation is explored. Each node in the search space contains a unique schedule of the sounds in the given time interval, and a score. An answer, when found, is in the form of a node. It is then converted to an audio format suitable for listening to.

The presentation evaluation implementation supports creation of all of the sound source types, however the sound sources that comprise a sound presentation must be earcons. This simplifies the details of the implementation. The object-oriented architecture of the application supports expandability for analysis of other types of sound sources.

The exclusion tables hold the overlap penalty values for all pairs of sounds in all possible overlapping positions. The evaluation of a given sound presentation consists of adding up the overlap penalties of all pairs of sounds in the presentation, and then adding to that the penalties (if any) derived from an earcon following another in sequence (computed on the fly) for all pairs of sounds in the presentation. This sum represents the score for a particular node in the n-dimensional search space.

The Computational Auditory Scene Synthesizer is useful because it contains much more sophisticated heuristics for avoiding perceptual problems than in the Centralized Audio Presentation System. Both the properties of overlapping sounds and sounds in sequence are considered. The implementation uses a set of algorithms to detect certain features of overlapping sections of two sounds to determine how well a given sound presentation will be perceived by the listener. These algorithms only compute penalties for the overlapping regions of the two earcons

	Offset 0	Offset 1	Offset 2	Offset 3	Offset 4	Offset 5...	Offset L_A
Earcon A	15.1	14.3	12.9	11.8	12.7	10.0	0.4
Earcon B	3.5	2.4	1.0	0.3	1.6	0.8	0.3
Earcon C	1.6	4.9	3.5	3.1	2.9	2.3	1.5
Earcon D	**4.2**	**3.8**	**1.9**	**0.7**	**0.4**	**0.3**	**0.2**

Exclusion Table for Earcon A (Earcon A always starts at Offset 0)

	Offset 0	Offset 1	Offset 2	Offset 3	Offset 4	Offset 5...	Offset L_B
Earcon A	3.1	2.3	3.9	1.8	1.4	0.9	0.3
Earcon B	12.5	11.4	11.3	10.3	8.7	7.8	1.3
Earcon C	5.9	5.6	4.5	2.9	2.4	2.1	0.8
Earcon D	**5.1**	**3.2**	**6.8**	**9.2**	**6.4**	**3.8**	**0.8**

Exclusion Table for Earcon B (Earcon B always starts at Offset 0)

	Offset 0	Offset 1	Offset 2	Offset 3	Offset 4	Offset 5...	Offset L_C
Earcon A	4.2	3.5	4.4	2.8	2.0	2.3	0.4
Earcon B	3.5	2.4	1.0	0.3	1.6	0.8	0.3
Earcon C	20.6	18.4	17.8	15.1	16.1	15.3	1.5
Earcon D	**4.1**	**2.0**	**1.4**	**1.7**	**1.8**	**1.3**	**0.5**

Exclusion Table for Earcon C (Earcon C always starts at Offset 0)

	Offset 0	Offset 1	Offset 2	Offset 3	Offset 4	Offset 5...	Offset L_D
Earcon A	**4.4**	**3.8**	**2.4**	**3.7**	**2.9**	**1.8**	**0.4**
Earcon B	**7.5**	**5.8**	**3.3**	**1.9**	**2.5**	**1.8**	**0.6**
Earcon C	**3.4**	**4.2**	**2.9**	**2.0**	**1.6**	**1.8**	**0.9**
Earcon D	**23.5**	**15.9**	**12.8**	**14.5**	**13.1**	**9.5**	**0.8**

Exclusion Table for Earcon D (Earcon D always starts at Offset 0)

Table 5.3: The exclusion tables after adding Earcon D. Boldfaced areas represent new additions.

being analyzed. The method allows for the easy addition of new heuristics as well. The heuristics used to detect conflicts between overlapping sounds are as follows:

- Common Onset of Notes
- Common Offset of Notes
- Pitch Crossings
- Common Harmonics
- Common Pitch Changes
- Timbre Similarities
- Average Pitch Separation

For sequential properties between two earcons, the following features are detected.

- Timbre Similarity
- Similarity of Pitch

In the following sections, the implementations of the heuristic algorithms are described in detail. Throughout the descriptions, surface plots of example search spaces are shown. In all of these examples, there are actually three earcons instead of two. Earcon A is always scheduled at offset 0, and the other two earcons B and C have varying offsets along the x- and y-axes, respectively. The resulting surface then, is one two-dimensional surface "slice" of a larger three-dimensional surface, where each slice of the three-dimensional surface corresponds to a similar surface plot but with the "fixed" earcon A at a different offset.

The advantage of presenting surface plots in the described fashion is that more interesting features emerge with the interaction of three earcons as opposed to two. However, the cases are generally still simple enough that the features of the surfaces can be readily identified as particular interactions between two specific earcons in the search space. For the purposes of all surface plot examples, the x-axis is defined to be the axis running from the origin (on the lower left of the surface plot) to the right and towards the reader. The y-axis runs from the origin away from the reader and to the right. The z-axis extends vertically from the origin.

If there are three earcons in the search space, a question arises as to exactly how the node score is computed when the exclusion tables only hold penalty values for pairs of earcons. This will be fully described in Section 5.4 A simple explanation for the purposes of this section is that the score for any node containing three earcons A, B, and C, is computed by adding the overlap penalties of (A and B), (A and C), and (B and C). Similarly, sequential penalties are added for sounds following each other in time.

5.3.1 Common Onset / Offset

The properties of common onset and common offset refer to the simultaneous starting and stopping of notes in two or more earcon sources.. Since it is known that the perceptual system tends to group together sonic events that start and stop simultaneously, this attribute is one that should be negated as much as possible.

The algorithm to compute the common onset and common offset penalties between two sounds uses the <Pitch, Duration> 2-tuple *EarconSource* data representation. By using this representation, the implementation becomes relatively simple. There are two note counters that function as indices into the 2-tuple array representation. Two time pointers are updated according to the <Duration> field of the 2-tuple representation. The time pointers are compared to see if they are approximately equal. If so, then a common onset/offset pair is present — both note counters are advanced and the time pointers are updated. Otherwise, the pointer that is further back in time gets moved ahead to the next note. The two time counters are then compared, and so on. Figure 5.8 illustrates the three cases of common onsets and offsets for earcons:

- A common onset between two notes at the beginning of the overlapped interval.
- A common onset/offset pair occurring in the middle of the overlapping earcon interval.
- A common offset between the final note of one earcon and a note of the other.

The following pseudo-code does not account for the special cases of common onset between two notes at the beginning of the overlapped interval, and common offset between the two final notes of the overlapped interval. These cases are identified and penalized in the actual implementation.

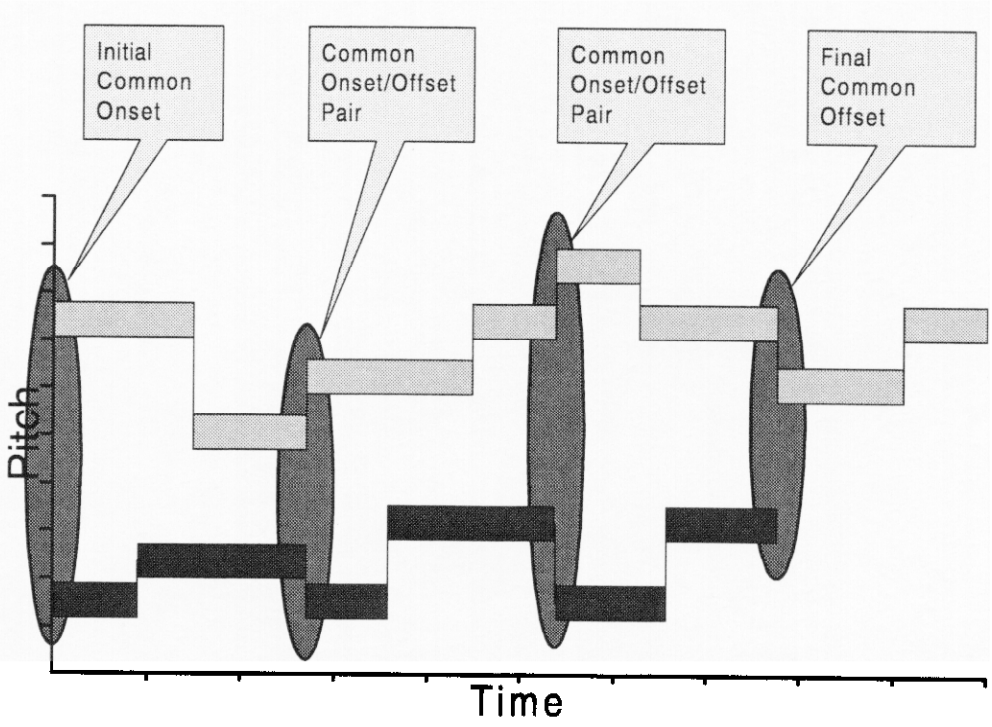


Figure 5.8: The common onsets and offsets of notes are counted by the onset/offset algorithm.

```
// Note: A starts first.
// Then B starts RelativePos blocks after A starts.
CommonOnsetOffset(Earcon2TUP A, Earcon2TUP B, Offset OffsetB)
                  returns Integer
{
    Penalty = 0

    // IndexA and IndexB are indices indicating the current
    // 2-tuple rep. note that is being examined.
    IndexA = 0
    IndexB = 0

    TimeA = 0   // A always starts at 0
    // Convert RelativePos from blocks to secs. for TimeB
    TimeB = OffsetB * 4096 / 44100

    While (More notes in A and in B)
    {
        // Check if a common onset/offset is present now
        dt = Abs(TimeA-TimeB)
        If (dt < .5)
        {
            // Common onset/offset.  Increment Penalty
            Penalty = Penalty + (-1 / (1 + e**(-(dt*20-5))) + 1)
            // Move both counters and pointers
            IndexA = IndexA + 1
            IndexB = IndexB + 1
            TimeA = TimeA + A[IndexA]
            TimeB = TimeB + B[IndexB]
        }
        Else
        {
            // One sound is ahead of the other.
            // There is no common onset/offset currently.
            // Step ahead the note pointer that is currently
            // further behind.
            If (TimeA < TimeB)
            {
                // Move ahead in A
                IndexA = IndexA + 1
                TimeA = TimeA + A[IndexA]
```

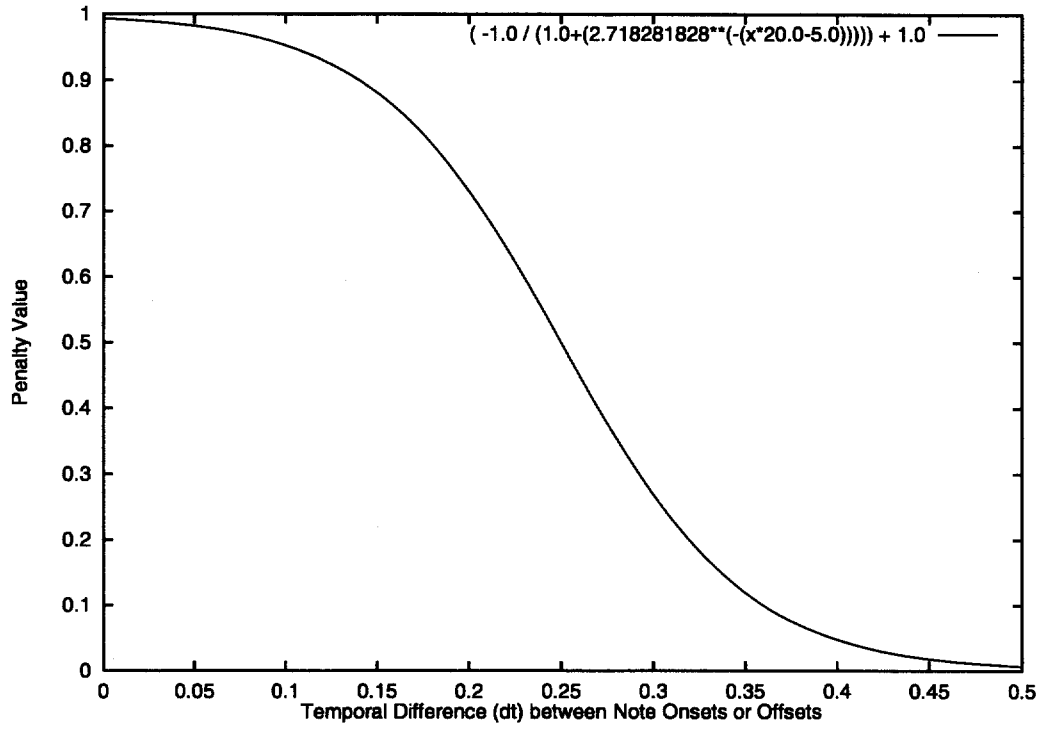


Figure 5.9: The common onset/offset sigmoid penalty function.

```
            }
            Else
            {
                // Move ahead in B
                IndexB = IndexB + 1
                TimeB = TimeB + B[IndexB]
            }
        }
    }
    Return Penalty
}
```

In the algorithm, the penalty is incurred if any two notes are less than 0.5 seconds apart in their start or stop times. The penalty should be much higher for notes that start simultaneously than for notes that start a few tenths of a second apart. That is why the sigmoidal function $-1/(1+e^{-(dt*20-5)})$ is used to actually increment the penalty for a given pair of notes with common onset or offset. The penalty function is shown in Figure 5.9.

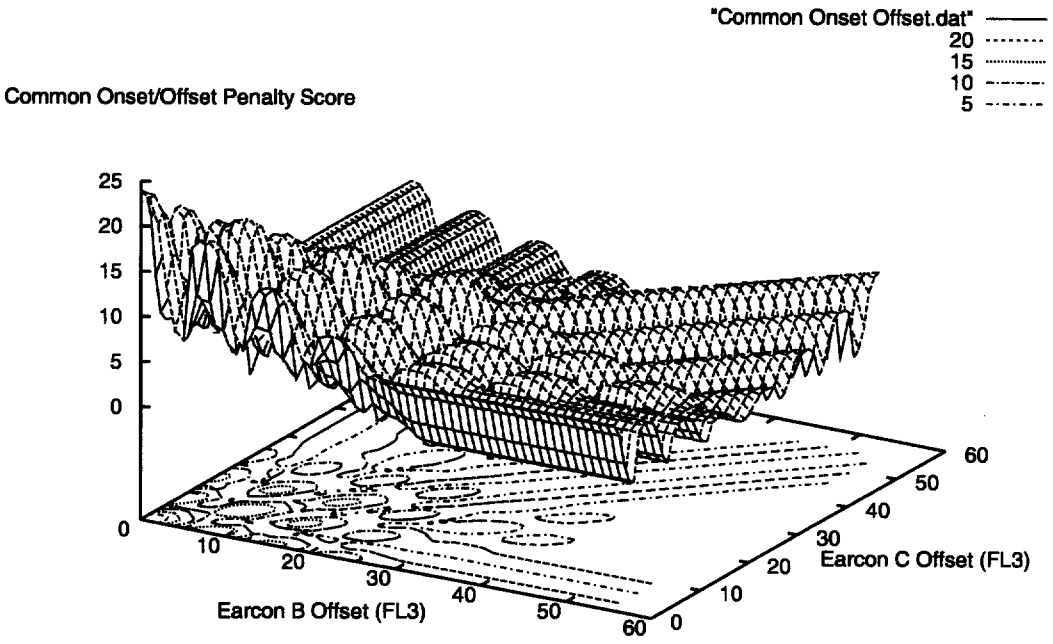


Figure 5.10: The search space for [SS8]. Each node is evaluated using only the common onset/offset algorithm.

In Figure 5.10, an example search space [SS8] is shown. [SS8] contains three identical earcons [FL3] containing four notes each, presented in an 8.0 second time interval. Earcon A is always fixed at offset 0, and the offsets of earcons B and C are varied along the x- and y-axes. The main diagonal ridge indicates the penalty from the two non-static earcons (earcons B and C) when their offsets are the same. In these cases, all four of the notes have common onsets and offsets between the two earcons. The ridges running parallel to the x- and y-axes indicate the penalty from the static earcon A and one of the other two earcons (corresponding to the earcon that is mapped to that axis). Of these ridges, the ridge closest to the axis is the highest because all the notes in the two earcons have common onset and offset. The next ridge inward corresponds to the case where the second note of B or C has common onset/offset with the first note of A. Similarly, the third note of B or C has common onset/offset with the second note of A, and the fourth note of B or C has common onset/offset with the third note of A. In all, there are three notes like this. The next ridge inwards indicates the case between A and B where two notes have common onset or offset. Each additional ridge indicates a conflict between A and B with one less note, and as such, the ridges get lower as they move away from the axis.

5.3.2 Harmonic Overlap

The harmonic overlap algorithm assigns penalties based on how many commonalities exist in the harmonic series of the overlapping notes of two earcons. The Pitch-per-Block representations of the *EarconSources* are used for this algorithm. The overlapping sections are aligned according to the current relative offsets. The overlapped region is then compared on a block by block basis. For each overlapping block, a penalty is incremented by an amount dependent upon the harmonic overlap of the pitches in the two blocks.

To assess a penalty for the two pitches in a given block, the fundamental frequencies of the two pitches are used. Let the frequency f_0 of the first pitch equal x, and f_0 of the second pitch equal y. Then, the harmonic series of x is $\{x, 2x, 3x, 4x, 5x, \ldots\}$ and the harmonic series of y is $\{y, 2y, 3y, 4y, 5y, \ldots\}$. A penalty for the current block is incremented for common values in both series. The amount of the penalty is weighted according to the distance between the common frequency and the closest fundamental frequency. This gives greater weight to common elements of the harmonic series that are closer to the fundamental frequency than to common elements that are further away from it. The largest penalty comes from two identical fundamental frequencies playing simultaneously. The next largest penalty comes from notes that are exactly one octave apart. In this case, the harmonics of the higher note correspond to every other harmonic of the lower note. The next worst case is that of two notes separated by an octave plus a perfect fifth. In this case, each harmonic of the higher note corresponds to every third harmonic of the lower note. These cases are illustrated in Figure 5.11

The common harmonics algorithm, in pseudo-code, is as follows:

```
// Note: EarconPPB refers to the Pitch-per-Block representation
//       of the given earcon.

HarmonicOverlap(EarconPPB A, EarconPPB B, Offset OffsetB)
                returns Integer
{
    Penalty = 0

    OffA = 0      // A starts first
    OffB = OffsetB
```

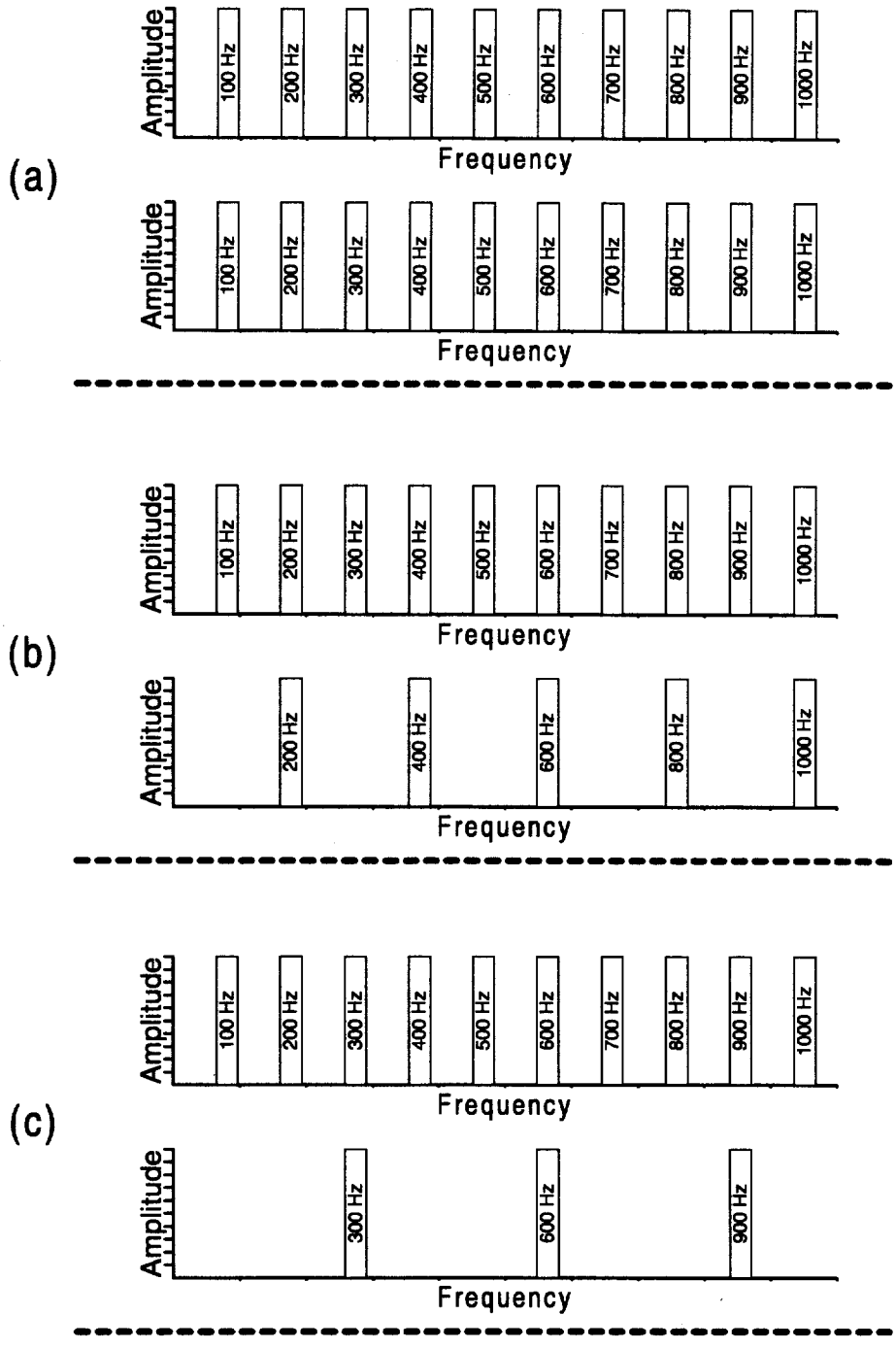


Figure 5.11: (a) The harmonics of two notes both with fundamental frequencies of 100Hz. (b) The harmonics of two notes with fundamental frequencies of 100Hz and 200Hz. (c) The harmonics of two notes with fundamental frequencies of 100Hz and 300Hz.

```
    For (i = 1 to Length of Overlap)
    {
        Integer Pitch_of_A = A[OffA]
        Integer Pitch_of_B = B[OffB]
        Penalty = Penalty + HarmonicPenalty(Pitch_of_A, Pitch_of_B)
    }
    Return Penalty
}

// Helper function for the HarmonicOverlap algorithm

HarmonicPenalty(Integer P1, Integer P2) returns Integer
{
    Penalty = 0

    LowerPitch = Min(P1, P2)
    HigherPitch = Max(P1, P2)

    For (Each Harmonic hLower in LowerPitch)
    {
        // hHigher is set to the harmonic in HigherPitch
        // that is in common with hLower.  If there is no
        // harmonic in common, hHigher is set to -1.
        hHigher = HarmonicPresentIn(HigherPitch, hLower)
        If (hHigher > -1)  // There is a common harmonic
        {
            h = Min(hHigher, hLower)
            Penalty = Penalty + e**(-h/4)
        }
    }
    Return Penalty
}
```

The function $e^{-h/4}$ is used as the penalty function, and is illustrated in Figure 5.12. This function approaches zero in a way such that only common harmonics that are less than approximately the 20^{th} harmonic of either pitch significantly contribute to the overall penalty value. Commonalities in harmonics beyond this point are ignored. Musical tones can certainly contain more than 20 harmonics, but the relative amplitudes of the higher harmonics generally are very small. Consequently, these harmonics do not affect the perception of the tone as significantly as the lower harmonics. The exponential decay of the penalty function reflects the relation-

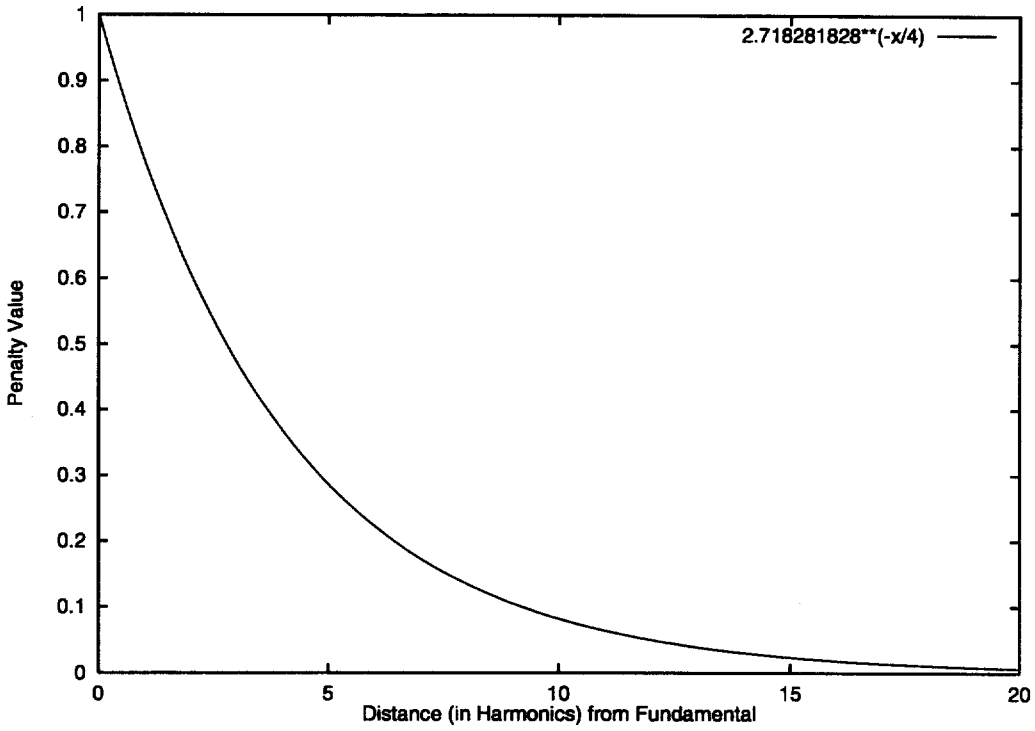


Figure 5.12: The penalty function used for common harmonics between overlapping pitches of two earcons.

ship between common harmonic number and perceptual weight. As the frequency of the common harmonic increases (away from the closer fundamental frequency), the contribution to the overall penalty computation exponentially decreases.

The actual algorithm is optimized to look at intervals in which there is no change of either note. These intervals often run over a number of blocks and as such some computation is saved since the penalty for each block in the interval will be constant (since `LowerPitch` and `HigherPitch` remain the same) and need not be recomputed.

An example search space [SS9] is displayed in Figure 5.13. [SS9] contains three identical earcons [FL2] containing four notes each, presented in an 4.0 second time interval. As usual, earcon A is always fixed at offset 0, and the offsets of earcons B and C are varied along the x- and y-axes. The structure of the search space is very regular. The highest penalty occurs when all three earcons have an offset of 0. Since there are three earcons playing the identical notes at the same time, all harmonics are in common between all earcons. The main diagonal ridge occurs because earcons B and C are scheduled with the same offsets. (See Figure 5.14a.)

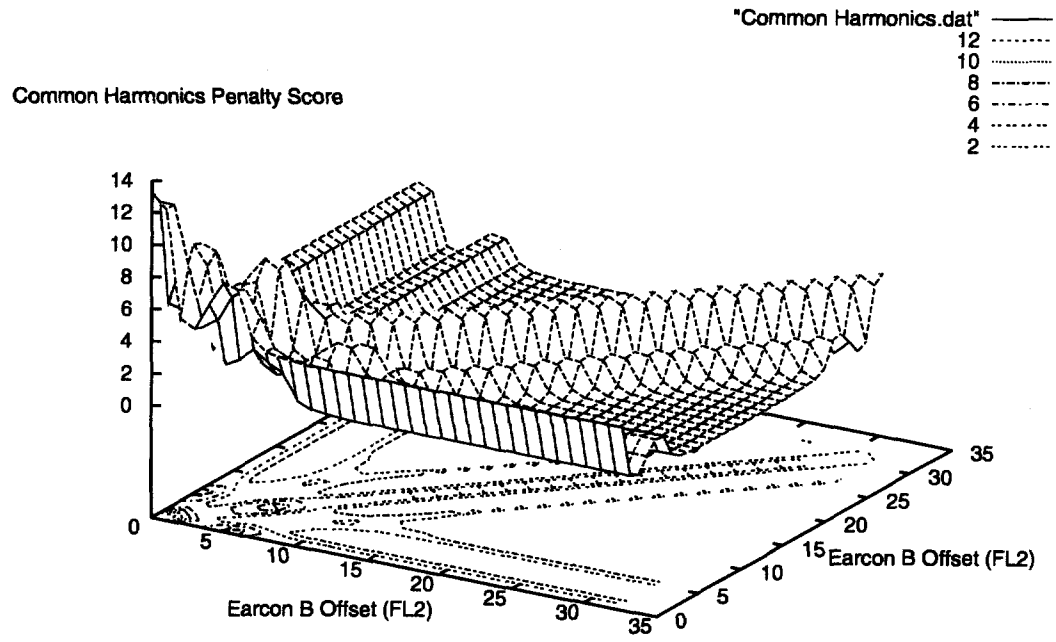


Figure 5.13: The search space [SS9] for a sound presentation. Each node is evaluated using only the common harmonics algorithm.

The main ridges parallel to the x- and y-axes occur because earcon A (always scheduled at offset 0) is interfering with either earcon C (also always at offset 0 for the x-axis ridge) or with earcon B (always at offset 0 for the y-axis ridge). The lower secondary ridges running parallel to the main ridges occur when the difference in offsets between two earcons is around 3. In this case, the second half of one earcon "GA" is overlapped with the first half of the other "GA" as in Figure 5.14b.

5.3.3 Timbre Similarity

Each *EarconSource* instance has a member that represents its timbre. In laying out a number of temporally overlapping earcons, a penalty is assessed proportionally to the similarity of the timbres and the length of the particular overlap. Timbre similarity values range from 0 to 1, where 0 indicates extremely dissimilar timbres, and 1 represents identical timbres. The timbre similarity value is then multiplied by the length of time that overlap occurs, to attain the penalty. For example, if earcon A and earcon B both use the *flute* timbre, and there is a 0.5 second overlap, then the penalty is $1.0 * 0.5$, for a final score of 0.5. If the same earcons were arranged such that there was a 1.5 second overlap, then the penalty would be

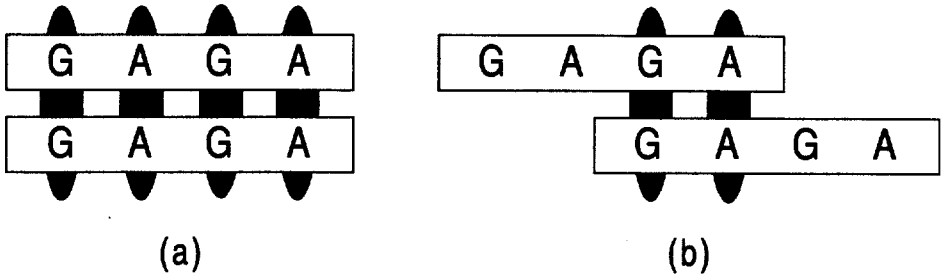


Figure 5.14: (a) All harmonics from all four notes are in common when these two earcons are played with the same relative offset. (b) All harmonics from two notes are in common when these two earcons are played with a particular staggered relative offset.

$1.0 * 1.5$, for a penalty score of 1.5. If earcon A uses a *flute* timbre and earcon B uses a *snare drum* timbre, the timbre similarity score might be 0.2. If there was a 1.5 second overlap between earcons A and B, the penalty would be $0.2 * 1.5$, or 0.3.

The question then remains, how can the similarity between two timbres be quantitatively measured? This problem has received much attention in recent years. Timbre is a perceptual quality that is exceedingly difficult to quantify. Grey has done considerable research on mapping timbres into multi-dimensional spaces, in which each axis in space represents a particular physical quality about the sound [Gre75]. One can then ascertain the distance between timbres. Those timbres that reside nearer to each other are perceptually more similar than those that are further apart in the space.

The approach implemented in the Comprehensive Auditory Scene Synthesizer is simpler. Although the algorithm is not as robust as others, it appears to function well at mapping timbre similarity to a single dimensional value, and it requires relatively little computation. The basis of this algorithm is that timbres with similar harmonic content tend to stream together as supported by the principle of spectral constancy [Bre90, pp. 98–103]. For example, the harmonics of three timbres are displayed in Figure 5.15. The top two have similar harmonic content; the bottom timbre has significantly different harmonic content. It would be expected that, in the absence of other conflicting cues, the stream segregation between the top two timbres would be poor. The stream segregation between the 1^{st} and 3^{rd} timbres, or the 2^{nd} and 3^{rd} timbres, would be stronger.

The approach in the implementation is to compare the frequency domain representations of all timbres in relation to all other timbres. At a given frequency, two timbres consist of a series of harmonics, and some inharmonic frequency compo-

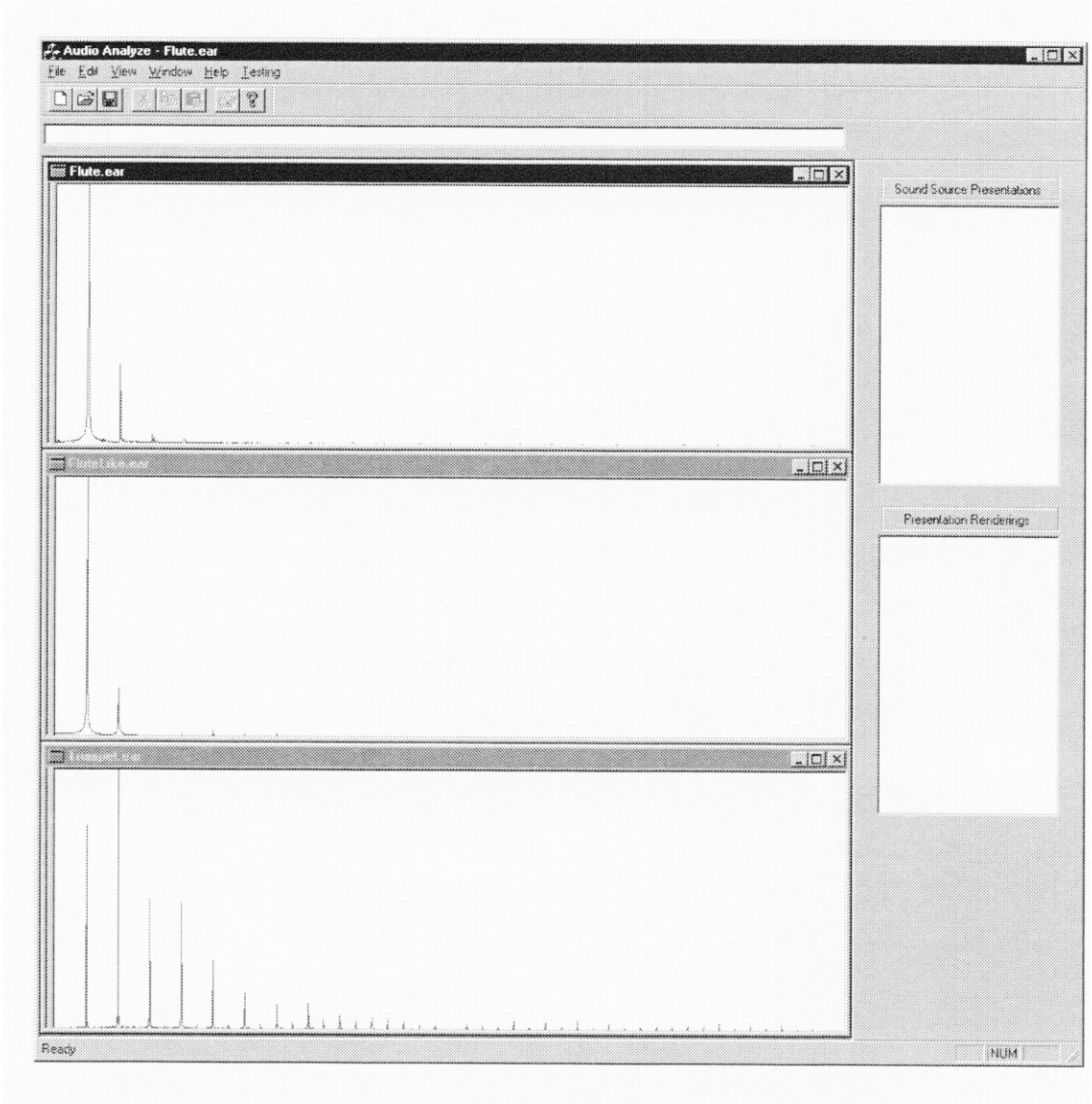


Figure 5.15: The harmonic series of three timbres.

nents as well. By computing the "difference" between the frequency components, one can arrive at an approximate "similarity" value between the two timbres.

In the implementation of this algorithm, a number of details needed to be worked out. First of all, what window of time in the timbre will be considered for comparing timbres? Since most of the notes comprising earcons are relatively short, the initial onset properties of a timbre will most likely have more importance in its perception than the sustaining section, since the sustain section of any given note is typically relatively short for earcons. The section of the timbre considered in the frequency domain is the first 0.2 seconds of the time domain representation at a fundamental frequency of 440Hz.

Secondly, since all of the timbres used in the implementation have a strong pitch attribute, there is not a great deal of inharmonic content to be considered in the frequency domain. Only harmonic frequencies were considered, up to the 50^{th} harmonic. The amplitudes of the 50 harmonics are compared as follows: first, the logarithmic value (in base e) is taken of each harmonic in each timbre. This makes a relative difference in amplitude (between the same harmonic of two timbres) more meaningful if one of the amplitudes is small. This is motivated by the reasoning that if a harmonic is present in only one of the timbres, it contributes more significantly to the timbre difference than if the harmonic were present in both timbres, but with somewhat different amplitudes. For example, If the 3^{rd} harmonic of timbre A has an amplitude of 0, and the 3^{rd} harmonic of timbre B has an amplitude of 2000, the difference is 2000. The difference in the log values of 0 and 2000 is about 7.5. (See Figure 5.16). However, if the relative amplitude difference of the 3^{rd} harmonics of A and B is still 2000, but derived from amplitudes such that $A = 6000$ and $B = 8000$, then the difference of the log values of 6000 and 8000 is about 0.5.

The log values of the first 50 harmonics are compared and the root mean squared values are summed. Let A_i and B_i represent the amplitudes of the i^{th} harmonic in the two timbres. Then,

$$timbreDiff = \sum_{i=0}^{49} \sqrt{\log(A_i)^2 - \log(B_i)^2} \tag{5.1}$$

The third complication with implementation of the timbre comparison algorithm is that different timbres may have different amounts of energy in them. A timbre is stored as a digital recording of a note. Therefore, a timbre created from a loud note will tend to have harmonics with greater amplitudes than a timbre created from a quieter sound. Even if these two timbres are identical, they will have a large *timbreDiff* value because of the large difference between the harmonics due to one timbre having more energy than the other. It is actually the shapes of the

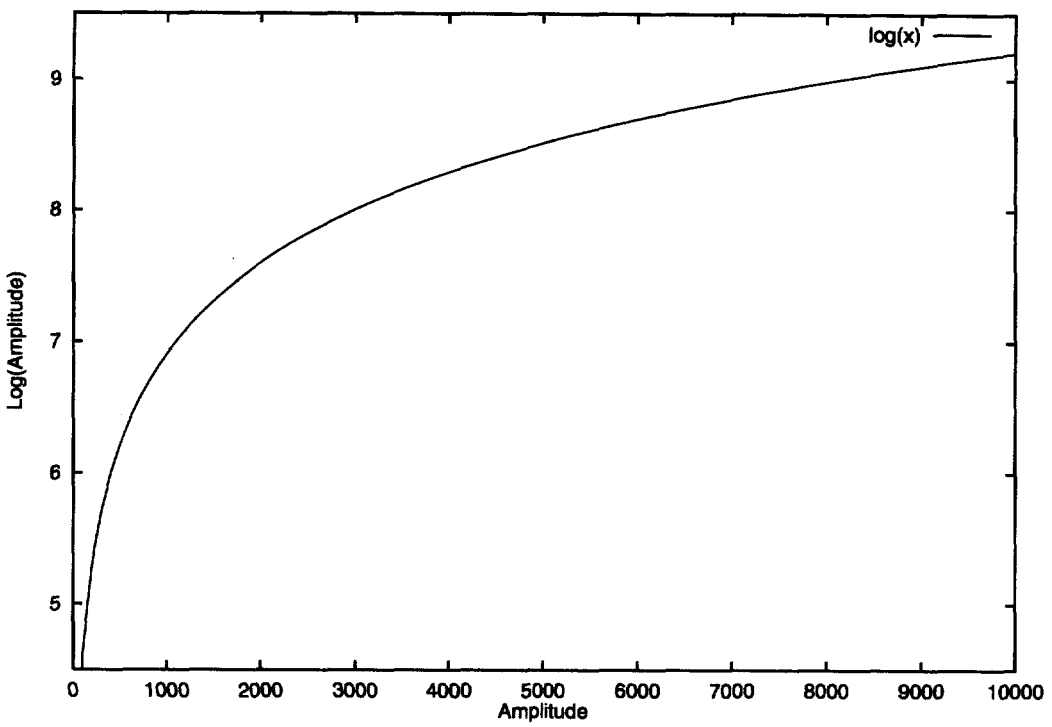


Figure 5.16: The relation between amplitude value and log value.

spectra that should be compared. To accomplish this, the process of computing Equation 5.1 is repeated but with a different constant k until a minimal value of *timbreDiff* value is found.

$$timbreDiff = \sum_{i=0}^{49} \sqrt{\log(A_i)^2 - \log(kB_i)^2}$$

Finally, the minimal value of *timbreDiff* is the actual value used. It is then put into the equation

$$timbrePenalty = overlap * e^{-timbreDiff}$$

where *overlap* is the number of blocks in which both of the timbres are present. The term $e^{-timbreDiff}$ normalizes the timbre difference penalty to a value between 0 and 1, as shown in Figure 5.17. A timbre difference value of 0 returns 1 for the timbre penalty value, and greater values of the timbre difference return exponentially smaller values. This function weights the conversion to penalty values such that the difference between a timbreDiff value of 0 and 1 is perceptually more significant than the difference between a *timbreDiff* value of 4 and 5.

The architecture for storing the timbre information is such that it will dynamically store as many new timbres as are introduced. When a new earcon is registered with the system, its timbre is identified. If the system has previously analyzed this timbre, then no further work needs to be done, and the earcon can be used immediately. If the timbre has not yet been analyzed, then during the earcon registration process, the new timbre is compared against each other timbre currently registered with the system. The resulting *timbreDiff* values are stored in a dynamically growing table, as shown in Table 5.4. The main diagonal values of the table are always 0 since there is no timbre difference for identical timbres. The table is also symmetric because all timbres are listed along the rows and columns. Therefore, only values above the main diagonal need be computed.

Since the timbre computations are performed during the earcon registration process (and not during the run-time decision making period), the timbre analysis algorithm can be replaced with a more computationally intensive method if it performed with better end results.

Figure 5.18 shows the search space [SS10] for three earcons, each with a unique timbre. The earcon with a trumpet timbre is fixed at offset 0. The earcons with

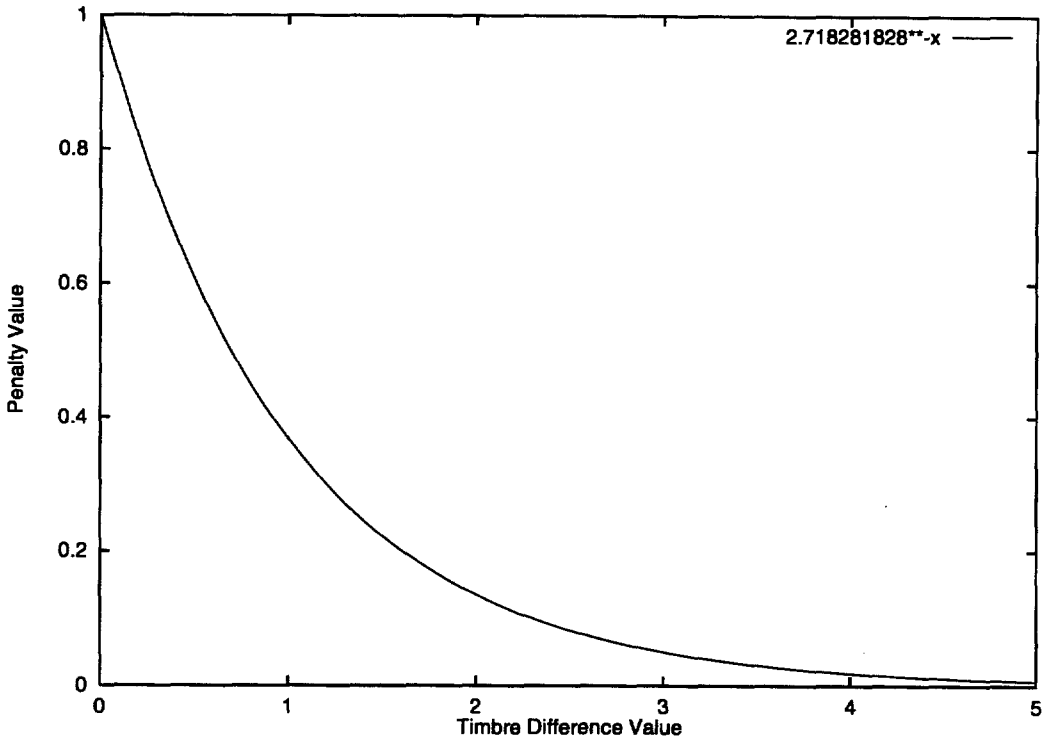


Figure 5.17: The negative exponential relation between the timbre difference value and the penalty value.

	Flute	Elec. Guitar
Flute	0.0	3.5
Elec. Guitar	3.5	0.0

	Flute	Elec. Guitar	Clarinet
Flute	0.0	3.5	1.2
Elec. Guitar	3.5	0.0	3.3
Clarinet	1.2	3.3	0.0

Table 5.4: The timbre table before a new timbre registration (top), and after a Clarinet timbre registration (bottom).

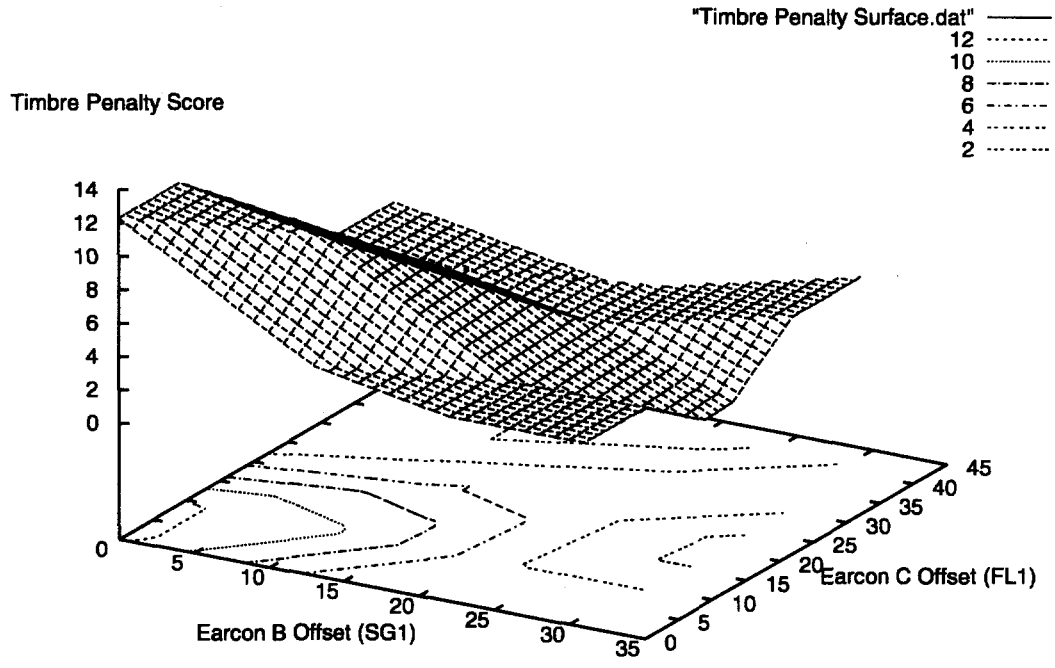


Figure 5.18: The search space [SS10] for a sound presentation containing three unique timbres. Each node is evaluated using only the timbre similarity algorithm.

steel guitar and flute timbres have varying offsets. The resulting search space has a somewhat linear downward sloped appearance from the origin to the furthest (x, y) values. This is primarily caused by the fact that the timbre penalties decrease linearly with decreased overlap periods of earcons. The deviations from a smooth surface are caused by the presence of earcon A that is fixed at offset 0. As more earcons are added to a sound presentation, the search space becomes increasingly irregular.

5.3.4 Pitch Crossings

The pitch crossing heuristic counts the number of times the pitches of two earcons cross each other. This is illustrated in Figure 5.19. The penalty for pitch crossings arises because it becomes unclear whether the two streams cross and continue or just meet and then diverge again. An example of this ambiguity is shown in Figure 5.20. This type of ambiguity is particularly avoided in the musical style of *counterpoint*, in which two or more melody lines are presented simultaneously to the listener [Fie92]. To a great extent, the same rationale applies here. By minimizing pitch crossing between concurrently playing earcons, the possibility of ambiguous auditory stream interpretations is also minimized.

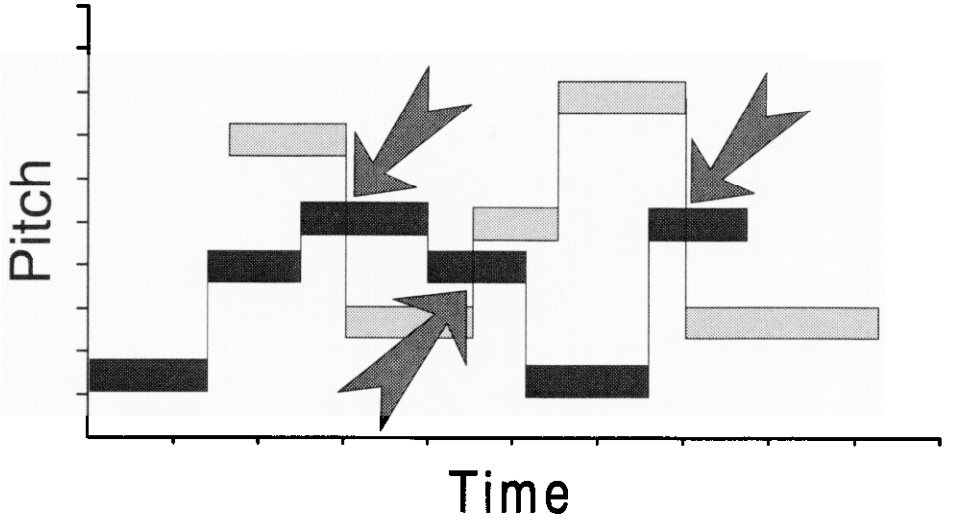


Figure 5.19: Two earcons that have three pitch crossings (indicated by the arrows).

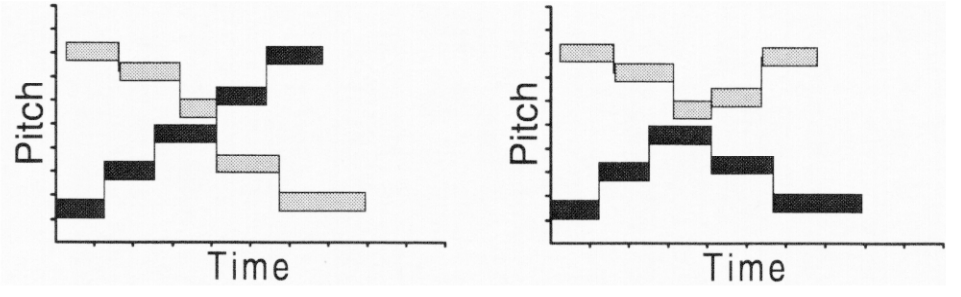


Figure 5.20: Two interpretations of the same notes from two earcons. Notes of the same shade are assigned to the same auditory stream. This type of ambiguity should be avoided if possible. The example is further complicated by the common onset/offset of the (possibly) pitch crossing third and forth notes.

An algorithm in pseudo-code that counts pitch crossing is relatively simple. The Pitch-per-Block representation is used, and each pitch crossing contributes a penalty value of 1. In the actual implementation of the pitch crossings, the <Pitch, Duration> 2-tuple representation is used. Since most notes span multiple blocks, pitch crossings can be checked by comparing the full interval of each note as opposed to each block in an earcon. This is a slightly more efficient method than the one presented in pseudo-code.

```
PitchCrossings(EarconPPB A, EarconPPB B, Offset OffsetB)
                returns Integer
{
    Penalty = 0

    // Set IndexA to the point in A where B begins
    IndexA = OffsetB
    IndexB = 0

    // Set OnTop to the earcon with the higher pitch
    Integer OnTop
    OnTop = GetHigherPitch(A, IndexA, B, Index B)

    // Loop through overlapped blocks looking for pitch crossings

    For (each block in the overlap region)
    {
        // Advance the block indices of the two earcons
        IndexA = IndexA + 1
        IndexB = IndexB + 1

        // See which earcon has the higher pitch now
        CurOnTop = GetHigherPitch(A, IndexA, B, Index B)

        // Has there been a pitch crossing since the
        // last blocks checked?
        if (CurOnTop != OnTop)  // Pitch crossing!
        {
            Penalty = Penalty + 1
            OnTop = CurOnTop  // Update OnTop
        }
    }
    Return Penalty
}
```

An example search space [SS11] containing the static earcon [TP5], as well as [SG3] and [SG4], is shown in Figure 5.21. In the surface plot, contour lines generally run parallel to the x- and y-axes. These indicate conflicts between the earcon fixed at offset 0, namely, [TP5], and the earcon along the axis perpendicular to the contour. In these cases, the earcon along the parallel axis has no significant additional penalty and so its position does not greatly affect the penalty scores when the other two earcons have a particular temporal relation. There are only two diagonal contour lines in the surface plot, each contributing only an additional value of 1. This indicates that there is little conflict between earcons B and C when they are in any particular relation to each other.

5.3.5 Closeness of Pitch Between Different Earcons

The Closeness-of-Pitch algorithm assigns a penalty for the overlapping section of two earcons based upon how close the pitches from the different sources are. If notes are, on average, distant in pitch for any given time in the overlapped interval, then they will more readily segregate into two separate streams. This streaming effect is further strengthened with the minimization of pitch crossings by the pitch crossing algorithm. Figure 5.22 shows two possible schedules for two earcons. Schedule (a) leads to closeness-of-pitch conflicts. Schedule (b) contains minimal conflicts.

The Closeness-of-Pitch algorithm uses the Pitch-per-Block data representation. The penalty is computed by iterating through all overlapping blocks and incrementing the penalty according to a function of the closeness of the two pitches. The pseudo-code is as follows:

```
ClosenessOfPitch(EarconPPB A, EarconPPB B, Integer OffsetB)
                 returns Integer
{
    Penalty = 0

    for (Index = next block in the overlap region)
    {
        // Store the difference in pitch in p.
        p = Abs(A[Index] - B[Index+Offset]
        Penalty = Penalty + -1 / (1 + e**(-0.8*p+5)) + 1
    }
    Return Penalty
}
```

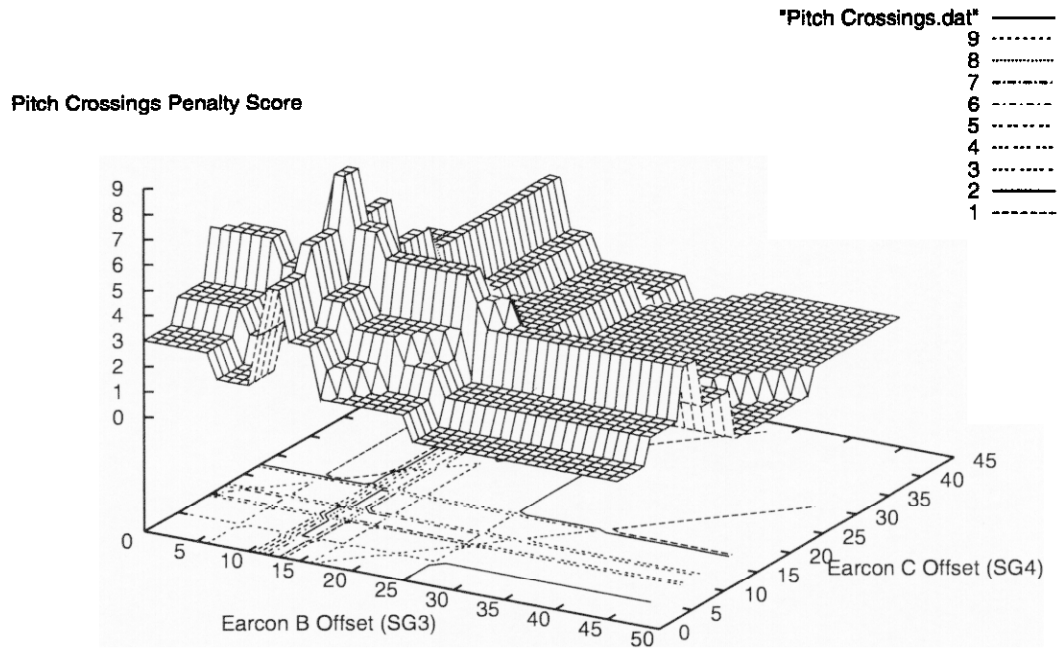


Figure 5.21: The search space [SS11] for a sound presentation. Each node is evaluated using only the pitch crossings detection algorithm.

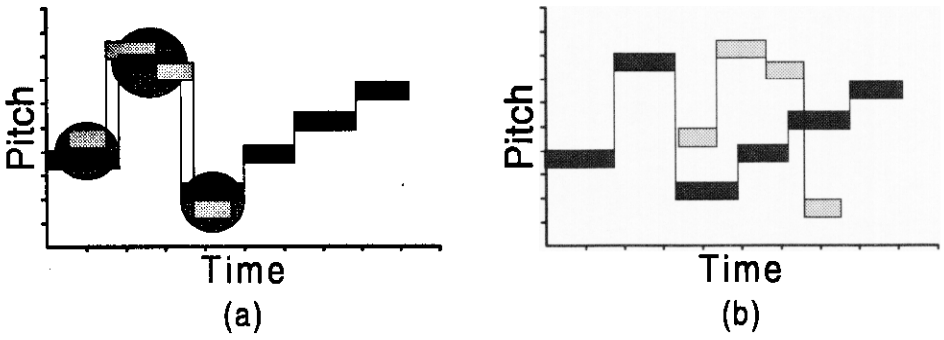


Figure 5.22: (a) The arrangement of two earcons causes three Closeness-of-Pitch conflicts, indicated with the circles. (b) This alternate arrangement of the same two earcons has no Closeness-of-Pitch conflicts.

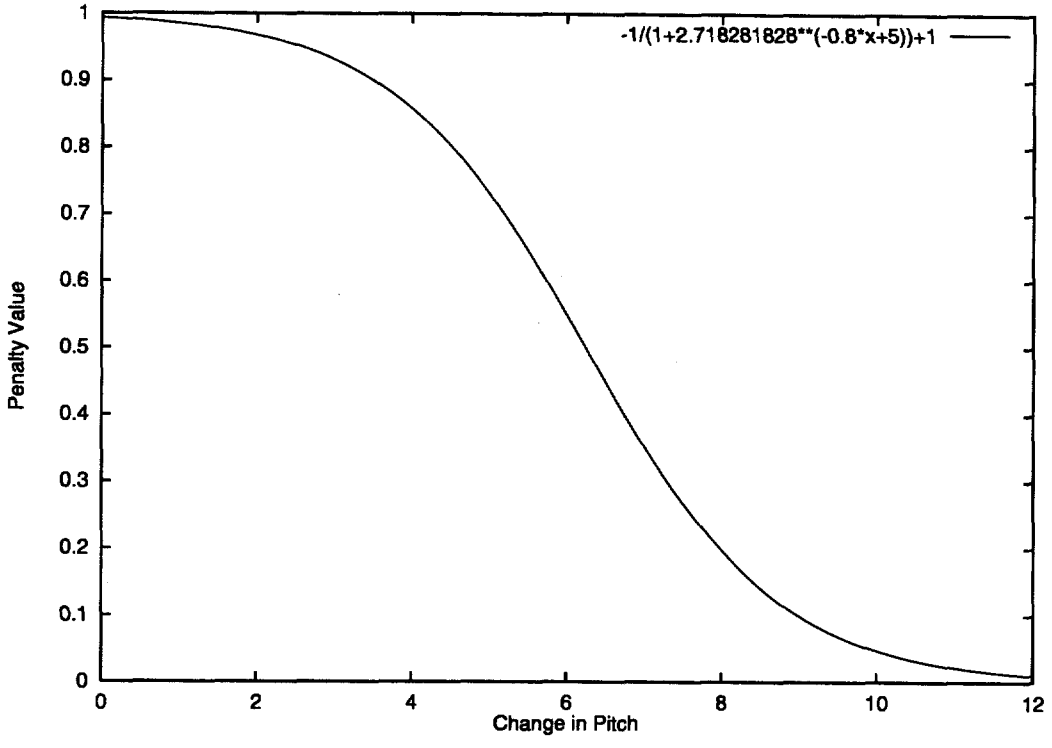


Figure 5.23: The penalty function for the Closeness-of-Pitch algorithm.

The penalty increment sigmoid function $-1/((1 + e^{-0.8*p+5}) + 1$ is shown in Figure 5.23.

An example search space [SS12] computed from only the Closeness-of-Pitch algorithm is shown in Figure 5.24. The earcons used are [SA1] (the static earcon fixed at offset 0), [BA1], and [SG2], all presented in a 5 second time interval. For most values of y, the value of x does not have a significant impact on the value of that (x, y) location. This is why most of the contour lines run parallel to the x-axis. This fact indicates that most of the penalty at each node comes from a conflict between the static earcon [SA1] and the earcon [SG2] mapped to the y-axis. The largest penalty in the search space occurs at the line $y = 8$, and since the values on this line are roughly equal, the conflict arises between the y-axis earcon [SG2] and the static earcon [SA1]. [SA1] begins with a low note that is then followed by a sequence of higher notes. These higher notes are close in pitch to the notes of [SG2]. The main conflict occurs then when [SA1] is at offset 0 and [SG2] is at offset 8. [BA1] has little contribution to the penalty because it consists of notes somewhat lower than the notes in [SA1] and [SG2]. The only significant penalty from [BA1] comes when the highest note of [BA1] (C3) plays concurrently with the lowest notes of [SA1] (F3) and [SG2] (G3). This occurs when all three earcons

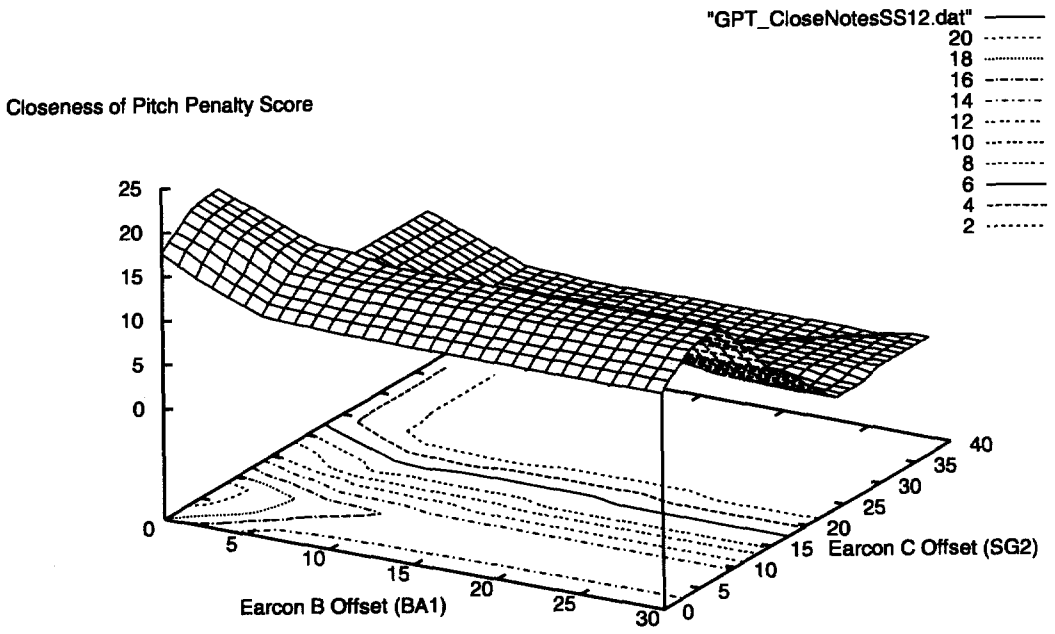


Figure 5.24: The search space [SS12] for a sound presentation. Each node is evaluated using only the Closeness-of-Pitch algorithm.

have the same relative offset. Since [SA1] is fixed at offset 0, the penalty will occur when the offsets of [BA1] and [SG2] are about equal and close to 0. In Figure 5.24, the penalty is seen as a slightly raised region in the area of $x < 7$.

5.3.6 Common Note Change Patterns

Parallel movement of pitches has long been used in music composition to group together various instrument parts of a composition. A prime example of such a work is Ravel's *Bolero*. When parallel movement of pitches is combined with common onset and offset of notes, the result is a strong fusion between the two (or more) musical fragments.

The grouping effect of common note change patterns between earcons must therefore be minimized in the Computational Auditory Scene Synthesizer. Figure 5.25 shows an example of the type of case that should be prevented.

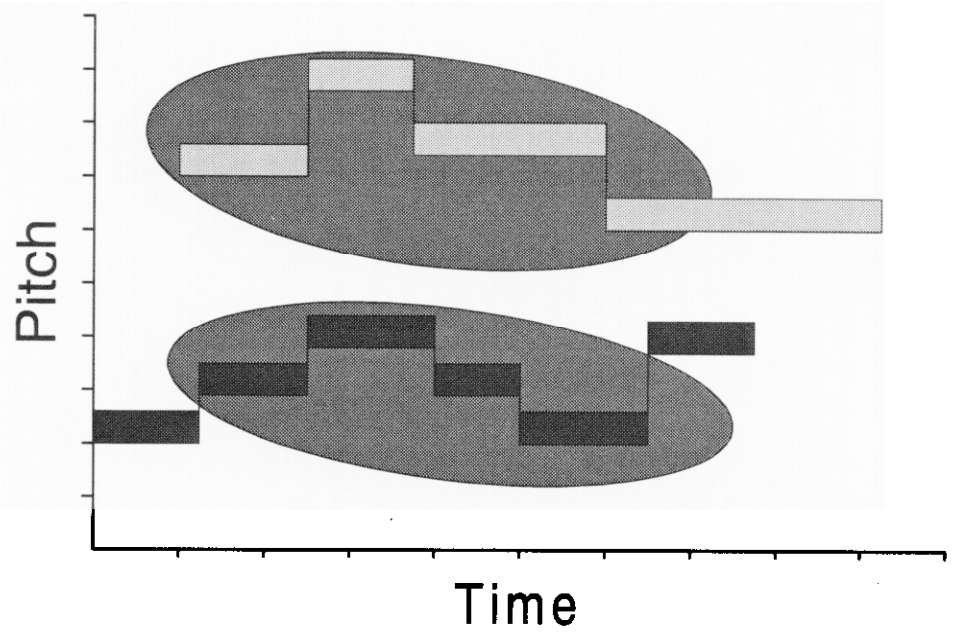
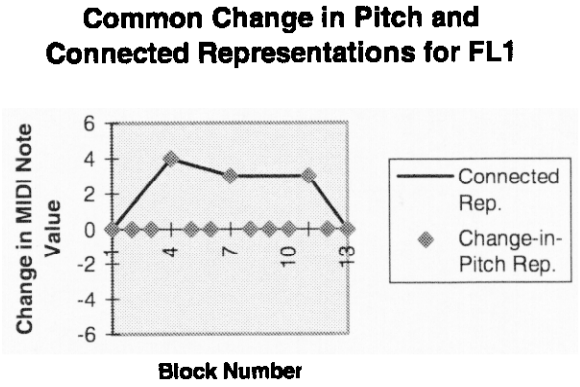


Figure 5.25: An example of a common note change pattern between two earcons is highlighted.

The detection of these types of patterns is not as straightforward as many of the other algorithms discussed. There are varying degrees to which the patterns could be described as "common." The definition is hard to quantify. Certain cases, at least, are obvious. The presence of two note patterns changing in identical ways warrants a penalty. A downward moving note pattern and upward moving note pattern occurring simultaneously should cause no penalty in this algorithm. Similar (but not exact) changes occurring simultaneously should result in a penalty but not as much as if the changes were exact. Finally, similar changes that occur slightly out of phase from each other should result in a penalty, but less so than for similar changes occurring simultaneously.

The approach taken to solve this problem is as follows:

- Start with the Change-in-Pitch-per-Block representations of the two earcons.
- Apply a method to Connect-the-Spikes of each Change-in-Pitch-per-Block representation.
- Compute a penalty based on the similarity of the two Connect-the-Spikes representations by comparing the values of the points at every overlapped block. Essentially, similarly shaped overlapping regions are penalized.



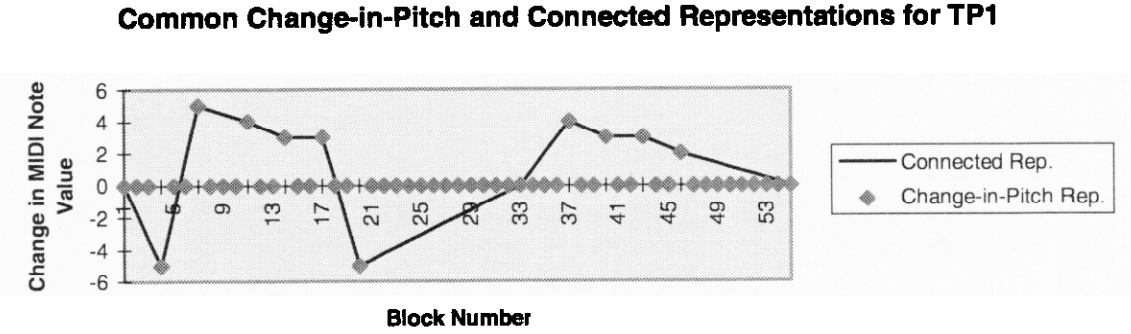


Figure 5.26: The Change-in-Pitch-per-Block and Connect-the-Spikes representations for both the [FL1] and [TP1] earcons.

The new Connect-the-Spikes representation is a modification of the Change-in-Pitch-per-Block representation. A line is drawn between each two consecutive non-zero values in the Pitch-per-Block representation. The only exceptions to this are the first line, drawn from 0 to the first non-zero value, and the last line, drawn from the last non-zero value to the last 0 value. Figure 5.26 shows examples of the Change-in-Pitch-per-Block and Connect-the-Spikes representations for the earcons [TP1] and [FL1].

The algorithm for the computation of the penalty is presented in pseudo-code.

```
// This algorithm uses the Connect-the-Spikes Representation of
// the two earcons A and B.

CommonNoteChangePatterns(EarconConRep A, EarconConRep B,
                         Offset OffsetB) returns Integer
{
    Penalty = 0

    // Iterate over all blocks in the overlap region between
    // earcons A and B.  B starts OffsetB blocks after A starts.

    for (i = all blocks in the overlap region)
    {
        Double Smaller   = MIN(A[i+OffsetB], B[i])
        Double Larger    = MAX(A[i+OffsetB], B[i])
        Double Dist      = Larger - Smaller

        NewPenalty = e**(-Dist)

        // Increment Penalty with the penalty value for
        // this block comparison iteration
        Penalty = Penalty + NewPenalty
    }
    Return Penalty
}
```

The penalty function works well in practice to return a value of between 0 and 1 for each overlapping block. The penalty value rapidly decreases as the distance between the values in the Connect-the-Spikes representation increases.

The Connect-the-Spikes representation of [FL1] appears similar to two subsections of the Connect-the-Spikes representation of [TP1], starting (in the [TP1] graph) at offsets 5 and 32 (See Figure 5.26). These similarities are depicted in the surface plot of the search space example in Figure 5.27. This surface plot is of the search space created by fixing [TP1] at offset 0, and varying the offsets for [FL1] along the x-axis and [FL1] along the y-axis. The time interval used was 6 seconds. Since the same earcon is mapped to both the x- and y-axes, the surface is symmetric about the line $y = x$.

Only the Common-Note-Change-Patterns algorithm was used to compute the surface. The ridge running diagonal to the x- and y-axes along the line $y = x$ is a result of [FL1] and [FL1] having the same offset. This means their pitch changes

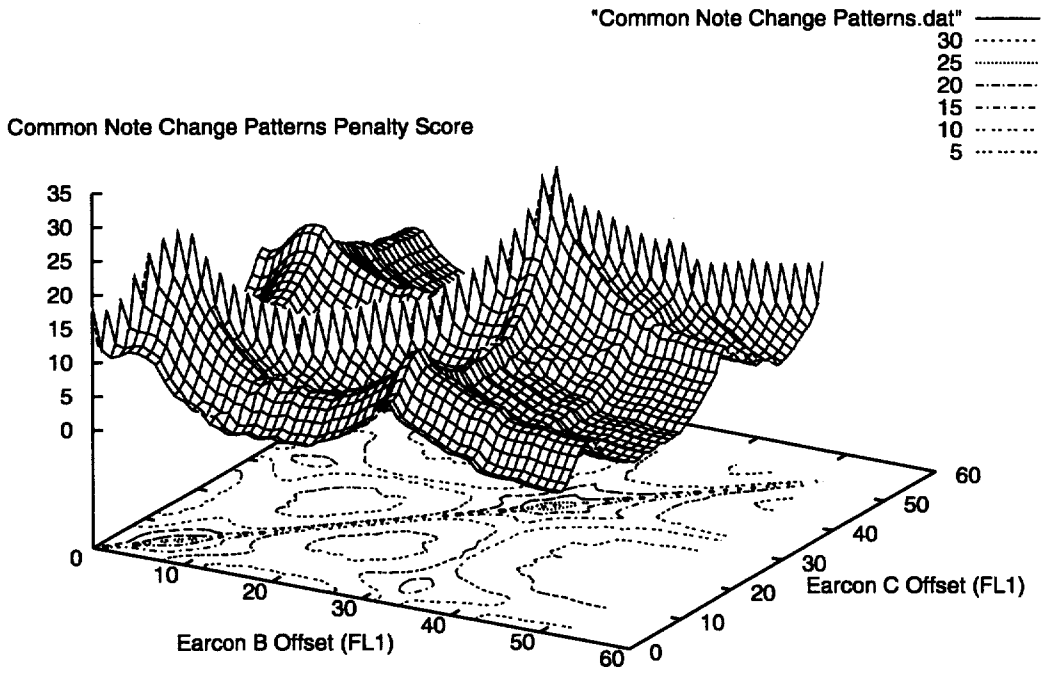


Figure 5.27: An example search space [SS13] created with the Common-Note-Change-Patterns algorithm.

coincide exactly, causing a significant penalty. There are two somewhat smaller ridges running parallel to the x-axis, and two smaller ridges running parallel to the y-axis. These ridges are present because of the penalty derived from aligning one of the [FL1] earcons with [TP1] . When earcon B ([FL1]) is at offset 5 or 32, a ridge is formed parallel to the x-axis, indicating a penalty associated with earcons A and B. Similarly, when earcon C ([FL1]) is at offset 5 or 32, a ridge is formed parallel to the y-axis indicating a penalty between earcons A and C. The high peaks at (5,5) and (32,32) represents the points at which all three earcons have a simultaneous and similar note change patterns.

5.3.7 Timbre and Pitch Similarity Between Consecutive Earcons

The Timbre Similarity between Consecutive Earcons penalty occurs when the timbre of the last note of an earcon A is similar to the timbre of the first note of another earcon B, and when A ends approximately when B starts. This situation introduces an ambiguity to the listener as to whether A and B are separate auditory events, or if A followed by B is actually one event. A similar situation arises when the pitch of the last note of A is close to the pitch of the first note of B, and again when A ends approximately when B starts. The combination of these two situations compounds the ambiguity. The following pseudo-code outlines the implemented method to assign a penalty based upon these situations.

```
// Note: A starts first.  B starts OffsetB blocks after A starts.

SeqPenalty(Earcon A, Earcon B, Offset OffsetB)
           returns Integer, Integer
{
    PitchPenalty = 0
    TimbrePenalty = 0

    EndOfA = Length(A)
    StartOfB = OffsetB
    Dist = ABS(EndOfA - StartOfB)

    LastNoteOfA = Last note of A
    FirstNoteofB = First note of B

    DiffPitch = ABS(LastNoteOfA - FirstNoteOfB)
```

```
    // Look up the timbre similarity value in the timbre table
    DiffTimbre = GetTimbreTableVal(A, B)

    PitchPenalty = PitchPenalty +
                   e**(-Dist/4) * e**((-DiffPitch/(12/4))/4)
    TimbrePenalty = TimbrePenalty + e**(-Dist/4) * e**-DiffTimbre

    Return PitchPenalty, TimbrePenalty
}
```

The multiplication of the two exponential functions balances the penalty values according to both the pitch or timbre similarity, and the distance between end and start times of the two earcons.

The exclusion tables hold penalty values for overlapping regions of pairs of earcons. Since the Timbre and Pitch Similarity between Consecutive Earcons penalty can occur when there is little or no overlap between two earcons, the values are not stored in the exclusion tables. These penalty values are actually computed "on the fly" as needed for a given arrangement of sounds, since they require only a relatively simple calculation. However, the sequential penalty values could be stored in a similar structure to the exclusion tables. Performance would increase slightly, at the cost of added memory space.

5.3.8 Penalty Value Computation for the Exclusion Tables

The values entered into the exclusion tables are calculated from the return values of the penalty algorithms for all overlaps of all earcons in a sound presentation. An exclusion table exists for each unique earcon in the presentation. It stores values that represent penalties associated with overlapping any earcon in the presentation with itself, at any non-negative relative offset, as shown in Table 5.2. To compute an exclusion table entry for earcon A that represents the penalty of playing earcon B concurrently with A where B has a relative offset to A of n blocks, the function *ComputeEntry*(A, B, n) is called.

The *ComputeEntry* function calls each of the penalty algorithms that operate on an overlapping range of two earcons and computes a weighted sum of their returned penalty values for the particular overlapped arrangement of A and B. Each overlap penalty algorithm returns a value less than the number of blocks in the overlapping region common to A and B, because each algorithm cannot assign a

penalty greater than 1 for each overlapping block. The penalty value $P_{A,B,o}$ represents the exclusion table penalty value entry for earcon A when earcon B begins playing o blocks after A begins. It is computed as follows:

$$P_{A,B,o} = w_1 * A_1(A,B,o) + w_2 * A_2(A,B,o) + w_3 * A_3(A,B,o) + \cdots + w_7 * A_7(A,B,o)$$

where

- $P_{A,B,o}$ is the weighted, summed penalty value for playing A at relative offset 0 and playing B at relative offset o ($o <$ Length(A), guaranteeing an overlap between A and B of at least 1 block)[3],
- w_n is the weight assigned to $A_n(A,B,o)$,
- $A_1(A,B,o)$ is the Common Onset Penalty,
- $A_2(A,B,o)$ is the Common Offset Penalty,
- $A_3(A,B,o)$ is the Harmonic Overlap Penalty,
- $A_4(A,B,o)$ is the Timbre Similarity Penalty,
- $A_5(A,B,o)$ is the Pitch Crossings Penalty,
- $A_6(A,B,o)$ is the Closeness of Pitch Penalty, and
- $A_7(A,B,o)$ is the Common Note Change Patterns Penalty.

The weights have been set through the examination of numerous example exclusion tables for sets of earcons. By inspection, the salient features between earcons are captured using these weights.

The computation of the exclusion tables happens before the search space is examined. It is done "off-line" as a optimized preprocessing step to allow the exploration of the search space to occur much faster.

[3]This penalty arises only because of the temporally overlapped region common to both A and B.

5.4 Heuristic Search Method

In the preceding sections, a number of different algorithms were described to compute penalties associated with relative temporal positions of two earcons. As the search space of a given sound presentation is explored, the current node being examined must have a score computed for it. This score should represent the overall perceptibility of the full auditory presentation that corresponds to the given node. The exclusion tables represent the penalties associated with any two earcons. However, each node in an n-dimensional search space consists of n earcons, where $n \geq 2$. Therefore, the method to compute the score of a given node containing more than two earcons is as follows:

- Consider each pair of earcons (A, B) in the presentation.
  - If the offset of B is greater than or equal to the offset of A, and if the relative schedules of A and B are such that there is at least one block of temporal overlap between them, then
    - Look in the exclusion table of A for the row corresponding to B, and get the n^{th} value of the row, where n is the offset of B relative to A.
    - Increment the score by this value.
  - Compute the sequential penalties for A and B regardless of the amount of overlap, and add this penalty to the node score.

If two earcons start at the same offset, the penalties between them should not be counted twice. This special case in accounted for in the actual implementation.

As an example, let the current node have three earcons:

- Earcon A is at offset 0 and is of length 8 blocks (It is scheduled from block 0 to 7)
- Earcon B is at offset 3 and is of length 3 blocks (It is scheduled from block 3 to 5)
- Earcon C is at offset 4 and is of length 7 blocks (It is scheduled from block 4 to 10)

This example is illustrated in Figure 5.28. The steps to compute the node score are as follows:

- For earcon A:
  - Go to the row corresponding to B in exclusion table A and get the value stored at location 3.[4] (The 1^{st} block of B starts playing at the same time as the 4^{th} block of A) Add this value to the score of the node.
  - Go to the row corresponding to C in exclusion table A and get the value stored at location 4. (The 1^{st} block of C starts playing at the same time as the 5^{th} block of A) Add this value to the score of the node.
- For earcon B:
  - Ignore A since A starts before B.
  - Go to the row corresponding to C in exclusion table B and get the value stored at location 1. (The 1^{st} block of C starts playing at the same time as the 2^{nd} block of B) Add this value to the score of the node.
- For earcon C:
  - Ignore A since A starts before C.
  - Ignore B since B starts before C.
- Compute sequential penalties (if any) for A and B. Add this value to the score of the node.
- Compute sequential penalties (if any) for A and C. Add this value to the score of the node.
- Compute sequential penalties (if any) for B and C. Add this value to the score of the node.

An example search space [SS14] is shown in Figure 5.29. It consists of [TP1] fixed at offset 0 and [CE1] and [SG2] along the x- and y-axes, respectively. All of the node scores in this example have been computed using both the concurrent algorithm penalties in the exclusion tables and the "on the fly" computed sequential penalties. The irregularity of the surface is typical for the search spaces.

For very simple search spaces like in Figure 5.29, all node scores can be computed. The global minimum represents the schedule of sounds that contains the predicted best presentation with the fewest perceptual conflicts between sounds. The challenge of larger spaces with more sounds is that the entire search space cannot

[4]Note that the exclusion table values begin at 0. The value at location 3 is the 4^{th} value in the row.

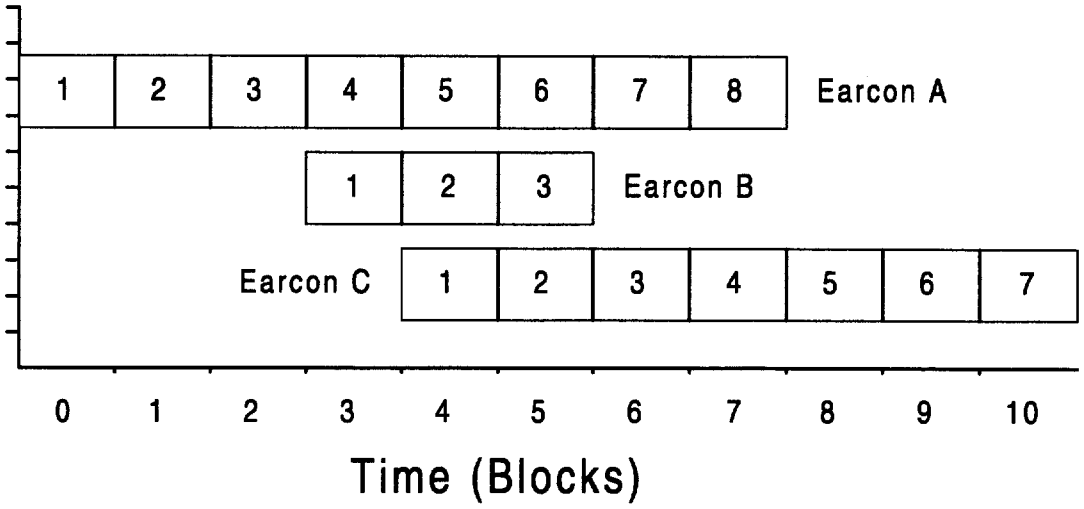


Figure 5.28: The schedule of three earcons in a node.

be examined because of the exponential growth of the scheduling possibilities. A heuristic search method is one way to find a low node score that is close to the global minimum, while only exploring a tiny fraction of the entire search space.

The exclusion table values are computed before beginning an examination of the search space. This allows the heuristic search algorithm run vastly faster, since most of the work of computing the score of a node in the search space is already done and stored in the exclusion tables. During the search, values are looked up in the tables rather than computed repeatedly from scratch. Since the sequential properties are less expensive to compute, the sequential ambiguity scores are not pre-computed and stored. Rather, the sequential penalty values are computed on demand during every evaluation of a node in the search space. Some computations are likely to be repeated unnecessarily, but the addition of the sequential penalty adds only a small constant amount of computation for each node evaluation of the concurrent penalty value.

The computational cost of evaluating the penalty score of a node is $O(n^3)$, where n is the number of earcons. This running time is derived from the examination of all pairs (A, B) of earcons in the presentation which is $(O(n^2)$ — and then for each pair, examine up to n rows of the exclusion table for A to find the row of entries corresponding to B. Then, a constant time reference is made to an element in that row. $O(n^2) * n = O(n^3)$. The sequential penalty calculation runs in $O(n^2)$ time and as such does not seriously impact the computational time as the problem sets grow large.

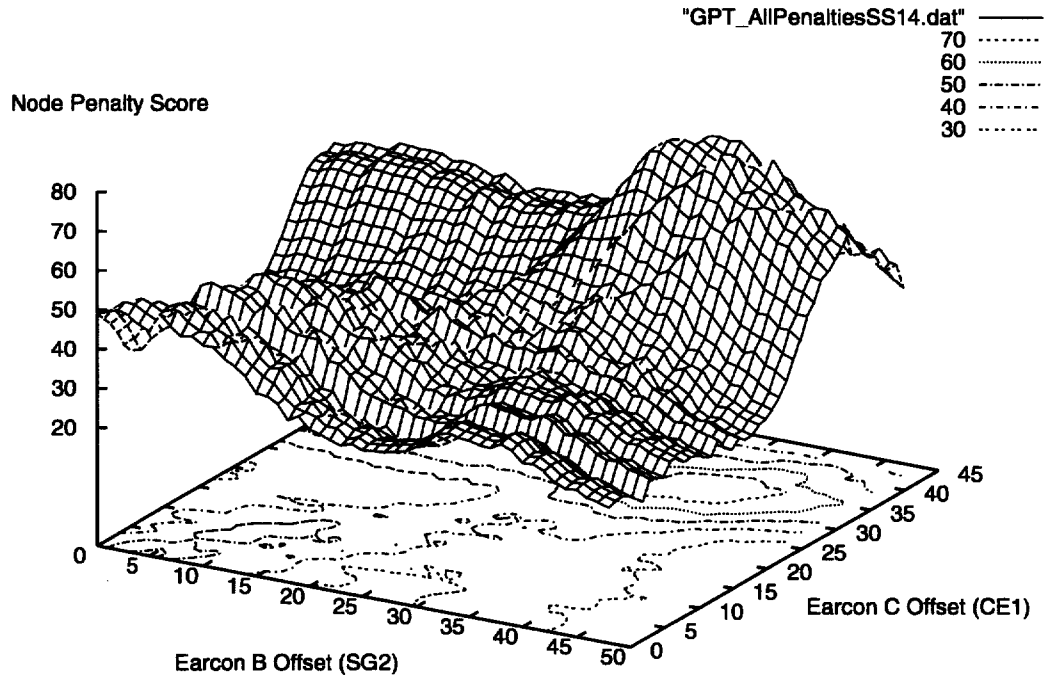


Figure 5.29: The search space [S14] computed using all the penalty algorithms.

Neither of these algorithm running times would matter if the entire search space needed to be examined, since the number of nodes in the search space increases exponentially with relation to n. For all but the most trivial problems, it is not feasible to search all possible nodes for the best answer. A heuristic search algorithm is needed to find a reasonably good answer quickly. Two techniques are implemented: *random exploration* and *hill climbing.*

In the random exploration technique, nodes are randomly examined until either k nodes have been examined (where k is specified by the user), or until the best found value does not improve for some number of additional nodes examined. This very simple technique finds reasonably good answers fairly quickly. This is most likely due to the very irregular nature of large search spaces. Some results from the random exploration technique will be compared against a slightly more sophisticated technique to be named *iterated hill climbing.*

Hill climbing is a heuristic search algorithm that finds a local maximum. Since the best answer in the search space is the global minimum, the hill climbing technique will be referred to as *hill rolling.* This algorithm is identical to hill climbing except that one comparison operator is changed. The hill rolling algorithm takes a search space and a starting location in that space as input, and returns a node in S that is a local minimum. It functions as follows:

```
HillRoll(SearchSpace S, Node StartPosition) returns Node
{
    Node CurrentNode = StartPosition
    Boolean AtLocalMinimum = FALSE

    While (AtLocalMinimum = FALSE)
    {
        Evaluate the score at CurrentNode
        Evaluate all neighboring nodes of CurrentNode

        If no neighboring nodes have a lower score
        {
            AtLocalMinimum = TRUE
        }
        Else
        {
            CurrentNode = Minimum(neighboring nodes)
        }
    }
    Return CurrentNode
}
```

This method works well for search spaces in which either only one local minimum exists or all local minima have approximately the same score. Since the algorithm terminates upon finding the first local minimum, the node score found will be the lowest (in the case of a search space with only one local minimum) or extremely close to the lowest (for a search space containing local minima that have roughly the same score). However, the search spaces for problems in this research can have many thousands of local minima with widely varying scores. The hill rolling approach would almost always terminate in a local minimum that is not necessarily very close in value to the global minimum.

To improve the results then, the *HillRoll* algorithm can be iteratively applied using random starting locations until the results do not improve considerably for some time, or until n nodes have been examined (where n is set by the user). This means a number of local minima are searched. Some of these minima may be large, some small. However, if no better minimum is found after consecutively searching K local minima (where K is sufficiently large), then the best result so far is likely not going to improve significantly, and the algorithm terminates. This approach works best if there are many local minima that are close in value to the global minimum.

A simplified version of the Iterated Hill Rolling algorithm, in pseudo-code, is as follows:

```
InteratedHillRoll(SearchSpace S) returns Node
// Returns the "best" solution it finds
{
    Integer Count = 0 // Num of minima visited without improvement
    Node BestMinimum
    Node CurrentMinimum

    // Choose start node randomly
    CurrentMinimum = HillRoll(S, RandomNode(S))
    BestMinimum = CurrentMinimum

    // K is the maximum number of local minima to examine
    While (Count < K)
    {
        CurrentMinimum = HillRoll(S, RandomNode(S))
        If (CurrentMinimum < BestMinimum)
        {
            Count = 0
            BestMinimum = CurrentMinimum
        }
```

```
        Else
        {
            Count = Count + 1
        }
    }
    Return BestMinimum
}
```

This algorithm works well in general, but cases might arise when the constant K is large and the search space has many local minima of widely scattered values. Under these conditions, it is possible that the *IteratedHillRoll* algorithm will take too long to terminate as results slowly improve over time. To account for this possibility, a constraint is present in the algorithm to terminate the search after n nodes have been examined. Of course, if the algorithm terminated due to this case, the solution would be worse than otherwise since the best found answer was likely still improving over time. If the results were not improving over time, the algorithm would already have terminated due to the condition of unimproving results for some number of local minima visited.

Is there any advantage that one of the search algorithms has over the other? It appears that the iterated hill rolling technique performs slightly better than the random approach. However, the improvement in results is relatively small. Tests were run on a number of sound presentations comparing the results of these two techniques over time. A maximum number of nodes n to search was set at a constant value. The "best results so far" of each algorithm were saved after every 100 searched nodes. The constant n was typically large, between 1,000,000 and 10,000,000 nodes depending on the size of the particular search space. The number of actual nodes in the search space was orders of magnitude larger than n — typically in the trillions to septillions, and sometimes much bigger still. The entire search space could therefore not generally be examined, so the global minimum is unknown in almost all of the problems. The worst found score is noted with each example. The knowledge of the highest score found gives a notion as to the range of scores present in a search space. This helps in determining how large an improvement in score is significant for a given problem.

The first example is a search space small enough to search thoroughly. In this example, [SS6] (containing 6 earcons) was examined in a 4.2 second time interval. The number of nodes in the search space is 330,126,720.[5] Table 5.5 and Figure 5.30

[5]This number is derived from (22 positions to schedule earcon 1) * (28 positions to schedule earcon 2) * (35 positions to schedule earcon 3) * (22 positions to schedule earcon 4) * (29 positions to schedule earcon 5) * (24 positions to schedule earcon 6).

1 EARCON		2 EARCONS		3 EARCONS	
Interval (Blocks)	Nodes in search space	Interval (Blocks)	Nodes in search space	Interval (Blocks)	Nodes in search space
20	$1^1 = 1$	20	$1^2 = 1$	20	$1^3 = 1$
21	$2^1 = 2$	21	$2^2 = 4$	21	$2^3 = 8$
22	$3^1 = 3$	22	$3^2 = 9$	22	$3^3 = 27$
23	$4^1 = 4$	23	$4^2 = 16$	23	$4^3 = 64$

Table 5.5: The size of search spaces as a function of earcons and time interval.

show the exponential growth of nodes in a search space as a function of the number of earcons, and the polynomial growth as a function of time interval. In Figure 5.30, each earcon is assumed to be of an average length of 20 blocks (about 2 seconds). The equation plotted is $(x-20)^y$, where x is the number of blocks in the time interval of the presentation, and y is the number of 20-block earcons. When x is constant, the function grows exponentially as $O(K^n)$, and when y is constant, the function has polynomial growth at the rate of $O(n^k)$.

The problem as stated took approximately 15 hours to search the entire space. Slightly over 20 million nodes per hour were evaluated on a 180MHz Intel Pentium Pro computer system. The global minimum was found to be 68.89, and the global maximum was found to be 278.25. The approximate position of this problem on the graph in Figure 5.30 is at (6 earcons, 45 blocks) — the exponential growth rate of the search spaces for more complex problems makes them impossible for any computer to exhaustively explore in a reasonable period of time.

The random and iterated hill rolling algorithms were used to find a minimum score node with at most 1,000,000 nodes searched. The random search algorithm found a best score of 74.65 after approximately 450,000 nodes searched. The iterated hill rolling method found a best score of 69.28 after approximately 225,000 nodes searched. This result was closer to the global minimum (within 0.39) than the best result of the random algorithm (within 5.75). Figure 5.31 illustrates the results of the algorithms on this problem, and the straight line at y=68.89 represents the global minimum of the search space. Both algorithms did find good answers after only a few thousand nodes. The range of values in the search space were from 68.89 to 278.25. After only 300 nodes searched, the iterated hill rolling algorithm found a best score of 89.73, and after 4100 nodes searched, found a best score of 74.53. The random algorithm was not far behind, finding a low score of 80.69 after only 600 nodes. Figure 5.32 shows the results of this problem relative to the global minima and maxima for a much smaller number of nodes searched (to 10,000 nodes searched).

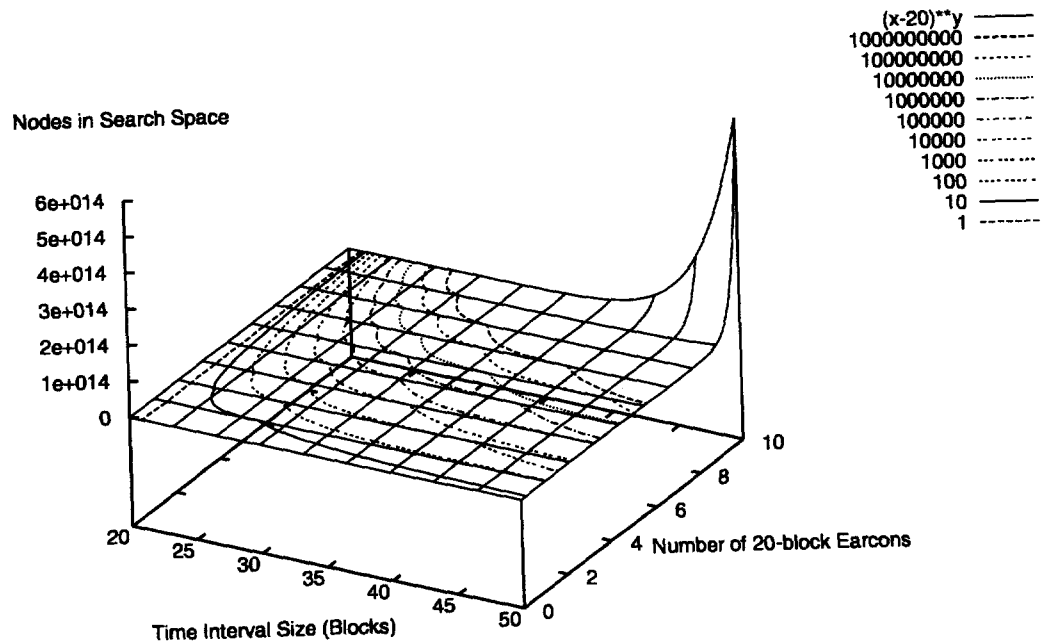


Figure 5.30: The explosive growth of search space size as a function of number of earcons and time interval size.

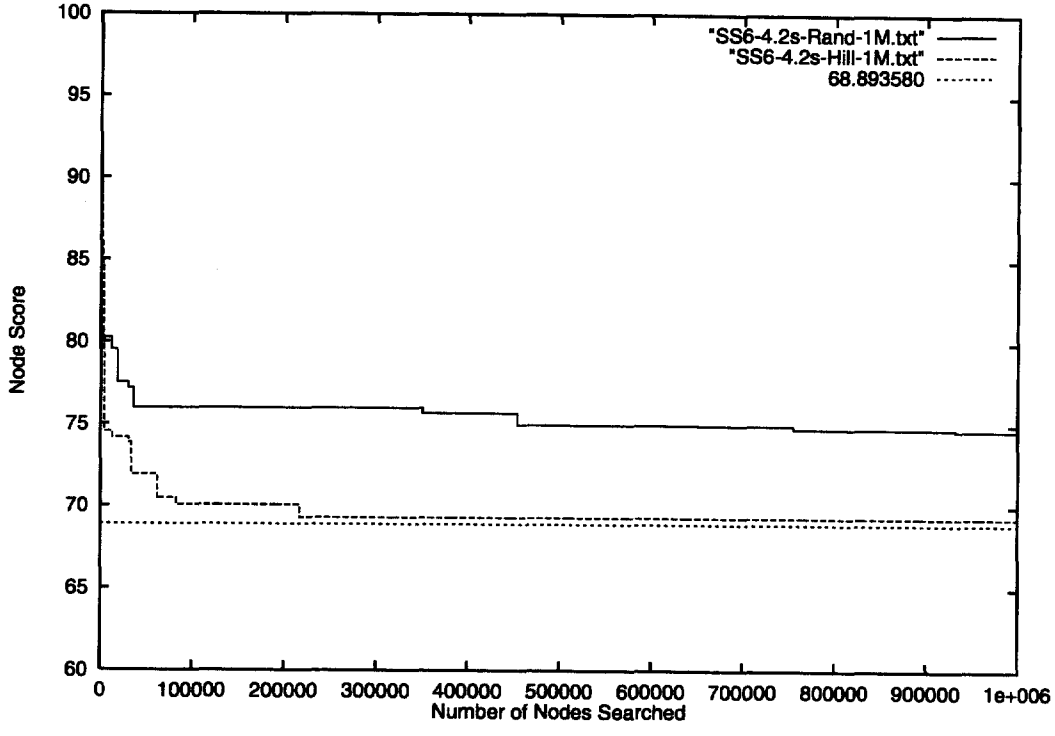


Figure 5.31: The best score found as a function of nodes searched (up to 1 million) for [SS6] in a 4.2 second interval. The straight line at 68.89 represents the global minimum of this search space.

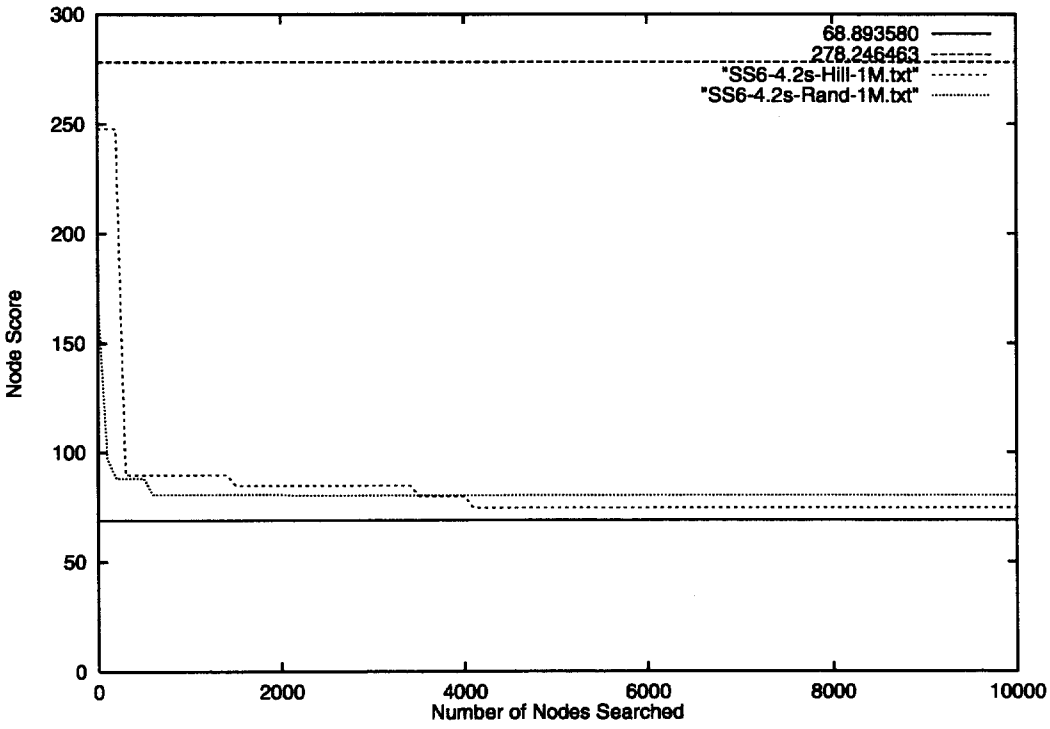


Figure 5.32: The best score found as a function of nodes searched (up to 10,000) for [SS6] in a 4.2 second interval. The straight line at 68.89 represents the global minimum of this search space, and the straight line at 278.25 represents the global maximum.

The next two examples are for larger search spaces that cannot be exhaustively searched. Figures 5.33 and 5.34 show the best answers found for each algorithm as a function of nodes searched. Figure 5.33 shows the results for the search space [SS5]. The relevant details of the exploration of search space 5 are as follows:

- 10 earcons in the search space
- 8 seconds in which to present the earcons
- 932,889,504,927,744,000 (approximately 933 quadrillion) nodes in the search space[6]
- 1,000,000 (1 million) nodes searched
- Worst Node score found (Random Alg.): 712.00
- Best Node score found (Random Alg.): 149.65
- Best Node score found (Iterated Hill Roll Alg.): 137.55
- Approximately 1/900000000000 (one nine-hundred-billionth) of search space explored

Figure 5.34 shows the results for [SS4] to 1 million nodes searched. The exploration of search space 4 had the following statistics:

- 12 earcons in the search space
- 15 seconds in which to present the earcons
- 51,116,675,834,540,603,750,400,000 (51 septillion) nodes in the search space[7]
- 10,000,000 (10 million) nodes searched

[6]This number is derived from (65 positions to schedule earcon 1) * (65 positions to schedule earcon 2) * (69 positions to schedule earcon 3) * (74 positions to schedule earcon 4) * (76 positions to schedule earcon 5) * (63 positions to schedule earcon 6) * (64 positions to schedule earcon 7) * (70 positions to schedule earcon 8) * (32 positions to schedule earcon 9) * (63 positions to schedule earcon 10).

[7]This number is derived from (140 positions to schedule earcon 1) * (140 positions to schedule earcon 2) * (144 positions to schedule earcon 3) * (149 positions to schedule earcon 4) * (151 positions to schedule earcon 5) * (138 positions to schedule earcon 6) * (139 positions to schedule earcon 7) * (145 positions to schedule earcon 8) * (107 positions to schedule earcon 9) * (140 positions to schedule earcon 10) * (140 positions to schedule earcon 11) * (138 positions to schedule earcon 12).

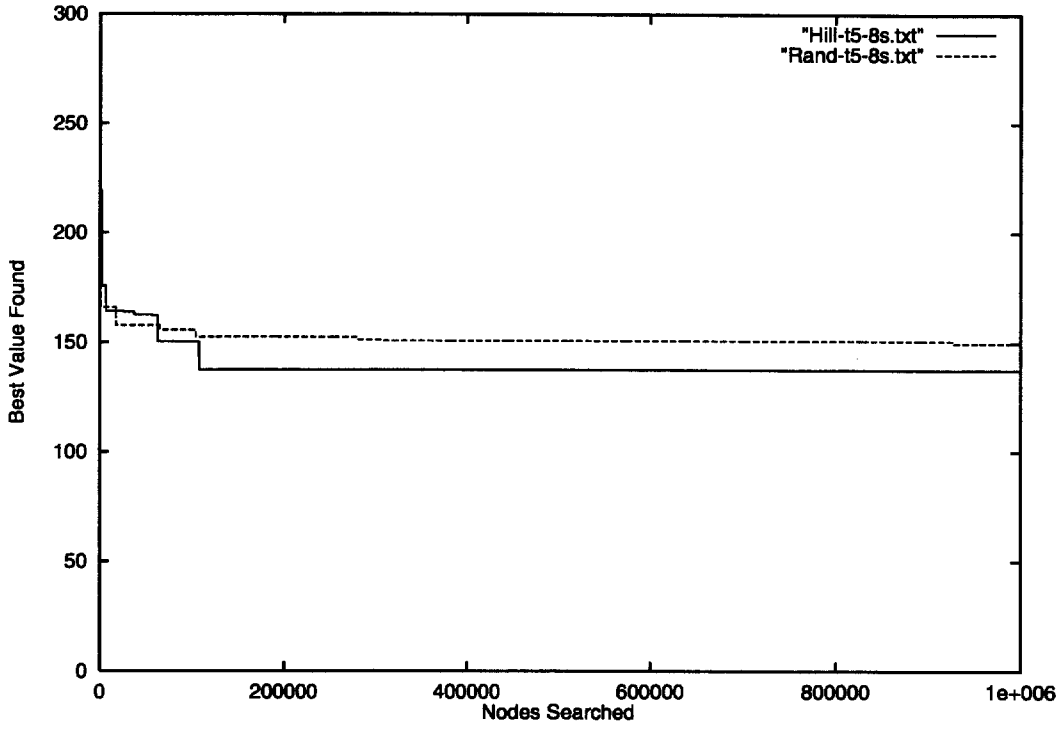


Figure 5.33: The best results of the two search algorithms on [SS5] in a 10 second time interval, as a function of nodes searched.

- Worst Node score found (Random Alg.): 1480.78
- Best Node score found (Random Alg.): 78.53
- Best Node score found (Iterated Hill Roll Alg.): 65.75
- Approximately 1/5000000000000000000 (one five-quintillionth) of search space explored

The results in the two graphs are similar to each other, and typical of the type of search spaces used in this research. The random search algorithm does slightly better for an extremely low number of nodes searched, but as more nodes are searched, the iterated hill rolling algorithm finds slightly better answers. The amount of difference in the answers is small, but consistently in all cases tested, the results are consistent — given a large number of nodes to examine, the iterated hill rolling algorithm finds better results. The improvement of the iterated hill rolling technique over the random search method was approximately 1.6% assuming that the worst node score is about equal to the global maximum in each search space.

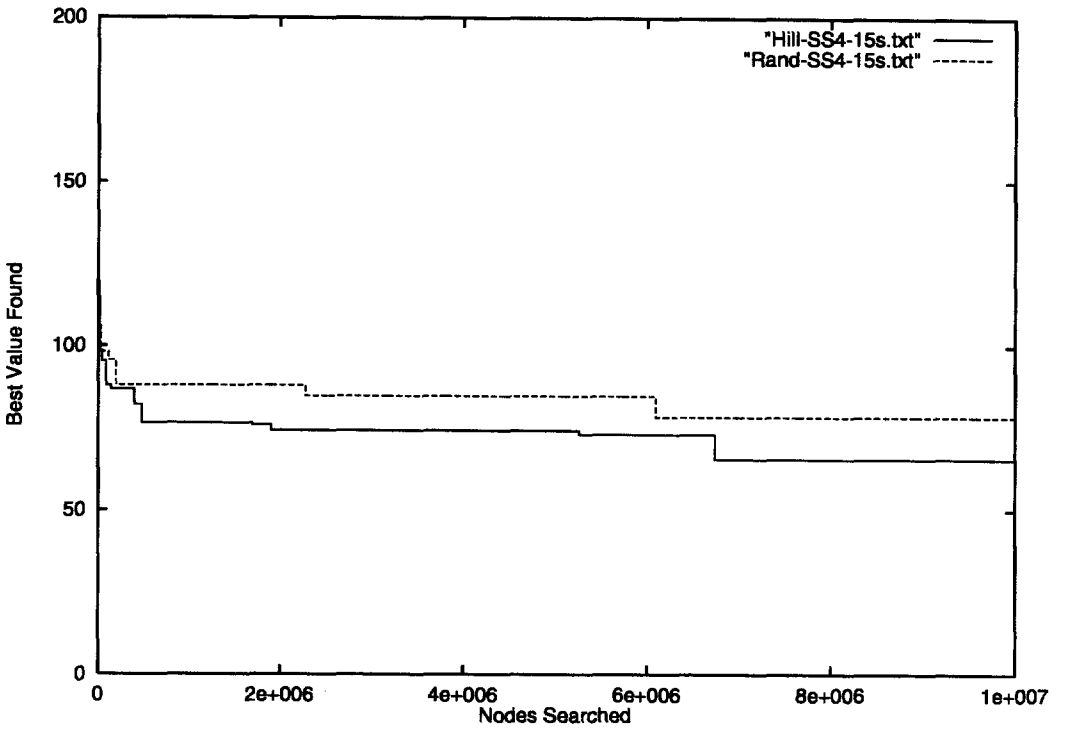


Figure 5.34: The best results of the two search algorithms on [SS4] in a 15 second time interval, as a function of nodes searched.

The percentage of total nodes searched in the larger examples is vastly smaller than the percentage of total nodes searched in the small example. As a result, the closeness to the global minimum in the small search space example probably does not scale to larger examples. However, it is still possible to conservatively gauge the closeness of an evaluated node to the global minimum in any example, large or small.

To show this, consider the following definitions:

Let G_{Max} be the global maximum in a search space. For non-trivial search spaces, this value is not known. Let G_{Min} be the global minimum in a search space. For non-trivial search spaces, this value is not known either. Let v be a node score in the search space. Since v is in the search space, certainly $G_{Min} \leq v \leq G_{Min}$. Define r to be the ratio of the distance between v and G_{Min} to the distance between G_{Max} and G_{Min}.

$$r = \frac{v - G_{Min}}{G_{Max} - G_{Min}}$$

It is the case that $0 \leq r \leq 1$, since $G_{Min} \leq G_{Max}$. The value r represents how close to the global minimum v is, relative to the global maximum. When $r = 0$, $v = G_{Min}$. When $r = 1$, $v = G_{Max}$.

Since the global minima and maxima cannot be known in general, they can both be conservatively under-approximated. Let A_{Min} and A_{Max} be the approximations. $A_{Min} = 0$ always. This is a conservative estimate of the global minimum. Certainly $G_{Min} \geq 0$, so $A_{Min} \leq G_{Min}$. A_{Max} is set to the largest node score ever found for the given search space. Since A_{Max} is a node score in the given search space, $A_{Max} \leq G_{Max}$. Let $\hat{r}$ be a ratio similar to r, only using the known values A_{Min} and A_{Max} in place of the unknown values G_{Min} and G_{Max}.

$$\hat{r} = \frac{v - A_{Min}}{A_{Max} - A_{Min}}$$

Since $A_{Min} = 0$,

$$\hat{r} = \frac{v}{A_{Max}}$$

If it can be shown that $\hat{r} \geq r$ (or $v/A_{Max} \geq (v - G_{Min})/(G_{Max} - G_{Min})$) in all cases, (given $0 \leq G_{Min} \leq v \leq A_{Max} \leq G_{Max}$ and $G_{Min} < G_{Max}$) then $\hat{r}$ represents a ratio greater than or equal to the ratio of how close v is to the global minimum G_{Min}. Note that $\hat{r}$ is a "conservative" estimate of r — the real ratio (which cannot be computed because G_{Min} and G_{Max} are unknown) may be much less, but certainly cannot be greater than $\hat{r}$.

Since $v \geq G_{Min}$, $G_{Max} > G_{Min}$, and $0 \geq (v - G_{Min})/(G_{Max} - G_{Min}) \geq 1$, then

$$\frac{v}{G_{Max}} \geq \frac{v - G_{Min}}{G_{Max} - G_{Min}} \tag{5.2}$$

Furthermore, since $A_{Max} \leq G_{Max}$, it follows that

$$\frac{v}{A_{Max}} \geq \frac{v}{G_{Max}} \tag{5.3}$$

By transitivity with Equations 5.2 and 5.3, the desired form is derived.

$$\frac{v}{A_{Max}} \geq \frac{v - G_{Min}}{G_{Max} - G_{Min}} \tag{5.4}$$

This is a useful result because by dividing a given node score v by the highest node score known for the same search space, the quotient obtained is a value between 0 and 1 that represents a safe estimate of how close v is to the global minimum, relative to the global maximum. The exact value of the ratio may be much smaller, but certainly it cannot be any larger.

For example, in the case of [SS4] with a 15 second time interval, $A_{Min} = 0$ (by definition) and $A_{Max} = 1480.78$. This value is the highest yet found for this search space. The best answer found by the iterated hill rolling algorithm after only 500,000 nodes was $v = 76.56$. This is *certainly* within 76.56 of the global minimum, and is also *at least* 1402.22 less than the global maximum. The best answer found after only 500,000 nodes, or approximately $1/100,000,000,000,000,000,000$ of the search space, is within $v/A_{Max} = 76.56/1480.78$, or 5%, of the global minimum. The answer may be closer than 5% within the global minimum, but is certainly not further than 5%.

A similar analysis of the [SS5] search space example guarantees that the best answer found is within 19% of the global minimum. The answer might be closer to than that to the global minimum, but there is no way to know for certain if it is. In general, more temporally crowded search spaces will yield scores that are further from the "absolute minimum" of 0 than less temporally crowded search spaces. This occurs because with more temporal overlap come higher average penalty values. The actual global minimum becomes increasingly greater than 0 as the overlap increases because any node score v becomes greater on average, as generally does the ratio v/A_{Max}.

5.5 Conclusion

The human auditory system perceptually segregates the numerous sounds that comprise the auditory scene. In the user interface, sounds can be dynamically analyzed and presented in such a way as to assist the auditory system in clearly segregating each individual sound source. This is done through the temporal adjustment of sounds relative to other concurrently playing sounds, and through the processing of the sounds to give each the quality of a unique spatial position and spatial modulation.

The temporal adjustment of sounds relative to each other is achieved through the use of heuristics. It appears that the perceptual system uses a large number of heuristics to resolve ambiguous auditory scene analysis problems, and so this approach seems very natural in the application of auditory scene analysis cues to the computer interface. Other possible approaches to solving this problem are the use of constraint-based rules to assist in the scheduling of sounds, and artificial neural networks that are trained using numerous examples of "good" and "bad" examples of sound presentations. A neural network certainly could be used for the evaluation of heuristics such as determining the degree of difference between timbres, the degree to which two *EarconSource* objects are separated by pitch, or the degree to which two sounds have common pitch changes.

Through the use of heuristics, a number of sounds can be analyzed and temporally arranged so as to minimize the perceptual interactions that can occur between them as they play concurrently. The specific interactions that are avoided are those which encourage auditory stream fusion between two unrelated sounds. The sound presentation with the fewest such interactions will have properties that allow the perceptual system to more clearly segregate each individual sound. This presentation will tend to include the following features more so than any other presentation in the search space:

- Earcons with similar timbres will not temporally overlap.
- Notes from different earcons will not start or stop at the same time.
- Pitch sequences from different earcons will not cross over each other.
- Sequences of pitches from different earcons will not change in similar ways at the same time.
- Concurrent notes from different earcons will not have similar harmonic content.

- The notes in overlapping earcons will be distant in pitch.
- Consecutively played earcons will not have similar timbres.
- The last note of an earcon will have a much different timbre than the first note of another earcon played soon after the first.
- The last note of an earcon will have a much different pitch than the first note of another earcon played soon after the first.

Many other heuristics aid in the process of auditory stream segregation, although the implemented set should cover most cases reasonably well. Nonetheless, new heuristics can easily be integrated into the Computational Auditory Scene Synthesizer. If a new heuristic requires a new data representation of a sound, the representation can be added in a straightforward manner.

The Computational Auditory Scene Synthesizer only does a full analysis of sounds that are of the *EarconSource* class. A useful extension to this work would be to have the algorithms operate on a new *ToneSequenceSource* class that would exist as a subclass of *SoundSource* and a superclass of *EarconSource* and *RawSource*. Any sound that is a sequence of tones can be integrated into all of the algorithms currently written. The hypothetical *ToneSequenceSource* need only have the representations that an *EarconSource* currently has. The <Note, Duration> representation can be computed from a recorded sample using pitch recognition algorithms. The timbre can also be analyzed from the recorded samples.

For such an approach to work, the recorded sound must have only one perceptual stream contained in it. For example, it would be possible to convert the samples of a flute playing a sequence of notes. However, if the recorded samples contained a flute and an oboe playing a duet, this method would produce poor results if the two instruments parts are supposed to be scheduled independently of each other to maximize perceptual distinctness. However, if the duet was to be perceived as a basic perceptual unit, then perhaps the results of this approach would be satisfactory. Problems might arise in the timbre discrimination algorithm, since it assumes the same timbre for the entire sound source.

There is one other possibility to integrate recorded sounds of multiple instruments into this implementation. Ellis, among others, has performed research in the area of Computational Auditory Scene Analysis [Ell96]. The goal of his work was to take a digital recording and isolate each individual source of sound. The recording can then be reconstructed to have only a subset of the sound sources contained in it, or a single sound source can be further analyzed in the absence of extraneous

sounds. For example, footsteps could be eliminated from a recording of a conversation between two people. Using this type of system, it may be feasible to take a recording of more than one instrument, and separate it into a number of recordings of one instrument each. Then, each individual sound source can be scheduled independently. This approach, of course, assumes that there is no significance in the temporal relation between the different sound sources. This is not the case for music, so it is unclear if this extension has any useful benefit.

The final and most useful extension to this implementation would be the design of general algorithms that analyze and operate on any *SoundSource* object. In this case, all sound features must be derived from only the spectral information of a given sound. An analysis algorithm would have to identify the salient frequency components of each sound, and then use heuristics that do not function only on sounds that are tone sequences. However, tonal information would be relevant to the presentation of the sounds. Analysis difficulties can arise in determining features such as relative onsets and offsets, pitch attributes, and extraneous frequency content.

Chapter 6

User Study

Do the penalty scores computed for a node in the Comprehensive Auditory Scene Synthesizer correlate with users' abilities to distinguish between the earcons contained in a given search space? In order to answer this question, a user study was conducted. Eight subjects were asked to listen to the auditory representations of numerous nodes of different search spaces. A total of nine search spaces were used. From each search space, six nodes were selected. The following process was used to select the nodes of a given search space. The Comprehensive Auditory Scene Synthesizer ran for a period of time and saved the lowest and highest scores found. The period of time generally lasted for about four days. The other four desired scores were linearly spaced between the high and low scores. The search space was examined until nodes were found that contained values within 0.05 of each desired score.[1] A further constraint on all examples used was that silence could not occupy more than 20% of the given time interval. The full set of examples consisted of nine groups of six instances each, or a total of 54 examples.

For each subject, the presentation order of the examples was randomized. Each subject alternately listened to an example and recorded how many distinct earcons he or she heard. The examples used and the raw user data are listed in Appendix C.

The average sound "overlap" factor of a particular search space can be determined by dividing the sum of the lengths of all sounds contained in it by the length of the time interval. For example, if the total length of earcons to presented in a 2.0 second interval is 5.0 seconds, then the overlap factor is 2.5. A search space that has a smaller overlap factor will, on average, have a smaller amount of temporal overlap than a search space with a larger overlap factor. The nine search spaces

[1] In one case ([SS17] presented in seven seconds with a desired score of 4962.5), the closest node score found was within only 0.2. However, since the node scores for this example ranged from 1493.5 to 5829.7, the score difference is insignificant.

used in the study had average overlap factors of roughly 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, and 5.0. A number of questions can be asked of user performance relating to average overlap factor:

- Do the predicted scores correlate to user performance at all average overlap factors?
- At what average overlap factor (if any) does intelligent scheduling of sound sources become necessary for improved user performance?
- At what average overlap factor (if any) does intelligent scheduling of sound sources become irrelevant to user performance?
- Is there a relation between average overlap factor and average user performance?

6.1 Analysis

In analyzing the data, the first goal was to determine if there was a correlation between the computed node score and the users' performance in identifying how many distinct earcons could be heard. Specifically, for the set of six nodes for a given sound presentation with a particular average overlap, the six computed perceptual scores were compared against the mean user performances. Both the computed scores for each node and user performance means and standard deviations are listed in Appendix C.

It was not expected that a possible relation between mean user performance and computed perceptual score would be linear, so a non-parametric correlation test was appropriate. The Spearman Rank Correlation Coefficient was used in all cases with a significance level of $\alpha = 0.05$. For each of the nine experiments (with varied average overlap factor), the computed perceptual scores and the average user performance were ranked. The null and alternative hypotheses were formulated as follows:

H_0: The two sets of ranks are independent
H_1: The two sets of ranks are not independent

The test was one-sided in that a correlation was defined to be positive only. For $n = 6$ and $\alpha = 0.05$, the critical value of r_s is 0.829. The following table shows the value r_s obtained for each set and the conclusion drawn from the value:

Example	Average Overlap	r_s	Reject H_0?
[SS22] in 9.3 sec.	1.0	0.829	Yes
[SS25] in 11 sec.	1.5	0.986	Yes
[SS15] in 10 sec.	2.0	0.829	Yes
[SS19] in 8 sec.	2.5	0.829	Yes
[SS20] in 7 sec.	3.0	0.943	Yes
[SS21] in 10 sec.	3.5	0.783	No
[SS17] in 7 sec.	4.0	0.771	No
[SS24] in 8 sec.	4.5	0.714	No
[SS23] in 5.9 sec.	5.0	0.429	No

The plots in Figure 6.1 graphically illustrate the relation between mean user performance and computed perceptual score. All error bars indicate the 95% confidence interval for each data point. The positive correlation was statistically significant for average overlaps up to and including 3.0, and not statistically significant for average overlaps of 3.5 or greater. However, if the data points for the two highest computed perceptual scores for the node with an average overlap of 3.5 are ignored, the ranks of the other four values positively correlate with the computed score. Furthermore, it is encouraging to note that in all average overlap cases but one (namely, the node with an average overlap of 5.0), mean user performance was best for the node that had the lowest computed perceptual score.

These results can be interpreted as follows: for average overlaps less than 3.5, the particular temporal layout of earcons is a large factor in users' abilities to distinguish individual earcons. For average overlaps between 3.5 and 4.5, intelligent temporal layout of earcons is still a factor in user performance. For the average overlap of 5.0 (and presumably for nodes with average overlaps greater than 5.0), user performance appears to be similar regardless of the particular temporal layout of the sounds. This may be the result of reaching a "saturation point" of auditory perception. At and beyond this point, user performance may be roughly the same regardless of the temporal layout of the earcons.

6.2 Prediction of User Performance

It is useful to have a method of predicting average user performance given a particular arrangements of sounds in a time interval. In a real-time intelligent audio scheduling system such as the Centralized Audio Presentation System, this method would provide a useful metric to assist in both the scheduling problem and

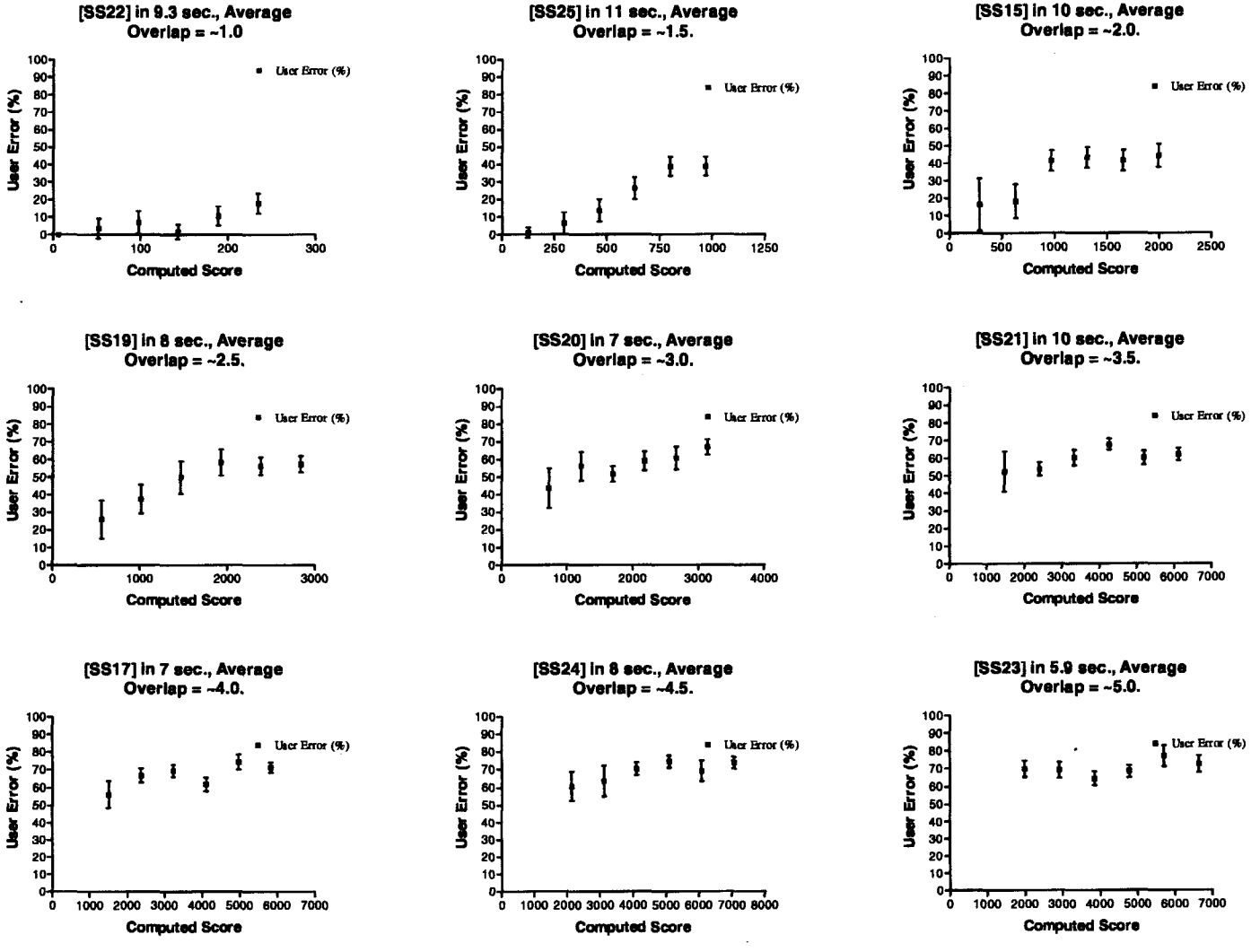


Figure 6.1: Average user error percentage as a function of computed score.

in the decision to delay or preempt particular auditory messages, dependent upon the "interference" each message adds to the currently audible sound presentation.

6.2.1 Prediction of User Error

In order to find a function to predict user performance, it is very helpful to have an idea of what shape function should be fit to the sampled data. It seems reasonable to start by considering user performance as a function of perceptual score per unit time.[2] One would expect that with very small perceptual score per unit time values, user error would be small in most cases. As perceptual score per unit time increases, more opportunities for error become available, and as such, more errors will occur on average. For all perceptual score per unit time values higher than some critical value, user error would be equally high indicating that the users' perceptual "saturation points" have been exceeded.

The hypothesized relationship between perceptual score per unit time and user error could suggest the use of a one phase exponential association function of the form $y = YMax * (1 - e^{-kx})$. Figure 6.2 shows a scatter plot of the mean user error for the scores of each node in the user study. A computed score is calculated from a node in a search space. When each computed score is divided by the associated node's interval length for the search space, the scatter plot in Figure 6.3 is attained. In Figure 6.3, each point represents mean user error for a node with the computed score per unit time. The plot indicates that there does indeed appear to be an exponential relation between the computed score per unit time and the mean user error. In Figure 6.3, the function $y = 72.0 * (1 - e^{0.0066x})$ is fit to the scatter plot. Using this function, mean user error u can be predicted given the computed score s of the node and the interval length l of the node.

$$u(s, l) = 72.0 * (1 - e^{0.0066s/l})$$

To evaluate the accuracy of the prediction function, further user performance data was collected. Users listened to two additional sets of examples and identified the number of distinct earcons that could be discerned. The two search spaces used in the additional testing were [SS18] presented in 18 seconds, and [SS16] presented in seven seconds. In Figure 6.4, the prediction function is shown along with the data points collected from the additional user testing. While the fit is not perfect, the results appear reasonable considering the small amount of data collected. With more extensive user testing, the prediction function could undoubtedly be refined further.

[2] A lower perceptual score predicts a lower user error. Similarly, a higher perceptual score predicts a higher user error.

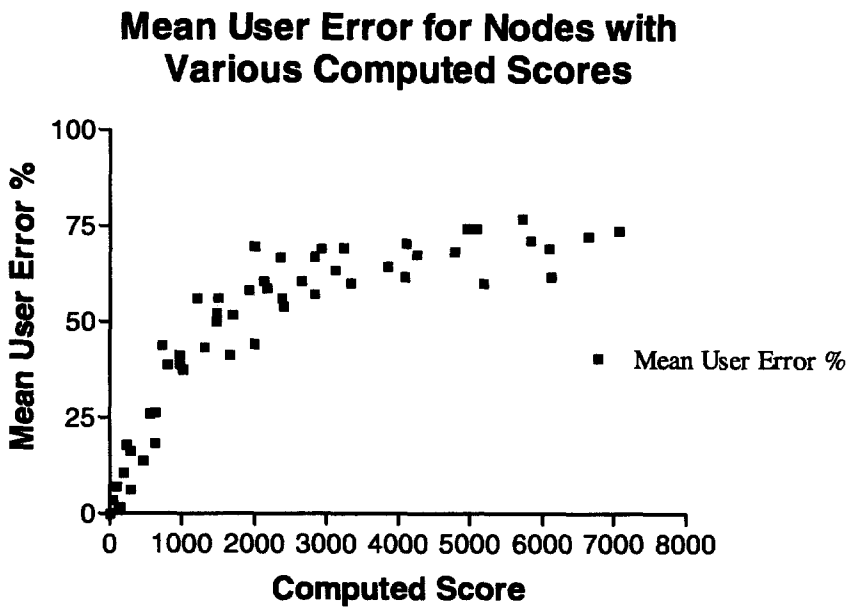


Figure 6.2: A scatter plot suggesting an exponential association between mean user error and computed score.

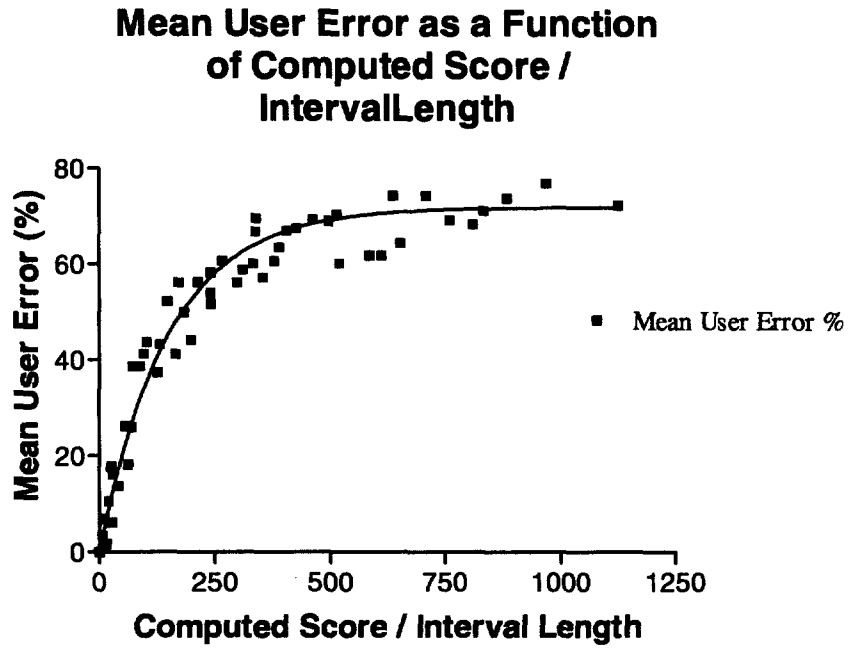


Figure 6.3: The exponential function $y = 72.0 * (1 - e^{0.0066x})$ is fit to the scatter plot of (computed score / interval length) and mean user error.

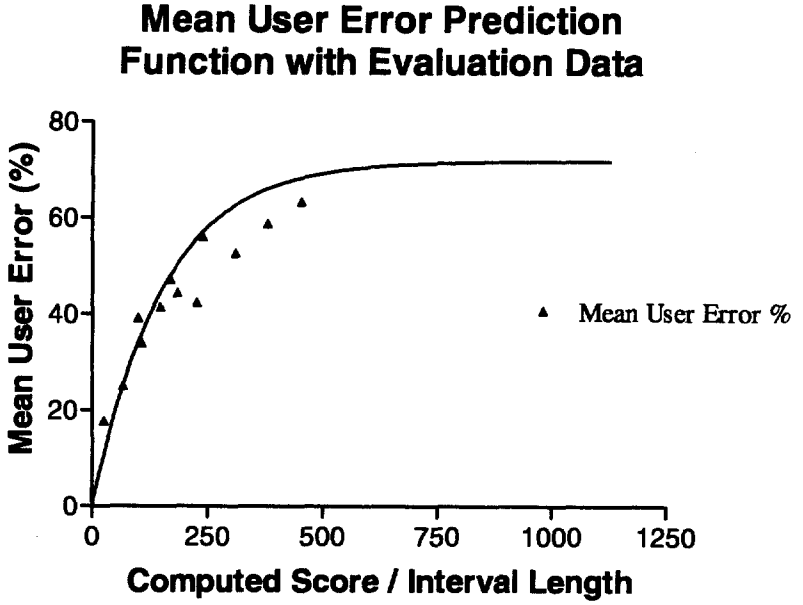


Figure 6.4: The prediction function $y = 72.0 * (1 - e^{0.0066x})$ is shown along with further user data collected to evaluate the accuracy of the function.

6.2.2 Prediction of the Lowest Node Score in a Search Space

In Chapter 5, a technique was described that conservatively approximates the closeness of a particular node score to the global minimum in the search space. The method describes where the node score lies in the interval between the lowest possible score (namely, 0) and the highest known node score for the particular search space. The method is not very useful for determining when a node score will not significantly improve given additional time. Empirical analysis of the search results for the examples used in the user study has shown a relation between the lowest node score found and the parameters of average overlap and interval length. For each search space, the lowest score found was plotted as a function of its overlap factor. The resulting scatter plot is illustrated in Figure 6.5. When each data point is divided by the interval length for its corresponding search space, the plot then appeared as in Figure 6.6. The data is very nicely fit by the polynomial equation $y = 16.06x + 16.82x^2$, which appears in Figure 6.6 as a solid curve.

It is now possible to define a prediction function *HTrans*, that computes the lowest node score likely to be found for a given search space in a given time interval. This function requires two parameters: the average overlap factor o for the search space, and the interval length l. The function definition is as follows:

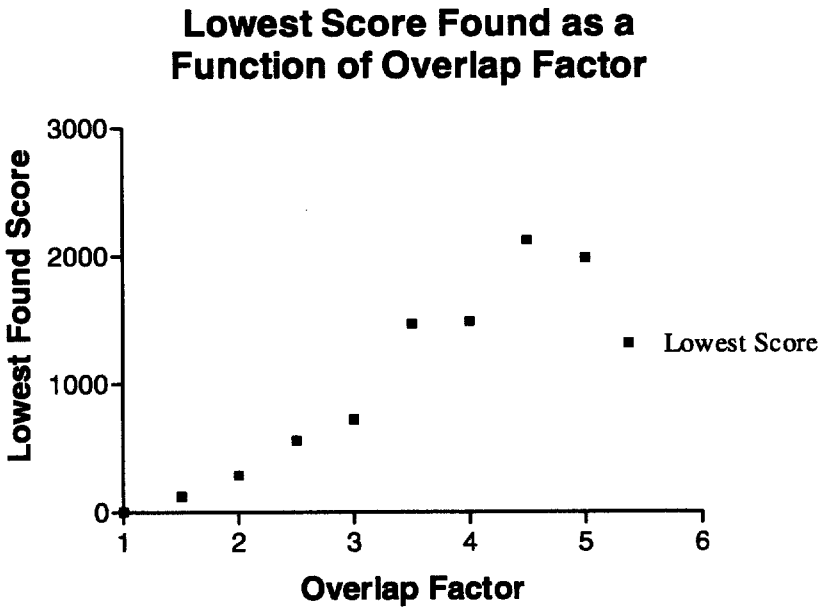


Figure 6.5: A scatter plot relating average overlap factor to lowest computed score.

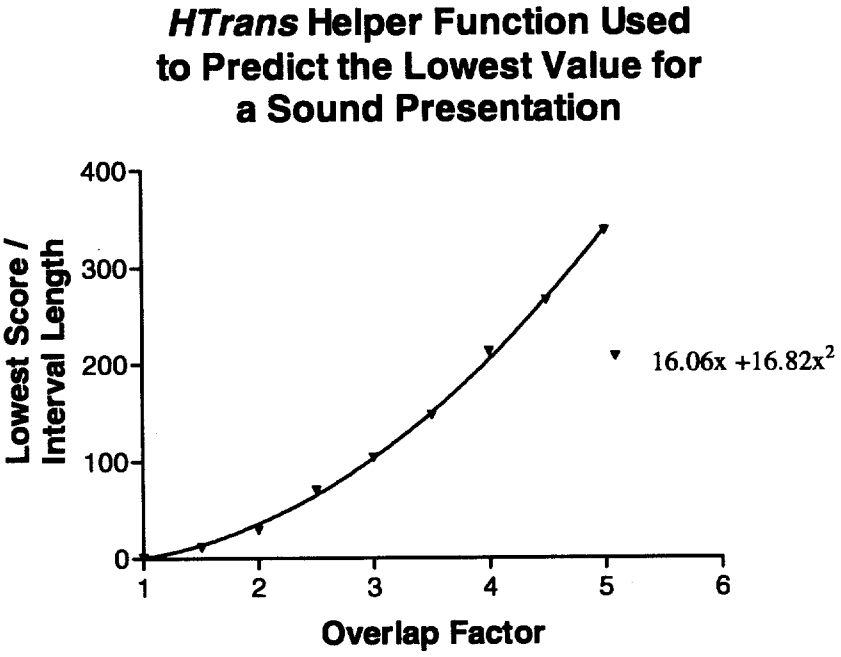


Figure 6.6: The best score found as a function of average overlap divided by corresponding interval length.

$$HTrans(o, l) = (-16.06o + 16.82o^2) * l$$

In order to evaluate the accuracy of the prediction function, two search spaces were examined to estimate their minimum computed scores. The first, [SS16] presented in 7.0 seconds, has an average overlap factor of 3.07. The function *HTrans*(3.07, 7.0) predicts the value 764.56 for the lowest score estimate for this problem. The actual lowest score found was 681.3. The prediction was within 11% of the lowest computed score. The second example is [SS17] presented in 12.0 seconds. The average overlap factor is 2.4. The function *HTrans*(2.4, 12.0) predicts the value 700.07. The lowest score actually found for this problem was 677.6. The prediction in this case was approximately 3% from the actual lowest found score.

In general, the prediction function should return results of this accuracy as long as the example being evaluated consists of a variety of different earcons.[3] The predicted score is dependent upon the average computed score per unit time for temporally overlapped earcons. This average computed score per unit time is then multiplied by the length of the time interval to get the final result.

6.2.3 Prediction of User Error for the Node Containing the Lowest Found Score

Given the data collected in the user study, a function *VTrans*(o) can be defined that relates the average overlap factor o to the mean user error percentage for the *lowest* score node of the corresponding search space. This is visually represented as a graph in Figure 6.7. The data points roughly correspond to a sigmoid function, although a line could be fit fairly accurately as well. The sigmoid function seems the better choice though, since user performance should be consistently good for low values of o, and as o get very large, the mean user error should asymptotically approach some value less than or equal to 100%. The sigmoid function fits the data points at 1.0 and 1.5, where users performed very well.

The best fit Boltzmann sigmoid function is as follows:

$$VTrans(o) = -4 + \frac{70.42}{1 + e^{\frac{2.647 - o}{0.6}}}$$

[3]Certain examples that contain many of the same or similar earcons may have no perceptibly clear presentation, regardless of the temporal arrangement of each individual earcon. In such anomalous cases, all scores in the search spaces are abnormally high, and the *HTrans* function will predict a lower value than that observed through user evaluation.

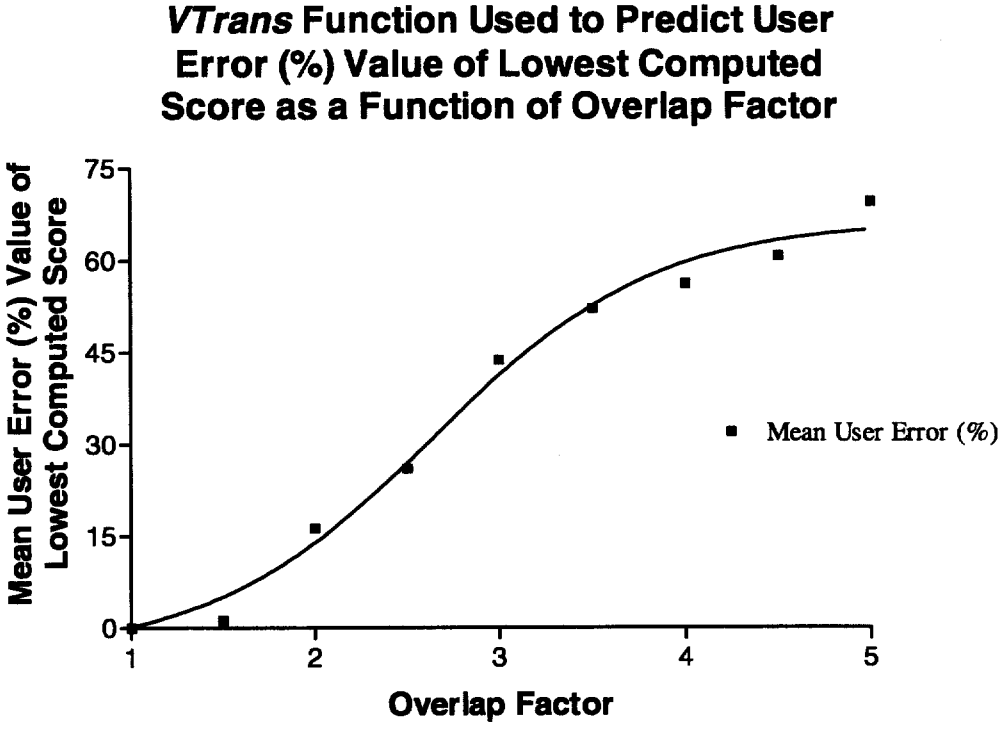


Figure 6.7: User error percentage as a function of average overlap. The Boltzmann Sigmoid function $y = -4 + 70.42/(1 + e^{(2.647-x)/0.6})$ is fit to the measured data points.

To evaluate the accuracy of the *VTrans* prediction function, the extra examples evaluated by users were used. The first example considered was [SS16] presented in a 7.0 second time interval, with an average overlap $o = 3.07$. The function $VTrans(3.07)$ equals 43.13 percent. The actual mean user error percentage for this example was 39.29. The predicted mean user error percentage was within 3.9 percentage points of the measured value. The second example considered was [SS18] presented in a 13 second time interval, with an average overlap factor of 1.84. The function $VTrans(1.84)$ equals 10.55 percent, while the actual measured mean user error percentage was 15.83. In this case, the predicted value was within 5.3 percentage points of the observed value.

6.3 Conclusion

In situations where the user must actively monitor multiple sound sources, a sound presentation regulated using the methods described in Chapter 5 will improve user perceptibility over an unregulated method. This has been demonstrated, through the results and analysis of the user study presented in this chapter, for sound presentations containing an average overlap of approximately 4.0 or less. For sound presentations containing higher average overlaps, the presentation regulation method did not significantly affect user performance.

It is possible to predict user error rates in recognition of earcons based upon the indicators of computed score, average overlap, and the temporal interval length. With this type of prediction model, the intelligent scheduling of sounds can be controlled to allow the user to receive a great deal of information concurrently through the auditory modality.

Chapter 7

Summary and Future Research

The goal of this research was to create a method for the intelligent presentation of dynamic temporally overlapping auditory messages. Two main approaches were studied: the use of numerous computer selectable auditory representations for the same message; and the dynamic temporal adjustment of auditory messages to minimize perceptual conflicts between overlapping sounds.

In the first approach, a number of applications present audio through a centralized control mechanism. That mechanism schedules and possibly dynamically alters each sound request so that the user can clearly distinguish between each individual sound comprising the computed auditory scene.

Such a system has a number of advantages for auditory user interface design:

1. The design of each sound used by a particular application is not constrained by the other sounds used. Each sound can be created without regard for possible auditory stream segregation problems that might arise between different auditory messages.

2. Possible conflicts are minimized between sounds arising from different applications in a multitasking environment. This problem cannot be solved by the sound designer since it is only known at run-time what other applications are being used.

3. Sounds created dynamically at run-time can be used in the computer interface. Such sounds are often created from data collected during the current execution of the application. Each dynamically created sound can be analyzed at run-time and presented to the user through the centralized control mechanism.

4. Users are guaranteed to receive an overall more perceptible combined auditory output from all running applications.

An architecture was designed that supports the intelligent scheduling of multi-representational auditory objects from numerous applications. A heuristic-based evaluation method provides the intelligence that chooses a particular auditory representation and exactly when it is to be played. Each request can contain preferences to specify the most desirable auditory representation, and parameters to provide guidance to the centralized control mechanism for its scheduling decisions.

There were two implementations, described in Chapters 4 and 5, that attempted to control the dynamic presentation of temporally overlapping auditory messages. The first, the Centralized Audio Presentation System, provided the real-time centralized control mechanism through which applications requested auditory output with certain constraints. Audio servers have offered a centralized method of playing sound, but have not ever performed analysis and intelligent scheduling with respect to sound perception issues. The Presentation System offered a number of benefits over this type of resource-oriented scheduler:

1. Each request sent from an application to the Presentation System can encode numerous auditory representations. This approach offers the flexibility of presenting the representation most suitable with respect to the current global auditory state.

2. Requests were received in real-time, and simple heuristics allowed the Presentation System to choose an appropriate representation very quickly.

3. Certain auditory representations could be modified in specific ways by the Presentation System in order to encourage greater stream segregation between all currently sounding auditory messages.

4. Each request had a particular priority associated with it. High priority requests must be presented very clearly to the user; low priority requests could be presented in a less clear manner, or in more extreme cases, ignored altogether by the Presentation System.

5. Each request also contained a maximum latency parameter. This is the time period in which the request must be filled. Requests that must be synchronized to graphics or sounded immediately would have a low latency. Requests that could be presented at any point in a larger time period would have higher latency values.

The second implementation that controlled the presentation of auditory messages was the Computational Auditory Scene Synthesizer. Through the use of a more sophisticated heuristic-based scheduling method, a set of sounds was scheduled in a given time period. Since the total length of the sounds was generally much greater than the length of the time period, temporal overlap between sounds was necessary. The actual scheduling was controlled purely through heuristics that were designed to avoid perceptual conflicts between sounds. The best schedule was the one that allowed the user to most easily discern each individual sound presented in the time period.

Since the entire search space for a given scheduling problem grows exponentially with the number of sounds to schedule, it was not feasible to evaluate all possible schedules to find the best one. a fast heuristic search technique was implemented to find a reasonably good schedule in a very short amount of time. The hybrid search technique of repeatedly applying hill climbing from random starting locations in the search space produced good schedules of sounds in very short amounts of time.

A user study was conducted to test the correlation between the heuristic-based predicted perceptual score of a sound presentation, and the performance of a user in identifying the number of discrete auditory messages present in that same presentation. For the examples with average sound overlaps of less than 3.5, a positive correlation was found. This indicates that the programmed heuristics do penalize many of the perceptual conflicts that arise between temporally overlapping sounds. The inconclusiveness of results for examples with average overlaps of 3.5 or greater demonstrates the need for further evaluation and refinement of each heuristic and perhaps the implementation of additional perceptual cues for auditory stream segregation, such as spatial location for each sound source.

As demonstrated by the example implementations, most of the research goals discussed in Chapter 1 have been met. Most importantly, the presentation of dynamic temporally overlapping auditory messages can be controlled to improve the user's perception of each individual sound. This control has be demonstrated in two ways: through the selection of the most appropriate auditory representation (if more than one representation exists), and through the intelligent temporal scheduling of sounds within the constraints of each individual sound request. The selection of auditory representations was used in the implementation of the Centralized Audio Presentation System, and an intelligent scheduling method was implemented in both the Presentation System and the Computational Auditory Scene Synthesizer. In the latter system, temporal constraints for auditory messages were not used, so that the heuristic scheduling algorithms could be tested and utilized to their full extent. It would be trivial to extend the Computational Auditory Scene Synthesizer to include temporal constraints on each auditory message. Es-

sentially, only a subset of the entire search space would require possible examination, thus decreasing overall search time.

It has been shown through the Computational Auditory Scene Synthesizer that the scheduling of sounds can be accomplished even when the search spaces become enormous because of combinatorial explosion. Examples have demonstrated that reasonably good schedules can be found very quickly, and therefore the heuristic-based method of evaluating an auditory scene is feasible even in a real-time environment.

Finally, the dynamic modification of auditory messages was explored in the Centralized Audio Presentation System through the use of register manipulation of earcons. It appears that this technique holds promise for auditory interface design, but further research is required to determine the applicability of the technique for real systems.

While the example implementations have demonstrated that the overall architecture for Computational Auditory Scene Synthesis can be realized, there are many issues left unresolved.

- The heuristic-based auditory scene synthesis scheduling method should be integrated into a new version of the Centralized Audio Presentation System. This would add a much more robust scheduling algorithm that would result in better auditory scene generation. It is unfortunate that this was never completed.

- The use of spatial audio was not implemented because of a lack of necessary hardware, but spatialization of sound is known to be a very strong cue in auditory scene analysis. The Comprehensive Auditory Scene Synthesizer could be programmed to present sounds in spatially distinct locations. A heuristic could be added to assess a penalty relative to the proximity of different simultaneously presented sounds. Furthermore, certain well known perceptual effects that cause confusion[1] could be identified and penalized accordingly.

- An experiment similar to the one performed in Chapter 6, only using a three-dimensional spatial audio system, would prove valuable in determining the effect of spatialized sound on the auditory stream segregation of multiple earcons. Each earcon contained in a given sound presentation could be rendered in a spatially distinct area of the listening space. It is very probable

[1]The best known effect of this nature is the mistake of identifying a sound behind the head as being in front of the head, and vice-versa. So, the case of presenting spatial sounds directly in front of and behind a user simultaneously can be penalized.

that users would perform better with the addition of spatial information, but an experiment to quantify the improvement would be necessary to draw any certain conclusions.

- The weights used for each perception heuristic were chosen through informal testing. A more formal approach to finding correct weights would be an important addition to the implementation.

- Each heuristic algorithm used to detect perceptual conflicts between sounds was never evaluated apart from others. The literature supports the importance of each heuristic, but the particular implementation was not evaluated by itself. Specifically, the penalty functions used are somewhat arbitrary. Experimental testing could help in the creation of more accurate implementations of each given heuristic.

- The set of perception heuristics was chosen because each is considered to be very important in the literature. Personal observation helped in the selection process as well. However, it is unclear how user performance would change given subsets of the implemented heuristics or with the addition of any new heuristics. Experimental testing of groups of heuristics would be instructive in finding out which heuristics are necessary and which can be ignored.

- It would be valuable to perform a user study that measured if the user could hear particular earcons in an overlapped earcon presentation. This type of study could help to find the point at which intelligent scheduling of sounds cannot overcome the masking effects of temporally overlapping sounds.

- The method of computational auditory scene synthesis has only been implemented to full analyze abstract earcons. It would be useful to extend the functionality of the implementation to operate on a larger class of sounds.

These problems, and others related to the presentation of dynamic overlapping auditory messages, will undoubtedly be studied and resolved in the near future. The necessity of creating more complex auditory computing environments will continue to compel research in this relatively new discipline of auditory user interfaces.

Appendix A

Earcons Referenced in the Text

The syntax for earcons described in this appendix is as follows:

[*Timbre*]([*Pitch*][*Duration*])+

Variables in square brackets are required; a + symbol indicates one or more of the expression to which it applies. Pitches are described using the following notation: the musical pitch is followed by a number indicating the octave. An octave of 0 indicates MIDI note values ranging from 0 to 11, or fundamental frequencies ranging from 8.18 Hz to 15.43 Hz. For each higher octave, the MIDI note values are incremented by 12, and the fundamental frequencies are doubled. So, the pitch G1 would be the note G played in octave number 1. This corresponds to the MIDI note value 18 and a fundamental frequency of 24.50 Hz. The pitch G2 corresponds to MIDI note value , and fundamental frequency 24.50 Hz * 2 = 49.00 Hz.

The note durations are given in terms of seconds.

Label	Timbre Used	Earcon Definition
[BA1]	Electric Bass	(C3 0.6) (C3 0.4) (G2 0.2) (B♭2 1.0)
[BA2]	Electric Bass	(C♯3 0.186) (D♯3 0.186) (F3 0.37) (D♯3 0.186) (F3 0.186) (F♯3 0.37)
[BA3]	Electric Bass	(C♯3 0.186) (D♯3 0.186) (F3 0.186) (F♯3 0.186) (G♯3 0.557)
[BA4]	Electric Bass	(G♯3 0.186) (F♯3 0.186) (F3 0.186) (D♯3 0.186) (C♯3 0.557)
[BA5]	Electric Bass	(B♭2 0.186) (G2 0.372) (A2 0.186) (B♭2 0.186) (F♯2 0.186) (G2 0.372)

Label	Timbre Used	Earcon Definition
[BA6]	Electric Bass	(E2 0.093) (E2 0.093) (E2 0.093) (F2 0.557) (G♯2 0.279) (A2 0.279)
[CE1]	Celeste	(E6 0.25) (F♯6 0.25) (G♯6 0.25) (B♭6 0.25) (C7 0.25) (D7 0.25) (E7 0.25) (F♯ 0.25)
[CE2]	Celeste	(C♯6 0.37) (D♯6 0.186) (F6 0.186) (D♯6 0.37) (F6 0.186) (F♯6 0.186)
[CE3]	Celeste	(G♯6 0.186) (F♯6 0.186) (F6 0.186) (D♯6 0.186) (C♯6 0.557)
[CE4]	Celeste	(C♯6 0.186) (F6 0.186) (F♯6 0.186) (D♯6 0.186) (G♯6 0.557)
[CE5]	Celeste	(E4 0.093) (A4 0.093) (E5 0.093) (E6 0.093) (B6 0.093) (C7 0.093) (B6 0.093)
[CE6]	Celeste	(B6 0.093) (E6 0.093) (E5 0.093) (A4 0.093) (E4 0.093) (D♯4 0.093) (E4 0.15)
[DG1]	Dist. Elec. Guitar	(A4 0.4) (G♯4 0.2) (F♯4 0.2) (D♯4 0.8)
[DG2]	Dist. Elec. Guitar	(C♯4 0.186) (D♯4 0.186) (F4 0.186) (F♯4 0.186) (G♯4 0.557)
[FL1]	Flute	(G5 0.3) (B5 0.3) (D6 0.3) (F6 0.3)
[FL2]	Flute	(G6 0.25) (A6 0.25) (G6 0.25) (A6 0.25)
[FL3]	Flute	(G6 0.7) (G6 0.7) (G6 0.7) (G6 0.7)
[FL4]	Flute	(C♯6 0.186) (D♯6 0.186) (F6 0.37) (D♯6 0.186) (F6 0.186) (F♯6 0.37)
[FL5]	Flute	(G♯6 0.186) (F♯6 0.186) (F6 0.186) (D♯6 0.186) (C♯6 0.557)
[SA1]	Saxophone	(F3 0.6) (A3 0.4) (B4 0.4) (D4 0.2) (C4 0.6)
[SA2]	Saxohpone	(C♯3 0.186) (D♯3 0.186) (F3 0.37) (D♯3 0.186) (F3 0.186) (F♯3 0.37)
[SA3]	Saxophone	(C♯3 0.186) (D♯3 0.186) (F3 0.186) (F♯3 0.186) (G♯3 0.557)
[SA4]	Saxophone	(G♯3 0.186) (F♯3 0.186) (F3 0.186) (D♯3 0.186) (C♯3 0.557)
[SA5]	Saxophone	(C5 0.372) (C5 0.093) (A5 0.372) (G♯5 0.186)
[SA6]	Saxohpone	(C5 0.372) (C5 0.093) (E5 0.372) (E♭5 0.186)
[SA7]	Saxophone	(C5 0.372) (C5 0.093) (C♯4 0.372) (C4 0.186)
[SA8]	Saxophone	(A5 0.186) (A5 0.186) (D5 0.186) (F5 0.124) (C5 0.124) (C♯5 0.124) (D5 0.372)
[SG1]	Steel Guitar	(E4 0.3) (G♯4 0.3) (B4 0.3) (D♯5 0.3) (E5 0.9)

Label	Timbre Used	Earcon Definition
[SG2]	Steel Guitar	(G3 0.25) (B3 0.25) (D4 0.125) (D♯4 0.125) (D4 0.125) (D♯4 0.125) (C4 0.5)
[SG3]	Steel Guitar	(E5 0.5) (C5 0.25) (G4 0.25) (C4 0.75)
[SG4]	Steel Guitar	(E4 0.25) (F♯4 0.25) (G♯4 0.5) (B4 0.25) (D♯5 0.25) (E5 0.5)
[SG5]	Steel Guitar	(C♯3 0.186) (D♯3 0.186) (F3 0.37) (D♯3 0.186) (F3 0.186) (F♯3 0.37)
[SG6]	Steel Guitar	(C♯3 0.186) (D♯3 0.186) (F3 0.186) (F♯3 0.186) (G♯3 0.557)
[SG7]	Steel Guitar	(G♯3 0.186) (F♯3 0.186) (F3 0.186) (D♯3 0.186) (C♯3 0.557)
[TP1]	Trumpet	(G5 0.3) (D5 0.3) (G5 0.3) (B5 0.3) (D6 0.3) (F6 0.3) (C6 1.2) (C6 0.3) (E6 0.3) (G6 0.3) (B♭6 0.3) (C7 0.9)
[TP2]	Trumpet	(B5 0.25) (B♭5 0.25) (B5 0.5) (C♯6 0.5) (D6 0.5)
[TP3]	Trumpet	(C♯6 0.25) (D♯6 0.25) (C♯6 0.25) (D♯6 0.25) (F6 1.0)
[TP4]	Trumpet	(G6 0.25) (F♯6 0.25) (G6 0.5) (A6 0.5) (B♭6 0.5)
[TP5]	Trumpet	(G4 0.3) (D5 0.3) (G5 0.3) (B5 0.3) (G5 0.3) (D5 0.3) (G4 0.3) (D5 0.3) (G5 0.3) (B5 0.3)
[TP6]	Trumpet	(C♯6 0.37) (D♯6 0.186) (F6 0.186) (D♯6 0.37) (F6 0.186) (F♯6 0.186)
[TP7]	Trumpet	(G♯6 0.186) (F♯6 0.186) (F6 0.186) (D♯6 0.186) (C♯6 0.557)
[TP8]	Trumpet	(C♯6 0.186) (F6 0.186) (F♯6 0.186) (D♯6 0.186) (G♯6 0.557)
[TT1]	Tom Tom Drum	(A4 0.186) (A5 0.186) (A4 0.186) (A5 0.186) (A4 0.139) (A5 0.139) (A4 0.139) (A4 0.139)
[TT2]	Tom Tom Drum	(A4 0.093) (A5 0.093) (A4 0.093) (A5 0.093) (A4 0.093) (A5 0.093) (A4 0.093) (A5 0.093) (A4 0.093) (A5 0.093) (A4 0.093) (A5 0.093) (A4 0.093)

Appendix B

Search Spaces Referenced in the Text

This table lists the search spaces referenced in the text. Each entry contains an unique identifier followed by the list of earcons contained in the presentation. Each earcon is marked by its identifier which can be referenced in Appendix A.

Search Space Identifier	Num. of Earcons in the Search Space	List of Earcons in the Given Search Space
[SS1]	8	[SA1], [CE1], [DG1], [FL1], [BA1], [SG1], [SG2], [TP3]
[SS4]	12	[SA1], [CE1], [FL1], [FL2], [DG1], [SG1], [SG2], [TP1], [TP2], [TP3], [TP4], [BA1]
[SS5]	10	[SA1], [CE1], [FL1], [FL2], [DG1], [SG1], [SG2], [TP1], [TP2], [BA1]
[SS6]	6	[SA1], [DG1], [FL2], [BA1], [SG2], [TP2]
[SS7]	7	[TP4], [CE1], [DG1], [SA1], [SG1], [SG2], [BA1]
[SS8]	3	[FL3], [FL3], FL3]
[SS9]	3	[FL2], [FL2], [FL2]
[SS10]	3	[TP2], [SG1], [FL1]
[SS11]	3	[TP5], [SG3], [SG4]
[SS12]	3	[SA1], [BA1], [SG2]
[SS13]	3	[TP1], [FL1], [FL1]
[SS14]	3	[TP1], [SG2], [CE1]

Search Space Identifier	Num. of Earcons in the Search Space	List of Earcons in the Given Search Space
[SS15]	13	[BA1], [CE1], [CE2], [CE3], [FL1], [FL5], [SA2], [SA4], [SG2], [SG3], [SG7], [TP6], [TP7]
[SS16]	14	[BA1], [BA2], [BA3], [BA4], [CE1], [CE2], [CE4], [DG1], [DG2], [FL5], [SG3], [TP8], [TT1], [TT2]
[SS17]	20	[BA3], [BA4], [CE1], [CE3], [CE5], [CE6], [DG2], [FL4], [FL5], [SA4], [SA6], [SA7], [SG4], [SG6], [SG7], [TP5], [TP6], [TP7], [TT1], [TT2]
[SS18]	15	[BA1], [BA1], [BA1], [BA1], [BA1], [FL1], [FL1], [FL1], [FL1], [FL1], [TP8], [TP8], [TP8], [TP8], [TP8]
[SS19]	12	[BA1], [BA2], [CE3], [CE4], [FL1], [FL4], [SA1], [SA3], [SG3], [TP2], [TP3], [TP4]
[SS20]	14	[BA1], [BA5], [BA6], [CE2], [CE3], [CE4], [SA5], [SA6], [SA8], [SG1], [SG2], [SG5], [TP2], [TT1]
[SS21]	22	[BA1], [BA4], [BA5], [BA6], [CE2], [CE3], [CE4], [CE5], [DG1], [SA1], [SA5], [SA6], [SA8], [SG1], [SG2], [SG4], [SG5], [TP2], [TP3], [TP4], [TP8], [TT1]
[SS22]	7	[BA4], [BA5], [CE2], [CE4], [CE5], [SA3], [SA4]
[SS23]	19	[BA1], [BA5], [CE1], [CE2], [CE3], [DG1], [DG2], [FL1], [FL5], [SA2], [SA4], [SG2], [SG3], [SG7], [TP2], [TP6], [TP7], [TT1], [TT2]
[SS24]	22	[BA1], [BA2], [BA3], [BA4], [CE1], [CE2], [CE4], [DG1], [DG2], [FL1], [FL5], [SA1], [SA2], [SA6], [SG1], [SG3], [SG4], [TP2], [TP4], [TP8]
[SS25]	10	[BA1], [BA5], [CE2], [CE3], [DG1], [FL1], [SA1], [SA8], [SG2], [SG4]

Appendix C

User Study Data

This table lists the examples given to each subject in the study. The first nine groups listed have an increasing average overlap, from approximately 1.0 to approximately 5.0, in 0.5 increments. The last two groups listed were used to evaluate the prediction function for user performance.

File Name of Example	Num of Earcons	Int Len (Secs.)	Int Len (Blks.)	Sum of Earcon Lens (Blks.)	Overlap Factor
ss22-9.3s-6.0.rnd	7	9.3	101	101	1.00
ss22-9.3s-51.7.rnd	7	9.3	101	101	1.00
ss22-9.3s-97.4.rnd	7	9.3	101	101	1.00
ss22-9.3s-143.1.rnd	7	9.3	101	101	1.00
ss22-9.3s-188.8.rnd	7	9.3	101	101	1.00
ss22-9.3s-234.5.rnd	7	9.3	101	101	1.00
ss25-11s-125.2.rnd	10	11.0	119	181	1.52
ss25-11s-294.5.rnd	10	11.0	119	181	1.52
ss25-11s-463.8.rnd	10	11.0	119	181	1.52
ss25-11s-633.0.rnd	10	11.0	119	181	1.52
ss25-11s-802.3.rnd	10	11.0	119	181	1.52
ss25-11s-971.6.rnd	10	11.0	119	181	1.52

File Name of Example	Num of Earcons	Int Len (Secs.)	Int Len (Blks.)	Sum of Earcon Lens (Blks.)	Overlap Factor
ss15-10s-287.8.rnd	13	10.0	109	209	1.92
ss15-10s-629.9.rnd	13	10.0	109	209	1.92
ss15-10s-972.0.rnd	13	10.0	109	209	1.92
ss15-10s-1314.1.rnd	13	10.0	109	209	1.92
ss15-10s-1656.2.rnd	13	10.0	109	209	1.92
ss15-10s-1998.3.rnd	13	10.0	109	209	1.92
ss19-8s-561.2.rnd	12	8.0	87	223	2.56
ss19-8s-1015.5.rnd	12	8.0	87	223	2.56
ss19-8s-1469.8.rnd	12	8.0	87	223	2.56
ss19-8s-1924.2.rnd	12	8.0	87	223	2.56
ss19-8s-2378.5.rnd	12	8.0	87	223	2.56
ss19-8s-2832.8.rnd	12	8.0	87	223	2.56
ss20-7s-726.3.rnd	14	7.0	76	234	3.08
ss20-7s-1208.1.rnd	14	7.0	76	234	3.08
ss20-7s-1689.9.rnd	14	7.0	76	234	3.08
ss20-7s-2171.8.rnd	14	7.0	76	234	3.08
ss20-7s-2653.6.rnd	14	7.0	76	234	3.08
ss20-7s-3135.4.rnd	14	7.0	76	234	3.08
ss21-10s-1473.6.rnd	22	10.0	109	380	3.49
ss21-10s-2402.3.rnd	22	10.0	109	380	3.49
ss21-10s-3331.1.rnd	22	10.0	109	380	3.49
ss21-10s-4259.8.rnd	22	10.0	109	380	3.49
ss21-10s-5188.6.rnd	22	10.0	109	380	3.49
ss21-10s-6117.3.rnd	22	10.0	109	380	3.49
ss17-7s-1493.5.rnd	20	7.0	76	312	4.11
ss17-7s-2360.7.rnd	20	7.0	76	312	4.11
ss17-7s-3228.0.rnd	20	7.0	76	312	4.11
ss17-7s-4095.2.rnd	20	7.0	76	312	4.11
ss17-7s-4962.5.rnd	20	7.0	76	312	4.11
ss17-7s-5829.7.rnd	20	7.0	76	312	4.11

File Name of Example	Num of Earcons	Int Len (Secs.)	Int Len (Blks.)	Sum of Earcon Lens (Blks.)	Overlap Factor
ss24-8s-2127.5.rnd	22	8.0	87	387	4.45
ss24-8s-3116.0.rnd	22	8.0	87	387	4.45
ss24-8s-4104.6.rnd	22	8.0	87	387	4.45
ss24-8s-5093.1.rnd	22	8.0	87	387	4.45
ss24-8s-6081.7.rnd	22	8.0	87	387	4.45
ss24-8s-7070.2.rnd	22	8.0	87	387	4.45
ss23-5.9s-1992.3.rnd	19	5.9	65	318	4.89
ss23-5.9s-2922.5.rnd	19	5.9	65	318	4.89
ss23-5.9s-3852.7.rnd	19	5.9	65	318	4.89
ss23-5.9s-4783.0.rnd	19	5.9	65	318	4.89
ss23-5.9s-5713.2.rnd	19	5.9	65	318	4.89
ss23-5.9s-6643.4.rnd	19	5.9	65	318	4.89
ss18-13s-318.0.rnd	15	13.0	141	260	1.84
ss18-13s-844.0.rnd	15	13.0	141	260	1.84
ss18-13s-1370.0.rnd	15	13.0	141	260	1.84
ss18-13s-1895.9.rnd	15	13.0	141	260	1.84
ss18-13s-2421.9.rnd	15	13.0	141	260	1.84
ss18-13s-2947.9.rnd	15	13.0	141	260	1.84
ss16-7s-681.3.rnd	14	7.0	76	233	3.07
ss16-7s-1178.1.rnd	14	7.0	76	233	3.07
ss16-7s-1674.9.rnd	14	7.0	76	233	3.07
ss16-7s-2171.8.rnd	14	7.0	76	233	3.07
ss16-7s-2668.6.rnd	14	7.0	76	233	3.07
ss16-7s-3165.4.rnd	14	7.0	76	233	3.07

The following table is a listing of the raw data obtained from the user study described in Chapter 5. The file names for each example are created as follows: the search space number (i.e., ss22) is entered, followed by a dash. Then, the length of the time interval is entered in seconds, (i.e., 9.3) followed by the letter *s* and a dash. Finally, the computed score for the example is entered (i.e., 006.0) followed by the extension *.rnd.* In the table, the lengths of time intervals and groups of earcons are expressed in blocks, where each block is 4096/44100 seconds.

File Name of Example	Comp. Score	Num. Heard	Real Num.	Int. Length	Earcon Len. Sum	User Number
ss22-9.3s-006.0.rnd	6.00	7	7	101	101	1
ss22-9.3s-006.0.rnd	6.00	7	7	101	101	2
ss22-9.3s-006.0.rnd	6.00	7	7	101	101	3
ss22-9.3s-006.0.rnd	6.00	7	7	101	101	4
ss22-9.3s-006.0.rnd	6.00	7	7	101	101	5
ss22-9.3s-006.0.rnd	6.00	7	7	101	101	6
ss22-9.3s-006.0.rnd	6.00	7	7	101	101	7
ss22-9.3s-006.0.rnd	6.00	7	7	101	101	8
ss22-9.3s-051.7.rnd	51.71	7	7	101	101	1
ss22-9.3s-051.7.rnd	51.71	8	7	101	101	2
ss22-9.3s-051.7.rnd	51.71	7	7	101	101	3
ss22-9.3s-051.7.rnd	51.71	7	7	101	101	4
ss22-9.3s-051.7.rnd	51.71	6	7	101	101	5
ss22-9.3s-051.7.rnd	51.71	7	7	101	101	6
ss22-9.3s-051.7.rnd	51.71	7	7	101	101	7
ss22-9.3s-051.7.rnd	51.71	7	7	101	101	8
ss22-9.3s-097.4.rnd	97.42	7	7	101	101	1
ss22-9.3s-097.4.rnd	97.42	6	7	101	101	2
ss22-9.3s-097.4.rnd	97.42	7	7	101	101	3
ss22-9.3s-097.4.rnd	97.42	6	7	101	101	4
ss22-9.3s-097.4.rnd	97.42	7	7	101	101	5
ss22-9.3s-097.4.rnd	97.42	7	7	101	101	6
ss22-9.3s-097.4.rnd	97.42	6	7	101	101	7
ss22-9.3s-097.4.rnd	97.42	8	7	101	101	8
ss22-9.3s-143.1.rnd	143.07	7	7	101	101	1
ss22-9.3s-143.1.rnd	143.07	7	7	101	101	2
ss22-9.3s-143.1.rnd	143.07	6	7	101	101	3
ss22-9.3s-143.1.rnd	143.07	7	7	101	101	4
ss22-9.3s-143.1.rnd	143.07	7	7	101	101	5
ss22-9.3s-143.1.rnd	143.07	7	7	101	101	6
ss22-9.3s-143.1.rnd	143.07	7	7	101	101	7
ss22-9.3s-143.1.rnd	143.07	7	7	101	101	8
ss22-9.3s-188.8.rnd	188.82	6	7	101	101	1
ss22-9.3s-188.8.rnd	188.82	7	7	101	101	2
ss22-9.3s-188.8.rnd	188.82	7	7	101	101	3
ss22-9.3s-188.8.rnd	188.82	6	7	101	101	4
ss22-9.3s-188.8.rnd	188.82	6	7	101	101	5

File Name of Example	Comp. Score	Num. Heard	Real Num.	Int. Length	Earcon Len. Sum	User Number
ss22-9.3s-188.8.rnd	188.82	6	7	101	101	6
ss22-9.3s-188.8.rnd	188.82	6	7	101	101	7
ss22-9.3s-188.8.rnd	188.82	6	7	101	101	8
ss22-9.3s-234.5.rnd	234.51	6	7	101	101	1
ss22-9.3s-234.5.rnd	234.51	8	7	101	101	2
ss22-9.3s-234.5.rnd	234.51	6	7	101	101	3
ss22-9.3s-234.5.rnd	234.51	6	7	101	101	4
ss22-9.3s-234.5.rnd	234.51	5	7	101	101	5
ss22-9.3s-234.5.rnd	234.51	6	7	101	101	6
ss22-9.3s-234.5.rnd	234.51	6	7	101	101	7
ss22-9.3s-234.5.rnd	234.51	5	7	101	101	8
ss25-11s-125.2.rnd	125.16	10	10	119	181	1
ss25-11s-125.2.rnd	125.16	10	10	119	181	2
ss25-11s-125.2.rnd	125.16	10	10	119	181	3
ss25-11s-125.2.rnd	125.16	10	10	119	181	4
ss25-11s-125.2.rnd	125.16	10	10	119	181	5
ss25-11s-125.2.rnd	125.16	10	10	119	181	6
ss25-11s-125.2.rnd	125.16	10	10	119	181	7
ss25-11s-125.2.rnd	125.16	9	10	119	181	8
ss25-11s-294.5.rnd	294.51	10	10	119	181	1
ss25-11s-294.5.rnd	294.51	9	10	119	181	2
ss25-11s-294.5.rnd	294.51	8	10	119	181	3
ss25-11s-294.5.rnd	294.51	9	10	119	181	4
ss25-11s-294.5.rnd	294.51	10	10	119	181	5
ss25-11s-294.5.rnd	294.51	10	10	119	181	6
ss25-11s-294.5.rnd	294.51	10	10	119	181	7
ss25-11s-294.5.rnd	294.51	9	10	119	181	8
ss25-11s-463.8.rnd	463.78	9	10	119	181	1
ss25-11s-463.8.rnd	463.78	10	10	119	181	2
ss25-11s-463.8.rnd	463.78	8	10	119	181	3
ss25-11s-463.8.rnd	463.78	8	10	119	181	4
ss25-11s-463.8.rnd	463.78	11	10	119	181	5
ss25-11s-463.8.rnd	463.78	8	10	119	181	6
ss25-11s-463.8.rnd	463.78	9	10	119	181	7
ss25-11s-463.8.rnd	463.78	8	10	119	181	8
ss25-11s-633.0.rnd	633.02	8	10	119	181	1
ss25-11s-633.0.rnd	633.02	8	10	119	181	2
ss25-11s-633.0.rnd	633.02	7	10	119	181	3

File Name of Example	Comp. Score	Num. Heard	Real Num.	Int. Length	Earcon Len. Sum	User Number
ss25-11s-633.0.rnd	633.02	7	10	119	181	4
ss25-11s-633.0.rnd	633.02	7	10	119	181	5
ss25-11s-633.0.rnd	633.02	8	10	119	181	6
ss25-11s-633.0.rnd	633.02	6	10	119	181	7
ss25-11s-633.0.rnd	633.02	8	10	119	181	8
ss25-11s-802.3.rnd	802.26	5	10	119	181	1
ss25-11s-802.3.rnd	802.26	6	10	119	181	2
ss25-11s-802.3.rnd	802.26	6	10	119	181	3
ss25-11s-802.3.rnd	802.26	7	10	119	181	4
ss25-11s-802.3.rnd	802.26	7	10	119	181	5
ss25-11s-802.3.rnd	802.26	6	10	119	181	6
ss25-11s-802.3.rnd	802.26	6	10	119	181	7
ss25-11s-802.3.rnd	802.26	6	10	119	181	8
ss25-11s-971.6.rnd	971.57	6	10	119	181	1
ss25-11s-971.6.rnd	971.57	7	10	119	181	2
ss25-11s-971.6.rnd	971.57	5	10	119	181	3
ss25-11s-971.6.rnd	971.57	6	10	119	181	4
ss25-11s-971.6.rnd	971.57	6	10	119	181	5
ss25-11s-971.6.rnd	971.57	6	10	119	181	6
ss25-11s-971.6.rnd	971.57	7	10	119	181	7
ss25-11s-971.6.rnd	971.57	6	10	119	181	8
ss18-13s-0318.0.rnd	318.01	15	15	141	260	1
ss18-13s-0318.0.rnd	318.01	12	15	141	260	2
ss18-13s-0318.0.rnd	318.01	14	15	141	260	3
ss18-13s-0318.0.rnd	318.01	11	15	141	260	4
ss18-13s-0318.0.rnd	318.01	13	15	141	260	5
ss18-13s-0318.0.rnd	318.01	14	15	141	260	6
ss18-13s-0318.0.rnd	318.01	12	15	141	260	7
ss18-13s-0318.0.rnd	318.01	10	15	141	260	8
ss18-13s-0844.0.rnd	844.04	10	15	141	260	1
ss18-13s-0844.0.rnd	844.04	12	15	141	260	2
ss18-13s-0844.0.rnd	844.04	12	15	141	260	3
ss18-13s-0844.0.rnd	844.04	9	15	141	260	4
ss18-13s-0844.0.rnd	844.04	15	15	141	260	5
ss18-13s-0844.0.rnd	844.04	10	15	141	260	6
ss18-13s-0844.0.rnd	844.04	11	15	141	260	7
ss18-13s-0844.0.rnd	844.04	12	15	141	260	8
ss18-13s-1370.0.rnd	1370.00	10	15	141	260	1

File Name of Example	Comp. Score	Num. Heard	Real Num.	Int. Length	Earcon Len. Sum	User Number
ss18-13s-1370.0.rnd	1370.00	10	15	141	260	2
ss18-13s-1370.0.rnd	1370.00	11	15	141	260	3
ss18-13s-1370.0.rnd	1370.00	8	15	141	260	4
ss18-13s-1370.0.rnd	1370.00	10	15	141	260	5
ss18-13s-1370.0.rnd	1370.00	10	15	141	260	6
ss18-13s-1370.0.rnd	1370.00	9	15	141	260	7
ss18-13s-1370.0.rnd	1370.00	11	15	141	260	8
ss18-13s-1895.9.rnd	1895.94	10	15	141	260	1
ss18-13s-1895.9.rnd	1895.94	9	15	141	260	2
ss18-13s-1895.9.rnd	1895.94	9	15	141	260	3
ss18-13s-1895.9.rnd	1895.94	8	15	141	260	4
ss18-13s-1895.9.rnd	1895.94	8	15	141	260	5
ss18-13s-1895.9.rnd	1895.94	10	15	141	260	6
ss18-13s-1895.9.rnd	1895.94	8	15	141	260	7
ss18-13s-1895.9.rnd	1895.94	11	15	141	260	8
ss18-13s-2421.9.rnd	2421.93	6	15	141	260	1
ss18-13s-2421.9.rnd	2421.93	10	15	141	260	2
ss18-13s-2421.9.rnd	2421.93	9	15	141	260	3
ss18-13s-2421.9.rnd	2421.93	8	15	141	260	4
ss18-13s-2421.9.rnd	2421.93	9	15	141	260	5
ss18-13s-2421.9.rnd	2421.93	8	15	141	260	6
ss18-13s-2421.9.rnd	2421.93	8	15	141	260	7
ss18-13s-2421.9.rnd	2421.93	9	15	141	260	8
ss18-13s-2947.9.rnd	2947.93	8	15	141	260	1
ss18-13s-2947.9.rnd	2947.93	10	15	141	260	2
ss18-13s-2947.9.rnd	2947.93	10	15	141	260	3
ss18-13s-2947.9.rnd	2947.93	7	15	141	260	4
ss18-13s-2947.9.rnd	2947.93	9	15	141	260	5
ss18-13s-2947.9.rnd	2947.93	8	15	141	260	6
ss18-13s-2947.9.rnd	2947.93	9	15	141	260	7
ss18-13s-2947.9.rnd	2947.93	8	15	141	260	8
ss15-10s-0287.8.rnd	287.77	13	13	109	209	1
ss15-10s-0287.8.rnd	287.77	12	13	109	209	2
ss15-10s-0287.8.rnd	287.77	12	13	109	209	3
ss15-10s-0287.8.rnd	287.77	7	13	109	209	4
ss15-10s-0287.8.rnd	287.77	8	13	109	209	5
ss15-10s-0287.8.rnd	287.77	13	13	109	209	6
ss15-10s-0287.8.rnd	287.77	12	13	109	209	7

File Name of Example	Comp. Score	Num. Heard	Real Num.	Int. Length	Earcon Len. Sum	User Number
ss15-10s-0287.8.rnd	287.77	10	13	109	209	8
ss15-10s-0629.9.rnd	629.89	11	13	109	209	1
ss15-10s-0629.9.rnd	629.89	10	13	109	209	2
ss15-10s-0629.9.rnd	629.89	11	13	109	209	3
ss15-10s-0629.9.rnd	629.89	8	13	109	209	4
ss15-10s-0629.9.rnd	629.89	12	13	109	209	5
ss15-10s-0629.9.rnd	629.89	13	13	109	209	6
ss15-10s-0629.9.rnd	629.89	10	13	109	209	7
ss15-10s-0629.9.rnd	629.89	10	13	109	209	8
ss15-10s-0972.0.rnd	972.01	7	13	109	209	1
ss15-10s-0972.0.rnd	972.01	8	13	109	209	2
ss15-10s-0972.0.rnd	972.01	8	13	109	209	3
ss15-10s-0972.0.rnd	972.01	8	13	109	209	4
ss15-10s-0972.0.rnd	972.01	7	13	109	209	5
ss15-10s-0972.0.rnd	972.01	9	13	109	209	6
ss15-10s-0972.0.rnd	972.01	8	13	109	209	7
ss15-10s-0972.0.rnd	972.01	6	13	109	209	8
ss15-10s-1314.1.rnd	1314.12	9	13	109	209	1
ss15-10s-1314.1.rnd	1314.12	6	13	109	209	2
ss15-10s-1314.1.rnd	1314.12	7	13	109	209	3
ss15-10s-1314.1.rnd	1314.12	7	13	109	209	4
ss15-10s-1314.1.rnd	1314.12	7	13	109	209	5
ss15-10s-1314.1.rnd	1314.12	8	13	109	209	6
ss15-10s-1314.1.rnd	1314.12	8	13	109	209	7
ss15-10s-1314.1.rnd	1314.12	7	13	109	209	8
ss15-10s-1656.2.rnd	1656.18	7	13	109	209	1
ss15-10s-1656.2.rnd	1656.18	9	13	109	209	2
ss15-10s-1656.2.rnd	1656.18	6	13	109	209	3
ss15-10s-1656.2.rnd	1656.18	8	13	109	209	4
ss15-10s-1656.2.rnd	1656.18	8	13	109	209	5
ss15-10s-1656.2.rnd	1656.18	7	13	109	209	6
ss15-10s-1656.2.rnd	1656.18	8	13	109	209	7
ss15-10s-1656.2.rnd	1656.18	8	13	109	209	8
ss15-10s-1998.3.rnd	1988.32	6	13	109	209	1
ss15-10s-1998.3.rnd	1988.32	9	13	109	209	2
ss15-10s-1998.3.rnd	1988.32	7	13	109	209	3
ss15-10s-1998.3.rnd	1988.32	6	13	109	209	4
ss15-10s-1998.3.rnd	1988.32	7	13	109	209	5

File Name of Example	Comp. Score	Num. Heard	Real Num.	Int. Length	Earcon Len. Sum	User Number
ss15-10s-1998.3.rnd	1988.32	7	13	109	209	6
ss15-10s-1998.3.rnd	1988.32	8	13	109	209	7
ss15-10s-1998.3.rnd	1988.32	8	13	109	209	8
ss19-8s-0561.2.rnd	561.25	11	12	87	223	1
ss19-8s-0561.2.rnd	561.25	7	12	87	223	2
ss19-8s-0561.2.rnd	561.25	8	12	87	223	3
ss19-8s-0561.2.rnd	561.25	7	12	87	223	4
ss19-8s-0561.2.rnd	561.25	10	12	87	223	5
ss19-8s-0561.2.rnd	561.25	10	12	87	223	6
ss19-8s-0561.2.rnd	561.25	8	12	87	223	7
ss19-8s-0561.2.rnd	561.25	10	12	87	223	8
ss19-8s-1015.5.rnd	1015.52	10	12	87	223	1
ss19-8s-1015.5.rnd	1015.52	6	12	87	223	2
ss19-8s-1015.5.rnd	1015.52	7	12	87	223	3
ss19-8s-1015.5.rnd	1015.52	7	12	87	223	4
ss19-8s-1015.5.rnd	1015.52	7	12	87	223	5
ss19-8s-1015.5.rnd	1015.52	8	12	87	223	6
ss19-8s-1015.5.rnd	1015.52	8	12	87	223	7
ss19-8s-1015.5.rnd	1015.52	7	12	87	223	8
ss19-8s-1469.8.rnd	1469.83	8	12	87	223	1
ss19-8s-1469.8.rnd	1469.83	4	12	87	223	2
ss19-8s-1469.8.rnd	1469.83	7	12	87	223	3
ss19-8s-1469.8.rnd	1469.83	5	12	87	223	4
ss19-8s-1469.8.rnd	1469.83	6	12	87	223	5
ss19-8s-1469.8.rnd	1469.83	7	12	87	223	6
ss19-8s-1469.8.rnd	1469.83	5	12	87	223	7
ss19-8s-1469.8.rnd	1469.83	6	12	87	223	8
ss19-8s-1924.2.rnd	1924.22	3	12	87	223	1
ss19-8s-1924.2.rnd	1924.22	6	12	87	223	2
ss19-8s-1924.2.rnd	1924.22	5	12	87	223	3
ss19-8s-1924.2.rnd	1924.22	4	12	87	223	4
ss19-8s-1924.2.rnd	1924.22	6	12	87	223	5
ss19-8s-1924.2.rnd	1924.22	5	12	87	223	6
ss19-8s-1924.2.rnd	1924.22	5	12	87	223	7
ss19-8s-1924.2.rnd	1924.22	6	12	87	223	8
ss19-8s-2378.5.rnd	2378.47	4	12	87	223	1
ss19-8s-2378.5.rnd	2378.47	6	12	87	223	2
ss19-8s-2378.5.rnd	2378.47	5	12	87	223	3

File Name of Example	Comp. Score	Num. Heard	Real Num.	Int. Length	Earcon Len. Sum	User Number
ss19-8s-2378.5.rnd	2378.47	5	12	87	223	4
ss19-8s-2378.5.rnd	2378.47	6	12	87	223	5
ss19-8s-2378.5.rnd	2378.47	5	12	87	223	6
ss19-8s-2378.5.rnd	2378.47	5	12	87	223	7
ss19-8s-2378.5.rnd	2378.47	6	12	87	223	8
ss19-8s-2832.8.rnd	2832.83	5	12	87	223	1
ss19-8s-2832.8.rnd	2832.83	5	12	87	223	2
ss19-8s-2832.8.rnd	2832.83	5	12	87	223	3
ss19-8s-2832.8.rnd	2832.83	4	12	87	223	4
ss19-8s-2832.8.rnd	2832.83	5	12	87	223	5
ss19-8s-2832.8.rnd	2832.83	6	12	87	223	6
ss19-8s-2832.8.rnd	2832.83	6	12	87	223	7
ss19-8s-2832.8.rnd	2832.83	5	12	87	223	8
ss16-7s-0681.3.rnd	681.33	12	14	76	233	1
ss16-7s-0681.3.rnd	681.33	6	14	76	233	2
ss16-7s-0681.3.rnd	681.33	8	14	76	233	3
ss16-7s-0681.3.rnd	681.33	8	14	76	233	4
ss16-7s-0681.3.rnd	681.33	8	14	76	233	5
ss16-7s-0681.3.rnd	681.33	13	14	76	233	6
ss16-7s-0681.3.rnd	681.33	7	14	76	233	7
ss16-7s-0681.3.rnd	681.33	6	14	76	233	8
ss16-7s-1178.1.rnd	1178.10	7	14	76	233	1
ss16-7s-1178.1.rnd	1178.10	7	14	76	233	2
ss16-7s-1178.1.rnd	1178.10	8	14	76	233	3
ss16-7s-1178.1.rnd	1178.10	6	14	76	233	4
ss16-7s-1178.1.rnd	1178.10	7	14	76	233	5
ss16-7s-1178.1.rnd	1178.10	10	14	76	233	6
ss16-7s-1178.1.rnd	1178.10	7	14	76	233	7
ss16-7s-1178.1.rnd	1178.10	7	14	76	233	8
ss16-7s-1674.9.rnd	1674.86	7	14	76	233	1
ss16-7s-1674.9.rnd	1674.86	7	14	76	233	2
ss16-7s-1674.9.rnd	1674.86	6	14	76	233	3
ss16-7s-1674.9.rnd	1674.86	4	14	76	233	4
ss16-7s-1674.9.rnd	1674.86	6	14	76	233	5
ss16-7s-1674.9.rnd	1674.86	6	14	76	233	6
ss16-7s-1674.9.rnd	1674.86	6	14	76	233	7
ss16-7s-1674.9.rnd	1674.86	7	14	76	233	8
ss16-7s-2171.8.rnd	2171.84	5	14	76	233	1

File Name of Example	Comp. Score	Num. Heard	Real Num.	Int. Length	Earcon Len. Sum	User Number
ss16-7s-2171.8.rnd	2171.84	7	14	76	233	2
ss16-7s-2171.8.rnd	2171.84	6	14	76	233	3
ss16-7s-2171.8.rnd	2171.84	5	14	76	233	4
ss16-7s-2171.8.rnd	2171.84	8	14	76	233	5
ss16-7s-2171.8.rnd	2171.84	9	14	76	233	6
ss16-7s-2171.8.rnd	2171.84	5	14	76	233	7
ss16-7s-2171.8.rnd	2171.84	8	14	76	233	8
ss16-7s-2668.6.rnd	2668.55	4	14	76	233	1
ss16-7s-2668.6.rnd	2668.55	6	14	76	233	2
ss16-7s-2668.6.rnd	2668.55	6	14	76	233	3
ss16-7s-2668.6.rnd	2668.55	6	14	76	233	4
ss16-7s-2668.6.rnd	2668.55	6	14	76	233	5
ss16-7s-2668.6.rnd	2668.55	5	14	76	233	6
ss16-7s-2668.6.rnd	2668.55	6	14	76	233	7
ss16-7s-2668.6.rnd	2668.55	7	14	76	233	8
ss16-7s-3165.4.rnd	3165.36	4	14	76	233	1
ss16-7s-3165.4.rnd	3165.36	6	14	76	233	2
ss16-7s-3165.4.rnd	3165.36	5	14	76	233	3
ss16-7s-3165.4.rnd	3165.36	5	14	76	233	4
ss16-7s-3165.4.rnd	3165.36	6	14	76	233	5
ss16-7s-3165.4.rnd	3165.36	4	14	76	233	6
ss16-7s-3165.4.rnd	3165.36	6	14	76	233	7
ss16-7s-3165.4.rnd	3165.36	5	14	76	233	8
ss20-7s-0726.3.rnd	726.32	10	14	76	234	1
ss20-7s-0726.3.rnd	726.32	6	14	76	234	2
ss20-7s-0726.3.rnd	726.32	7	14	76	234	3
ss20-7s-0726.3.rnd	726.32	6	14	76	234	4
ss20-7s-0726.3.rnd	726.32	9	14	76	234	5
ss20-7s-0726.3.rnd	726.32	11	14	76	234	6
ss20-7s-0726.3.rnd	726.32	7	14	76	234	7
ss20-7s-0726.3.rnd	726.32	7	14	76	234	8
ss20-7s-1208.1.rnd	1208.13	8	14	76	234	1
ss20-7s-1208.1.rnd	1208.13	7	14	76	234	2
ss20-7s-1208.1.rnd	1208.13	5	14	76	234	3
ss20-7s-1208.1.rnd	1208.13	5	14	76	234	4
ss20-7s-1208.1.rnd	1208.13	7	14	76	234	5
ss20-7s-1208.1.rnd	1208.13	6	14	76	234	6
ss20-7s-1208.1.rnd	1208.13	7	14	76	234	7

File Name of Example	Comp. Score	Num. Heard	Real Num.	Int. Length	Earcon Len. Sum	User Number
ss20-7s-1208.1.rnd	1208.13	4	14	76	234	8
ss20-7s-1689.9.rnd	1689.94	7	14	76	234	1
ss20-7s-1689.9.rnd	1689.94	6	14	76	234	2
ss20-7s-1689.9.rnd	1689.94	7	14	76	234	3
ss20-7s-1689.9.rnd	1689.94	6	14	76	234	4
ss20-7s-1689.9.rnd	1689.94	8	14	76	234	5
ss20-7s-1689.9.rnd	1689.94	7	14	76	234	6
ss20-7s-1689.9.rnd	1689.94	7	14	76	234	7
ss20-7s-1689.9.rnd	1689.94	6	14	76	234	8
ss20-7s-2171.8.rnd	2171.78	6	14	76	234	1
ss20-7s-2171.8.rnd	2171.78	5	14	76	234	2
ss20-7s-2171.8.rnd	2171.78	6	14	76	234	3
ss20-7s-2171.8.rnd	2171.78	6	14	76	234	4
ss20-7s-2171.8.rnd	2171.78	6	14	76	234	5
ss20-7s-2171.8.rnd	2171.78	6	14	76	234	6
ss20-7s-2171.8.rnd	2171.78	4	14	76	234	7
ss20-7s-2171.8.rnd	2171.78	7	14	76	234	8
ss20-7s-2653.6.rnd	2653.60	8	14	76	234	1
ss20-7s-2653.6.rnd	2653.60	5	14	76	234	2
ss20-7s-2653.6.rnd	2653.60	5	14	76	234	3
ss20-7s-2653.6.rnd	2653.60	5	14	76	234	4
ss20-7s-2653.6.rnd	2653.60	6	14	76	234	5
ss20-7s-2653.6.rnd	2653.60	5	14	76	234	6
ss20-7s-2653.6.rnd	2653.60	5	14	76	234	7
ss20-7s-2653.6.rnd	2653.60	5	14	76	234	8
ss20-7s-3135.4.rnd	3135.40	4	14	76	234	1
ss20-7s-3135.4.rnd	3135.40	5	14	76	234	2
ss20-7s-3135.4.rnd	3135.40	5	14	76	234	3
ss20-7s-3135.4.rnd	3135.40	4	14	76	234	4
ss20-7s-3135.4.rnd	3135.40	4	14	76	234	5
ss20-7s-3135.4.rnd	3135.40	5	14	76	234	6
ss20-7s-3135.4.rnd	3135.40	4	14	76	234	7
ss20-7s-3135.4.rnd	3135.40	6	14	76	234	8
ss21-10s-1473.6.rnd	1473.55	15	22	109	380	1
ss21-10s-1473.6.rnd	1473.55	9	22	109	380	2
ss21-10s-1473.6.rnd	1473.55	11	22	109	380	3
ss21-10s-1473.6.rnd	1473.55	7	22	109	380	4
ss21-10s-1473.6.rnd	1473.55	9	22	109	380	5

File Name of Example	Comp. Score	Num. Heard	Real Num.	Int. Length	Earcon Len. Sum	User Number
ss21-10s-1473.6.rnd	1473.55	15	22	109	380	6
ss21-10s-1473.6.rnd	1473.55	9	22	109	380	7
ss21-10s-1473.6.rnd	1473.55	9	22	109	380	8
ss21-10s-2402.3.rnd	2402.25	10	22	109	380	1
ss21-10s-2402.3.rnd	2402.25	12	22	109	380	2
ss21-10s-2402.3.rnd	2402.25	10	22	109	380	3
ss21-10s-2402.3.rnd	2402.25	9	22	109	380	4
ss21-10s-2402.3.rnd	2402.25	10	22	109	380	5
ss21-10s-2402.3.rnd	2402.25	10	22	109	380	6
ss21-10s-2402.3.rnd	2402.25	11	22	109	380	7
ss21-10s-2402.3.rnd	2402.25	9	22	109	380	8
ss21-10s-3331.1.rnd	3331.12	7	22	109	380	1
ss21-10s-3331.1.rnd	3331.12	10	22	109	380	2
ss21-10s-3331.1.rnd	3331.12	9	22	109	380	3
ss21-10s-3331.1.rnd	3331.12	8	22	109	380	4
ss21-10s-3331.1.rnd	3331.12	8	22	109	380	5
ss21-10s-3331.1.rnd	3331.12	8	22	109	380	6
ss21-10s-3331.1.rnd	3331.12	10	22	109	380	7
ss21-10s-3331.1.rnd	3331.12	10	22	109	380	8
ss21-10s-4259.8.rnd	4259.81	8	22	109	380	1
ss21-10s-4259.8.rnd	4259.81	8	22	109	380	2
ss21-10s-4259.8.rnd	4259.81	6	22	109	380	3
ss21-10s-4259.8.rnd	4259.81	6	22	109	380	4
ss21-10s-4259.8.rnd	4259.81	7	22	109	380	5
ss21-10s-4259.8.rnd	4259.81	7	22	109	380	6
ss21-10s-4259.8.rnd	4259.81	7	22	109	380	7
ss21-10s-4259.8.rnd	4259.81	8	22	109	380	8
ss21-10s-5188.6.rnd	5188.57	8	22	109	380	1
ss21-10s-5188.6.rnd	5188.57	11	22	109	380	2
ss21-10s-5188.6.rnd	5188.57	8	22	109	380	3
ss21-10s-5188.6.rnd	5188.57	8	22	109	380	4
ss21-10s-5188.6.rnd	5188.57	9	22	109	380	5
ss21-10s-5188.6.rnd	5188.57	8	22	109	380	6
ss21-10s-5188.6.rnd	5188.57	9	22	109	380	7
ss21-10s-5188.6.rnd	5188.57	9	22	109	380	8
ss21-10s-6117.3.rnd	6117.26	7	22	109	380	1
ss21-10s-6117.3.rnd	6117.26	9	22	109	380	2
ss21-10s-6117.3.rnd	6117.26	9	22	109	380	3

File Name of Example	Comp. Score	Num. Heard	Real Num.	Int. Length	Earcon Len. Sum	User Number
ss21-10s-6117.3.rnd	6117.26	9	22	109	380	4
ss21-10s-6117.3.rnd	6117.26	8	22	109	380	5
ss21-10s-6117.3.rnd	6117.26	7	22	109	380	6
ss21-10s-6117.3.rnd	6117.26	9	22	109	380	7
ss21-10s-6117.3.rnd	6117.26	9	22	109	380	8
ss17-7s-1493.5.rnd	1493.51	10	20	76	312	1
ss17-7s-1493.5.rnd	1493.51	12	20	76	312	2
ss17-7s-1493.5.rnd	1493.51	7	20	76	312	3
ss17-7s-1493.5.rnd	1493.51	6	20	76	312	4
ss17-7s-1493.5.rnd	1493.51	9	20	76	312	5
ss17-7s-1493.5.rnd	1493.51	9	20	76	312	6
ss17-7s-1493.5.rnd	1493.51	9	20	76	312	7
ss17-7s-1493.5.rnd	1493.51	8	20	76	312	8
ss17-7s-2360.7.rnd	2360.67	8	20	76	312	1
ss17-7s-2360.7.rnd	2360.67	7	20	76	312	2
ss17-7s-2360.7.rnd	2360.67	5	20	76	312	3
ss17-7s-2360.7.rnd	2360.67	6	20	76	312	4
ss17-7s-2360.7.rnd	2360.67	6	20	76	312	5
ss17-7s-2360.7.rnd	2360.67	7	20	76	312	6
ss17-7s-2360.7.rnd	2360.67	7	20	76	312	7
ss17-7s-2360.7.rnd	2360.67	7	20	76	312	8
ss17-7s-3228.0.rnd	3227.96	6	20	76	312	1
ss17-7s-3228.0.rnd	3227.96	5	20	76	312	2
ss17-7s-3228.0.rnd	3227.96	7	20	76	312	3
ss17-7s-3228.0.rnd	3227.96	5	20	76	312	4
ss17-7s-3228.0.rnd	3227.96	7	20	76	312	5
ss17-7s-3228.0.rnd	3227.96	6	20	76	312	6
ss17-7s-3228.0.rnd	3227.96	6	20	76	312	7
ss17-7s-3228.0.rnd	3227.96	7	20	76	312	8
ss17-7s-4095.2.rnd	4095.17	7	20	76	312	1
ss17-7s-4095.2.rnd	4095.17	8	20	76	312	2
ss17-7s-4095.2.rnd	4095.17	7	20	76	312	3
ss17-7s-4095.2.rnd	4095.17	7	20	76	312	4
ss17-7s-4095.2.rnd	4095.17	9	20	76	312	5
ss17-7s-4095.2.rnd	4095.17	9	20	76	312	6
ss17-7s-4095.2.rnd	4095.17	7	20	76	312	7
ss17-7s-4095.2.rnd	4095.17	7	20	76	312	8
ss17-7s-4962.5.rnd	4962.34	5	20	76	312	1

File Name of Example	Comp. Score	Num. Heard	Real Num.	Int. Length	Earcon Len. Sum	User Number
ss17-7s-4962.5.rnd	4962.34	7	20	76	312	2
ss17-7s-4962.5.rnd	4962.34	4	20	76	312	3
ss17-7s-4962.5.rnd	4962.34	5	20	76	312	4
ss17-7s-4962.5.rnd	4962.34	5	20	76	312	5
ss17-7s-4962.5.rnd	4962.34	6	20	76	312	6
ss17-7s-4962.5.rnd	4962.34	5	20	76	312	7
ss17-7s-4962.5.rnd	4962.34	4	20	76	312	8
ss17-7s-5829.7.rnd	5829.66	6	20	76	312	1
ss17-7s-5829.7.rnd	5829.66	5	20	76	312	2
ss17-7s-5829.7.rnd	5829.66	6	20	76	312	3
ss17-7s-5829.7.rnd	5829.66	5	20	76	312	4
ss17-7s-5829.7.rnd	5829.66	7	20	76	312	5
ss17-7s-5829.7.rnd	5829.66	6	20	76	312	6
ss17-7s-5829.7.rnd	5829.66	5	20	76	312	7
ss17-7s-5829.7.rnd	5829.66	6	20	76	312	8
ss24-8s-2127.5.rnd	2127.47	13	22	87	387	1
ss24-8s-2127.5.rnd	2127.47	9	22	87	387	2
ss24-8s-2127.5.rnd	2127.47	6	22	87	387	3
ss24-8s-2127.5.rnd	2127.47	7	22	87	387	4
ss24-8s-2127.5.rnd	2127.47	8	22	87	387	5
ss24-8s-2127.5.rnd	2127.47	10	22	87	387	6
ss24-8s-2127.5.rnd	2127.47	8	22	87	387	7
ss24-8s-2127.5.rnd	2127.47	8	22	87	387	8
ss24-8s-3116.0.rnd	3116.01	10	22	87	387	1
ss24-8s-3116.0.rnd	3116.01	12	22	87	387	2
ss24-8s-3116.0.rnd	3116.01	5	22	87	387	3
ss24-8s-3116.0.rnd	3116.01	6	22	87	387	4
ss24-8s-3116.0.rnd	3116.01	7	22	87	387	5
ss24-8s-3116.0.rnd	3116.01	7	22	87	387	6
ss24-8s-3116.0.rnd	3116.01	9	22	87	387	7
ss24-8s-3116.0.rnd	3116.01	8	22	87	387	8
ss24-8s-4104.6.rnd	4104.55	7	22	87	387	1
ss24-8s-4104.6.rnd	4104.55	6	22	87	387	2
ss24-8s-4104.6.rnd	4104.55	5	22	87	387	3
ss24-8s-4104.6.rnd	4104.55	6	22	87	387	4
ss24-8s-4104.6.rnd	4104.55	7	22	87	387	5
ss24-8s-4104.6.rnd	4104.55	8	22	87	387	6
ss24-8s-4104.6.rnd	4104.55	6	22	87	387	7

File Name of Example	Comp. Score	Num. Heard	Real Num.	Int. Length	Earcon Len. Sum	User Number
ss24-8s-4104.6.rnd	4104.55	7	22	87	387	8
ss24-8s-5093.1.rnd	5093.13	5	22	87	387	1
ss24-8s-5093.1.rnd	5093.13	5	22	87	387	2
ss24-8s-5093.1.rnd	5093.13	5	22	87	387	3
ss24-8s-5093.1.rnd	5093.13	5	22	87	387	4
ss24-8s-5093.1.rnd	5093.13	7	22	87	387	5
ss24-8s-5093.1.rnd	5093.13	5	22	87	387	6
ss24-8s-5093.1.rnd	5093.13	6	22	87	387	7
ss24-8s-5093.1.rnd	5093.13	7	22	87	387	8
ss24-8s-6081.7.rnd	6081.69	7	22	87	387	1
ss24-8s-6081.7.rnd	6081.69	4	22	87	387	2
ss24-8s-6081.7.rnd	6081.69	6	22	87	387	3
ss24-8s-6081.7.rnd	6081.69	7	22	87	387	4
ss24-8s-6081.7.rnd	6081.69	9	22	87	387	5
ss24-8s-6081.7.rnd	6081.69	6	22	87	387	6
ss24-8s-6081.7.rnd	6081.69	7	22	87	387	7
ss24-8s-6081.7.rnd	6081.69	8	22	87	387	8
ss24-8s-7070.2.rnd	7070.16	6	22	87	387	1
ss24-8s-7070.2.rnd	7070.16	7	22	87	387	2
ss24-8s-7070.2.rnd	7070.16	6	22	87	387	3
ss24-8s-7070.2.rnd	7070.16	6	22	87	387	4
ss24-8s-7070.2.rnd	7070.16	6	22	87	387	5
ss24-8s-7070.2.rnd	7070.16	6	22	87	387	6
ss24-8s-7070.2.rnd	7070.16	4	22	87	387	7
ss24-8s-7070.2.rnd	7070.16	5	22	87	387	8
ss23-5.9s-1992.3.rnd	1992.25	7	19	65	318	1
ss23-5.9s-1992.3.rnd	1992.25	5	19	65	318	2
ss23-5.9s-1992.3.rnd	1992.25	5	19	65	318	3
ss23-5.9s-1992.3.rnd	1992.25	5	19	65	318	4
ss23-5.9s-1992.3.rnd	1992.25	5	19	65	318	5
ss23-5.9s-1992.3.rnd	1992.25	7	19	65	318	6
ss23-5.9s-1992.3.rnd	1992.25	7	19	65	318	7
ss23-5.9s-1992.3.rnd	1992.25	5	19	65	318	8
ss23-5.9s-2922.5.rnd	2922.46	7	19	65	318	1
ss23-5.9s-2922.5.rnd	2922.46	5	19	65	318	2
ss23-5.9s-2922.5.rnd	2922.46	5	19	65	318	3
ss23-5.9s-2922.5.rnd	2922.46	5	19	65	318	4
ss23-5.9s-2922.5.rnd	2922.46	7	19	65	318	5

File Name of Example	Comp. Score	Num. Heard	Real Num.	Int. Length	Earcon Len. Sum	User Number
ss23-5.9s-2922.5.rnd	2922.46	7	19	65	318	6
ss23-5.9s-2922.5.rnd	2922.46	5	19	65	318	7
ss23-5.9s-2922.5.rnd	2922.46	6	19	65	318	8
ss23-5.9s-3852.7.rnd	3852.70	6	19	65	318	1
ss23-5.9s-3852.7.rnd	3852.70	7	19	65	318	2
ss23-5.9s-3852.7.rnd	3852.70	7	19	65	318	3
ss23-5.9s-3852.7.rnd	3852.70	5	19	65	318	4
ss23-5.9s-3852.7.rnd	3852.70	8	19	65	318	5
ss23-5.9s-3852.7.rnd	3852.70	7	19	65	318	6
ss23-5.9s-3852.7.rnd	3852.70	7	19	65	318	7
ss23-5.9s-3852.7.rnd	3852.70	7	19	65	318	8
ss23-5.9s-4783.0.rnd	4782.96	5	19	65	318	1
ss23-5.9s-4783.0.rnd	4782.96	7	19	65	318	2
ss23-5.9s-4783.0.rnd	4782.96	6	19	65	318	3
ss23-5.9s-4783.0.rnd	4782.96	6	19	65	318	4
ss23-5.9s-4783.0.rnd	4782.96	7	19	65	318	5
ss23-5.9s-4783.0.rnd	4782.96	6	19	65	318	6
ss23-5.9s-4783.0.rnd	4782.96	6	19	65	318	7
ss23-5.9s-4783.0.rnd	4782.96	5	19	65	318	8
ss23-5.9s-5713.2.rnd	5713.18	3	19	65	318	1
ss23-5.9s-5713.2.rnd	5713.18	6	19	65	318	2
ss23-5.9s-5713.2.rnd	5713.18	3	19	65	318	3
ss23-5.9s-5713.2.rnd	5713.18	3	19	65	318	4
ss23-5.9s-5713.2.rnd	5713.18	5	19	65	318	5
ss23-5.9s-5713.2.rnd	5713.18	4	19	65	318	6
ss23-5.9s-5713.2.rnd	5713.18	5	19	65	318	7
ss23-5.9s-5713.2.rnd	5713.18	6	19	65	318	8
ss23-5.9s-6643.4.rnd	6643.41	4	19	65	318	1
ss23-5.9s-6643.4.rnd	6643.41	7	19	65	318	2
ss23-5.9s-6643.4.rnd	6643.41	5	19	65	318	3
ss23-5.9s-6643.4.rnd	6643.41	6	19	65	318	4
ss23-5.9s-6643.4.rnd	6643.41	6	19	65	318	5
ss23-5.9s-6643.4.rnd	6643.41	4	19	65	318	6
ss23-5.9s-6643.4.rnd	6643.41	5	19	65	318	7
ss23-5.9s-6643.4.rnd	6643.41	5	19	65	318	8

The next table lists user scores along with the mean and standard deviations for each example. This information provides a measure of the variation of user performance in each example. All values in the table have been rounded to the nearest tenth to fit better on the page.

Comp. Score	User1 % Err	User2 % Err	User3 % Err	User4 % Err	User5 % Err	User6 % Err	User7 % Err	User8 % Err	Mean	S.D.
1.0 overlap, [SS22] in 9.3 sec.										
6.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
51.7	0.0	14.3	0.0	0.0	14.3	0.0	0.0	0.0	3.6	6.6
97.4	0.0	14.3	0.0	14.3	0.0	0.0	14.3	14.3	7.1	7.6
143.1	0.0	0.0	14.3	0.0	0.0	0.0	0.0	0.0	1.8	5.1
188.8	14.3	0.0	0.0	14.3	14.3	14.3	14.3	14.3	10.7	6.6
234.5	14.3	14.3	14.3	14.3	28.6	14.3	14.3	28.6	17.9	6.6
1.5 overlap, [SS25] in 11 sec.										
125.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	10.0	1.3	3.5
294.5	0.0	10.0	20.0	10.0	0.0	0.0	0.0	10.0	6.3	7.4
463.8	10.0	0.0	20.0	20.0	10.0	20.0	10.0	20.0	13.8	7.4
633.0	20.0	20.0	30.0	30.0	30.0	20.0	40.0	20.0	26.3	7.4
802.3	50.0	40.0	40.0	30.0	30.0	40.0	40.0	40.0	38.8	6.4
971.6	40.0	30.0	50.0	40.0	40.0	40.0	30.0	40.0	38.8	6.4
2.0 overlap, [SS15] in 10 sec.										
287.8	0.0	7.7	7.7	46.2	38.5	0.0	7.7	23.1	16.3	17.7
629.9	15.4	23.1	15.4	38.5	7.7	0.0	23.1	23.1	18.3	11.6
972.0	46.2	38.5	38.5	38.5	46.2	30.8	38.5	53.9	41.3	7.0
1314.1	30.8	53.9	46.2	46.2	46.2	38.5	38.5	46.2	43.3	7.0
1656.2	46.2	30.8	53.9	38.5	38.5	46.2	38.5	38.5	41.3	7.0
1998.3	53.9	30.8	46.2	53.9	46.2	46.2	38.5	38.5	44.2	8.0
2.5 overlap, [SS19] in 8 sec.										
561.2	8.3	41.7	33.3	41.7	16.7	16.7	33.3	16.7	26.0	12.9
1015.5	16.7	50.0	41.7	41.7	41.7	33.3	33.3	41.7	37.5	10.0
1469.8	33.3	66.7	41.7	58.3	50.0	41.7	58.3	50.0	50.0	10.9
1924.2	75.0	50.0	58.3	66.7	50.0	58.3	58.3	50.0	58.3	8.9
2378.5	66.7	50.0	58.3	58.3	50.0	58.3	58.3	50.0	56.2	5.9
2832.8	58.3	58.3	58.3	66.7	58.3	50.0	50.0	58.3	57.3	5.3

Comp. Score	User1 % Err	User2 % Err	User3 % Err	User4 % Err	User5 % Err	User6 % Err	User7 % Err	User8 % Err	Mean	S.D.
3.0 overlap, [SS20] in 7 sec.										
726.3	28.6	57.1	50.0	57.1	35.7	21.4	50.0	50.0	43.8	13.5
1208.1	42.9	50.0	64.3	64.3	50.0	57.1	50.0	71.4	56.2	9.7
1689.9	50.0	57.1	50.0	57.1	42.9	50.0	50.0	57.1	51.8	5.1
2171.8	57.1	64.3	57.1	57.1	57.1	57.1	71.4	50.0	58.9	6.3
2653.6	42.9	64.3	64.3	64.3	57.1	64.3	64.3	64.3	60.7	7.6
2832.8	71.4	64.3	64.3	71.4	71.4	64.3	71.4	57.1	67.0	5.3
3.5 overlap, [SS21] in 10 sec.										
1473.6	31.8	59.1	50.0	68.2	59.1	31.8	59.1	59.1	52.3	13.5
2402.3	54.5	45.5	54.5	59.1	54.5	54.5	50.0	59.1	54.0	4.5
3331.1	68.2	54.5	59.1	63.6	63.6	63.6	54.5	54.5	60.2	5.3
4259.8	63.6	63.6	72.7	72.7	68.2	68.2	68.2	63.6	67.6	3.8
5188.6	63.6	50.0	63.6	63.6	59.1	63.6	59.1	59.1	60.2	4.7
6117.3	68.2	59.1	59.1	59.1	63.6	68.2	59.1	59.1	61.9	4.2
4.0 overlap, [SS17] in 7 sec.										
1493.5	50.0	40.0	65.0	70.0	55.0	55.0	55.0	60.0	56.3	9.2
2360.7	60.0	65.0	75.0	70.0	70.0	65.0	65.0	65.0	66.9	4.6
3228.0	70.0	75.0	65.0	75.0	65.0	70.0	70.0	65.0	69.4	4.2
4095.2	65.0	60.0	65.0	65.0	55.0	55.0	65.0	65.0	61.9	4.6
4962.3	75.0	65.0	80.0	75.0	75.0	70.0	75.0	80.0	74.4	5.0
5829.7	70.0	75.0	70.0	75.0	65.0	70.0	75.0	70.0	71.3	3.5
4.5 overlap, [SS24] in 8 sec.										
2127.5	40.9	59.1	72.7	68.2	63.6	54.5	63.6	63.6	60.8	9.7
3116.0	54.5	45.5	77.3	72.7	68.2	68.2	59.1	63.6	63.6	10.3
4104.6	68.2	72.7	77.3	72.7	68.2	63.6	72.7	68.2	70.5	4.2
5093.1	77.3	77.3	77.3	77.3	68.2	77.3	72.7	68.2	74.4	4.2
6081.7	68.2	81.8	72.7	68.2	59.1	72.7	68.2	63.6	69.3	6.8
7070.2	72.7	68.2	72.7	72.7	72.7	72.7	81.8	77.3	73.9	4.0
5.0 overlap, [SS23] in 5.9 sec.										
1992.3	63.2	73.7	73.7	73.7	73.7	63.2	63.2	73.7	69.7	5.4
2922.5	63.2	73.7	73.7	73.7	63.2	63.2	73.7	68.4	69.1	5.2
3852.7	68.4	63.2	63.2	73.7	57.9	63.2	63.2	63.2	64.5	4.7
4783.0	73.7	63.2	68.4	68.4	63.2	68.4	68.4	73.7	68.4	4.0
5713.2	84.2	68.4	84.2	84.2	73.7	78.9	73.7	68.4	77.0	6.9
6643.4	78.9	63.2	73.7	68.4	68.4	78.9	73.7	73.7	72.4	5.4

Bibliography

[Alb94] M. C. Albers. The Varèse system: Hybrid auditory interfaces, and satellite-ground control: Using auditory icons and sonification in a complex, supervisory control system. In G. Kramer and S. Smith, editors, *Proceedings of the Second International Conference on Auditory Display, ICAD '94*, pages 1–13, Santa Fe, New Mexico 87501, 1994. Santa Fe Institute.

[All83] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[Aro91] B. Arons. The design of audio servers and toolkits for supporting speech in the user interface. *Journal of the American Voice I/O Society*, 9:27–41, March 1991.

[Bar94] S. Barrass. A perceptual framework for the auditory display of scientific data. In G. Kramer and S. Smith, editors, *Proceedings of the Second International Conference on Auditory Display, ICAD '94*, pages 131–145, Santa Fe, New Mexico 87501, 1994. Santa Fe Institute.

[Beg93] D. R. Begault. Call sign intelligibility using a spatial audio display. Technical Memorandum TM104014, National Aeronautics and Space Administration, 1993.

[Beg94a] D. R. Begault. *3D Sound for Virtual Reality and Multimedia.* AP Professional, Boston, MA, 1994.

[Beg94b] D. R. Begault. Head-up auditory displays for TCAS advisories: A preliminary investigation. *Human Factors*, 1994.

[BGK92] M. M. Blattner, R. M. Greenberg, and M. Kamegai. Listening to turbulence: An example of scientific audiolization. In M. M. Blattner and R. Dannenberg, editors, *Multimedia Interface Design*, Frontier, pages 87–102. ACM Press / Addison Wesley, New York, NY, 1992.

[Bly82] S. Bly. *Sound and Computer Information Presentation.* PhD thesis, University of California, Davis, Davis, CA, 1982. Published as Lawrence Livermore National Laboratory Technical Report UCRL-53282.

[BM91] M. W. Beavouis and R. Meddis. A computer model of auditory stream segregation. *Journal of Experimental Psychology*, 43:517–541, 1991.

[BPG94] M. M. Blattner, A. L. Papp III, and E. P. Glinert. Sonic enhancement of two-dimensional graphics displays. In G. Kramer, editor, *Auditory Display: Sonification, Audification, and Auditory Interfaces*, volume XVIII, pages 447–470, Reading, MA, 1994. Santa Fe Institute Studies in the Sciences of Complexity, The Addison-Wesley Publishing Company, The Advanced Book Program.

[Bra67] H. Brant. Space as an essential aspect of music composition. In E. Schwartz and B. Childs, editors, *Contemporary Composers on Contemporary Music.* Holt, Rinehart & Winston, New York, NY, 1967.

[Bre78] A. S. Bregman. Auditory streaming: Competition among alternative organizations. *Perception & Psychophysics*, 23:380–387, 1978.

[Bre90] A. S. Bregman. *Auditory Scene Analysis: The Perceptual Organization of Sound.* The MIT Press, Cambridge, MA, 1990.

[Bre94] S. A. Brewster. *Providing a Structured Method for Integrating Non-Speech Audio into Human-Computer Interfaces.* PhD thesis, University of York, Heslington, York, UK, 1994.

[Bre95] A. S. Bregman. Constraints on computational models of auditory scene analysis, as derived from human perception. *Journal of the Acoustical Society of Japan*, 16(3):133–136, May 1995.

[Bro92] G. J. Brown. *Computational Auditory Scene Analysis: A Representational Approach.* PhD thesis, University of Sheffield, Sheffield, UK, 1992.

[BSG89] M. M. Blattner, D. A. Sumikawa, and R. M. Greenberg. Earcons and icons: Their structure and common design principles. *Human-Computer Interaction*, 4(1):11–44, 1989.

[BW82] E. M. Burns and W. D. Ward. Intervals, scales, and tuning. In D. Deutsch, editor, *The Psychology of Music*, Academic Press Series in Cognition and Perception, pages 241–269. Academic Press, Inc. (Harcourt Brace Jovanovich, Publishers), Orlando, FL, 1982.

[BWE93] S. A. Brewster, P. C. Wright, and A. D. N. Edwards. An evaluation of earcons for use in auditory human-computer interfaces. In *Proceedings of the INTERCHI '93 Conference on Human Factors in Computer Systems*, Amsterdam, The Netherlands, April 1993.

[CB92] M. P. Cooke and G. J. Brown. Computational auditory scene analysis: Exploiting the continuity illusion. In J. Pittman, editor, *Proceedings of the Fourth Australian International Conference on Speech Science and Technology*, pages 8–13, 1992.

[Cho80] J. M. Chowning. Computer synthesis of the singing voice. In *Sound Generation in Winds, Strings, Computers*, volume 29. Kungl. Musical Academy, Stockholm, 1980. Royal Swedish Academy of Music.

[Coh94] J. Cohen. Monitoring background activities. In G. Kramer, editor, *Auditory Display: Sonification, Audification, and Auditory Interfaces*, volume XVIII, pages 499–531, Reading, MA, 1994. Santa Fe Institute Studies in the Sciences of Complexity, The Addison-Wesley Publishing Company, The Advanced Book Program.

[Coo93] M. P. Cooke. *Modeling Auditory Processes and Organisation.* Cambridge University Press, Cambridge, MA, 1993.

[Deu82] D. Deutsch, editor. *The Psychology of Music.* Academic Press Series in Cognition and Perception. Academic Press, Inc. (Harcourt Brace Jovanovich, Publishers), Orlando, FL, 1982.

[Dow73] W. J. Dowling. Rhythmic groups and subjective chunks in memory for melodies. *Perception & Psychophysics*, 14:37–40, 1973.

[Ell96] D. P. W. Ellis. *Prediction-Driven Computational Auditory Scene Analysis.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1996.

[FB95] S. Flinn and K. S. Booth. The creation, presentation, and implications of selected auditory illusions. Technical Report TR-95-15, University of British Columbia Computer Science Department, 1995.

[Fie92] M. Fields. Gems of music composition: Parallel fifths and octaves, 1992. Author's electronic mail address: fieldszip.eecs.umich.edu.

[FK94] W. T. Fitch and G. Kramer. Sonifying the body electric: Superiority of an auditory over a visual display in a complex, multivariate system. In G. Kramer, editor, *Auditory Display: Sonification, Audification, and Auditory Interfaces*, volume XVIII, pages 307–325, Reading, MA, 1994.

Santa Fe Institute Studies in the Sciences of Complexity, The Addison-Wesley Publishing Company, The Advanced Book Program.

[FK96] S. P. Frysinger and K. Kramer, editors. *Proceedings of the Third International Conference on Auditory Display, ICAD '94*, Palo Alto, CA, 1996. Xerox PARC.

[FM33] H. Fletcher and W. A. Munson. Loudness: Its definition, measurement and calculation. *Journal of the Acoustical Society of America*, 5:82–88, 1933.

[Gav86] W. W. Gaver. Auditory icons — using sound in computer interfaces. *Human-Computer Interaction*, 2(2), 1986.

[Gav89] W. W. Gaver. The SonicFinder: An interface that uses auditory icons. *Human-Computer Interaction*, 4(1), 1989.

[Gav93] W. W. Gaver. Synthesizing auditory icons. In *Proceedings of the INTERCHI '93 Conference on Human Factors in Computer Systems*, Amsterdam, The Netherlands, April 1993.

[Gav94] W. W. Gaver. Using and creating auditory icons. In G. Kramer, editor, *Auditory Display: Sonification, Audification, and Auditory Interfaces*, volume XVIII, pages 417–446, Reading, MA, 1994. Santa Fe Institute Studies in the Sciences of Complexity, The Addison-Wesley Publishing Company, The Advanced Book Program.

[GB93] E. P. Glinert and M. M. Blattner. Programming the multimodal interface. In *Proceedings of the ACM Multimedia '93*, pages 189–206, Anaheim, CA, 1993. Association for Computing Machinery, ACM Press.

[Gre75] J. M. Grey. An exploration of musical timbre. Technical Report STAN-M-2, Center for Computer Research in Music and Acoustics, Department of Music, Stanford University, 1975.

[GSO91] W. W. Gaver, R. B. Smith, and T. O'Shea. Effective sounds in complex systems: The ARKola simulation. In Scott P. Robertson, Gary M. Olson, and Judith S. Olson, editors, *Proceedings of the CHI '91 Conference on Human Factors in Computer Systems*, pages 85–90, New Orleans, LA, 1991. ACM Special Interest Group on Computer Human Interaction.

[Hay94] C. Hayward. Listening to the earth sing. In G. Kramer, editor, *Auditory Display: Sonification, Audification, and Auditory Interfaces*, volume XVIII, pages 369–404, Reading, MA, 1994. Santa Fe Institute Studies in the Sciences of Complexity, The Addison-Wesley Publishing Company, The Advanced Book Program.

[JF89] S. D. Jones and S. M. Furer. The construction of audio icons and information cues for human-computer dialogs. In T. Megaw, editor, *Contemporary Ergonomics: Proceedings of the Ergonomics Society's 1989 Annual Conference*, pages 436–441, Reading, UK, 1989. Taylor & Francis.

[JF94] J. A. Jackson and J. M. Francioni. Synchronization of visual and aural parallel program performance data. In G. Kramer, editor, *Auditory Display: Sonification, Audification, and Auditory Interfaces*, volume XVIII, pages 291–306, Reading, MA, 1994. Santa Fe Institute Studies in the Sciences of Complexity, The Addison-Wesley Publishing Company, The Advanced Book Program.

[JKW81] M. R. Jones, G. Kidd, and R. Wetzel. Evidence for rhythmic attention. *Journal of Experimental Psychology: Human Perception and Performance*, 7:1059–1073, 1981.

[JNZM93] J. A. Johnson, B. A. Nardi, C. L. Zarmer, and J. R. Miller. ACE: Building interactive graphical applications. *Communications of the ACM*, 36(4):41–55, April 1993.

[KK80] J. Kerman and V. Kerman. *Listen.* Worth Publishers, Inc., New York, NY, third edition, 1980.

[Kra94a] G. Kramer, editor. *Auditory Display: Sonification, Audification, and Auditory Interfaces*, volume XVIII, Reading, MA, 1994. Santa Fe Institute Studies in the Sciences of Complexity, The Addison-Wesley Publishing Company, The Advanced Book Program. The Proceedings of the First International Conference on Auditory Display, ICAD '92.

[Kra94b] G. Kramer. Some organizing principles for representing data with sound. In G. Kramer, editor, *Auditory Display: Sonification, Audification, and Auditory Interfaces*, volume XVIII, pages 185–221, Reading, MA, 1994. Santa Fe Institute Studies in the Sciences of Complexity, The Addison-Wesley Publishing Company, The Advanced Book Program.

[KS94] G. Kramer and S. Smith, editors. *Proceedings of the Second International Conference on Auditory Display, ICAD '94*, Santa Fe, New Mexico 87501, 1994. Santa Fe Institute.

[LM90] D. Lunney and R. C. Morrison. Auditory presentation of experimental data. In E. J. Farrell, editor, *Extracting Meaning from Complex Data: Processing, Display and Interaction*, volume 1259, pages 140–146, 1990.

[LPC90] L. F. Ludwig, N. Pincever, and M. Cohen. Extending the notion of a window system to audio. *Computer*, 23(8):66–72, 1990.

[Man84] D. L. Mansur. Graphs in sound: A numerical data analysis method for the blind. Master's thesis, University of California, Davis, 1984. Published as Lawrence Livermore National Laboratory Technical Report UCRL-53548.

[MBJ85] D. L. Mansur, M. M. Blattner, and K. I. Joy. Soundgraphs — a numerical data analysis method for the blind. In *Proceedings: 18th Hawaii International Conference on System Sciences*, pages 163–174, 1985.

[McA84] S. McAdams. *Spectral Fusion, Spectral Parsing, and the Formation of Auditory Images.* PhD thesis, Stanford University, Stanford, CA, 1984.

[Mey57] L. B. Meyer. *Emotion and Meaning in Music.* University of Chicago Press, Chicago, IL, 1957.

[Moo82] B. C. J. Moore. *An Introduction to the Psychology of Hearing.* Academic Press, London, England, second edition, 1982.

[Myn94a] E. D. Mynatt. Auditory presentation of graphical user interfaces. In G. Kramer, editor, *Auditory Display: Sonification, Audification, and Auditory Interfaces*, volume XVIII, pages 533–555, Reading, MA, 1994. Santa Fe Institute Studies in the Sciences of Complexity, The Addison-Wesley Publishing Company, The Advanced Book Program.

[Myn94b] E. D. Mynatt. Designing with auditory icons. In G. Kramer and S. Smith, editors, *Proceedings of the Second International Conference on Auditory Display, ICAD '94*, pages 109–119, Santa Fe, New Mexico 87501, 1994. Santa Fe Institute.

[PB94] A. L. Papp III and M. M. Blattner. A centralized audio presentation system. In G. Kramer and S. Smith, editors, *Proceedings of the Second International Conference on Auditory Display, ICAD '94*, pages 231–239, Santa Fe, New Mexico 87501, 1994. Santa Fe Institute.

[PB96] A. L. Papp III and M. M. Blattner. Dynamic presentation of asynchronous auditory output. In *Proceedings of the ACM Multimedia '96*, pages 109–116, Boston, MA, 1996. Association for Computing Machinery, ACM Press.

[Pis41] W. Piston. *Harmony.* W. W. Norton & Co., Inc., New York, NY, 1941.

[Ram96] T. V. Raman. Emacspeak — direct speech access. In *The Second Annual ACM Conference on Assistive Technologies*, pages 64–71, Vancouver, British Columbia, Canada, 1996. Association for Computing Machinery, ACM Press.

[Ras78] R. A. Rasch. The perception of simultaneous notes such as in polyphonic music. *Acoustica*, 40:21–33, 1978.

[RFJ+90] D. A. Rabenhorst, E. J. Farrell, D. H. Jameson, T. D. Linton, and J. A. Mandelman. Complementary visualization and sonification of multidimensional data. In E. J. Farrell, editor, *Extracting Meaning from Complex Data: Processing, Display and Interaction*, volume 1259, pages 147–153, 1990.

[RK92] J. D. Reichbach and R. A. Kemmerer. Soundworks: An object-oriented distributed system for digital sound. *IEEE Computer*, pages 25–37, March 1992.

[RP82] R. A. Rasch and R. Plomp. The perception of musical tones. In D. Deutsch, editor, *The Psychology of Music*, Academic Press Series in Cognition and Perception, pages 135–147. Academic Press, Inc. (Harcourt Brace Jovanovich, Publishers), Orlando, FL, 1982.

[SC91] C. Scaletti and A. B. Craig. Using sound to extract meaning from complex data. In *Extracting Meaning from Complex Data: Processing, Display, Interaction II*, volume 1459, pages 207–219. SPIE, 1991.

[Sch51] A. Schoenberg. *Style and Idea.* Williams & Norgate, London, 1951.

[SE96] R. D. Stevens and A. D. N. Edwards. An approach to the evaluation of assistive technology. In *The Second Annual ACM Conference on Assistive Technologies*, pages 32–36, Vancouver, British Columbia, Canada, 1996. Association for Computing Machinery, ACM Press.

[SGP91] S. G. Smith, G. Grinstein, and R. Pickett. Global geometric, sound, and color controls for iconographic displays. In *Extracting Meaning from Complex Data: Processing, Display and Interaction II*, volume 1459, pages 192–206, 1991.

[Sla81] A. W. Slawson. The musical control of sound color. *Canadian University Music Review*, 3:67–79, 1981.

[Smi90] S. Smith. Representing data with sound. In *Proceedings of the Visualization 1990 Conference.* IEEE Computer Society Press, 1990.

[Sum85] D. A. Sumikawa. Guidelines for the integration of audio cues into computer interfaces. Master's thesis, University of California, Davis, 1985. Published as Lawrence Livermore National Laboratory Technical Report UCRL-53656.

[Van77a] L. P. A. S. Van Noorden. Minimum differences of level and frequency for perceptual fission of tone sequences ABAB. *Journal of the Acoustical Society of America*, 61:1041–1045, 1977.

[Van77b] L. P. A. S. Van Noorden. *Temporal Coherence in the Perception of Tone Sequences.* PhD thesis, Eindhoven University of Technology, NL, 1977.

[Wen92] E. M. Wenzel. Three-dimensional virtual acoustic displays. In M. M. Blattner and R. Dannenberg, editors, *Multimedia Interface Design*, Frontier, pages 257–288. ACM Press / Addison Wesley, New York, NY, 1992.

[Wen94] E. M. Wenzel. Spatial sound and sonification. In G. Kramer, editor, *Auditory Display: Sonification, Audification, and Auditory Interfaces*, volume XVIII, pages 127–150, Reading, MA, 1994. Santa Fe Institute Studies in the Sciences of Complexity, The Addison-Wesley Publishing Company, The Advanced Book Program.

[Wes79] D. L. Wessel. Timbre space as a musical control structure. *Computer Music Journal*, 3:45–52, 1979.

[Wil89] S. M. Williams. STREAMER: A prototype tool for computational modeling of auditory grouping effects. Technical Report CS-89-31, Department of Computer Science, University of Sheffield, Sheffield, UK, 1989.

[Wil94] S. M. Williams. Perceptual principles in sound grouping. In G. Kramer, editor, *Auditory Display: Sonification, Audification, and Auditory Interfaces*, volume XVIII, pages 95–125, Reading, MA, 1994. Santa Fe Institute Studies in the Sciences of Complexity, The Addison-Wesley Publishing Company, The Advanced Book Program.

[Yeu80] E. S. Yeung. Pattern recognition by audio representation of multivariate analytical data. *Analytic Chemistry*, 52:1120–1123, 1980.

Technical Information Department • **Lawrence Livermore National Laboratory**
University of California • Livermore, California 94551