AudioStreamer: Leveraging The Cocktail Party Effect for Efficient Listening

by

Atty Thomas Mullins

B.A., Linguistics
Stanford University
Stanford, CA
1985

SUBMITTED TO THE PROGRAM IN MEDIA ARTS AND SCIENCES, SCHOOL OF ARCHITECTURE AND PLANNING, IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN MEDIA TECHNOLOGY

AT THE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February 1996

Signature of Author ______________________________
Program in Media Arts and Sciences
January 19, 1996

Certified by ______________________________
Christopher M. Schmandt
Principal Research Scientist
MIT Media Laboratory

Accepted by ______________________________
Stephen A. Benton
Chairperson
Departmental Commitee on Graduate Students
Program in Media Arts and Sciences

AudioStreamer: Leveraging The Cocktail Party Effect for Efficient Listening

by

Atty Thomas Mullins

SUBMITTED TO THE PROGRAM IN MEDIA ARTS AND SCIENCES,
SCHOOL OF ARCHITECTURE AND PLANNING, ON JANUARY 19, 1996 IN PARTIAL
FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN MEDIA TECHNOLOGY

AT THE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
FEBRUARY 1996

Abstract

The focus of this thesis is the design and implementation of a system for efficiently browsing large audio databases. Although audio is rich and highly expressive, from a retrieval standpoint its serial nature makes it inefficient for storing unstructured data. AudioStreamer takes advantage of our ability to monitor multiple streams of audio simultaneously by presenting a listener with three simultaneous streams of spatialized audio. AudioStreamer enhances our ability to selectively attend to a single stream of audio by allowing the listener to alternately bring each stream into "focus". AudioStreamer attempts to increase the efficiency of browsing by cueing the listener to switch attention at salient times. Issues of attention, non-visual interfaces, and "figure and ground" in audio are also explored.

Thesis Supervisor: Christopher M. Schmandt
Title: Principal Research Scientist

This work was supported by the News in the Future consortium.

Thesis Committee

Thesis Advisor ______________________________

Christopher M. Schmandt
Principal Research Scientist
MIT Media Laboratory

Reader ______________________________

Walter Bender
Associate Director for Information Technology
MIT Media Laboratory

Reader ______________________________

Hiroshi Ishii
Associate Professor of Media Technology
Program in Media Arts and Sciences

Acknowledgments

I would like to thank the following people, without whom this thesis would not have been possible:

My advisor Chris Schmandt for his encouragement, support, and friendship during the writing of this thesis.

My readers, Walter Bender and Hiroshi Ishii for their time and comments.

Andy Wilson and the folks at Crystal River Engineering for loaning me the Beachtrons and assisting in the implementation and debugging of the Sun client code.

Deb Roy, Chris Horner, Charla Lambert, and Jordan Slott for direct contributions to this thesis.

Jeff Herman, Eric Ly, Lisa Stifelman, Barry Arons, Matt Marx for their insightful comments and suggestions.

Minoru Kobayashi and Brenden Maher for providing feedback from users of AudioStreamer.

Linda Peterson for her unwavering support and guidance.

My spouse Lida for her devotion and for keeping me sane while raising two wonderful sons.

Contents

Figures

1. Introduction

> We thrive in information-thick worlds because of our marvelous capacities to select, edit, single out, structure, highlight, group, pair, merge, harmonize, synthesize, focus, organize, condense, reduce, boil down, choose, categorize, catalog, classify, list, abstract, scan, look into, idealize, isolate, discriminate, distinguish, screen, pigeonhole, pick over, sort, integrate, blend, inspect, filter, lump, skip, smooth, chunk, average, approximate, cluster, aggregate, outline, summarize, itemize, review, dip into, flip through, browse, glance into, leaf through, skim, refine, enumerate, glean, synopsize, winnow the wheat from the chaff, and separate the sheep from the goats.
>
> (Tufte 1990)

Movie sound tracks, lectures, books on tape, voice mail, snippets of bird song -- the wealth of audio information stored in digital form is truly astonishing. Driven by technologies such as CD-ROM, DAT, and CD-AUDIO, the amount and variety of published audio information available to the average consumer is enormous and constantly increasing. In addition to published audio, our lives are filled with "ubiquitous audio" of many sorts: meetings, conversations, the news broadcast playing on the radio, phone calls, etc. According to current estimates, a year's worth of office conversation requires approximately two Gigabytes of storage (Schmandt 1994). The combination of writeable mass storage devices, efficient compression techniques, and computers capable of manipulating audio make recording and storage of all of the audio in one's environment a possibility.

Unfortunately, the largely unstructured and dynamic nature of this accumulation of audio makes subsequent retrieval a tedious task. This thesis addresses that problem. In short, the goal is the design of an application that allows a listener to browse through a large database of recorded speech in search of the parts that are of interest.

We have all browsed through books and magazines, the racks of CDs at the local music shop, or the shelves at the grocery store. In these examples browsing is a visual activity; we take advantage of the abilities of our visual system to find things of interest. By capitalizing on the perceptual abilities of our auditory system the application demonstrated in this thesis attempts to bring some of the properties of visual browsing (primarily simultaneous access) into the auditory domain. In the process, issues of the design of non-visual interfaces, the nature of listening, and our ability to attend to simultaneous auditory events is explored.

1.1 The Utility of Speech

Speech is a rich and expressive medium (Chalfonte 1991). A transcript of a speech by JFK, while moving, hardly conveys power of a recording of the actual event. Transcripts are convenient for retrieval and storage, but fail to capture the emotion encoded in speech by parameters such as pitch, amplitude, and rate. Millions of years of evolution have given us the ability to convey the most subtle of ideas and emotions with intonation, hesitation, and volume. In addition, local dialects, foreign accents, and manner of speaking are indicators of our geographic origin and form the basis of the social fabric that binds us all together.

From a pragmatic standpoint speech has many advantages as well. Audio is a background channel. I can listen to the radio news on the subway, driving in my car, jogging along the Charles River, or while reading a book. With some limitation I can listen while my hands and eyes are busy. Speech is also easy to produce; most of us learn to speak at an early age, and almost none of us have the ability to type at the speed with which we speak. Finally, audio is accessible remotely by existing technology such as telephones. At the time of this writing I have the ability to call MIT from Arizona and listen to recorded news broadcasts, a synthesized voice rendering of my email, my voice mail, my calendar, and my rolodex, all for the price of a phone call (Schmandt 1993a).

Although speech as a datatype has many advantages, it also has significant liabilities that necessitate a careful examination of potential voice based applications. Primary among the liabilities is that speech is slow. A normal speaking rate is about 180 words per minute, while some people can speak as quickly as 225 words per minute. In contrast, reading rates are much higher; generally 350 to 500 words per minute and some as high as 1000 words per minute. Speech is also serial, temporal, and bulky; only a single word is transmitted at a time, once spoken it is gone, and a minute of telephone-quality speech requires about 1/2 Megabyte of storage.

Most systems that successfully use voice as an information storage medium employ sophisticated presentation techniques combined with carefully designed user interfaces. Several of these presentation techniques place a high cognitive load on the listener. The presentation technique discussed in this thesis is no exception and thought must be given to ensure that it is properly situated with respect to a listener's needs.

1.2 Research Challenges

The two primary research contributions of this thesis are: 1) an efficient method for browsing audio that is scaleable with respect to the amount of available structure in the audio; 2) a user interface that immerses the user in the listening process and minimizes the need for interacting with the system.

1.2.1 Presentation

The efficient presentation of audio information for browsing is the foremost contribution of this thesis. AudioStreamer is designed for use in real world situations with actual audio recordings or live audio feeds. These sources of audio vary widely with regard to the amount of explicit structure associated with them. AudioStreamer is designed to take advantage of structure if it is available, but is still effective even in the absence of any structure.

In its final form AudioStreamer presents three simultaneous audio streams (recordings of news broadcasts) to a listener over conventional stereo headphones. The streams are spatially separated using three dimensional audio processing hardware. By spatially separating the streams AudioStreamer allows the listener to easily attend to one of the streams while ignoring the others. The listener sits in a chair and switches attention from one stream to another browsing for items of interest. Once the listener locates something interesting he can attend to that stream until something interesting occurs on another stream.

The efficiency of this simultaneous listening increases if structure is supplied to AudioStreamer in the form of pointers to story boundaries and speaker change boundaries within a news broadcast. At these potentially interesting points AudioStreamer plays a tone and temporarily increases the gain of the corresponding stream to elicit an attention shift on the part of the listener. The listener can choose to continue to attend to that stream or switch attention to another stream if the item was not of interest.

1.2.2 User Interface Design

Several different user interface designs were implemented during the development of AudioStreamer. The presentation technique used in AudioStreamer places a high cognitive load on the listener; as a result the interface required careful design so as not to overload the user by requiring frequent interaction with the system. In order to accomplish this, several assumptions about how the system would be used had to be made. In addition, several input modalities were evaluated in order to find a combination that was compatible with the cognitive requirements placed on the listener.

AudioStreamer's user interface is also an experiment in the design of a non-visual interface (an interface that doesn't rely on a visual display). Several initial designs were implemented and refined. A non-visual interface was selected for two reasons. First, non-visual interfaces eliminate the need for a keyboard or display. As a result the system can be implemented on a small and portable platform, allowing it to be used in a variety of environments. Secondly, a non-visual interface allows the listener to concentrate on listening, rather than diverting his attention by requiring him to look at a screen and type on a keyboard.

The final version of the interface uses a Polhemus head tracking system to determine the listener's head position in real-time. By simply looking in the direction of a stream the listener "selects" it and AudioStreamer responds by temporarily increasing the gain of that stream (similar to what occurs at story and speaker boundaries). Repeated selection of a stream causes AudioStreamer to assume that the listener is very interested in the information playing on that stream. This increased interest is reflected by increasing the gain further and by extending the length of time during which that stream is selected.

1.3 Overview of Document

The overall structure of this thesis flows from basic principles to the details of the actual implementation. Wherever possible, visual analogies to auditory principles have been used to aid the reader's understanding of the basic auditory concepts on which AudioStreamer relies.

Chapter two discusses previous attempts at browsing audio of various types. The parallel presentation technique used in AudioStreamer is outlined and put in context.

Chapter three discusses the basic approach taken by AudioStreamer and the underlying perceptual and cognitive principles on which it is based. Emphasis is given to a discussion of "The Cocktail Party Effect" and the nature of attention and memory.

Chapter four outlines the initial design and implementation of the AudioStreamer.

Chapter five discusses the refinements to the initial design based on feedback from several users.

Finally, chapter six reviews the thesis and outlines further research possibilities.

The appendices include annotated code listing of the major software components of AudioStreamer.

2. Browsing Audio

> **browse** ... **a**) to skim through a book reading at random passages that catch the eye. **b**) to look over or through an aggregate of things casually esp. in search of something of interest.
>
> (Webster 1993)

This chapter presents a general taxonomy of audio, with particular emphasis on the amount of structure that is available with various sources. An overview of previous work related to browsing audio is also presented.

2.1 Sources of Audio

As mentioned in Chapter One there are many sources of audio available to today's consumers. This section presents a partial ordering of those sources with respect to how much structure they contain. A given piece of audio is said to be structured if the content of the audio, in the form of a transcript, is available for processing. Semi-structured audio indicates that some information about the audio is known, the date and time of recording or the number of speakers for example, but the actual content is unknown. Finally, unstructured audio has little or no explicit information associated with it and the burden of extracting structure lies with the system.

Structure is very important to the design of an efficient and widely applicable browser. Ideally, a browser will be effective to some degree in the absence of any structure at all, but will also be able to take advantage of structure when it is available, with correspondingly better results. AudioStreamer falls into this category.

Figure 2-1 arranges various audio sources according to how much explicit structure they contain. In this depiction, recordings of conversations and meetings fall on the least structured end of the scale, whereas transcribed audio and voice notes make up the most structured audio. In between, there are a wide variety of sources with varying amounts of structure. For a more comprehensive overview of the sources of recorded audio see Arons 1994.

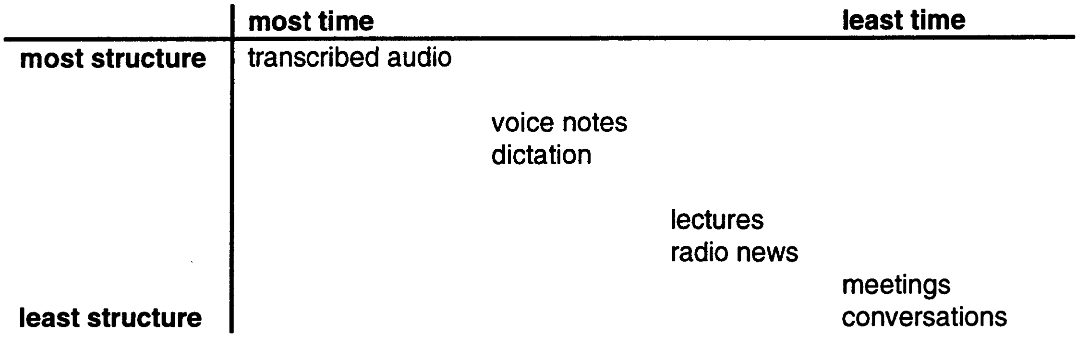

Figure 2-1 Sources of audio arranged according to structure and the total amount of time required to produce them.

These sources could also be ordered by the total amount of time that was required to produce them. Transcribed audio requires a large amount of time, in addition to producing the audio, someone actually had to listen to the entire presentation, recorded or live, and transcribe it into text. On the other hand, spontaneous conversation requires relatively little time on the part of the participants; the content is generally free form and only conversational rules such as turn taking and linguistic factors such as pausing and intonation are in effect.

The ideal situation, the "holy grail" so to speak, is to have automatic transcription of meetings or conversations. Unfortunately, in practice this turns out to be very difficult (Hindus 1994). One issue is that recognition of spontaneous natural speech is not possible yet. Even the more constrained problem of keyword spotting is difficult when applied to spontaneous speech. Other issues, such as associating utterances with participants turns out to present a serious technological challenge as well. In the meantime designers of audio browsers have to rely on efficient presentation techniques that do not require structure, but that can make use of any structure that is available.

2.2 Related Work

Many systems have been developed to browse and skim audio. This section gives an overview of several noteworthy systems. Each subsection begins with a discussion of general techniques and then presents some systems that make use of those techniques. User interface design and implementation are also touched upon.

2.2.1 Unstructured Audio

One of the most useful techniques for efficiently presenting unstructured audio is time-scaling or time-compression. Put simply, time-compression involves presenting audio in faster than real-time. The idea is similar to playing a 33 rpm record at 45 rpm, without the associated pitch shift. A large number of algorithms have been developed for time-compressing speech while minimizing the loss of intelligibility and comprehension. One example is silence removal or shortening where silences in the speech signal are detected and shortened or removed altogether. The most effective and efficient techniques tend to rely on removing temporal redundancy in the speech signal (Arons 1992a). A good example is selective sampling using the *synchronized overlap add method* (Roucus 1985), where successive segments of the speech are overlapped and averaged, essentially removing redundant pitch periods without altering the pitch and magnitude of the signal.

It appears that a compression of 50% (twice normal speed) is the perceptual limit on the maximum amount that a signal can be compressed without a significant degradation in comprehension (Gerber 1974). However, for purposes of browsing (where every word need not be understood or the material is familiar) much higher compression ratios are possible. For example, the author has informally experimented with a compression ratio of 2.8 with good success at locating phone numbers in voice mail messages.

The Speech Research Group at the MIT Media Lab has developed a suite of audio applications that make extensive use of time-compression (Schmandt 1993b). The voice mail application allows a user to listen to and author voice messages at the desktop. The application provides the user with interactive control of the rate of compression, which can be increased or decreased in 25% increments by pressing a button. The interface is primarily graphical and input is via button presses or selection from the mouse.

A news based application has also been developed as part of the Phoneshell telephone information system (Schmandt 1993a). A user can call into Phoneshell over the telephone and select from a set of recent radio news broadcasts. The user can adjust the speed of playback by pressing keys on the telephone keypad. This application is interesting not only because it illustrates the utility of telephone based information systems, but also because it involves browsing large pieces of recorded audio. It points out some of the shortcomings of using time compression with long audio documents, even at a compression ratio of two it still takes 15 minutes to listen to a 30 minute news broadcast. In many instances additional listening efficiency is desirable, which leads to the next section.

2.2.2 Extracting Structure

Consider a 10 minute portion of the sound track of your favorite film. It almost certainly contains music, environmental background noise (a car driving by, footsteps, doors slamming, explosions), and voice based sounds (laughter, speech, singing). Now let's assume you are very interested in action films and you are writing a book, "Hollywood TNT: A History of Pyrotechnics in Theater and Film." In order to be diligent in your research you would like to watch several hundred explosions from a large sample of films. Obviously an explosion detector of some kind would be quite useful (unless of course you are watching "Terminator 2" where simply pressing "play" will do the job). An efficient detector will likely take advantage of the soundtrack, extracting portions of audio where there are large changes in amplitude accompanied by lots of high frequency energy.

This explosion detector can be said to segment the audio into two sets, segments that contain explosions and those that do not. One can envision other segmenters that might be useful for browsing audio. For example, a music segmenter can be used to browse an audio soundtrack. In "Indiana Jones and the Last Crusade" of the first 40 minutes only seven minutes do not contain music (Hawley 1993). It turns out that these seven minutes contain set up dialog that is integral to the plot. By simply skipping from non-music section to non-music section, a viewer could get a good idea of what was about to happen.

For truly efficient browsing finer resolution is desirable. In addition to segmenting the audio, an efficient browser also needs to characterize the segments. For example, for a given segment of speech, we would like to know: Is a male or female speaking? Is the speaker emphasizing a point? Who is speaking? Have we heard the current speaker before? And finally, what is the speaker saying? Algorithms have been developed for answering all of these questions with varying results. Keyword spotting in particular has met with little success in browsing due to its high false alarm rate. However, as is discussed below, several systems have been developed that make use of pause, pitch, and speaker information to great effect.

SpeechSkimmer is an audio-only user interface for interactively skimming recorded speech (Arons 1993). The interface is designed to allow a listener to interactively browse a pre-processed speech recording. The system makes use of relatively simple speech processing techniques, pause and pitch detection, to find emphasized portions of a recording such as a lecture. The input device is a touch pad that allows the listener to move between several levels of detail once an interesting segment of audio is found. SpeechSkimmer also seamlessly integrates time-compression, allowing a listener to play back portions of audio at increased speed.

Another system developed in the Speech Research Group at MIT allows a listener to browse recordings of the BBC World Service's daily news broadcasts (Roy

1995). The system uses pause information to find story boundaries with surprising success. Recently, the system has been augmented with speaker differentiation software that can detect when the speaker changes (Roy 1995). Based on these two sources of information, the system can segment the audio based on stories and speakers. This allows the detection of "actualities", reports from on scene correspondents, which often have very high information content. Again, time-compression is used to present the segments of potential interest as rapidly as possible.

Systems that are based on extracting structure have the great advantage of working with audio "as is" -- some machine processing is involved, but human intervention is minimal. However, given that the detection and characterization algorithms are seldom error free, user interfaces to such systems must be carefully designed to allow the listener to quickly weed out the errors and move on to the next segment. Time-compression can almost always be used in conjunction with these systems to improve their time-efficiency.

2.2.3 Explicit Structure

A segment of audio with explicit structure has additional structured information associated with it. This structure is generally in the form of links, a transcription, or some other datatype that can be manipulated by computer. For example, a digital recording of the evening news broadcast might have an associated table of pointers to the beginnings of stories. A more common example is the "index" function on a hand-held tape recorder that can be used to mark the beginning of each chunk of recorded audio by inserting a cue tone. The purpose of the structure is to allow alternate principled ways of navigating through the audio, e.g., by story or by speaker, rather than just from beginning to end.

The explicit structure is frequently authored by one or more persons and can be quite labor intensive. For example, a transcript of a half hour evening news program is about 10 pages long. Occasionally the author will be the consumer as well. Increasingly, however, the information is published to be consumed by others. For instance, more and more television shows are being closed captioned. The closed captioning information can be subsequently parsed to build a structured datatype that represents the content of the audio. This datatype can be used to navigate through the audio.

Browsing explicitly structured audio can be extremely efficient, although it will only be effective to the degree that the associated content is complete and accurate. This places a great burden on the authoring process and makes it practical only in limited domains. If the information is accurate and the system is well designed, good results will follow. Below are outlined several systems that fall into this category. Most of them are multi-layered, making use of explicit structure as well as structure derived from processing, e.g., typed links combined with pause based skipping.

HyperSpeech is a speech-only hypermedia system that was designed to be an experiment in the design of non-visual interfaces (Arons 1991). In the system recorded interviews were manually segmented by topic. Each segment or *node* was logically connected to other nodes using a set of typed links. The user could follow a *name* link to hear a comment by a particular speaker. Following a *more* link causes a more in-depth segment on a particular topic to be played. A speech recognizer accepts spoken input from the user and output consists of a combination of recorded and synthesized speech. One of the primary lessons learned from the implementation of HyperSpeech was that authoring hypermedia networks is often the most difficult aspect of building hypermedia systems.

VoiceNotes is an application that allows a user to author and manage a database of short spontaneous voice recordings (Stifelman 1993). The system itself is a handheld device that allows a user to manage lists of related audio segments. The user interacts with the system using a combination of button and speech input and output consists of recorded speech and non-speech audio feedback.

NewsTime explores issues of the design and implementation of a graphical interface to recorded audio news programs (Horner 1993). The data for the system consists of the audio portion of television news broadcasts. The recorded broadcasts are segmented by story and speaker using closed captioning information. In addition, segments of audio are classified by topics such as weather, sports, and finance, using a keyword matching algorithm. The user can navigate through the broadcast using a combination of mouse and keyboard input. Time-compression and pause based skipping are also integrated.

2.3 Auditory Displays

The recent development of the capability to binaurally present three-dimensional audio has stimulated a great deal of research into auditory displays (Foster 1991). This research has been primarily geared towards virtual reality or telecommunication applications (Fisher 1988, Wenzel 1990, Kendall 1990, Koizumi 1992). AudioStreamer is not a virtual reality system, but it does make use of a three-dimensional auditory display.

One system in particular has some relevance for the design of AudioStreamer. Cohen built a system for managing three dimensional audio "windows" (Cohen 1991). The system, MAW (multidimensional audio windows), uses a gestural front end based on the DataGlove and an auditory display based on the Convolvotron (Foster 1988). A user of MAW can manually manipulate audio sources in three-space using a gestural interface. Audio sources are modified using *filtears*, signal processing that superimposes information on audio signals, to indicate their current status. Sources can be moved in three-space by "grabbing" them with the DataGlove and placing them in a new position. For example, the user can grab an audio source at which point it is *muffled* (low-pass filtered) to indicate that the source has been grabbed. Using gestures, the user can then place the source at a new location. The user can also point to an audio source, giving it the *spotlight*, at which point it is made more perceptually prominent as a prelude to performing some action. Finally, a source can be *accented* which is similar to giving it the spotlight except that the accent is persistent.

AudioStreamer makes use of some of these enhancement techniques to indicate auditory focus. In its final implementation AudioStreamer also uses head gestures to interact with the audio streams.

2.4 Summary

This chapter described several approaches to browsing audio. The systems that were outlined were differentiated based on whether they were designed to work with structured or unstructured audio. Browsers designed for unstructured audio attempt to take advantage of people's perceptual abilities in order to present audio in faster than real time. Browsers designed for structured audio make use of the structure in order to present the audio based on *content.* This chapter outlined the relative advantage of these approaches. Unstructured browsing works with audio of many kinds and does not require special processing of the audio. Structured browsing is generally more accurate and efficient than unstructured browsing, but it requires extraction or authoring of the structure.

AudioStreamer takes a hybrid approach. The system can present unstructured audio in a time efficient manner. However, when structure is available, such as speaker or story boundaries, AudioStreamer uses this information to alert the listener that a potentially interesting piece of information is about to be played. In this way, AudioStreamer combines the advantages of both approaches while minimizing the disadvantages. AudioStreamer is also an experiment in the design of an interface to an audio information display. Previous uses of three-dimensional audio have focused primarily on virtual reality systems. AudioStreamer, on the other hand, explores the use of three-dimensional audio to improve the efficiency of browsing, resulting in a unique style of user interface.

3. Approach

> Of all the senses, hearing most resembles a contraption some ingenious plumber has put together from spare parts.
>
> (Ackerman 1990)

This chapter consists of an overview of the approach used in AudioStreamer with particular emphasis on the parallel presentation method and the perceptual and cognitive abilities that underlie it. The first section gives a brief overview of AudioStreamer and how it differs from other audio browsing systems. The second section discusses the cocktail party effect and outlines some early experiments. The third section discusses selective attention and ways of enhancing it. The fourth section gives an overview of how humans localize sounds in space. The fifth section talks about the notion of peripheral listening; whether we can listen to several things at once. The chapter concludes with a discussion of how the limitations of peripheral listening influence the design of AudioStreamer.

3.1 What is AudioStreamer ?

AudioStreamer is an audio browser that makes use of a novel presentation technique; multiple streams of audio are presented to a listener simultaneously. The listener wears stereo headphones over which three spatially separated channels of audio are played. The listener can interact with the individual streams of audio, alternately bringing each into auditory focus as segments of interesting speech are played on that stream. The goal of parallel presentation is to allow a listener to find segments of audio that are of interest in a time-efficient manner. Once an interesting piece of audio is found the user interface allows the listener to hear it in detail. Eventually, the listener can again return to browsing in search of other interesting segments of audio.

Listening to multiple simultaneous channels of audio is similar to channel surfing in the visual domain -- alternating between several television channels, often in quick succession, using a remote control. In the case of AudioStreamer we have multiple channels of audio playing different material simultaneously and the ability to bring one into focus relative to the others. The experience is very much like being in a room full of talking people and having the ability to selectively eavesdrop on a conversation at will.

Three channels of audio (as opposed to say five) was chosen for one reason. It appears that the cognitive load of listening to simultaneous channels increases with the number of channels. Stifelman did an experiment where she presented multiple channels of audio to listeners (Stifelman 1994). The listeners performed two tasks simultaneously, listening comprehension on a primary channel and target monitoring on the non-primary channel or channels. Subjects were presented with either one or two non-primary channels. In the experiment separation was achieved by presenting one channel to the left ear, another channel to the right ear, and a third channel to both ears at the same gain.

There was a clear decline in performance between the two channel and three channel condition. Stifelman concludes that "... it seems that increasing the number of channels beyond three would cause a further decrease in performance." She goes on to say, "The question becomes, how do we go beyond two simultaneous channels without seriously impairing performance?"

This thesis seeks to answer that question. In Stifelman's experiment listeners in the three channel condition encountered difficulty in several areas. First, they had difficulty in focusing on the primary passage. Secondly, once their attention switched to a non-primary channel listeners had difficulty switching their attention back to the primary channel. AudioStreamer uses certain "attentional enhancements", such as true spatial separation, cue tones, and a "focus" mechanism to improve the ability to attend to one channel and to allow listeners to easily switch attention between several channels.

Data for AudioStreamer consists primarily of the audio portion of television news broadcasts and radio news broadcasts. News was chosen because of its variable content and because it was readily available. It should be noted that AudioStreamer was designed to be data independent -- the system itself and the presentation techniques that it employs work with many types of audio ranging from voice mail to recorded lectures.

AudioStreamer does not attempt to make a priori judgments about what the listener might be interested in hearing. The goal is to present all of the audio as efficiently as possible and to allow the listener to decide the relative value of listening to one stream as opposed to another. AudioStreamer attempts to avoid one of the prominent shortcomings of browsers, in particular news browsers, which rely on filtering of one form or another (Malone 1987). Due to excessive filtering, items of potential interest are discarded and serendipity is lost. This is not to say that AudioStreamer cannot take advantage of information regarding the potential interest of a piece of audio to a listener, it is simply that the system does not make those judgments.

Parallel presentation is similar to time compression: both can operate on audio with various levels of structure; both focus on making listening more time-efficient; and both can be combined with other processing techniques. Although time compression and parallel presentation are not mutually exclusive, both techniques place a high cognitive load on the listener combining them requires careful consideration.

AudioStreamer's user interface is designed to allow the user to be immersed in the listening process. This is accomplished by minimizing the need for interaction with the system and by making the method of interaction reinforce the listening process (as will be explained later). The prevailing design theme was "devote cycles to listening, not interacting." In order to minimize the need to interact, the system design incorporates some assumptions about how it will be used. The primary goal is to browse audio; switching attention between multiple streams as interesting things are presented. As such, the system is biased towards browsing as opposed to detailed listening. However, the listener does have the ability to listen to a particular stream in detail, which requires more interaction with the system.

A highly evolved visual browser served as a model for the initial design of AudioStreamer -- the front page of a newspaper. Newspaper front pages have all of the properties that AudioStreamer strives to capture in the audio domain. Newspapers rely on *simultaneous presentation* of information, allowing the reader to quickly pick and choose among several articles. The articles themselves are structured to have *multiple levels of detail.* Each article has a headline and is written in a journalistic style -- most of the salient information is presented in the first few lines or paragraphs and the detailed information is presented later in the article. Finally, the articles are *ordered by relative importance*, which is indicated by position on the page, size of the headline, and length of the article. Some of these properties are present in AudioStreamer by virtue of using news broadcasts as data. In particular, news broadcasts present stories in a specific order, where the "top" stories are presented early in the broadcast. AudioStreamer goes farther and presents these stories simultaneously and at multiple levels of detail.

3.2 "The Cocktail Party Effect"

Our auditory systems are amazingly effective and efficient at analyzing the "auditory scene" that surrounds us. At any given time we have little difficulty in determining how many sound sources there are, what their perceptual characteristics are, and where they are located in space. All of this information is determined from the effects of our vibrating ear drums on the hairs in a fluid filled tube in our inner ear known as the cochlea. To gain a true appreciation for the miraculous nature of this process consider the following analogy from "Auditory Scene Analysis" (Bregman 1990).

> The difficulties that are involved in scene analysis often escape our notice. This example can make them more obvious. Imagine that you are at the edge of a lake and a friend challenges you to play a game. The game is this: Your friend digs two narrow channels up from the side of the lake. Each is a few feet long and a few inches wide and they are spaced a few feet apart. Halfway up each one, your friend stretches a handkerchief and fastens it to the sides of the channel. As waves reach the side of the lake they travel up the channels and cause the two handkerchiefs to go into motion. You are allowed to look only at the handkerchiefs and from their motions to answer a series of questions: How many boats are there on the lake and where are they ? Which is the most powerful one ? Which one is closer ? Is the wind blowing ? Has any large object been dropped suddenly into the lake ?
>
> Solving this problem seems impossible, but it is a strict analogy to the problem faced by our auditory systems. The lake represents the lake of air that surrounds us. The two channels are our two ear canals, and the handkerchiefs are our eardrums. The only information that the auditory system has available to it, or ever will have is the vibrations of these two eardrums. Yet it seems able to answer questions very like the ones that were asked by the side of the lake: How many people are talking ? Which one is louder, or closer ? Is there a machine humming in the background ? We are not surprised when our sense of hearing succeeds in answering these questions any more than we are when our eye, looking at the handkerchief, fails.

Parallel presentation of audio relies heavily on our ability to attend one speaker in the presence of other speakers and of background noise. This phenomenon is known as “the cocktail party effect” and was studied in detail by Colin Cherry in the early 1950s (Cherry 1953, Cherry 1954). Cherry’s experiments focus on the ability of the auditory system to attend to one of several competing messages. Cherry suggests several factors that contribute to this ability:

- Distinct spatial origins.
- Correlation with visual events: lip reading, gestures, etc.
- Different speaking voices, mean pitches, male female, etc.
- Differing accents.
- Transition probabilities: content and syntactic constraints.

All of these factors except the last can be eliminated by recording two messages from the same speaker and mixing them on magnetic tape. In Cherry’s initial set of experiments two such messages were played to a group of subjects and they were asked to “shadow” one of them; repeat words or phrases from the recording in real-time. The subject matter of each message was distinct and both messages were presented to both ears. The subjects could replay the tape repeatedly, the task being to separate one of the messages. Subjects were able to carry out this task with relatively few errors. However, the subjects reported that it was very difficult. Cherry observed of one subject, “He would shut his eyes to assist concentration. Some phrases were repeatedly played over by him, perhaps 10 to 20 times, but his guess was right in the end.”

In a follow-up experiment pairs of inseparable messages were constructed. The messages were composed by selecting clichés from speeches in newspapers and stringing them together. Cherry reports the following example (Cherry 1953):

> I am happy to be here today to talk to the man in the street. Gentleman, the time has come to stop beating about the bush - we are on the brink of ruin, and the welfare of the workers and of the great majority of the people is imperiled
> ...

Subjects were observed to read out whole clichés at a time in roughly equal numbers from both messages. Message separation appeared impossible. To explain this phenomenon Cherry proposed that listeners store sets of transition probabilities in their memories that allows them to predict sequences of words. The task of separating messages in the second experiment was much more difficult because of the high predictability of the clichés, recognition of one or two words allowed the subject to predict the whole cliché, combined with the low probability of one cliché following another.

Cherry's second set of experiments concerned dichotic presentation of two different messages to a listener. In dichotic presentation the listener wears headphones. One message is played to the left ear and a different message is played to the right ear. Dichotic presentation can be viewed as an extreme case of spatial separation. In these experiments the subjects had no difficulty shadowing the target message and rejecting the other message.

Cherry then asked what information, if any, was retained from the message played to the "rejected" ear. A series of experiments was designed to determine the answer. The conclusion of these experiments was that "the rejected signal has certain statistical properties recognized, but the detailed aspects, such as the language, individual words, or semantic content are unnoticed." Subjects were able to determine that the signal played to the rejected ear was speech. A change of voice, from male to female, was also almost always identified, as was playing a 400 Hz tone. Subjects were not able to identify any word or phrase heard in the rejected ear, were not able to identify the language, and did not notice if the speech was played backward.

3.3 Selective Attention

Cherry's experiments were designed to determine the extent of our ability to attend to one speech signal in the presence of another speech signal. The question arises of how to enhance this ability. Many experiments have been carried out to determine what acoustic cues are important in stream segregation.

Cherry's initial experiment points to one of the strongest cues for stream segregation -- spatial location. In the dichotic listening task subjects had little difficulty in rejecting one message and attending to the other. Dichotic listening can be thought of as an extreme case of spatial separation. Other experiments investigate the effect of spatial location on stream segregation directly and lend additional support for this conclusion (Spieth 1954, Webster 1954).

The effectiveness of spatial location as an acoustic cue for stream segregation can be partially explained by the notion of the binaural masking level difference or BMLD. According to Blauert (Blauert 1983), "... a desired signal S with a certain direction of incidence is less effectively masked by an undesired noise N from a different direction when the subjects listen binaurally." For example, in a typical binaural masking experiment a tone and a noise is presented to a listener binaurally over headphones. The intensity of the noise is increased until the tone can no longer be heard. Now the tone in one ear is shifted 180 degrees out of phase with respect to the tone in the other ear while leaving the gain of noise unchanged. The tone will become audible again. The difference between the intensity of the noise needed to mask the tone in the control condition and in the out of phase condition is the BMLD.

In another example, a mixture of a tone and a noise is presented to a single ear. Again the intensity of the noise is increased until the tone can no longer be heard. Then only the noise is presented to the other ear. The tone once again becomes audible. From these examples it can be concluded that masking is less effective when the target signal and the masking signal are *perceived* to come from two different locations.

If the auditory system groups sounds by their locations, then by the principle of "psychophysical complementarity", the perceptual representation of spatial location should map to the physical representation of spatial location (Shepard 1981). Several experiments by Rhodes confirm that auditory spatial information is indeed represented analogically (Rhodes 1986). Unlike vision and touch, spatial information is not mapped topographically at the auditory receptor surface. Rather it is pitch that is mapped topographically. However, in Rhodes' experiments a linear relationship between the spatial separation of sound sources and the time it takes to switch attention among them was observed. From this Rhodes concludes, "These results parallel those obtained for shifts of visual attention and are consistent with the view that auditory spatial information, like visual spatial information, may be represented topographically or analogically." Bregman has pointed out that this result might be erroneous (Bregman 1990). He says that the increase in time with distance between sound sources might be due to listeners trying to determine the label for the sound source. In Rhodes' experiment, the sound sources were labeled by numbers and the listeners might have counted their way between sound sources. Bregman goes on to write, "The correct interpretation of this particular experiment is unclear, but there is indirect evidence to support the theory behind it."

Another consideration is the minimum spatial separation that is needed to discriminate between two sound sources. Early experiments found that localization blur ranged from approximately 1.5-5 degrees for various types of sounds presented sequentially (Blauert 1983). However, for simultaneous presentation, the situation of interest, the minimum recommended distance is 60 degrees for accurate localization (Divenyi 1989).

Several other cues for streaming are relevant to this thesis. Frequency and pitch have been found to have an impact on stream segregation (Spieth 1954, Egan 1954, Brokx 1982). Differentially filtering two competing signals into separate frequency bands, or introducing a pitch difference between them appears to improve a listener's ability to attend to one of them. Finally, the relative intensities of the signals can have an impact on the degree to which one signal masks the other in both monaural and dichotic listening tasks (Egan 1954). For a general overview of the acoustic cues involved in streaming and their potential for system design see Arons 1992b.

From an ecological point of view our auditory systems have evolved so as not to rely on any single cue for stream segregation. Rather it appears that all cues are weighed and combined in an effort to create a coherent representation of reality. According to Bregman (Bregman 1990):

> ... the human auditory system does not give overriding importance to the spatial cues for belongingness but weighs these cues against all others. When the cues all agree, the outcome is a clear perceptual organization, but when they do not, we can have a number of outcomes.

As an example of a situation where spatial cues might be at odds with other cues, such as frequency or pitch, consider a highly reverberant environment such as a handball court. In this case spatial cues for streaming become unusable and instead the auditory system relies on other cues to group sounds.

From a system design perspective it is clear that the palette of cues to chose from is very rich. As will be seen in the next chapter, during the initial design of AudioStreamer several different combinations of cues were experimented with in order to achieve the desired effect.

3.4 Localization

Since spatial location is one of the most important determinants of stream segregation, it is worth spending some time discussing how humans locate sounds in space. Much of the research done on sound localization derives from the "duplex theory" (Rayleigh 1907). The duplex theory holds that there are two primary localization cues: interaural time differences (ITDs) and interaural intensity differences (IIDs). ITDs arise from the fact that sounds from sources located to one side of a listener arrive at the closer ear sooner than at the far ear (see Figure 3-1). ITDs were thought to be responsible primarily for localization at low frequencies due to phase ambiguities at higher frequencies. IIDs are the result of head shadowing and were thought to be responsible for localization at high frequencies (see Figure 3-2).

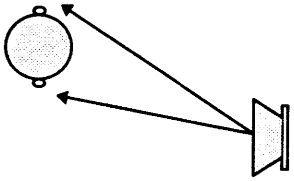

Figure 3-1 Interaural time differences. Sounds from sources off to one side arrive first at the closer ear.

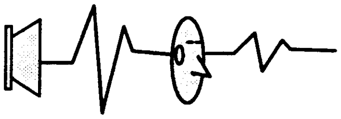

Figure 3-2 Interaural intensity differences. Sources are louder in the closer ear due to the shadowing effect of the head.

Several deficiencies with the duplex theory have been illuminated. First, people are able to localize sounds on the vertical median plane. On this plane interaural differences are minimal, yet localization is not difficult. Secondly, when sounds are presented to listeners over headphones *externalization* is absent. The sounds appear to come from within the listener's head even though appropriate interaural cues are present.

As it turns out the main shortcoming of the duplex theory is that it fails to take into account the spectral shaping caused by the outer ears or pinnae. The experiments that lead to the development of the duplex theory were performed with pure sine waves, which may explain why the frequency dependent filtering of the pinnae was not discovered. Unlike pure sine waves, sounds with several frequency components are differentially filtered by the pinnae depending on their location. For example, for a complex sound at a given spatial location, frequency component A might be more attenuated than frequency component B. If the sound source is moved to a different location, just the opposite might be true, B is more attenuated that A. This differential filtering allows the auditory system to discriminate between sounds with different spatial origins.

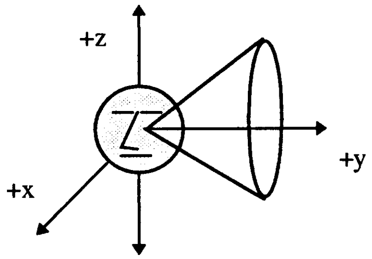

Figure 3-3 Cone of confusion. All sounds emanating from any location on the surface of a cone originating at the center of the listener's head, result in identical interaural time, intensity, and phase *differences* (assuming that the head is a perfect sphere).

Current theories of localization hold that IIDs, ITDs, and pinnae effects all contribute to localization ability and accuracy. In particular, pinnae effects are thought to be primarily responsible for front/back position, elevation , and externalization of sounds (Wenzel 1992). In addition, it appears that head motion is also important. According to Buser and Imbert (Buser 1991):

> Sound localization with head movements eliminated scarcely corresponds to any natural situation except perhaps the case of a transient sound arriving unexpectedly with no chance of subsequent exploratory movements. In more commonplace occurrences, the subject has time to carry out head movements that help localize the source not only with respect to the median plane but also within that plane (up/down) and to make front/back judgments with equal precision.

A common localization error is the front-back reversal; a sound source in front of the listener is perceived as coming from behind the listener. This error appears to follow from the notion of "cones of confusion" (see Figure 3-3). A cone of confusion illustrates that there are areas in space around the listener that result in identical IIDs and ITDs for sources located in that area. Several experiments have shown that allowing listeners to move their heads during localization substantially reduced front-back reversals and improved localization accuracy (Thurlow 1967).

Sound localization is considered to be a primary source of selective pressure in the evolution of mammalian hearing (Masterton 1969). The ability to determine the origin of a sound is of obvious advantage to an animal. Our hearing and vision have evolved in concert and can be said to be complementary. Once a sound is localized, an alerting or orienting response tends to follow. We orient our eyes such that the source of the sound can be seen. Similarly, we tend to become habituated to continuous and uninformative sounds and relegate them to the background (the humming of the hard drive in my laptop is an example). As will be seen in the following sections, AudioStreamer makes use of both of these properties.

3.5 Peripheral Listening

The idea of peripheral listening is analogous to that of peripheral vision, or to be more precise peripheral *seeing*. Our visual system is structured such that only a portion of the information that arrives at our retinas is seen in detail at any given time. At the moment I am reading individual letters as I am typing them on my laptop computer. If I stop typing and do not move my eyes, I can see the desk lamp to my left and the letter box to my right, albeit with poor resolution. Even though I do not have to shift my eyes, I do have to shift my attention slightly to see the lamp and the box. The desk lamp and letter box are said to occur in my peripheral vision.

The acuity of peripheral vision is far less than the acuity of primary vision. I see the desk lamp and letter box because I am familiar with this room. In actuality, I am only seeing the broad outlines and shapes of the objects in my peripheral vision. If I was placed in an unfamiliar setting and asked to describe what was in my peripheral vision I would only be able to respond with the general visual characteristics of objects. Reading, for instance, in peripheral vision would be an impossibility.

Certain events that occur in peripheral vision will elicit an automatic attention shift on the part of the viewer. For example, motion or a dramatic and rapid change in intensity, such as a light turning on, will cause an involuntarily, and often brief, attention shift. In other words, discontinuities of sufficient magnitude are also detected in peripheral vision.

The situation in peripheral listening appears to be similar to that of peripheral seeing. Only the gross acoustical characteristics of nonattended auditory channels are perceived and certain discontinuities on those channels cause involuntary attention shifts. Detailed information from those channels is not available to the listener. Cherry's initial shadowing experiments confirm this view. As mentioned above, when asked about the message that was played to the rejected ear, subjects were only able to recall general characteristics. More detailed information such as the content of the message, or even the language, were not recalled.

Subsequent experiments by various researchers have both supported and questioned some of Cherry's results. Most of these experiments were designed to support one of two competing models of attention. The *early selection* model of attention holds that selection among channels is based on the physical characteristics of the information arriving at the sensors. Meaning is only extracted from channels that have already been selected. On the other hand, the *late* *selection* model contends that some processing occurs on all channels, and that selection is based on a combination of meaningful and physical properties

Moray, a proponent of early selection, performed a dichotic listening experiment where a passage was played to one ear and a list of words was repeatedly played to the other ear (Moray 1959). Subjects were instructed to shadow the passage and to try to remember as much as possible from the other ear. As in Cherry's experiments, subjects were not able to recall any of the words on the list. In a second experiment, Moray did discover that commands prefixed with the subjects' name that were played to the rejected ear were heard in about one third of the trials.

Treisman and Geffen performed a dichotic listening experiment in which subjects were required to shadow a primary message while attempting to detect target words in the rejected message (Treisman 1967). As expected subjects performed poorly. In a similar experiment, Lawson used tone bursts instead of words as targets (Lawson 1966). She found that there was no difference between the detection of a tone in the primary and in the rejected message. Subjects did well in both cases. Citing these results Moray concludes (Moray 1970), "As we have seen there has been agreement until now that [auditory] selection, however it may occur, seems to block the processing of verbal ('complex') signals and leave 'simple' signals relatively undisturbed."

Norman, a proponent of late selection, criticizes Cherry's results on two grounds. First, in Cherry's experiments, subjects were required to shadow the primary message, which requires intense cognitive effort on the part of the subject. According to Norman (Norman 1976), "... a simple tabulation of how accurately the subject's spoken words agree with the presented material does not even begin to tell how much attention is diverted to the task." He goes on to cite the subject's lack of memory for the content of the primary message as farther proof of the difficulty of shadowing.

Norman's second criticism centers on the time delay before asking subjects to recall nonattended material. Both Cherry and Moray waited before asking their subjects how much of the nonattended material they remembered. Norman conducted an experiment in which subjects were asked to shadow English words that were presented to one ear (Norman 1969). Two-digit numbers were presented to the other ear and subjects were tested on their recall of the numbers. It was found that if shadowing was interrupted within twenty seconds of the presentation of a number, subjects did have some recall of the digits. Norman concludes, "... verbal material presented on nonattended channels gets into short-term memory, but is not transferred to long-term memory."

3.6 Summary

The previous three sections gave an overview of many experiments on the nature of the cocktail party effect, audition, selective attention and the structure of memory. The question is, what does all of this mean for the design of AudioStreamer? While it is tempting to apply the results of these experiments directly to the design of AudioStreamer, there are potential pitfalls in doing so. The experiments reviewed were conducted under controlled laboratory conditions. They were carefully designed to answer specific questions about the nature of psychological processes. Any direct translation of the results into real world situations could undermine the assumptions on which the results were based.

Despite this potential shortcoming, it seems clear that some of the results are relevant to the design of AudioStreamer. First, under certain conditions, listeners have the ability to attend to a single audio stream when several are playing simultaneously. From an implementation perspective, this can be achieved by spatially separating the streams, differentially filtering the streams into separate frequency bands, modifying the pitch of the streams, or some combination of the these. Furthermore, one stream can be made more prominent relative to the others, brought into focus so to speak, by applying the above techniques, or by modifying its relative intensity.

The issue of peripheral listening is more complicated. For practical purposes, only the gross acoustical characteristics of nonattended streams are processed by the listener. Highly salient information, such as a listener's name, does occasionally attract a listener's attention. In addition, certain discontinuities in the gross acoustical characteristics of nonattended streams, such as a speaker change or a change in intensity, attract a listener's attention. As Norman pointed out, some meaningful information is processed and stored in short-term memory. However, this has little impact on the design of AudioStreamer, since listeners only recalled the information if the listening task was interrupted.

As will be seen in the next chapter, the design of AudioStreamer takes advantage of selective attention and peripheral listening to allow a listener to browse multiple streams of audio. A listener's selective attention is enhanced by spatially separating the three streams of audio using digital processing techniques. The user interface provides the listener with the capability to alternately bring one stream into auditory focus by increasing its relative gain and makes it easy for a listener to switch attention between the streams.

The result is that the listener browses the three audio streams simultaneously, leading to a time saving and a corresponding overall increase in efficiency. In a study by Webster and Thomas on the ability of air control tower operators to respond to two overlapping messages the authors write (Webster 1954), "Another interesting point is that even under the worst possible encoding scheme, namely two simultaneous voice messages, an average of 60 percent of each of the two message identifications was received, resulting in a greater total information intake per unit time ..." They go on to say, "... this is only true for those parts of the messages that have low informational content." In a summary of various selective listening experiments, Broadbent writes (Broadbent 1958), "For the moment, we may note that the statement 'one cannot do two tasks at once' must depend on what is meant by the word 'task'." In the case of AudioStreamer 'task' means 'browse'. Since browsing does not imply detailed listening, one would expect that listeners can browse several streams at once.

At a base level the user of AudioStreamer can switch attention between streams at will. Sometimes the switch is the result of the listener losing interest in the stream currently being attended to. At other times the switch occurs because the listener's attention is attracted by something that's happening in the listener's peripheral hearing (a speaker change, a bomb blast, a familiar name being spoken, etc.). Furthermore, AudioStreamer can notify the listener about optimal times to switch attention to another stream, if that information is supplied to the system, as is discussed in the next chapter.

The bottom line is that listeners have a limited capacity to process information. A given listener can only process a certain maximum quantity of audio per unit time. If more than the maximum quantity is presented, some information is not processed. The great advantage of AudioStreamer is that it provides the listener with an efficient means of choosing which audio to process.

4. Initial Design

> Many things difficult to design prove easy to performance.
>
> (Johnson 1988)

The last chapter gave an overview of the perceptual and cognitive experiments and theories that influenced the design of AudioStreamer. It is clear from that discussion that humans do have the ability to listen to several streams of audio at once provided that detailed listening is not a requirement. AudioStreamer attempts to take advantage of this fact while minimizing any liabilities of simultaneous presentation (a high cognitive load for example).

This chapter gives an overview of the initial design and implementation of AudioStreamer. The chapter begins with a discussion of the hardware platform. The next section discusses the spatialization technology used by AudioStreamer. The third section discusses data gathering and processing. The fourth section presents a discussion of the user interface design including a section on the spatial layout of the sound sources, how figure and ground are indicated, how the system makes suggestions, and the two alternative input modalities used in the initial design.

4.1 Hardware Platform

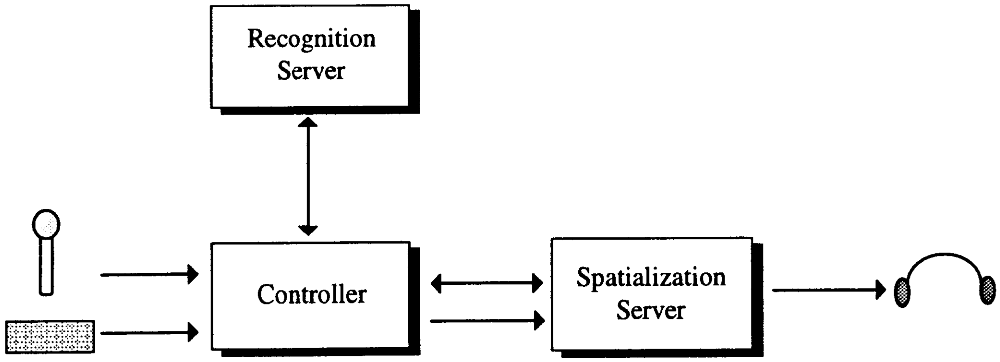

Figure 4-1 AudioStreamer Initial Hardware Configuration.

Figure 4-1 displays an overview of the hardware configuration used in AudioStreamer. The system consists of three primary parts: a *controller*, a Sun SPARCstation that handles input from the user (both button and speech) and serves as an audio server; a *recognition server*, a second SPARCstation that handles speech recognition; and a *spatialization server*, a PC (Intel 486 33 MHz) containing two Beachtron™ audio cards that perform audio spatialization. The remainder of this section will discuss the function of each of these components in greater detail.

The controlling SPARCstation plays several roles. It takes speech and keystroke input from the listener, interprets it, and routes the commands to the spatialization server via an RS-232 serial connection. Keystroke input is interpreted locally and sent to the spatialization server via another RS-232 serial connection. Speech input is sent to the recognition server for recognition via a TCP/IP connection. The result of recognition is passed back to the controller for interpretation and subsequent transfer to the spatialization server.

The controller also serves as the primary source of audio in the system. In addition to passing speech input to the recognition server for recognition, the controller sends audio to the spatialization server for presentation to the listener. In this role the controller makes use of multiple processes running the SPARCstation based audio server developed by the Speech Research Group at the MIT Media Lab (Arons 1992c). The SPARCstation audio server is based on a distributed architecture and allows multiple applications to share limited audio resources in an asynchronous and event-driven environment.

Finally the controller plays a utility role. It opens and closes connections to the recognition server and the spatialization server as needed. Error handling and cleanup also take place on the controller.

The recognition server runs on a separate SPARCstation and communicates with the controller over a TCP/IP connection. The recognition server is an implementation of a client/server model developed in the Speech Research Group at the Media Lab at MIT (Ly 1993). This model allows several distributed clients to make asynchronous connections to a particular recognition engine. In the case of AudioStreamer the recognition engine is from Texas Instruments (Wheatley 1992). The TI recognizer supports speaker-independent, connected-speech, large vocabulary recognition. The controller passes a buffer of input speech to the recognizer for recognition. The recognizer passes back a recognized utterance as well as additional data such as a confidence value.

The spatialization server controls the spatial position, acoustic characteristics, and presentation of the sound sources in AudioStreamer by relaying commands from the controller to the spatialization hardware (see the next section). In addition, the spatialization server maintains a model of the listener's head position and physical characteristics such as interaural separation. The controller sends interpreted commands and three channels of audio to the spatialization server. The audio channels are spatialized, mixed, and presented to the listener over conventional stereo headphones. The server code was developed by Crystal River Engineering for use with its line of three-dimensional audio products (CRE 1993). The client code was also developed by Crystal River and modified by the author to run under Sun-OS 4.1.3 on a SPARCstation.

4.2 Spatializing Audio

The last chapter outlined the three most salient acoustic cues for spatial location: interaural time differences, interaural intensity differences, and pinnae effects. Interaural time and intensity differences generally provide lateralization cues to the listener, while pinnae effects are responsible for externalization. Several researchers at the NASA Ames Research Center have developed a three-dimensional audio display based on the Convolvotron signal processing hardware (Wenzel 1992). The Convolvotron takes into account all three acoustic cues for localization and the results are surprisingly realistic.

AudioStreamer relies on the Beachtron signal processing hardware developed by Crystal River Engineering to spatialize audio in real-time (CRE 1993). The Beachtron is a less expensive variant of the Convolvotron. However it uses a similar synthesis technique based on the head-related transfer function (HRTF): the direction-dependent, acoustic effects imposed on an incoming signal by the outer ears. HRTFs are measured by placing small probe microphones in a listener's ears near the eardrums. The listener is seated in an anechoic chamber in the center of a spherical array of 74 loudspeakers arranged at equal intervals. A 4 Hz train of 75 acoustic clicks is played from each of the speakers in turn and the response of the probe microphones is recorded, averaged and digitized. The result is 75 FIR filters represented in the time domain for both the left and right ears.

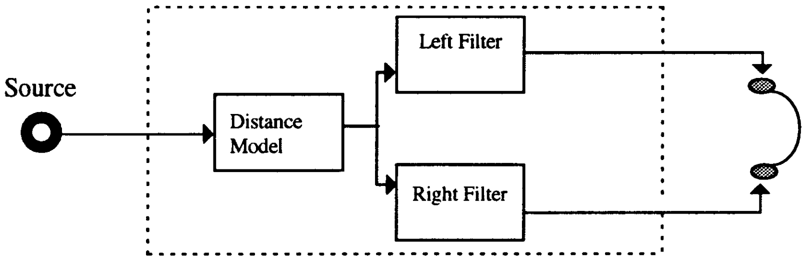

Figure 4-2 Spatialization processing on the Beachtron.

To render sounds at specific spatial locations the filters are first combined into left ear and right ear listener specific "location filters". At run time these location filters are downloaded into the memory of the Beachtron. An incoming audio signal is first pre-processed by a distance model that simulates atmospheric loss (see Figure 4-2). The output is then convolved with the left and right filters. The resulting spatialized signal is presented to the listener over stereo headphones.

Each Beachtron can simultaneously spatialize two sources. Both sources are processed as described above and the results are mixed before presentation to the listener. In addition the Beachtron supports external auxiliary inputs which are also mixed before presentation. As a result up to eight Beachtrons can be used simultaneously, allowing 16 sources to be spatialized. Two Beachtrons are installed in 16-bit ISA slots on the spatialization server.

4.3 Data Gathering

AudioStreamer presents three simultaneous recordings of news broadcasts to the listener. The broadcasts are automatically collected and processed each evening. Currently, the audio portions of the ABC Nightly News and CBS Evening News, and the evening broadcast of the BBC World Service is used.

Figure 4-3 depicts how the evening television news broadcasts are recorded and processed. A NeXT computer is connected to the audio output of a VCR with a cable television feed. In addition, the NeXT computer is connected to the output of a video line 21 decoder that also gets its input from the VCR. The unblanked portion of video line 21 transmits an encoded composite data signal that carries the closed captions for the news broadcast (Lentz 1980).

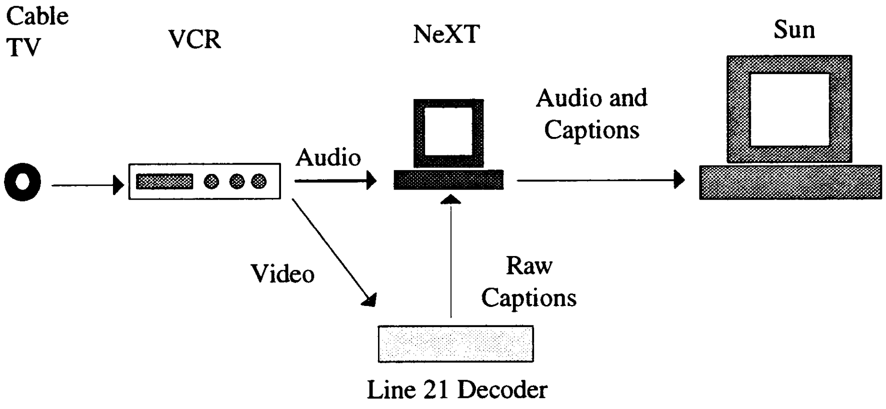

Figure 4-3 A diagram of television news processing.

In addition to containing transcriptions of the audio portion of the news broadcasts, closed captions also contain information about speaker changes and story boundaries. Speaker changes are indicated by the presence of a '>>' in the captions and story boundaries are marked with a '>>>'. A MACH process (getcaption) running on the NeXT collects the captions and time stamps each story boundary and speaker change marker as it is encountered. Another MACH process (recordtofile) digitally records the audio portion of the broadcast and stores it as an audio file. Since both processes are started simultaneously the actual boundaries and the time-stamped markers are roughly synchronized.

More accurate synchronization is achieved once the audio file and caption file are transferred to the Sun using an algorithm developed by Horner (Horner 1994). The algorithm relies on the presence of the significant pauses that occur during speaker and story changes. On the Sun the audio file is analyzed by a pause detection algorithm and significant pauses are marked. The output is compared to the time-stamped markers. The markers are adjusted to coincide with the preceding significant pause provided that the pause and time-stamp occur within a pre-defined window of time. The result is the near elimination of synchronization delay between the audio and captions.

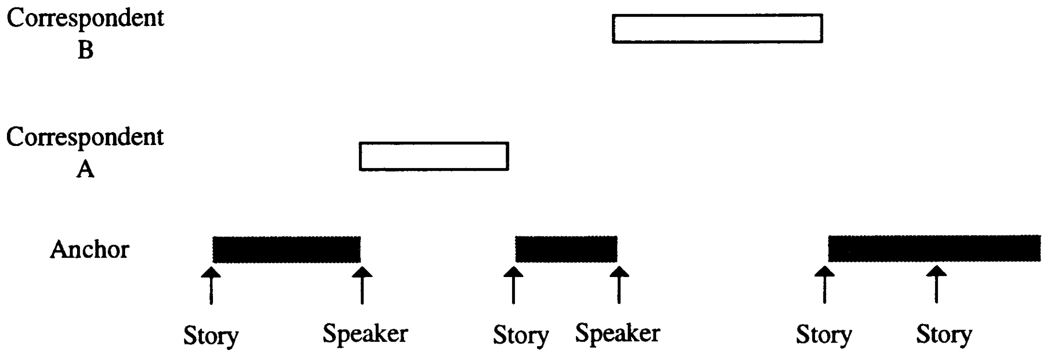

Figure 4-4 The structure of television news broadcasts.

A separate Perl script running on the Sun (sndnote_to_streamer) takes the output of the synchronization process and converts into a form useable by AudioStreamer. Figure 4-4 shows a graphical representation of the final product. The audio signal is segmented by story and speaker. Actualities (reports by a correspondent “on the scene”) are often the most interesting portion of a story. Actualities are generally indicated by the presence of a speaker change within a story. As will be described below, AudioStreamer uses this information to make suggestions to the listener about the presence of potentially interesting material on a particular stream.

Radio news is collected and processed in a different way. The audio is automatically recorded every evening on a Sun SPARCstation connected to a radio tuner. Since radio news is not closed captioned other information is used to segment the signal into interesting chunks. It is conceivable that future radio broadcasts will have text based content information transmitted with them on a side-band. But for the moment structure has to be extracted directly from the audio. Roy developed and implemented a speaker indexing system that segments an incoming audio signal by speaker and assigns a unique label to each speaker (Roy 1995). The recorded BBC radio broadcasts are processed by the speaker indexing system and the results are converted to a form compatible with

AudioStreamer. Although story boundaries are a more reliable indicator of when an attention shift will be most fruitful, speaker changes are often associated with the introduction of new and interesting information.

4.4 User Interface Design

This section gives an overview of the initial user interface design. Simplicity was the overriding concern in the initial design. The implementation was guided by some of the perceptual and cognitive theories presented in the last chapter. The resulting system is "perceiver controllable" as defined by Tufte -- the user can interact with the system by simply shifting attention, additional interaction is not necessary (Tufte 1990). The next chapter will discuss the weaknesses of the initial design and how they were addressed.

4.4.1 Spatial Layout

Figure 0-5 shows the spatial layout of the sound sources in AudioStreamer. A total of three streams of audio are playing simultaneously. The sound sources all appear to be 60 inches from the listener's head and are separated by 60 degrees. A 60 degree separation was chosen to facilitate selective attention while minimizing the time to switch attention between the streams (see chapter three for a more detailed discussion).

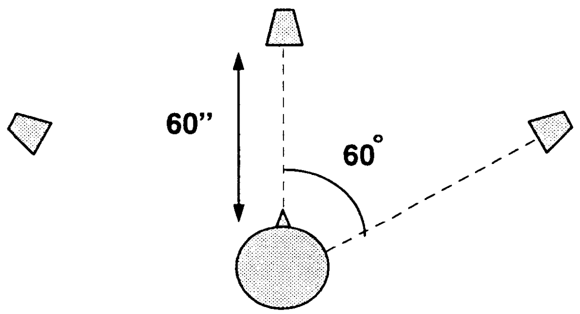

Figure 4-5 Spatial layout of sound sources in AudioStreamer

In the initial implementation the sound sources are not fixed in space, but move in unison with the listener's head. As mentioned in the last chapter there is evidence to suggest that front-back reversals are reduced if head motion relative to the sound sources is allowed. Head motion relative to the sound sources implies that they are fixed in space. Front-back reversals generally occur when discreet sounds are played to a listener. In the case of AudioStreamer the sound sources are playing continuously and front-back reversals are of little concern. Nonetheless, the subsequent iteration of the user interface supports head tracking and the listener has the option of fixing the sound sources in space.

The spatial layout of the sound sources is controlled by the Swindows software module (see appendix). Swindows imports the metaphor of graphical windows into the audio domain. Swindows supports up to N simultaneous sources, where N is hardware limited (N = 4 in the current system). Sources can be placed anywhere in the perceptual three-space of the listener. Sources can be *opened* - turned on; *closed* - turned off; *reshaped* - made louder or quieter; *moved*; and *destroyed.* The module is written in ANSI C and was designed to be portable to any system that supports Crystal River Engineering's Beachtron Client Protocol.

4.4.2 Figure and Ground in Audio

AudioStreamer allows the listener to put one stream into *focus*, making it more acoustically prominent than the other streams. The goal of allowing the listener to select an auditory focus is similar to allowing a viewer to select a visual focus in a graphical system -- attending to focused items is easier than attending to unfocused ones. In a graphical system, focus can be represented in various ways; color, brightness, opacity and blur are examples. In the auditory domain the palette is different. During the design of AudioStreamer several methods of indicating auditory focus was explored including: pitch, spatial location, frequency, and gain. Each of these methods will be discussed in turn.

Pitch shifting a stream to indicate focus was tested and eliminated for several reasons. First, a pitch shift alone did not make the stream acoustically prominent enough relative to the other streams. Second, the resulting signal sounded too artificial to listeners. Filtering focused and unfocused streams into separate frequency bands (hi pass and low pass) was tested and eliminated for similar reasons. Filtering did not provide enough separation between the streams to make it easy for a listener to attend to one of them. In addition, the filtered streams changed the quality of the voices (since some of the data was discarded during filtering) in undesirable ways.

Changing the spatial location of the focused stream by moving it closer to and in front of the listener had more promise than either pitch shifting or filtering because the focused stream was far more prominent than the unfocused streams. However, the movement from unfocused position to focused position was nearly instantaneous and the resulting discontinuity disrupted the listening process and was confusing to listeners. One possible solution was to move the source in small increments simulating a smooth trajectory from unfocused to focused position. Unfortunately the resulting time delay could be perceived as sluggish system response, so modifying the spatial location of the focused stream was also eliminated.

Increasing the gain of the focused stream relative to the unfocused stream does not exhibit any of the above shortcomings. The resulting signal is not distorted and is more prominent that the other streams. Also, the listening process is not disrupted and system response is instantaneous. As a result the initial design of AudioStreamer used an increase in gain (specifically 10 dB) to indicate focus. As is described below, during audio presentation the listener can put a stream into focus, switch the focus from one stream to another, or remove the focus altogether.

4.4.3 Making Suggestions

During the data gathering process the recorded audio is augmented with additional information. Television news is augmented with story boundary information derived from closed captions and radio news is augmented with speaker change markers produced by the speaker indexing system. At the end of the data gathering process this information is converted to a format compatible with AudioStreamer. This format consists of a list of pointers into the associated audio file. The pointers refer to the beginning of potentially interesting segments. When AudioStreamer is started the audio file and the list of pointers can be passed as arguments. At run time AudioStreamer uses this list to cue the listener to shift attention at salient times.

To cue the listener AudioStreamer plays a 400 Hz tone on the stream whenever a new segment begins playing. The tone is played at the same gain as the segment. A 400 Hz tone was chosen because in target detection experiments during a dichotic listening task a 400 Hz tone was almost always detected when played to the non-attended ear (see last chapter). During audio presentation the occurrence of these tones elicits an involuntary attention shift to the stream that played the tone. The shift may be of short duration if the material is of little interest to the listener. However, if the material has some import the listener may choose to continue attending to that stream or even give it the focus.

It should be noted that this process is completely data driven. An audio file and an optional list of pointers to interesting segments can be passed to AudioStreamer. The pointers can come from any source; closed captions, speaker indexing, hand coding, etc. Since the 400 Hz cue tone always leaks through the attention barrier, the listener has the option of switching attention to another stream at the beginning of a segment. The result is an increase in browsing efficiency.

In the absence of the list of pointers the listener can actively shift attention among the streams in search of material. The listener's attention may also be attracted by discontinuities in non-attended streams. Audio browsing is still accomplished, although it is not as efficient as when pointers to segments are provided; assuming that the pointers actually point to new information. In this sense AudioStreamer is scaleable -- as the quality and quantity of information on the location of interesting material increases so too does the efficiency of browsing. However, even in the absence of any collateral information, parallel browsing is more efficient than linear browsing.

4.4.4 Interacting with AudioStreamer

The listener interacts with the first iteration of AudioStreamer using speech commands, keyboard commands, or a combination of both. During a typical listening session the listener is seated in a chair wearing a pair of stereo headphones and an AKG head mounted microphone. The listener starts AudioStreamer on the controlling Sun SPARCstation in order to browse the previous evening's news broadcasts. All three streams begin playing simultaneously from their respective spatial locations. The stream to the left of the listener plays the CBS Evening News, the center stream plays the BBC World Service, and the stream to the right plays the ABC Nightly News. Initially all three streams have the same gain and no stream is in focus. At this point the listener can sit back and shift attention from one stream to another without giving explicit commands to the system. Alternatively, the listener can repeatedly cycle through the streams in turn giving each one the focus until something is found. Once the listener finds something of interest the corresponding stream can be put into focus for more detailed listening.

Speech	Keyboard	Function
“Start Listening”	‘l’	Accept commands
“Stop Listening”	‘o’	Ignore commands
“Left”	‘1’	Focus left stream
“Center”	‘2’	Focus center stream
“Right”	‘3’	Focus right stream
“Off Left”	‘F1’	Turn off left stream
“Off Center”	‘F2’	Turn off center stream
“Off Right”	‘F3’	Turn off right stream
“Rotate”	‘r’	Rotate stream clockwise
“Unfocus”	‘u’	Remove focus
“Quit”	‘q’	Exit and clean up

Table 1 Speech commands and keyboard equivalents.

Periodically as segment boundaries are crossed the cue tone is played on one of the streams. After the cue tone the listener can shift attention to that stream, give that stream the focus, or ignore the tone completely. The listener has the option of putting a stream in focus at any time. In addition, if very detailed listening is desired the listener can turn off the other streams until the interesting segment is finished playing. This process continues until all three streams are finished playing or the listener stops audio presentation.

Table 1 shows a list of speech commands and their keyboard equivalents. Speech commands are recorded on the controlling Sun SPARCstation and are subsequently sent to the recognition server for recognition. The results are sent back to the controller for interpretation. A hybrid speech/keyboard approach was decided upon because in some situations speech input is more appropriate and in others keyboard commands are more efficient. For example, when cycling through the streams giving each one the focus keyboard commands are more efficient; it is much easier to repeatedly type "1 2 3" than it is to say "Left Center Right" over and over again. On the other hand, when the listener is browsing by shifting attention only, he can sit back in a chair with eyes closed and concentrate on the task. To select a stream by giving it the focus the listener only has to say "left", "center", or "right". This is much less disruptive to the listening task than looking at the keyboard and typing a key.

The user interface to the initial system is non-intuitive and crude. The primary focus of the initial design was to evaluate the presentation mechanism to determine if it was possible for a listener to easily focus on a single stream in the presence of others, and if it was possible to switch attention between them without undue effort. It appears that this is the case. User interface design issues were addressed in the redesign.

4.5 Summary

The initial design of AudioStreamer was a "proof of concept" exercise. Informal testing with various researchers at the MIT Media Lab and with representatives of the News in the Future consortium confirmed that people are indeed capable of browsing audio with AudioStreamer. User reaction varied widely. Several people reported that they thought the task was incredibly difficult. One person reported being "overwhelmed by too much noise, I kept switching attention between streams but couldn't settle down on any one." In contrast another user said, "That was great, at first I had difficulty, but once I made one stream louder I had no problem concentrating on it." In general the reaction was split. Some users had difficulty sorting out the auditory space, while others adapted relatively quickly. One user provided an insight that proved extremely useful. After listening for several minutes he reported, "it's hard but if I look in the direction of the source it's easier to concentrate on." The redesign of AudioStreamer incorporated this insight into the design of its user interface.

By spatially separating the streams and by providing a method of placing a stream in focus, AudioStreamer allows the listener to easily monitor simultaneous streams of audio. The next chapter outlines a modification of the presentation technique that makes use of the structure of radio and television news broadcasts. Based on this structure AudioStreamer can predict when potentially interesting information will be played on a stream and can cue the listener accordingly. As a result attention switches are much more productive.

In the next chapter the second iteration of the user interface is also presented. The updated version makes use of the same building blocks, Swindows for instance, as the initial version, but the user interface was modified based on feedback from several users. The primary complaint among users of the initial version was that interacting with the system detracted from listening. The version of the user interface discussed in the next section seeks to solve this problem.

5. Refining the Design

The human ear offers not just another hole in the body, but a *hole in the head.*

(Kahn 1992)

Chapter Four outlines the initial design and implementation of AudioStreamer. This chapter discusses refinements and changes that were made to the initial design in response to suggestions by various users of the system. The first section outlines how the notion of focus was redefined based on user feedback. Section two relays how the manner in which suggestions are made by the system was modified in the redesign. Section three talks about how non-speech audio feedback is incorporated in the redesign. Finally, section four presents two new methods for obtaining user input.

5.1 Redefining "Focus"

Listening to three simultaneous streams of audio requires a great deal of mental effort. For optimal browsing results most of the listener's cognitive resources should be devoted to the listening task. Unfortunately, most users of the initial version of AudioStreamer operated in what can be called "interaction mode" -- cycling through the streams one at a time, giving each the focus in turn. In cycle mode the listener invests a great deal of effort in interacting with the system. Interaction mode can be contrasted with "listening mode" where the listener switches attention from stream to stream without explicitly giving any stream the focus until more detailed listening is desired. Listeners in interaction mode also had a tendency to only give each stream the focus for a very short time, whereas in listening mode the listener's attention tended to linger on a particular stream for a longer period.

As was discussed in Chapter three, listening imposes a high cognitive load on the listener. One often cited advantage of speech based systems is that they leave or hands and eyes free for other purposes. Unfortunately, simultaneous presentation requires so much concentration on listening task that little capacity remains for other activities. Stifelman reports that, "... it seems there would be little cognitive capacity remaining for any additional activity -- subjects' eyes were not open to look at other things, they were tightly shut in deep concentration." The redesign of AudioStreamer seeks to maximize the cognitive resources devoted to listening by minimizing the amount of interaction that is required of the listener. This goal manifests itself in several ways. First, the system is biased towards browsing -- in the absence of user input the system automatically returns to the browse state where all streams are playing and none is in focus. Second, the system assumes that interaction with a stream is an indication of the listener's interest in the current segment playing on that stream. In other words, since interaction requires effort, the more the listener interacts with a stream, the more effort he has expended on listening to that stream, and the system responds accordingly by giving that stream more prominence relative to the other streams. The remainder of this section will give a detailed account of the way in which the version two of AudioStreamer implements these ideas.

In the initial implementation of AudioStreamer the listener explicitly interacted with system by giving a stream the focus, removing the focus, or switching the focus from stream to stream. In the redesign the notion of auditory focus was reimplemented to reflect the system's bias towards browsing. Instead of having the focus be either on or off, version two allows the focus to vary with time. For example, when the listener gives the focus to the left stream the system responds by immediately increasing the gain of that stream by 10 dB. Thereafter the gain begins a smooth linear decay back to the background level. The decay takes place over a period of approximately seven seconds giving the listener an opportunity to decide if the current segment is of interest.

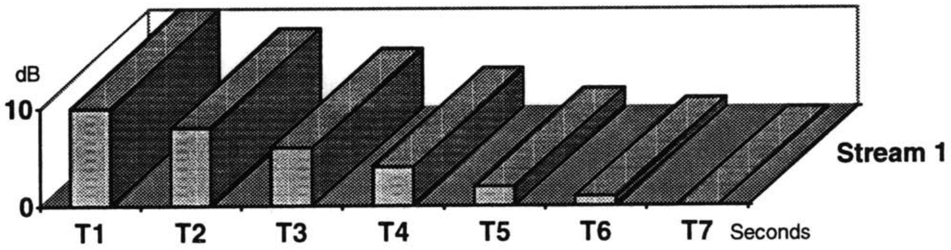

Figure 5-1 Linear decay of gain (representing focus) with time.

Figure gives a graphic representation of the process. The vertical axis represents gain and the horizontal axis represents time. At time T1 the listener gives stream one the focus. By time T7 the stream is no longer in focus. By allowing the focus to decay in this manner the listener is freed from having to explicitly remove the focus from a stream. Given that most users operating in search mode only gave the focus to a stream for a short time this represents a net gain in resources for listening. It should be noted that the listener is still able to explicitly remove the focus at which point the gain immediately returns to background level. Also the listener can switch focus to another stream and again the gain is immediately reduced.

The implementation of focus in version two was also influenced by the idea that more interaction with a stream implies greater interest in that stream. Rather than having a single degree of focus, version two introduces the concept of *levels of focus.* Figure 5-2 illustrates a sample interaction with a stream. At time T3 the listener gives the stream the focus. The system responds by giving that stream focus level one indicated by a 10 dB gain increase and approximately seven second decay time. At time T5 the listener selects the stream *before* the gain has decayed to background level. This time the system responds by setting the focus to level two, which has a higher initial gain and slower decay than level one. If the listener had let the focus decay to background level the system would have responded to the reselection by setting the focus to level one again. At time T8 the listener selects the stream again giving it level three focus and at time T13 the stream is selected once more giving it level four, the highest level, focus. Using system defaults, level three focus has a higher initial gain than level two and an even slower decay. At level four focus the initial gain is set high and does not decay at all. Instead the other two streams are paused and the listener can concentrate solely on the selected stream. The assumption is that level four focus indicates very strong interest in the segment playing on that stream and the listener wishes to listen to it in detail.

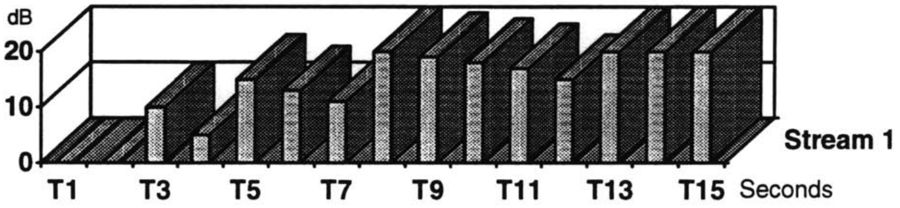

Figure 5-2 The interaction of focus levels and gain.

The user can set the initial gain and decay rate for all levels of focus at run-time by specifying a configuration file in the appropriate format (see appendix). In addition the user can specify whether unfocused streams should be paused at level four. Table 2 shows the default values used by AudioStreamer. The values implement the notion that as the listener moves up the levels of focus more detailed listening is desired at each step. It should be noted that these values represent the authors listening preference and were not derived experimentally. In practice each listener will want to choose their own values.

Several additional modifications were made to the way focus was implemented. First, focus only lasts through the current segment playing on a stream. Since continuity among adjacent segments is not ensured, focus level should be a property of the segment not the stream, since there is no guarantee that a listener will be interested in the next segment. Second, version two uses a *focus window* -- a period of time during which the current focus level is persistent. For example, assume that the listener has the center stream in focus at level three and something attracts his attention to the right stream (a familiar voice for instance). At this point the listener gives the focus to the right stream only to discover that the segment playing on the center stream is more interesting. Rather than requiring the listener to reset the focus on the center stream to level three by going through level one and level two, AudioStreamer automatically returns to level three on the center stream, provided that the listener switched the focus back to the center stream within a predefined period of time (the default is five seconds). As with the initial gain and rate of decay of the various focus levels, the listener can set the extent of the focus window in a configuration file.

Focus Level	Default
One	Initial Gain = 10 dB
	Decay time = 7 seconds
Two	Initial Gain = 12.5 dB
	Decay time = 15 seconds
Three	Initial Gain = 15 dB
	Decay time = 20 seconds
Four	Initial Gain = 15 dB
	No decay, other streams paused.

Table 2 Default focus parameters.

In version one all streams began playing simultaneously and users were observed to immediately give the focus to all of the streams several times. When asked about this one user remarked that he was overcome by “a wall of sound” and was cycling through the streams “to sort things out." To eliminate the need for this initial flurry of activity version two employs a staggered st`rt. The center stream begins playing first, followed by the right stream five seconds later, and after an additional five seconds the left stream begins playing. This allows the listener to gradually sort out the spatial layout of the sources and to “lock on” to each stream at the beginning of the initial segment. While a staggered start does not directly impact the implementation of focus it does appear to reduce the number of times listeners set the focus at the beginning of audio presentation.

5.2 Making Suggestions - Part 2

The redesign of AudioStreamer augments the way in which suggestions are made to the listener. The initial version of AudioStreamer played a 400 Hz tone at the beginning of each segment to inform the listener of some potentially interesting information playing on a stream. Version two takes this one step farther by increasing the gain and letting it decay in addition to playing a cue tone.

There are two reasons for this change. First, without interacting with the system at all the listener hears the beginning of each segment at a higher gain. Figure 5-3 illustrates this phenomenon. As segment boundaries are crossed each stream plays the cue tone and increases the gain 10 dB by default. The gain decays over a period of seven seconds (the default) just as if the listener had put that stream in focus. In practice simultaneous segment boundaries on multiple streams rarely occur (AudioStreamer can be modified to ensure that this *never* occurs) so the listener can concentrate on switching attention from stream to stream at the beginning of each segment. Broadcast news stories are generally written to quickly get the user's attention by presenting the theme of the story as soon as possible. By making the beginning of a segment more prominent AudioStreamer maximizes the chances of attracting the listener's attention at a time when decisions about the potential interest of a story can be made very quickly.

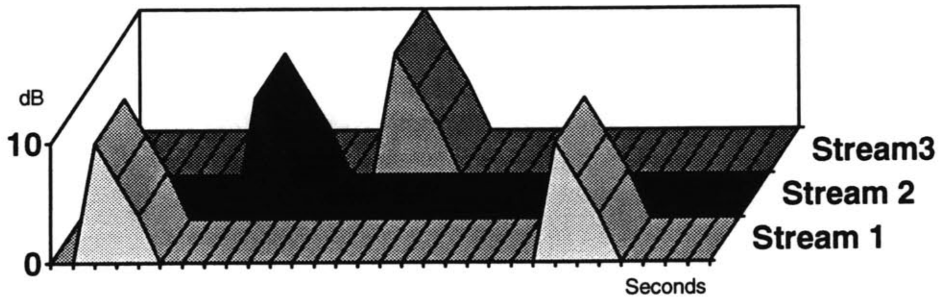

Figure 5-3 System produced gain increases at segment boundaries.

The second reason for using gain changes in conjunction with tones relates to levels of focus. If the listener has a stream in focus at level three and a segment boundary is crossed on another stream, background gain levels do not make the stream prominent enough, relative to the focused stream, for the listener to easily switch attention to determine what the new segment is about. By increasing the gain and allowing it to decay the listener can attend to it long enough to decide if it is interesting without changing the focus.

It should be noted that the initial gain change and decay at segment boundaries is *not* implemented as a focus level. If the listener changes the focus to a stream during the initial gain change the system responds by setting the focus level to one, not two. To avoid confusion, focus is solely controlled by the listener. The system can make suggestions about when focus changes might be productive, but the final decision rests with the listener.

5.3 Non-Speech Audio Feedback

AudioStreamer makes use of non-speech audio feedback in various places. Audio feedback is used because AudioStreamer does not have a visual display. Feedback is further limited to non-speech audio for several reasons. First, AudioStreamer's auditory space is already filled to capacity with speech output. By using non-speech audio, relevant messages are passed to the listener without unnecessarily cluttering the auditory space. Secondly, non-speech audio is generally more concise and compact than speech feedback. Since the listener's cognitive capacity is at a premium, any time savings translate into a net gain in efficiency. Feedback is kept to a minimum and is only used when the result of a listener's actions is not apparent. Also, all feedback is inline. In other words, feedback relating to a particular stream appears to emanate from the spatial location of that stream.

Type of Feedback	Function
Gain Increase and Decay	Corresponds to changes in focus level. Also marks the beginning of a segment. Generated in response to listener commands and automatically by AudioStreamer.
Boundary Tones	400 Hz tone at the beginning of a segment. Designed to inform the listener of potentially interesting material. Occurs in conjunction with gain changes.
Level Tones	An auditory indication of focus level. Focus level is indicated by playing short duration tones of different pitch on a stream using the Proteus synthesizer on the Beachtron.

Table 3 Types of non-speech audio feedback used in AudioStreamer.

Table 3 shows the three types of feedback employed by AudioStreamer. Increasing the gain and letting it decay is used in AudioStreamer to temporarily make a stream more prominent. Gain changes occur in two instances: the listener explicitly gives a stream the focus; and the system automatically increases the gain at segment boundaries. Tone feedback is used in two instances. AudioStreamer automatically generates a short duration 400 Hz tone at segment boundaries. In addition in version two of AudioStreamer tones are generated in response to a change in focus level. Each Beachtron audio processor has an on board Proteus 32-voice music synthesizer with a full MIDI interface (TBS 1992). AudioStreamer uses the synthesizer to play a tone to the listener every time the focus level is changed. The tones aid the listener in keeping track of the current focus level on a particular stream (several listeners complained that they did not know the current focus level after a selection). Table 4 shows the various focus levels and their corresponding tones.

In response to a change in focus level, AudioStreamer generates the appropriate MIDI sequence and passes it to the Beachtron via the serial connection. The tone itself is actually played on the fourth audio source -- two Beachtron cards support four spatialized audio sources, three of them are used to present news broadcasts to the listener. Before the tone is played the fourth audio source is moved to coincide with the spatial location of the source on which the tone is to be played. From the listener's perspective the tone appears to emanate from the selected source.

Focus Level	Tone
Level one	Middle C
Level two	Middle C + 1 Octave
Level three	Middle C + 2 Octaves
Level Four	Middle C + 3 Octaves
Unfocus	Middle C - 1 Octave.

Instrument #12: Vibraphone
Velocity: 63

Table 4 Focus levels and their corresponding tones.

5.4 User Input

A primary design goal for version two of AudioStreamer was to minimize the need for user input. While this goal was achieved, the user still needs to periodically interact with the system to effectively browse the audio. Ideally, the mode of interaction should reinforce the listening process, not detract from it. Toward that end, version two supports two new input methods: head pointing and gesture.

5.4.1 Head Pointing

Version two of AudioStreamer allows the listener to give the focus to a stream by turning his head towards the direction of the stream. Head orientation is computed using a Polhemus 3Space Isotrak head tracking system (Polhemus 1992). The Isotrak uses low-frequency magnetic field technology to determine the position and orientation of a sensor in relation to a magnetic source. The source is placed directly in front of the seated listener and the sensor is mounted on top of the stereo headphones the listener wears during audio presentation. Both the source and the sensor are connected to the Polhemus systems electronics unit (SEU) which contains the hardware and software to compute the position and orientation of the sensor with six degrees of freedom. The SEU is connected to a serial port on the controlling Sun SPARCstation. Approximately 30 times per second the SPARCstation polls the SEU to determine the current orientation of the listener's head. AudioStreamer makes decisions about which stream is in focus based on the orientation of the listener's head.

Figure 5-4 shows the layout of the auditory interaction space from the listener's point of view. As in version one, each stream is placed six feet from the listener, horizontally separated from the other streams by 60 degrees. In addition, in version two, the center source is elevated 30 degrees with respect to the listener. By elevating his head 20 degrees or more the listener gives the focus to the center stream. Similarly, by rotating his head to the left or right by 20 degrees the focus is given to the left or right stream respectively. The 20 degree cone in the center is the neutral area where no commands are interpreted. Finally, if the listener tilts his head down 20 degrees the focus is removed. The listener is not required to return his head position to the neutral area after command. For example, to give the focus to the left stream and then the center stream, the listener rotates his head to the left 20 degrees and the rotates up and towards the center stream to give it the focus.

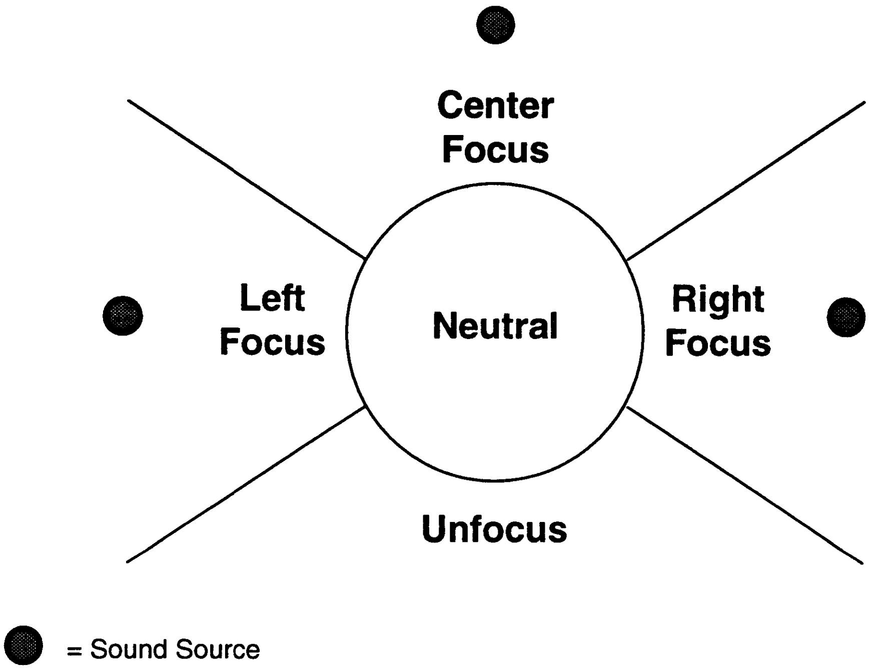

Figure 5-4 Partitioning of the auditory interaction space from the listener's point of view.

Using head pointing as opposed to speech or button input reinforces the listening process. Several users reported that it was easier to attend to a stream by looking in its direction. Since head-pointing does not require the listener to shift modes, as in switching from listening to speaking, or typing, for example, his attention can remain concentrated on the stream of interest in a natural way.

A useful by-product employing the Polhemus head tracking technology is that it allows AudioStreamer to fix the sources in the auditory space. The Beachtron maintains a real-time model of the listener's head. In addition to size and interaural separation, the model also takes into consideration the position and orientation of the head. At runtime the listener can request that AudioStreamer update this model. In this mode AudioStreamer continues to poll the SEU 30 times per second, but in addition to using head orientation to make focus decisions, AudioStreamer passes this information to the Beachtron. The result is a very realistic simulation of fixed audio sources in the listener's perceptual environment.

5.4.2 Gesture -- The Smart Chair

A second interface to AudioStreamer was developed for version two as an example of an experimental application of electric field (EF) sensing technology to human-computer interfaces. A chair was outfitted with four EF receivers and a transmitter developed by the Physics and Media Group at the MIT Media Laboratory (Zimmerman 1995). The sensors are used to interpret gestures made by the user to determine which stream, if any, should be in focus. An EF transmitter was mounted under the fabric in the seat of a standard padded chair. The EF transmitter couples low-frequency energy into the listener making him an EF emitter. Two sets of two receivers were placed in the head rest and arm rests of the chair. These receivers measure relative head rotation and hand proximity. The listener can give a stream the focus by using head rotation, hand motion, or a combination of both. Hand signals are interpreted as follows. In the neutral position both hands are in contact with the arm rests. Lifting the left hand gives the focus to the left stream. Lifting the right hand puts the right stream in focus. The center stream is given the focus by simultaneously lifting both hands. The focus can be removed by dropping both hands along the sides of the chair.

Head gestures have a similar syntax as head pointing with the Isotrak. In the neutral position the listener is facing forward without touching the headrest. The left stream is given the focus by rotating the head to the left and the right stream is given the focus by rotating the head to the right. The center stream is put in focus by touching the head to the head rest, and the focus is removed by tilting the head down and away from the headrest.

Using EF technology has an advantage over traditional head tracking technology, such as the Isotrak, in that it does not require that wires be connected to the user. Unfortunately, for the purposes of AudioStreamer, it has one drawback -- only relative head position and orientation is detected, not absolute orientation and position. This manifests itself in requiring the listener to return to the neutral position after a command is issued. In a preliminary version of the interface this was not a requirement and users frequently gave inadvertent commands to the system while unwittingly moving their hands or heads a little bit. The problem was minimized by requiring them to return to the neutral position after each command. Unfortunately, the resulting interface restricted the user's movements too severely.

Of the two methods of input just outlined, hand signals and head rotation, head rotation appears to be most effective. Hand signals suffer from the same drawback as speech and button input by detracting from listening. On the other hand, like head pointing, head rotation appears to reinforce the listener's ability to attend to the selected stream.

There are several drawbacks to the smart chair interface that warrant discussion. First, due to the initial implementation of the EF sensing device the system had to be laboriously recalibrated for each user. In effect the system had to be tuned based on a persons physical dimensions. Secondly, the interface restricts a listener's movements far too much. Users often erroneously put a stream in focus by moving their head small amounts or by unconsciously lifting their arms slightly. In contrast, the head pointing interface does not restrict a user's movements more than a conventional graphical interface would. A more sophisticated redesign of the smart chair interface may achieve more favorable results.

5.5 Summary

User reaction to version two of AudioStreamer was mixed but generally more favorable that for version one. In the initial version listeners tended to spend a large amount of time sorting out the auditory space. A great deal of effort seemed to be expended on interacting with the system. In version two listeners who made use of the head pointing interface spent more time sitting in the chair with their eyes closed concentrating on the listening task. One user remarked that "when the other channel beeped I couldn't help but listen to it." Another remarked that he felt that he was in a "noisy ballroom" and preferred to listen to the radio in a quiet setting. Finally one listener said, "that was hard, but I can imagine how hard it would be if the sounds all came from the same place." Also several people reported having difficulty in localizing the sounds. This last problem might be solved by using more sophisticated spatialization hardware.

The redesign of AudioStreamer successfully addressed some of the problems presented by the initial design. The goal of maximizing the amount of the user's cognitive resources that are devoted to listening was achieved in two ways: by reducing the amount of interaction to a minimum; and by implementing an input device that reinforces the listening process (the head pointing interface). Also selective attention to streams of interest was increased by the use of automatic gain increases at segment boundaries. Of the three interfaces that were presented in this chapter head pointing was most well received. However, a more elaborate implementation of the head rotation interface would probably achieve similar results.

6. Conclusion

> ... no silence exists which is not pregnant with sound.
>
> (Cage 1961)

This final chapter presents some thoughts on areas for future research and discusses the goals of parallel presentation of audio and how they are achieved. Section one discusses ways in which AudioStreamer's browsing efficiency can be enhanced by extracting structure from *within* segments. Section two discusses some potential applications for AudioStreamer and spatial audio. Section three reviews some possible experiments on the interaction of time-compression and parallel presentation as well as the extent of training effects. The chapter concludes with a summary of the thesis and a recap of the goals of AudioStreamer.

6.1 Browsing within Segments

AudioStreamer uses structure (story and speaker change boundaries) to inform the listener of the occurrence of new and potentially interesting information. The granularity of this information is at the segment level -- a new story is beginning or the speaker has changed. Browsing within a segment is the responsibility of the listener and AudioStreamer does not give any explicit hints as to what might be interesting. AudioStreamer can be enhanced in several ways to enable more efficient intra-segment browsing. Two such enhancements are outlined below.

6.1.1 Closed-Captions and Filtering

AudioStreamer only makes use of a small fraction of the information that is available in the closed-captions of a television news broadcast. While the story and speaker boundary markers are used to great advantage, the actual content of the audio, as a transcript, is discarded.

Rather than throwing this information away, AudioStreamer can use it to modify the order in which stories are presented to the listener based on a user profile. The content of a story can be represented using a standard vector space model based on text from closed-captions (Salton 1989). Story content is extracted by recognizing individual words in the closed-captions, eliminating common words included on a stop list, and using the remaining words to construct a weighted vector representing the content of the story.

Similarly, the listener profile is also represented as a weighted vector of words that is constructed automatically and changes as the listener's interest changes (see below). To order the stories for presentation a similarity measure between each story vector and the profile vector is computed (using the cosine of the angle between the story vector and the profile vector). The stories are rank ordered based on this similarity metric and are presented to the listener in that order.

The listener profile vector is adjusted after each listening session using a form of relevance feedback. In traditional relevance feedback a query vector is modified based on a user's feedback relating to the relevance of a particular document vector to the query. In the case of AudioStreamer the profile vector is modified based on *how long* a particular story was in focus during audio presentation (by appropriately weighting the story vector and taking the sum of the profile vector and weighted story vector). The computation is carried out for each story that is presented to the listener. Relevance feedback of this form allows the profile vector to track the changing interests of the listener. This is extremely important for news presentation since new and interesting stories emerge repeatedly.

Serendipity -- finding a story of interest purely by chance -- is retained in the system in two ways. First, the system only reorders the stories. It does not eliminate any. If the listener chooses to listen to all of the stories in the broadcasts, interesting stories not selected for by the listener profile are still heard. Secondly, genetic algorithms that introduce a chance element into the filtering process can be incorporated in the design to prevent over specialization of the listener profile. Sheth has developed such a system for filtering network news groups with positive results (Sheth 1994).

The success of this system is based in large part on the accuracy of user modeling and is subject to the same requirements. For example, a large thesaurus is essential since people use various words to describe similar concepts. This is further aggravated by the fact that news stories often import words from other languages that may not be in common usage or make use of proper nouns that can be misinterpreted. For instance, people interested in war would like to hear stories that contain the word "Jihad", but people interested in organic gardening probably do`not care about stories that contain "Apple today announced the shipment of a new line of Macintosh."

6.1.2 Emphasis Detection

In SpeechSkimmer, Arons developed a method of segmenting a speech recording based on emphasis detection (Arons 1994). Emphasized portions of speech often signal the presence of important material or indicate the introduction of new material. Arons' algorithm divided a speech recording into a number of windows (100 10ms frames). The standard deviation and number of frames with an F0 above a certain threshold are used as indicators of emphasis.

AudioStreamer can use this information to make suggestions to the listener with a story or speaker segment. A list of emphasis points can be converted to a segment file (see appendix) and subsequently passed to AudioStreamer. Making use of emphasis information does not require any change to AudioStreamer since the design is data driven.

6.2 Applications

This section outlines some ideas on further applications for parallel presentation. Several of the applications were implemented during the redesign of AudioStreamer.

6.2.1 Browsing Voice Mail

AudioStreamer is designed to browse databases of stored audio. One such database that occurs naturally in an office environment is voice mail. Busy executives frequently find themselves returning to the office to face the task of wading through a backlog of voice mail. Time sensitive or important information that may be contained in some of the messages generally requires prompt responses, especially given that time is in general limited. Unfortunately, those messages are often buried in a collection of less important messages. AudioStreamer can be used to browse through the collected voice mail messages to determine who called and which messages need immediate responses. Detailed listening to all of the messages can be done at the receiver's leisure.

As part of the redesign of AudioStreamer a configuration file tuned to voice mail was developed (see appendix). Voice mail messages tend to be much shorter than news stories. AudioStreamer's parameters needed to be modified to reflect this difference. The configuration file (named vmail_config) implements shorter delay times in the staggered start as well as a faster gain decay rate at all levels of focus. Since listeners generally do not have time to reach the higher focus levels during short voice mail messages, vmail_config disables focus level three and four.

A Perl script was written to take a user's voice mail box and generate an appropriate segment file for AudioStreamer. The script first sorts the messages by date (from most recent to oldest). After sorting the messages, the script builds a segment file where each piece of voice mail is treated as its own segment. During playback the listener first hears the most recent messages in parallel followed by older messages. This process continues until all messages are played back, or the listener stops the system. Listening to voice mail in parallel is useful for people who receive lots of mail of varying priority. By listening to several messages at once the listener can quickly sort out the messages that need an immediate response. Also, the author observed that messages from important people (one's spouse for example) tended to stand out because of the familiarity of the voice.

As a follow-up experiment the Perl script was generalized to build a segment file from any directory of audio files. The user passes a directory path to the script and the script generates a segment file and a configuration file. The segment file is built by treating all of the audio files in the directory as segments. The configuration file is optimized to the *average* length of the audio files in the directory. This average length is used to compute appropriate decay rates for the various focus levels. In practice this optimization method is simple and works quite well. However the results can be skewed by one or two exceptionally long or short audio files -- a decay taking place over 30 seconds does not make sense for a 20 second piece of audio. These problems can be overcome by implementing a more sophisticated optimization heuristic.

6.2.2 Parallel Fast-Forward

Another application of parallel presentation involves browsing long audio files such as a recorded lecture. Based on extracted structure, such as pauses, emphasis, or auxiliary information such as when slides where changed, it is conceivable that a long recording can be divided into cohesive units. A modified version of AudioStreamer can be used to present the units in parallel. This has several advantages over traditional sequential fast-forward. First, since time-compression is not used, the listener hears each unit in an unaltered form -- the rhythm and emphasis of the speaker remains intact. Second, by playing units in parallel the listener can compare the relative information content of each unit as it is playing. This is useful in the case of a long lecture where the listener may only be interested in portions of the recording.

6.2.3 Browsing Video

A significant amount of the content of a video tape, of a documentary for example, is contained on its audio track. It follows that to effectively browse the content of a video the audio must be browsed as well. Parallel presentation of audio can be combined with simultaneous video presentation in the design of a very powerful browser. The browser derives its power from the synergy of presenting information on two perceptual channels, auditory and visual. Visual corroboration of auditory information and vice versa has the potential to increase the overall information transmission rate.

As an example, the application might look like the following. The viewer is placed in front of three video displays that are simultaneously playing three different segments of video. The viewer listens to three parallel audio tracks spatially separated to coincide with the placement of the video displays. By switching attention between the parallel audio and video pairs the viewer browses the video in faster than real-time.

6.2.4 Navigating Audio-Only Hypermedia

One of the drawbacks of audio-only hypermedia is a lack of reference points for navigation. Users of HyperSpeech, for example, have a tendency to become lost in the network, repeatedly visiting nodes that have been heard before. One method of minimizing this inefficiency is to use "audio bread crumbs" -- cue tones that indicate that a node was visited before. Spatialized audio can also be used to increase the efficiency of exploring the network. By assigning each node a unique position in the listener's three-dimensional auditory space and by allowing the listener to move thorough the space, the system can leverage the users spatial memory. To reduce disorientation farther, several nodes can be playing simultaneously allowing the listener to choose where to go next without interacting with the system.

6.3 Further Research

This section briefly outlines two areas for further research: The interaction of time-scaling and parallel presentation and how browsing efficiency under parallel presentation improves with training.

6.3.1 Time Scaling and Parallel Presentation

As an experiment, version two of AudioStreamer was modified to incorporate time-compression at focus level four. If the listener gives a stream focus level four the other streams are paused as before. In addition, AudioStreamer gradually increases the time-compression of the selected stream from 1.0 to 2.0 times real-time. Once the end of the segment is reached the compression ration is reset to 1.0 and the other streams begin playing again.

A more interesting area of research is to determine how time-scaling interacts with parallel presentation when both techniques are used in unison. The assumption is that time-scaling and parallel presentation are to some degree complementary cognitive activities. A controlled experiment can be designed to ascertain the extent of this complementarity and its effect on browsing efficiency. It would be interesting to determine if there is a time-scaling factor where browsing efficiency dramatically increases or decreases. For example, it is conceivable that when presenting three simultaneous streams of information-dense audio, a scaling factor of 0.5 (half normal speed) will result in a net gain in time efficiency. Three simultaneous streams at half speed is still faster than three streams in sequence. Another possibility is that for news broadcasts a uniform scaling factor of 1.25 might be appropriate. On the other hand, perhaps it is feasible to only compress one stream while the others are played back at normal speed.

6.3.2 Training Effects

Intelligibility and comprehension of time-compressed speech increase with training. Novice listeners quickly adapt to a compression ration of 50%, but with more training much higher compression is possible (Orr 1965). Parallel presentation is expected to exhibit the same kind of behavior (this is informally confirmed by the experience of the author). A controlled experiment can be designed to determine the extent of the increase in browsing efficiency as well as the amount of time that is required to achieve competence.

6.3.3 Language and Content

Language and content impacts the usefulness of AudioStreamer in several ways. In this implementation of AudioStreamer broadcast news was chosen as data to present to a listener. All three channels of audio presented similar high-density information. In Webster and Thompson experiment (see chapter three) they reported that the number correct responses was much lower for material having equal density per channel than for material where the density varied from channel to channel. An experiment could be designed to more exactly determine the benefits of playing different information on the various channels. For example, it is conceivable that listeners could monitor a news broadcast, a baseball game, and a radio show such as "A Prairie Home Companion" and miss little of the interesting information.

6.4 Summary

Speech is a wonderfully rich channel of communication. Text has expressive power as well but pales in comparison to the range of information that can be transmitted with the power of the human voice. Rhythm, meter, intonation, emphasis, accent, and hesitation are the vehicles whereby words are given their power and impact. Unfortunately, for the designers of computer systems speech also has significant liabilities. Naturally occurring speech is slow. Recorded speech is difficult to browse, bulky and opaque. This research addresses these liabilities in a novel and principled way.

The initial implementation of AudioStreamer took some basic ideas about parallel presentation and embodied them in a system for browsing audio. While the implementation was rudimentary in many ways, it immediately became clear that some users could successfully browse three simultaneous channels of audio. Based on feedback from users of version one, a more sophisticated system was developed. The redesign focused on building a user interface that reinforces the listening processes. Informal reports suggest that this goal was achieved. Several styles of interaction were explored with the head-pointing interface being particularly successful. Although some users found version two of AudioStreamer difficult to listen to, most thought that with a little exposure browsing became much easier.

Simultaneous presentation of spatialized audio has the potential to solve many problems in current audio-based systems. During the design and implementation of both versions of AudioStreamer it quickly became apparent that as many questions were being posed as were being answered. Novel avenues for further research constantly presented themselves. In the past, spatial audio has been primarily used in the entertainment and virtual reality fields. This thesis is part of a movement to apply these technologies to the rich arena of solving real-world problems facing all of us in a society where time is increasingly at a premium.

A. Software Documentation

This appendix provides developer's documentation for using the Swindows and Streams software modules. The following appendices outlined the scripts used by AudioStreamer and the format of the segment files.

Swindows

SUMMARY

The Swindows, "sound windows", software module implements a windowing style abstraction on top of the CRE Client module developed by Crystal River Engineering and the author. Users of Swindows can "open", "close", and "move" sound windows in three dimensions.

Familiarity with the Speech Group's audio library and audio server is required for using this module. See the appropriate documentation.

DEPENDENCIES

The start_sources script (see Appendix B) must be executed at runtime before Swindows will be operational.

INCLUDES

The include file swindows.h must be included in application programs.

STRUCTURES

The swindow structure caches information that is required by various functions at run time.

```
typedef struct
        {
        int id;                  /* Unique ID for window (fd of server) */
        int b_id;                /* ID used by the Beachtron            */
        int focus;               /* True if swindow is the focus        */
        float gain;              /* Current gain                        */
        float last_gain;         /* Current non-focus gain              */
        float slocation[3];      /* Spherical location of swindow       */
        float blocation[6];      /* Location in Beachtron coordintates  */
        } swindow;
```

EXTERNAL INTERFACE

The following is a listing of exported functions from the Swindows module.

```
extern int
sw_init();
```

Initializes the swindow system. Allocates memory for maximum number of windows (hardware limited). Set default location and gain.
Returns true on success FALSE on failure

```
extern int
sw_kill();
```

Cleans up and exits Swindow system. Stops playing windows and reclaims storage.
Returns true on success FALSE on failure

```
extern int
sw_main_loop();
```

Main event processing loop for Swindows. Handles callbacks etc.
Returns TRUE on success FALSE on failure.

```
extern swindow
*sw_open();
```

"Opens" a sound window and returns it. The gain and location of the window are set to their defaults. Returns NULL if no more windows can be allocated.

```
extern int
sw_close(swindow *sw);
```

Closes the sound window and returns it to the free list.
Returns true on success FALSE on failure

```
extern void
sw_move(swindow *sw, float r, float theta, float phi);
```

Moves the sound window to <r, theta, phi> in spherical coordinates. The origin is at the listener's head.

```
extern void
sw_move_head(float *location);
```

Moves the listener's head in Beachtron coordinates.

```
extern int
sw_give_focus(swindow *sw, float focus_gain);
```

Gives the focus to the sound window by setting the gain to focus_gain.
Returns true on success FALSE on failure

```
extern int
sw_unfocus();
```

Removes the current focus if there is one.
Returns true on success FALSE on failure

```
extern void
sw_set_gain(swindow *sw, float gain);
```

Sets the gain of the sound window to gain.

```
extern float
sw_get_gain(swindow *sw);
```

Returns the current gain of the swindow.

```
extern int
sw_default_gain();
```

Set the gain on all sound windows to the default gain.
Returns TRUE on success FALSE on failure

```
extern int
sw_play(swindow *sw, char *filename);
```

Plays the audio file pointed to by filename in the sound window.
Returns TRUE on success FALSE on failure

```
extern int
sw_halt(swindow *sw);
```

Stops playing audio on the sound window.
Returns TRUE on success FALSE on failure

```
extern int
sw_off();
```

Turns off all of the sound windows.
Returns TRUE on success FALSE on failure

```
extern int
sw_pause(swindow *sw);
```

Pauses the sound window until and sw_continue command.
Returns TRUE on success FALSE on failure

```
extern int
sw_continue(swindow *sw);
```

Continues playing a paused sound window.
Returns TRUE on success FALSE on failure

```
extern int
sw_pause_till_done(swindow *sw);
```

Turns off the sound window until the current audio segment finishes. Continues playing queued segments if there are any.
Returns TRUE on success FALSE on failure

```
extern int
sw_flush_queue(swindow *sw);
```

Flushes the audio server event queue without turning off the current segment.
Returns TRUE on success FALSE on failure

```
extern int
sw_halt_and_flush_queue(swindow *sw);
```

Stops playing audio on a sound window and flushes the audio server event queue.
Returns TRUE on success FALSE on failure

```
extern int
sw_set_queueing(swindow *sw, int on);
```

Turns the event queue on and off for swindow.
Returns TRUE on success FALSE on failure

```
extern int
sw_register_callback(char *event, void (*cb) (), char *data);
```

Registers a callback on an event with the audio server.
Returns TRUE on success FALSE on failure.

Streams

SUMMARY

The previous section discussed the Swindows software module. Swindows handles the spatial layout of windows in the three-space of a listener. The Streams module on the other hand implements the user interface for AudioStreamer. In addition to managing the spatial location of sound sources in space (via Swindows) the Streams module also manages time-varying events such as the rate of gain decay at a particular focus level.

DEPENDENCIES

The restart_streamer script must be executed at run time in order for the Streams module to be operational.

INCLUDES

The include file stream.h must be included in application programs.

CONSTANTS

The following constants set the default values for the Streams module. All of these default values can be overridden using a configuration file (see appendix C). The default values represent the preferences of the author.

The gain constants set the gains for the various focus levels. The gains can be tuned to a particular listener's hearing ability and preferences. The gains for turning a source off and for the MIDI tones are also set here. Level 0 here represents the values used by

Gain Constant	Value
LEVEL_0_LEFT_GAIN	(float)0.0
LEVEL_0_MIDDLE_GAIN	(float)0.0
LEVEL_0_RIGHT_GAIN	(float)2.5
LEVEL_1_LEFT_GAIN	(float)10.0
LEVEL_1_MIDDLE_GAIN	(float)10.0
LEVEL_1_RIGHT_GAIN	(float)12.5
LEVEL_2_MIDDLE_GAIN	(float)15.0
LEVEL_2_LEFT_GAIN	(float)15.0

LEVEL_2_RIGHT_GAIN	(float)17.5
LEVEL_3_MIDDLE_GAIN	(float)15.0
LEVEL_3_LEFT_GAIN	(float)15.0
LEVEL_3_RIGHT_GAIN	(float)17.5
LEVEL_4_MIDDLE_GAIN	(float)15.0
LEVEL_4_LEFT_GAIN	(float)15.0
LEVEL_4_RIGHT_GAIN	(float)17.5
OFF_GAIN	(float)-120.0
MIDI_GAIN	(float)-1.0

The rate constants set the default increments (in dB) by which the gain decays at a particular focus level. For example at focus level 1 the left sources gain decays by 0.04 for every clock tick. There are approximately 25 clock ticks per second.

Rate Constant	Value
LEVEL_1_LEFT_RATE	(float)0.04
LEVEL_1_MIDDLE_RATE	(float)0.04
LEVEL_1_RIGHT_RATE	(float)0.04
LEVEL_2_MIDDLE_RATE	(float)0.02
LEVEL_2_LEFT_RATE	(float)0.02
LEVEL_2_RIGHT_RATE	(float)0.02
LEVEL_3_MIDDLE_RATE	(float)0.01
LEVEL_3_LEFT_RATE	(float)0.01
LEVEL_3_RIGHT_RATE	(float)0.01
LEVEL_4_MIDDLE_RATE	(float)0.0
LEVEL_4_LEFT_RATE	(float)0.0
LEVEL_4_RIGHT_RATE	(float)0.0

The start delay constants are used to stagger the start times of the streams. The values are in ticks.

Start Constant	Value
MIDDLE_START_DELAY	25
RIGHT_START_DELAY	125
LEFT_START_DELAY	245

The TSMS constants are used to control the time-compression at focus level four. TSMS_TICKS is the time interval between compression increases. MAX_TSMS_RATIO defines the maximum time compression. TSMS_RATE defines the increment in the the time-compression ratio increases.

Time-compression Constant	Value
TSMS_TICKS	80
TSMS_RATIO	1.0

MAX_TSMS_RATIO	1.4
TSMS_RATE	0.1

STRUCTURES

The Streams module uses three primary structures: segment, thread, and stream. Each will be described below.

The segment structure holds data relevant to a particular segment playing on a stream. Segments are portions of an audio file.

```
typedef struct
        {
        char audio_file[MAX_LINE_LEN];      /* Audio file for segment    */
        int  start;                         /* Start time in milliseconds */
        int  stop;                          /* Stop time in milliseconds */
        int  duration;                      /* Duration in milliseconds  */
        } segment;
```

Sequences of segments are called threads. The thread structure is used to organize various segments (not necessarily from the same audio file) into a coherent whole.

```
typedef struct
        {
        VARRAY *segments;               /* Array of segments in this thread  */
        int   num_segments;             /* Total number of segments in thread */
        int   current_segment_index;    /* The index of the current segment   */
        } thread;
```

The stream structure is the primary organizing structure in the Streams module. It contains all of the information relevant to a particular stream at run time
.

```
typedef struct
        {
        swindow *sw;                /* Sound window for this stream              */
        thread *thread;             /* Pointer to the thread playing             */
        segment *current_segment;   /* Pointer to current segment on this stream */
        double  last_tsms_ratio;    /* Play back speed before focus shift        */
        double  tsms_ratio;         /* Play back speed                           */
        double  tsms_rate;          /* Rate of play back speed change            */
        int     tsms_ticks;         /* Number of ticks at this tsms              */
        float   delayed_gain;       /* Delayed gain on the stream                */
        float   gains[5];           /* Array of gains                            */
        float   rates[5];           /* Array of rates                            */
float  last_gain;              /* Gain before focus shift                */
int    last_gain_level;        /* Gain level before focus shift          */
int    gain_level;             /* Current gain level                     */
int    gain_delay_ticks;       /* Clock ticks to delay gain by           */
int    start_delay_ticks;      /* Clock ticks to delay start by          */
int    segment_index;          /* Index to current segment               */
int    attention;              /* Was the last tone a feedback tone      */
long   paused_pos;             /* Position of paused stream, or -1       */
long   start_pos;              /* Start position of the segment          */
long   stop_pos;               /* Current stop position                  */
time_t focus_time;             /* Last time this stream was in focus     */
struct timeb  *start_time;     /* Time the segment began playing         */
} stream;
```

GLOBAL VARIABLES

The following are the global variables used in the Streams module.

```
stream *left, *center, *right;
```

Pointers to the left, center, and right stream respectivley.

```
thread *t1, *t2, *t3;
```

Pointers to the first, second, and third thread. t1 is playing on the center stream, t2 plays on the right stream, and t3 plays on the left stream.

EXTERNAL INTERFACE

The following is a list of exported functions from the Streams module.

```
extern void
s_initialize(char *filename1, char *filename2, char *filename3,
        char *config_filename, int output, int head, int gaze,
        int fish, int fb);
```

Initialization functions for the Streams system. Audio files and associated segment files (see appendix C) are pointed to by filename1, filename2, filename3. Config_filename points to the configuration file if there is one. Finally output, head, gaze, fish, and fb are Boolean values that turn on/off output to the screen, the smart chair interface, the fish interface, and MIDI feedback respectively.

```
extern void
s_kill();
```

Stops all streams, reclaims storage, cleans up, and exists the Streams system.

```
extern void
s_give_focus(stream *s);
```

Bumps the focus level on the stream.

```
extern void
s_unfocus();
```

Removes the focus if there is one.

```
extern void
s_next_segment(stream *s, int direction);
```

Advances to the next segment queued to play on the stream or reverses to he previous segment depending on the value of direction.

```
extern void
s_rotate();
```

Rotates the sound sources in a clockwise direction: Left->Center->Right.

B. Scripts and Utilities

The following is a listing of scripts and interfaces used to execute and manage AudioStreamer's various components. The scripts are primarily written in Perl. Exceptions are noted in the appropriate functional description.

MAIN ROUTINES

restart_streamer
arguments: none

function: Convenience routine that restarts AudioStreamer. Existing audio servers and stereoclients are killed and restarted. Calls kill_sources and start_sources.

start_sources
arguments: none

function: Starts four audio servers and two stereoclients for AudioStreamer.

kill_sources
arguments: none

function: Kills existing audio servers and stereoclients and does general cleanup.

restart_r_server
arguments: none

function: Convenience routine that kills the recognition server and restarts it.

start_r_server
arguments: none

function: Starts the speech recognition server and sets AudioStreamer up for recognition.

kill_r_server
arguments: none

function: Kills the recognition server and does general cleanup.

listentest

arguments: -c Following string names a configuration file
-o Output gain and focus data to screen
-h Fix sources in space using Polhemus
-g Turn on head tracking (gaze) interface
-f Turn on the fish (smart chair) interface
-s Turn on the speech interface
-t Turn on tone feedback for focus levels

function: Starts AudioStreamer in demo mode. Calls *listen* and passes the arguements to it. If no arguments are passed to listentest AudioStreamer starts up with the keyboard interface (which is always available).

listen

arguments right segment file (string)
center segment file (string)
left segment file (string)
configuration file name (string)
output, true/false (string)
fix sources, true/false (string)
head tracking, true/false (string)
smart chair, true/false (string)
speech, true/false (string)
tone feedback, true/false (string)

function: C routine that starts AudioStreamer. Assumes that restart_streamer has been executed. Sets up AudioStreamer with the appropriate interface. Connects to stereoclients and starts playing audio. Manages keyboard interface.

start_server

arguments: -p port
-d device

function: Convenience routine that starts an audio server using the given port and device.

start_stereo

arguments: -l Named pipe for the left channel
-r Named pipe for the right channel
-d Name of device.

function: Convenience routine that calls steroclient (wriiten by Jordan Slott) and passes the arguments.

ADDITIONAL APPLICATIONS

stream_directory
arguments: Name of directory

function: Browse a directory of sound file using AudioStreamer. The head tracking interface is used by default. Assumes that directory_to_streamer has been run to construct an appropriate segment file.

vmailtest
arguments: Same as listentest

function: Demo interface to voice mail browser. Assumes that directory_to_streamer has been run on the voice mail directory. Optional configuration file vmail_config can be passed in as the -c argument.

UTILITIES

adjust_directory
arguments: Name of directory

function: Adjusts the sound level on a directory of sound files using adjust_sound_level (written by Chris schmandt).

directory_to_streamer
arguments: Name of directory

function: Constructs appropriate AudioStreamer segment files for the directory. Also calls adjust_directory to normalize the sound levels of the directories audio files.

index_to_streamer
arguments: Name of speaker index file.

function: Converts the output of the speaker indexing algorithm to an AudioStreamer segment file.

sndnote_to_streamer
arguments: Name of sndnote file

function: Converts the output of the closed-caption synchronization algorithm to and AudioStreamer segment file.

speaker_index

arguments: Program (ABC, BBC, CBS, etc)
Directory

function: Convenience routine that transfers the audio file and associated speaker index file and converts the index file into an AudioStreamer segment file (using index_to_streamer).

CLOSED CAPTION PROCESSING

All of the following routines were written in C by Chris Horner and Alan Blount of the MIT Media Laboratory with the exception of the transfer_news script. Some of the routines were converted to run on the Sun SPARCStation by the author. For a more detailed description see Horner 1993.

capcon

arguments: filename
seconds

function: Convenience routine that calls getcaption and recordtofile to capture an audio file and its closed-captions. The raw closed-captions are further passed through a series of filters to remove control information.

getcaption

arguments: seconds

function: Gets captions from a serial port connected to a video line 21 decoder. Runs for the requested number of seconds. Speaker and story boundaries are time-stamped as they are encountered. The output is a file of raw captions and associated time-stamps.

recordtofile

arguments: filename
seconds

function: Records audio to a file for the requested number of seconds. Used in conjunction with getcaption to capture the audio and closed captions for a new broadcast.

cap2db

arguments: none

function: A lex parser that converts a filtered closed-caption file to and auxfile format.

paws

arguments: sound file name
silence level
silence length

function: Performs speech and silence detection for the sound file using the requested parameters. Results are written to an aux file.

syncs

arguments: sound file name

function: Synchronizes closed-caption time-stamps and significant silences to more accurately reflect actual speaker and story boundaries.

transfer_news

arguments: Program name

function: Transfers any new news that exist for the program. Processes the Run speech/silence detection and the synchronization algorithm. The output is converted to an AudioStreamer segment file.

C. File Formats

AudioStreamer make use of two types of files configuration files and segment files. Configuration files are used to override AudioStreamer's default settings. Segment files pass information regarding potential points of interest to AudioStreamer so that the system can cue the listener. Below each format is outlined in turn.

CONFIGURATION FILES

All of the defaults values for the parameters of the Streams module (see STREAMS section of appendix A) can be tailored to a specific listener's tastes using a configuration file. At run time the configuration file can be passed to AudioStreamer (using listen or listentest). A configuration file consists of a series of entries of the following form:

<white space> name of constant: <white space> default value <CR>

The name of the constant to be changed comes first followed by a colon. The comes the new value followed by a carriage return. White space is ingnored. Lines in the configuration file can be commented out by preceding them with a sharp sign ('#'). Constant names are normalized to lower case at read time so case errors are not an issue.

During the development of AudioStreamer several configuration file were created for different users of the system. In addition a configuration file tailored to listening to voice mail was developed. That file (vmail_config) is included below as an example of how to properly write a configuration file.

```
##      vmail_config: created May 1994 by Atty Mullins
##      Configuration file that tunes streamer for vmail listening.
##      In other words it is tuned for short messages.

        level_0_left_gain:          0.0
        level_0_middle_gain:        -1.0
        level_0_right_gain:         2.5
        level_1_left_gain:          10.0
        level_1_middle_gain:        08.0
        level_1_right_gain:         14.0
        level_2_middle_gain:        13.0
        level_2_left_gain:          15.0
        level_2_right_gain:         20.0
        level_3_middle_gain:        13.0
        level_3_left_gain:          15.0
        level_3_right_gain:         20.0
        level_4_middle_gain:        13.0
```

```
level_4_left_gain:        15.0
level_4_right_gain:       20.0
level_1_left_rate:        0.40
level_1_middle_rate:      0.40
level_1_right_rate:       0.40
level_2_middle_rate:      0.30
level_2_left_rate:        0.30
level_2_right_rate:       0.30
level_3_middle_rate:      0.0
level_3_left_rate:        0.0
level_3_right_rate:       0.0
level_4_middle_rate:      0.0
level_4_left_rate:        0.0
level_4_right_rate:       0.0
middle_start_delay:       5
right_start_delay:        25
left_start_delay:         45
tone_time_interval:       3
tsms_ticks:               20
tsms_ratio:               1.0
max_tsms_ratio:           1.4
tsms_rate:                0.1
```

SEGMENT FILES

Segment files contain information on which audio files to play on a particular stream. In addition they contain optional pointers to interesting segments within a particular audio file. Segment files have the following format:

```
<audio file name 1>
optional segment pointers
<audio file name 2>
optional segment pointers
......
```

Names of audio files are delimited by angle brackets. Following each audio file entry there is an optional list of segment pointers (if they are omitted the *entire* audio file is treated as a single segment). The segment pointers consist of a starting point and ending point for the segment in milliseconds. For example, the following entry would play three segments from the audio file *example.snd.*

```
<example.snd>
1000   50000
80000 100000
55000 90000
```

At run time AudioStreamer will play these three segments in the order they are listed on the requested stream (left, center, or right).

Several utilities build segment files automatically (as an example see sndnote_to_streamer in appendix B). Segment files can also be constructed by hand.

References

Ackerman 1990 Ackerman, D. *A Natural History of the Senses.* Vintage, 1990.

Arons 1991 Arons, B. HyperSpeech: Navigating in speech-only hypermedia. In *Proceedings of HyperText.* New York: ACM, 1991.

Arons 1992a Arons, B. Techniques, perception, and applications of time-compressed speech. In *Proceedings of 1992 Conference, American Voice I/O Society*, pp. 169-177, 1992.

Arons 1992b Arons, B. A review of the cocktail party effect. *Journal of the American Voice I/O Society.* Volume 12, 1992.

Arons 1992c Arons, B.A. Tools for building asynchronous servers to support speech and audio applications. In *Proceedings of the ACM symposium on User Interface Software and Technology.* New York: ACM 1992.

Arons 1993 Arons, B. SpeechSkimmer: interactively skimming recorded speech. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST).* ACM SIGGRAPH and ACM SIGCHI. ACM, 1993.

Arons 1994 Arons, B. Interactively Skimming Recorded Speech. Ph.D. dissertation, MIT, February 1994.

Blauert 1983 Blauert, J. *Spatial Hearing: The Psychophysics of Human Sound Localization.* MIT Press, 1983.

Bregman 1990 Bregman, A.S. *Auditory Scene Analysis: The Perceptual Organization of Sound.* MIT Press, 1990.

Broadbent 1958 Broadbent, D.E. *Perception and Communication.* Pergamon Press, 1958.

Brokx 1982 Brokx, J.P.L. and Nooteboom, S.G. Intonation and perceptual separation of simultaneous voices. *Journal of Phonetics.* Volume 10, pp. 23-26, 1982.

Buser 1991 Buser, P. and Imbert, M. *Audition.* Translated by R.H. Kay. MIT Press, 1991.

Cage 1961 — Cage, J. *Silence.* Wesleyan University Press, 1961.

Chalfonte 1991 — Chalfonte, B.L., R.S. Fish, and R.E. Kraut. Expressive richness: a comparison of speech and text as media for revision. In *Proceedings of the Conference on Computer Human Interaction*, pp. 21-26. ACM, April 1991.

Cherry 1953 — Cherry, E.C. Some Experiments on the recognition of speech, with one and two ears. *Journal of the Acoustic Society of America.* Volume 25, pp. 975-979, 1953.

Cherry 1954 — Cherry, E.C., Taylor, W.K. Some further experiments on the recognition of speech, with one and two ears. *Journal of the Acoustic Society of America.* Volume 26, pp. 554-559, 1954.

Cohen 1991 — Cohen, M. and Ludwig, L.F. Multidimensional audio window management. *International Journal of Man-Machine Studies.* Volume 34, pp. 319-336, 1991.

CRE 1993 — The Beachtron ™: Three Dimensional Audio for PC-compatibles. Crystal River Engineering Inc. 1993.

Divenyi 1989 — Divenyi, D.L., and Oliver, S.K. Resolution of steady-state sounds in simulated auditory space. *Journal of the Acoustical Society of America.* Volume 85, pp. 2042-2052, 1989.

Egan 1954 — Egan, J.P., Carterette, E.C. and Thwing, E.J. Some factors affecting multi-channel listening. *Journal of the Acoustic Society of America.* Volume 26, pp. 774-782, 1954.

Fisher 1988 — Fisher, S.S., Wenzel, E.M., Coler, C., and McGreevy, M.W. Virtual interface environment workstations. In *Proceedings of the Human Factors Society.* Volume 32, pp. 91-95, 1988.

Foster 1988 — Foster, S.H. *Convolvotron™ User's Manual.* Crystal River Engineering. 1988

Foster 1991 — Foster, S.H., Wenzel, E.M., and Taylor, R.M. Real time sythesis of complex acoustic environments. In *Proceedings of the ASSP (IEEE) Workshop on Applications of Signal Processing to Audio and Acoustics.* 1991.

Gerber 1974 — Gerber, S.E. Limits of speech time compression. In *Time-Compressed Speech.* Edited by S. Duker. Scarecrow, 1974.

Hawley 1993 Hawley, M. Structure out of Sound. Ph.D. dissertation, MIT, Sep. 1993.

Hindus 1993 Hindus, D., Schmandt, C., and Horner, C. Capturing, structuring, and representing ubiquitous audio. *ACM Transactions on Information Systems.* Volume 11, No. 4, pp. 376-400. October 1993.

Horner 1993 Horner, C. NewsTime: A Graphical User Interface to Audio News. Masters Thesis, MIT Media Lab, June1993.

Johnson 1988 Johnson, S. *The History of Rasselas, Prince of Abissinia.* Oxford University Press, 1988.

Kahn 1992 Kahn, D, and Whitehead, G. *Wireless Imagination: Sound, Radio, and the Avant-Garde.* MIT Press, 1992.

Kendall 1990 Kendall, G.S. Visualization by ear: Auditory imagery for scientific visualization and virtual reality. In *Dream Machines for Computer Music Production.* Edited by A. Wolman and M. Cohen. School of Music, Northwester University, 1990.

Koizumi 1992 Koizumi, N., Cohen, M., and Aoki, S. Design of virtual conferencing environments in audio telecommunication. In *Proceedings of the 92nd Annual Audio Engineering Society Convention.* 1992.

Lawson 1966 Lawson, E. Decisions concerning the rejected channel. *Quarterly Journal of Experimental Psychology.* Volume 18, pp. 260-265, 1966.

Lentz 1980 Lentz, J., Sillman, D., Thedick, H., and Wetmore, E. Television captioning for the deaf: signal and display specifications. *Report no. E-7709-C, Engineering and Technical Operations Department,* Public Broadcasting Service, Washington DC, 1980.

Ly 1993 Ly, E., Schmandt, C., and Arons, B. Speech recognition architectures for multimedia environments. In *Proceedings of 1993 AVIOS Conference.* San Jose, 1993.

Malone 1987 Malone, T.W., Grant, K.R., Turbak, F.A., Brobst, S.A., and Cohen, M.D. Intelligent information sharing systems. *Communications of the ACM.* Volume 30, pp. 390-402, 1987.

Masterson 1969 Masterson, R.B., Heffner, H.E., and Ravizza, R.J. The evolution of human hearing. *Journal of the Acousitcal Society of America.* Volume 45, pp. 966-985, 1969.

Moray 1959 Moray, N. Attention in dichotic listening: Affective cues and the influence of instructions. *Quarterly Journal of Experimental Psychology.* Volume 11, pp. 56-60, 1959.

Moray 1970 Moray, N. *Attention: Selective Processes in Vision and Hearing.* Academic Press, 1970.

Norman 1969 Norman, D. Memory while shadowing. *Quarterly Journal of Experimental Psychology.* Volume 21, pp. 85-93, 1969.

Norman 1976 Norman, D. *Memory and Attention.* John Wiley and Sons, 1976.

Orr 1965 Orr, D.B., Friedman, H.L., and Williams, J.C. Trainability of listening comprehension of speeded discourse. *Journal of Educational Psychology.* Volume 56, pp. 148-156. 1965.

Polhemus 1992 Polhemus Inc. *3Space Isotrak User's Manual.* Polhemus Inc., June 1992.

Rayleigh 1907 Lord Rayleigh. On our perception of sound direction. *Philosophical Magazine.* Volume 13, pp. 214-232, 1907.

Roy 1995 Roy, D.K. NewsComm: A Hand-Held Device for Interactive Access to Structured Audio. Masters Thesis, MIT Media Lab, June 1995.

Rhodes 1986 Rhodes, G. Auditory attention and the representation of spatial information. *Perception and Psychophysics.* Volume 42, pp. 1-14, 1982.

Roucus 1985 Roucus, S., and Wilgus, A.M. High quality time-scale modification for speech. In *Proceedings of the Internationsal Conference on Acoustics, Speech, and Signal Processing,* IEEE, pp. 493-496, 1985.

Roy 1995 Roy, D. NewsComm: A Hand-Held Device for Interactive Access to Structured Audio. Master's Thesis, MIT Media Lab. May, 1995.

Salton 1989 Salton, G. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison Wesley Publishing, 1989.

Schmandt 1994 Schmandt, C. *Voice Communication with Computers.* Van Nostrand Rheinhold, 1994.

Schmandt 1993a Schmandt, C. Phoneshell: the telephone as computer terminal. In *Proceedings ACM Multimedia* 93, pp. 373-382. New York: ACM, 1993.

Schmandt 1993b Schmandt, C. From desktop audio to mobile access: opportunities for voice in computing. In *Advances in Human-Computer Interaction.* Edited by H.R. Hartson and D. Hix. Ablex Publishing Corporation, 1993.

Shepard 1981 Shepard, R.N. Psychophysical complementarity. In *Perceptual Organization.* Edited by M. Kubovy and J.R. Pomerantz. Erlbaum, 1981.

Sheth 1994 Sheth, B. A Learning Approach to Personalized Information Filtering. Master's Thesis, Department of Electrical Engineering and Computer Science, MIT, February1994.

Spieth 1954 Spieth, W., Curtis, J.F., Webster, J.C. Responding to one of two simultaneous messages. *Journal of the Acoustic Society of America.* Volume 26, pp. 391-396, 1954.

Stifelman 1993 Stifelman, L. J., Arons, B., Schmandt, C., and Hulteen, E.A. VoiceNotes: A speech interface for a hand-held voice notetaker. In *Proceedings of INTERCHI.* ACM: New York: 1993.

Stifelman 1994 Stifelman, L.J. The cocktail party effect in auditory interfaces: a study of simultaneous presentation. MIT Media Laboratory Technical Report, September 1994.

TBS 1992 Turtle Beach Systems. *MultiSound User's Guide.* Version 1.2, Turtle Beach Systems, 1992.

Thurlow 1967 Thurlow, W.R., and Runge, P.S. Effects of induced head movements on localization of direction of sound sources. *Journal of the Acoustical Society of America.* Volume 42, pp. 480-488, 1967.

Tufte 1990 Tufte, E. *Envisioning Information.* Chesire, CT: Graphics Press, 1990.

Treisman 1967 Treisman, A.M. and Geffen, G. Selective attention: Perception or response ? *Quarterly Journal of Experimental Psychology.* Volume 19, pp. 1-17, 1967.

Webster 1954 Webster, J.C., Thompson, P.O. Responding to both of two overlapping messages. *Journal of the Acoustic Society of America.* Volume 26, pp. 396-402, 1954.

Webster 1993 Merriam-Webster. *Merriam-Webster's Colligiate Dictionary* - 10 Edition. Springfield, MA: Merriam-Webster Incorporated, 1993.

Wenzel 1990 Wenzel, E.M., Stone, P.K., Fisher, S.S., and Foster, S.H. A system for three-dimensional acoustic "visualization" in a virtual environment workstation. In *Proceedings of the IEEE Visualization '90 Conference.* pp. 329-337, 1990.

Wenzel 1992 Wenzel, E.M. Localization in virtual acoustic displays.*Presence.* Volume 1, pp. 80-107, 1992.

Wheatley 1992 Wheatley, B., Tadlock, J., and Hemphill, C. Automatic efficiency improvements for telecommunications application grammars. *First IEEE workshop on Interactive Voice Technology for Telecommunications Applications*, 1992.

Zimmerman 1995 Zimmerman, T., Smith, J.R., Paradiso, J.A., Allport, D., and Gershenfeld, N. Applying electric field sensing to human-computer interfaces. In *Proceedings of the Conference on Computer Human Interaction.* ACM, 1995.

4538-40