INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Towards the Sonification of the World Wide Web: SprocketPlug

Elijah Breder

Faculty of Music
McGill University
Montreal, Canada

March 1997

A thesis submitted to the Faculty of
Graduate Studies and Research
in partial fulfilment of the degree of
Master of Arts

Abstract

The goal of the thesis presented herein is to provide an overview of current issues in auditory display design and to suggest how these issues may be applied to the development of applications for the World Wide Web (WWW). The software developed as part of this thesis, the SprocketPlug plug-in for Netscape Navigator, provides a tool for exploring various auditory display techniques at three levels of WWW development: HTML, Javascript, and Java.

The strength of SprocketPlug is that it enables WWW developers to incorporate interactive spatialized sound as an integrated component of WWW documents and applications. The implementation of SprocketPlug is based on currently available technology: the Netscape plug-in architecture, Netscape LiveConnect, and the Apple SoundSprocket.

Résumé

Le but fondamental de la thèse suivante est de fournir une sommaire de la recherche en interface audio et de suggérer comment appuyer ces principes aux development des applications pour le World Wide Web (WWW). Le logiciel développer comme partie de cette thèse, SprocketPlug, permer d'explorer plusieurs concpets des interfaces audio au trois niveaux de programmation pur le WWW: HTML, Javascript, et Java.

L'avantage de SprocketPlug est qu'il permet aux programmeurs d'incorporer le son interactif en 3D comme une partie intégrante des applications et documents du WWW. SprocketPlug est baser sur des technologies disponible aujourd'hui: l'architecture plug-in de Netscape, Netscape LiveConnect, et Apple SoundSprocket.

Acknowledgements

I would like to thank my thesis supervisor Bruce Pennycook, who has provided continuous support throughout the course of my thesis research. Besides being an important source of information, his creative outlook on current technologies helped shape my own ideas. I would also like to thank my fellow colleagues Mark Ballora, David Mcintyre, Elbert McLaughlin, and Kyle Dawkins. Their input was most valuable not only because they were fellow students, but because they were friends who were always open for discussion.

TABLE OF CONTENTS

1. INTRODUCTION

The thesis presented herein will explore the current state of auditory display research and discuss how this body of knowledge may be applied to the World Wide Web (WWW). Important issues regarding auditory perception, auditory display design, and the current state of sound tools for the WWW will be discussed. It will be shown that although these tools are rapidly achieving high levels of sophistication, their main purpose is for sound delivery and do not meet the demands of an integrated auditory display component.

The software developed as part of this thesis, the SprocketPlug plug-in, provides WWW developers with interactive spatialized sound services which can be integrated at three important levels of WWW programming: HTML, Javascript, and Java. Rather than being an outside helper application or stand alone sound player (although it can be used as such), SprocketPlug is designed to be an integrated part of WWW development. After presenting the functionality of the thesis software, demonstrations will be discussed and assessed.

2. A SOUND BASIS

Until recently, sound has remained an underutilized component in human-machine/computer interactions. Although we use our remarkable listening abilities in our day to day lives, the use of audio in human-computer interfaces has been relegated to a few system beeps or "cute" sound effects. Sound has always taken a back seat to graphics development and remains an afterthought in many system designs. By discussing various aspects of sound and our auditory system, this chapter will describe how computer interfaces and user interactions may incorporate sound as a functional element of the computer system. The chapter will conclude with some possible explanations of why the use and development of audio has lagged behind graphic development at the computer interface.

2.1 Important Attributes Of Sound

Sound is a powerful communicative medium. It describes the physical world around us, informs us of ongoing events, and alerts us to take action. Very often all this information is presented to us at once, yet we are still able to decode and understand this conglomerate of acoustic information.

Imagine the following: You're driving your car with a passenger next to you. The radio is on at low volume, and you are engaged in conversation with the passenger. The radio catches your attention because the weather forecast is on. While you're listening to the weather report you hear a strange sound come from the car's engine. You decide to pull over and see if there is anything wrong. You pull to the side of the road and are about to get out of the car when you hear an ambulance siren behind you. You decide to wait for the ambulance to pass to get out of the car. You open the hood and hear some hissing. Being an expert mechanic, you have an idea of what could cause such a sound. You are able to locate the source of the sound and fix the problem. You get back in the car and continue on to your destination.

This scenario points out some very important aspects of sound and our listening abilities. Sound is omnidirectional and transparent. It describes the physics of events in our environment. We are able to monitor a number of simultaneous sound sources and shift our auditory attention to any source at will, all while performing other activities. We are also able to interpret the meaning of various sounds and take action based on these understandings.

The omnidirectional nature of sound allows us to hear what we can not see (Kramer 1994b, 4). While vision has a rather narrow area of focus and requires us to face the object to perceive it, we are able to hear sound all around us. This gives sound the ability to draw our attention to events outside our field of view.

Sound is also transparent: auditory objects do not occlude one another. Bregman (1990) describes this situation as follows:

> The auditory world is like the visual world would be if all objects were very, very, very transparent and glowed in sputters and starts by their own light, as well as reflecting the light of their neighbors (Bregman 1990, 37).

This attribute of sound allows hidden objects to be heard even though they are not seen.

Perhaps the most important aspect of sound is its ability to convey information about events in the environment. The following example describing a computer desktop from Gaver (1986) illustrates:

> The file hits the mailbox, causing it to emit a characteristic sound. Because it is a large message, it makes a rather weighty sound. The crackle of paper indicates a text file - if it had been a compile program, it would have clanged like metal. The sound comes from the left and is muffled. The mailbox must be in the window behind the one that is currently opened on the left side of the screen. And the echoes sound like a large empty room, so the load on the system must be fairly low. All this information from one sound!

As Gaver (1989) states, "sound is produced by the interaction of materials at a location in an environment". This points out the multidimensional nature of auditory information. Not only can a given sound inform us of the probable source, but it can also point out salient features of the source, the

materials involved, and the environment in which the event occurred.

2.2 The Capabilities of Our Auditory System

The human auditory system has evolved to take advantage of the various aspects of sound described above in order to help us make sense of the world around us. While the system is not responsive to all existing stimuli (the human audio range is approximately 20 Hz to 20 kHz), it has in essence tuned itself to the most frequent and ecological valid auditory characteristics of the environment (Bregman 1990, 13). Important aspects of the human auditory system include pressure and frequency sensitivity, temporal acuity, pattern recognition, localization abilities, and attentional capacities.

Our auditory system has developed finely-tuned responses to a wide range of stimuli. The first of these is pressure sensitivity. Although levels above 100dB are undesirable , the most intense sound we can hear without damaging our ears has a level of about 120dB above the faintest sound we can detect (Moore 1982, 47). This corresponds to an intensity ratio of 1 000 000 000 000:1 (a pressure amplitude of approximately 2×10^{-5} N/m^2). What is remarkable here is that while we are responsive to a wide dynamic range, the amplitudes of soundwaves are extremely small fluctuations in atmospheric pressure (where atmospheric pressure corresponds to 10^5 N/m^2). The minimum pressure fluctuation to which we are sensitive to corresponds to less than a change of one billionth of atmospheric pressure while the threshold of pain is still less than one one-thousandth of atmospheric pressure (Rossing 1990, 85).

For frequencies lying between 20Hz and 20kHz, the auditory system has a finely tuned frequency response as well. The critical band theory is commonly used to explain frequency resolution by describing the peripheral auditory system as a bank of bandpass filters, with continuously overlapping center frequencies (Moore 1982, 85). Begault (1994) describes this system as follows:

> A complex sound is analyzed by the ear with a bank of 24 filters, each tuned with a successive center frequency and bandwidth so as to cover to the entire audio range (just like a graphic equalizer). The size of the critical band approximates a 1/3 octave bandwidth; harmonics falling within a critical bandwidth will be integrated in such a way that the strongest harmonic will mask other harmonics within the same band, more so than if these other harmonics were in other bands.

Begault goes on to point out how this system is made more complex by the variable nature of these bandwidths. The width and placement of each band depend on various factors such as pressure level and spatial orientation of the listener. In addition, frequency sensitivity varies across the audible range. It must be noted that most humans do not respond to the entire audio range and frequency sensitivity deteriorates with age. This deterioration mostly affects the transients and high partial of sounds.

The auditory system is also capable of great temporal acuity, which is especially important for sound localization. Studies have shown that for frequencies below 1.5kHz, the auditory system localizes sound sources by detecting the on-set time difference (or phase difference) between the two ears of an incoming waveform. This corresponds to interaural time differences (ITD) in the range of 0.005 to 1.5 milliseconds (Begault 1994, 44). Similarly, the precedence effect describes how ITD values are used to discriminate a sound source from its echoes. If the echoes occur within 35 milliseconds of the direct sound, the direction of the sound is associated with the wavefront which first arrived at the ear - the direct sound. Echoes within this time frame tend to reinforce the direct sound (Rossing 1990, 462). In other words, they do not distract us from localizing the direct sound. We begin hearing independent onsets at around 20 milliseconds and for continuously repeated sounds, we may perceive rhythmic regularities and even pitch. Pitch will be perceived if onset times are in the broad range of .05 milliseconds, producing 20kHz tone, to 50 milliseconds resulting in a 20Hz tone (Kramer 1994b, 5).

The auditory system's ability to establish the direction of a given sound source complements the omnidirectional nature of sound. By examining the time, intensity, and timbral differences between the two ears, we are able to

determine the direction of a particular sound and hence, the position of the source. While vision focuses in a very specific direction, audition can monitor sonic objects and events from all around the listener. Very often our ears act like pointers for our eyes - they tell the eyes where to look.

Our ability to learn and recognize familiar sounds is facilitated by our auditory pattern recognition skills and auditory memory. We are able to distinguish and categorize a great number of sounds and pick out a familiar voice in a crowd. Often our survival depends on this ability: for example, hearing an approaching vehicle or the cracking branches of a prowling predator. We also use pattern matching skills when we attempt to classify unfamiliar sounds by comparing them to known sounds. This happens when someone says "it sounds like" when describing the unfamiliar sound.

The attentional capacities of the auditory system allow us to address the issue of sound's transparency and its omnidirectional nature. We are able to focus on some auditory objects and leave others in the background. This allows us to cope with our lack of "ear-lids" (Kramer 1994, 13) and to tune into important events. Related to the auditory system's attentional capacities are its monitoring abilities. Although we can focus our auditory attention on a particular object, we can simultaneously monitor a number of background events. Changes in these background events are detected by the auditory system, informing us that we might want to shift our attention to that particular event. A well known example of this is the "cocktail party effect" where a person engaged in one conversation can tune into another conversation upon hearing their own name (Cherry 1953, and Eysenck and Keane 1990).

Our auditory monitoring abilities also give rise to auditory gestalt formations. These describe our perceptual system's ability to hear a complex sound field as whole, without necessarily directing attention to its component parts (Kramer 1994, 8). For example, when listening to a choir, we are able to hear it as a single entity even though it is made up of many individual voices often singing in mutli-part harmony.

Auditory gestalts may be grouped into synthetic and analytic listening, first described by Helmholtz (Helmholtz 1859, adapted from Bregman 1990,

314-315). Synthetic listening interprets auditory percepts as generally as possible. For example, hearing the murmur of a crowded room rather than a large number of individual voices. Analytic listening takes place when we try to identify individual components of an auditory scene. An example is when we try to listen to a specific instrument in an orchestra.

2.3 Why Use Sound?

The simplest answer to the question of why sound should be used is because it forms an integral part of normal human experience (Gaver 1989). Many diverse professions even rely on sound a great deal: mechanics listen to automobile engines and doctors listen to heartbeats (Gaver 1989). People also listen to the states of various machinery they use. For example, drivers listen to their cars' engines, and computer users listen to their modems connecting and hard drives whirring to confirm read and write operations.

Furthermore, sound production is prevalent in today's computer technology and can be found in even the simplest home computer systems. In the past few years, computer audio components have greatly improved, going from simple low fidelity system beeps to full bandwidth, CD quality audio. The entertainment industry has embraced this technology and relies on it to deliver top-selling products. Video games are approaching Hollywood style productions in both the visual and audio aspects. Yet these resources remain untapped by the more general and "serious" computing systems in areas such as business and engineering.

The issue of noise and sound pollution is an important one. Great measures are taken to make the workplace as noise free as possible. Wouldn't all these computers making sounds create distractions and a chaotic atmosphere? Wouldn't the office turn into a kind of noisy video game arcade? Indeed it would, unless an organized and systematic approach to auditory display design is adopted. By studying the task at hand and designing an appropriate acoustic environment, we stand to benefit from auditory displays by exercising greater control over the sounds around us (Buxton

1989).

A great deal of auditory display research is devoted to the use of non-speech sounds. Another common question asked of auditory display researchers is: Why not just use speech to convey messages and information? Although speech may be appropriate in certain situations, the disadvantages seem to outweigh the advantages. First, speech requires more processing on the part of the listener than familiar sounds to be understood. For example, your response to a fire alarm is probably quicker than to some one shouting "Everybody out, there's a fire!". Similarly (to use the automobile example again), the mechanic learns a great deal more about the state of an engine by listening to it, rather than listening to someone explain the problem. Speech messages also add to the noise pollution problem by interfering and competing with real world speech.

Sound is also important because it compliments visual information. Practically all visual events and actions in the real world are accompanied by sonic events. Not only do we see an airplane fly overhead, but we hear it as well. A collision between objects is affirmed by the sound of the two objects hitting each other. This is part of the natural world (being the result of the physics of events) which we expect and depend on. Human-computer interactions can incorporate these types of sonic assurances by accompanying user actions and system events with appropriate sounds. Sounds can not only heighten the sense of realism and engagement, but may increase user enthusiasm and the perceived quality of a given system (Kramer 1994, 10).

2.4 The Slow Adoption Of Auditory Displays

In spite of the powerful communicative aspects of sound, the development and use of audio in computing environments has always been dominated by graphic developments. This section will offer several explanations why this has been the case.

As vision is the principle means of acquiring information in our society, the development of computer system visual display was of high

priority. CRT display technology and graphic user interfaces have taken great strides in research and development. Although these developments have parallels in the audio industry, it is only recently that sophisticated audio features have been incorporated into computer systems. While the computer entertainment industry has welcomed and embraced these technologies, general computing systems have practically ignored them, using the expanded audio capabilities of the hardware merely for improved system beeps. There is however, a growing interest in more sophisticated uses of audio by the general computing public, largely due to the increased availability and accessibility of high-fidelity audio computer hardware and software.

The lack of extensive scientific auditory research also accounts for audio lagging behind graphic development. As Bregman (1990) points out, up until the nineteen-sixties, vision had received almost all the attention. Studies on vision would have included treatments on lower-level psychophysical and physiological aspects, as well as higher-level perceptual and cognitive processes. In contrast, the few studies on audition focused only on basic physical properties of sound and perhaps some psychophysical aspects of our auditory system.

Bregman goes onto to offer some possible explanations for the lack of auditory research. One explanation suggests that "the fathers of Gestalt psychology, who opened up the whole question of perceptual organization, had focused on vision and never quite got around to audition" (Bregman 1990, 2). Another explanation has to do with the visual arts: "the desire for accurate portrayal led to an understanding of the cues for distance and certain facts about projective geometry." (Bregman 1990, 2). Although one might argue that music could have provided these same opportunities, music (and sound in general) is inherently abstract and more difficult to understand. There also wasn't the need to reproduce objects from the environment as there was in drawing, painting, and other visual arts.

Begault (1994) offers some explanations as well: First, audio is expendable because it isn't necessary for the successful operation of most computer systems. Next, the audio that is incorporated tends to be added as an

afterthought of system design. Consequently, the sounds are of low quality and quickly become annoying and even jarring (e.g. talking cars and piercing microwave oven timers). Lastly, audio manipulation and high fidelity audio equipment have remained the domain of musicians and electronic composers. As stated above however, the proliferation of high quality consumer oriented audio products has given the general public and research community exciting tools for auditory exploration.

3. AUDITORY DISPLAY DESIGN

The addition of sound to computer interfaces is not simply a matter of arbitrarily assigning sounds to actions and events. The design of auditory displays requires careful investigation into how sound can function as an integral and informative component of a given system. This chapter will discuss current approaches to a structured framework of auditory display design.

3.1 Preliminary Considerations

3.1.1 Assessment of Audio Related Problems

Sounds are inherently transient and lack persistence because they can only exist in time (Gaver 1989). Although sounds are suitable for the display of changing events, they are only available for limited amounts of time. As Kramer (1992, 13) states, "simultaneous comparisons and reminiscences [of auditory events] are problematical... [and] may produce cacophonous and incomprehensible results." In contrast, most visual objects are static and exist over time, allowing us to sample and examine them for as long as required. In particular, the auditory presentation of spatial extent and volumetric data is problematic (Kramer 1992, 13). The absolute and relative sizes of an auditory representation of objects are difficult to represent in sound.

Kramer (1994b) discusses several well known properties of sound in terms of auditory displays. These include the lack of absolute values of auditory variables, lack of orthogonality, and other factors such as user limitations. Although we are responsive to minute changes in various sound attributes such as amplitude, pitch, and timbre, we can not determine the absolute values these changes represent. Kramer compares this with XY graphs, where points and values are easily obtained visually. Although people with perfect pitch may have an advantage at determining absolute pitch values, similar abilities in amplitude and timbre detection are not known.

Kramer (1994b) also discusses how the lack of orthogonality of auditory parameters contribute to a lack of precision. When mapping data variables to the physical attributes of sound a change in one variable may be perceived as affecting a second variable even though this change is not found in the raw data.

User limitations present problems as well. Kramer (1994b) suggests that individual differences concerning auditory display interpretability may be no more problematic than similar differences concerning vision (e.g. color blind versus tone deaf). However, listener shortcomings regarding complex sonic environments are poorly understood.

3.1.2 Auditory Scene Analysis

Auditory scene analysis is concerned with the perception of complex sonic environments. Bregman (1990, 641) describes the central issue of auditory scene analysis as follows:

> "Although we need to build to separate mental descriptions of the different sound-producing events in our environment, the pattern of acoustic energy that is received by our ears is a mixture of the effects of the different events."

Auditory scene analysis examines how our auditory system extracts elements of this acoustic mixture and "groups them so that each group has been derived from the same environmental event" (Bregman 1990, 641). Essentially, it studies the sequential and spectral integration of spectral elements which give rise to the perceptual phenomenon of auditory streams.

An auditory stream is a perceptual unit which groups related environmental happenings. Bregman (1990) explains the difference between a sound and an auditory stream as follows: While a sound is a distinct event, a stream may incorporate more than one sound in its description of an event. He gives the example of a music performance where the sounds of an accompanying piano and a singer create a coherent unit or stream. In this way the musical performance forms a perceptual unit which is separate from other ongoing events such as coughing in the audience or the shuffling of program notes . Sound also "refers indifferently to the physical sound in the

world and our mental experience of it" (Bregman 1990, 10), while a stream is a perceptual representation which "acts as a center for our description of an acoustic event" (Bregman 1990, 10). In this way, Bregman explains, an auditory stream plays a similar role in auditory mental experiences as objects do in vision: "it acts as a center for our description of an event" (Bregman 1990, 10).

Auditory streams are formed by the auditory system through analysis of incoming acoustic signals in both the temporal and spectral domains. Bregman (1990) refers to these as sequential integration and spectral integration respectively. While Gestalt psychologists were primarily concerned with visual perception, many of the Gestalt factors which promote grouping in visual perception, influence sequential and spectral integration. Williams (1994) outlines these factors and relates them to auditory grouping as follows:

1 - similarity

Components which share the same attributes are perceived as related. Auditory grouping factors include common onset, common offset, common frequency, common frequency modulation, common amplitude modulation, and timbre.

2 - proximity

Components close to each other are more likely to be grouped together. Auditory grouping factors include temporal proximity, frequency proximity, and spatial location.

3 - good continuation

Components that display smooth transitions from one state to another are perceived as related. Auditory grouping factors include proximity in time of offset of one component with the onset of another, frequency proximity of consecutive components, constant glide trajectory of consecutive components, and smooth transitions from one state to another state for the same

parameter.

4 - habit or familiarity

Relationships between components that have been attributed in the past will preferably be assigned the same meaning when they occur again. Knowledge of familiar patterns and complex structures are stored in what cognitive psychology calls schemas. Once learned, schemas may be used "top-down" to assist in understanding a complex acoustic signal. The activation of a particular schema depends on the level of familiarity and the closeness with which it matches new, incoming auditory evidence. Schemas operate for particular classes of signals such as music, speech, machine noises, and other familiar sounds of our environment. (Bregman 1990, 10)

5 - belongingness

A component can normally form part of only one disjunctive object at a time and its percept is relative to the rest of the figure to which it belongs. Related to this is the principle of exclusive allocation. In terms of auditory grouping, this principle states that once a given sound is allocated to one stream, it tends to be excluded from other possible streams.

6 - common fate

Components that undergo the same kind of changes at the same time are perceived as related. Auditory grouping factors include common onset, common offset, common frequency modulation, and common amplitude modulation.

7 - closure

Incomplete forms tend to be completed. In audition, this refers to our ability to perceive the continuity of a given sound even in the presence of a masking sound. In order for this to be successful, the auditory system must have sufficient energy at the appropriate frequencies to stimulate the same

parts of the auditory system as the missing, or masked, components.

Although these seven points have been well defined in terms of visual perception, Williams (1994) states that "it is becoming evident that the gestalts identified as being of prime importance in vision research may not be directly reflected in audition".

In many situations, auditory streams do not result from a simple summation of factors which influence grouping. Complex interactions arise due to the competitiveness and collaboration between these factors (Bregman 1990). Bregman (1990) describes this process of competition and collaboration in the following way:

> It is as if each acoustic dimension could vote for a grouping, with the number of votes it cast being determined by the degree of similarity in that dimension and on the importance of the dimension. Then the streams whose elements were grouped by the most votes would be formed. (Bregman 1990, 652)

Bregman goes on to say that "such a voting system would be valuable in a natural environment in which it is not guaranteed that sounds that resemble one another in one or two ways will always have arisen from the same source" (Bregman 1990, 652).

Bregman (1990) also discusses the effects and consequences of streaming. Among these, the most important ones have to do with attention and the computation of with-in stream emergent properties. As Bregman states, it is easy to follow and focus our attention on a stream because "an integrated stream is the natural subject for an act of attention" (Bregman 1990, 10). Due to this, and the principles of belongingness and exclusive allocation described above, elements of one stream will less likely interfere with those of another stream when a stream's emergent properties are calculated. These are global features of a stream formed by higher levels of processing when lower level perceptual units are grouped.

Auditory scene analysis is important to the study and design of auditory displays because it provides researchers with an understanding of how complex acoustic signals may be interpreted by the listener. One of the

most important attributes of auditory displays is the intelligibility of simultaneously occurring sound streams: system alerts, background processes, and user action assurances. These auditory streams need to be detected and unambiguously identified by the user. By taking into account the factors which promote auditory grouping, auditory displays can be designed to provide a coherent sonic environment, modeled on real-world listening.

3.1.3 3D Sound

The ability to localize sound in three-dimensional space is a very important aspect of real-world auditory perception. It provides for a sense of situational awareness and self-orientation by allowing us to estimate the position of sounds which may be outside the current field of view. This section will present aspects of human localization and how auditory displays may benefit by incorporating sound spatialisation techniques.

3.1.3.1 Interaural Time and Intensity Differences

Interaural difference cues are probably the most important localization cues we use to localize sound sources on the horizontal plane. From an evolutionary standpoint, this makes perfect sense: humans are terrain-based animals whose auditory system has been optimized through evolution to deal with terrain-based sound sources, including those sources which are outside the field of view. The horizontal placement of our ears maximizes interaural differences for sound waves emitted by a source on the horizontal plane. Our auditory system has the ability to detect interaural differences in phase, amplitude envelope onset, and intensity. By minimizing these differences with head movement, we are able to direct our focal vision to items of interest which may not be in our current field of view.

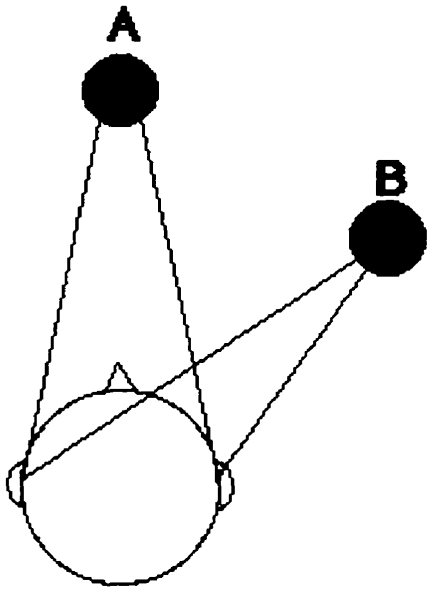
One of the first people to study and explain binaural localization of sound was Lord Rayleigh who, in 1876, performed experiments to determine his ability to localize sounds of different frequencies (Rossing 1990, 75). He found that lower frequencies were much harder to localize than higher frequencies. His explanation was that high frequency sounds coming from one side of the head produce a more intense sound in the ear closest to the

source (ipsilateral ear) than in the opposite ear (contralateral ear). In these cases, the head casts a "shadow" on the contralateral ear, thereby reducing a given sound's intensity. This did not occur for lower frequencies because the wavelengths were long enough to diffract around the head. In a second localization experiment performed in 1907, Rayleigh went on to show how the diffraction of soundwaves around the head caused interaural phase differences. He proposed that these phase differences were use in the localization of low frequencies.

Modern experiments have investigated the role of interaural time and intensity differences and have confirmed Rayleigh's initial findings. Figure 3-1 shows the travel path of sound waves for two sources. We see that for the source directly in front of the listener (source A), the sound waves reach the two ears at the same time. In this case the interaural time and intensity differences are minimized (they are not exactly equal due to the asymmetry of the human head and ears). However, the sound waves emitted by the second source to the right of the listener (source B), will produce significant interaural differences.

In general, if source B is below approximately 1 kHz, localization will be dependent on the interaural phase or time differences (ITD). If the source is greater than about 1.5 kHz (wavelengths are now smaller than the diameter of the head), the interaural intensity differences will be used (IID). This head shadow effect increases with increasing frequency (Middlebrooks and Green 1991).

figure 3-1 - interaural difference



The use of IIDs and ITDs by the auditory system is commonly referred to as the "duplex" theory of localization. The theory suggests that IIDs and ITDs operate over exclusive frequency regions. Although in the laboratory it is relatively easy to estimate the boundary point (around 1.5 kHz) where the system switches from using ITDs to IIDs, the interaction of the two mechanisms in the real world are not fully understood. There have been studies which investigated the role ITD of amplitude envelope onset times in the higher frequency region (Begault 1994, 472). In the frequency region above 1.5 kHz, the phase relationship between the two ears leads to an ambiguous situation: it is hard to tell which is the leading soundwave. These studies have shown that if an amplitude envelope is imposed on the test signal, the auditory system is able to detect the differences in envelope onset times, thus providing useful ITD cues.

Sound sources in the real world typically contain frequency components above and below the cutoff (about 1.5 kHz) suggested by the

duplex theory. It is quite likely that the auditory system does not really rely on any one mechanism for localization. Rather, all available information is used to provide the most suitable answer. The use of pan pots on conventional stereo mixing boards illustrates how amplitude changes independent of the sources' frequency content are sufficient in separating and placing individual sounds on a horizontal plane (the stereo field). Amplitude differences between the left and right channels are interpreted by the listener as various spatial locations. For example, a sound can appear to move across the horizon by continuously varying the amplitude difference of the left and right channels (Begault 1994).

3.1.3.2 The Precedence Effect

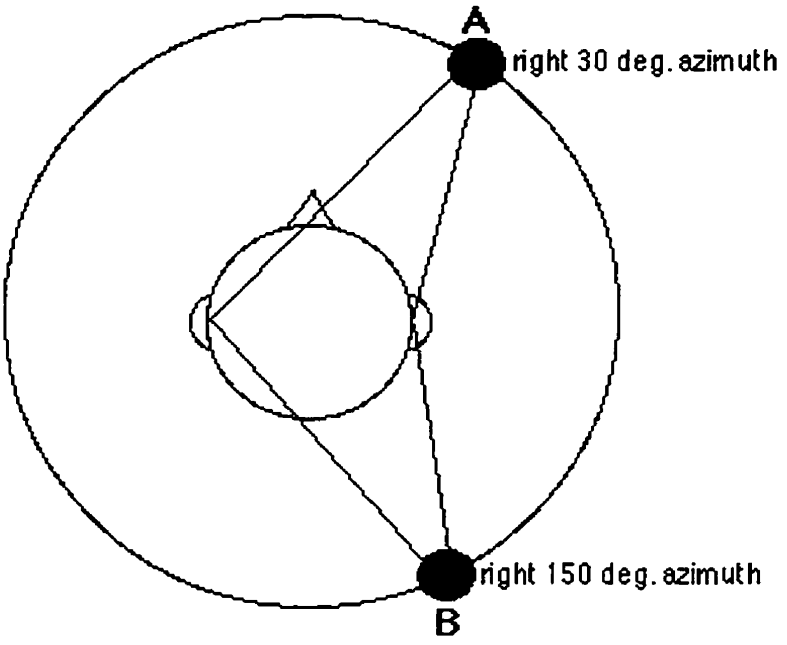
The precedence effect (other names include the Haas effect and Blauert's "the law of the first wavefront") describes the auditory system's ability to localize a sound source within a reverberant environment. Localization experiments have studied the precedence effect by delaying one side of the left-right pair (either through headphones or loudspeakers) and noting the perceptual effects on the listener when varying the delay time. Results show that as the delay time is increased from 1.5 msec. to 10 msec., the virtual sound position will be associated with the undelayed channel but its width will seem to increase. At some point between 10 msec. and 40 msec., depending on the sound source, a distinct echo will be heard coming from the delayed channel. However, the original event is still perceived as coming from the undelayed channel. In terms of real world localization, the Precedence effect explains how we are able to localize the original sound source (or direct signal) in spite of potentially confusing reflections and echoes (Begault 1994, 46).

3.1.3.3 Head Movement in Localization

In real-world perception and localization, our head acts as a pointer, helping us integrate information from both our visual and auditory senses. We use auditory information to locate and focus on particular objects which may or may not be part of the current visual scene. Head movements help us

minimize interaural differences, and essentially point us in the right direction. The following example (adapted from Begault, 1994) shows how head movements are used to locate a source at right 150 degrees azimuth which may potentially be confused with a source at right 30 degrees azimuth. Figure 3-2 illustrates the situation.

figure 3-2 - minimizing IID and ITD with head movement



At first the interaural difference cues suggest that the source is to the right of the listener. As the listener starts turning his/her head towards the right, if the interaural differences are minimized, then the source must be in front. If, on the other hand, the differences increase, then the source is further in back.

Head movements are also very important in front and back disambiguation. Studies have shown that the listener is able to integrate changes in IIDs, ITDs, as well as spectral changes, due to head movement and use this information in localization judgments (Begault 1994, 50). A simpler

example of the importance of quick judgments based on head movements is "if I don't see it but hear it, it must be in back."

3.1.3.4 The Doppler Effect

The Doppler effect (Rossing 1990, 42) is another important localization cue. It is the perceived pitch change of sound sources caused by listener and/or sound source motion. If the the listener and source are moving towards each other, there will be an increase in perceived pitch. On the other hand, if the two are moving away from each other, the perceived pitch will be lower. The situation is explained as follows (adapted from Rossing, 1989): If a given stationary sound source emits 100 sound waves per second, a stationary listener will count exactly 100 waves per second. If the listener starts moving towards the source, he/she will meet more soundwaves per second, thereby increasing the perceived pitch. If the listener is moving away from the source, less soundwaves will be received per second, resulting in a lowering of perceived pitch. Similarly, the listener receives more sound waves per second as a sound source is moving towards him/her, and less if the sound source is moving away.

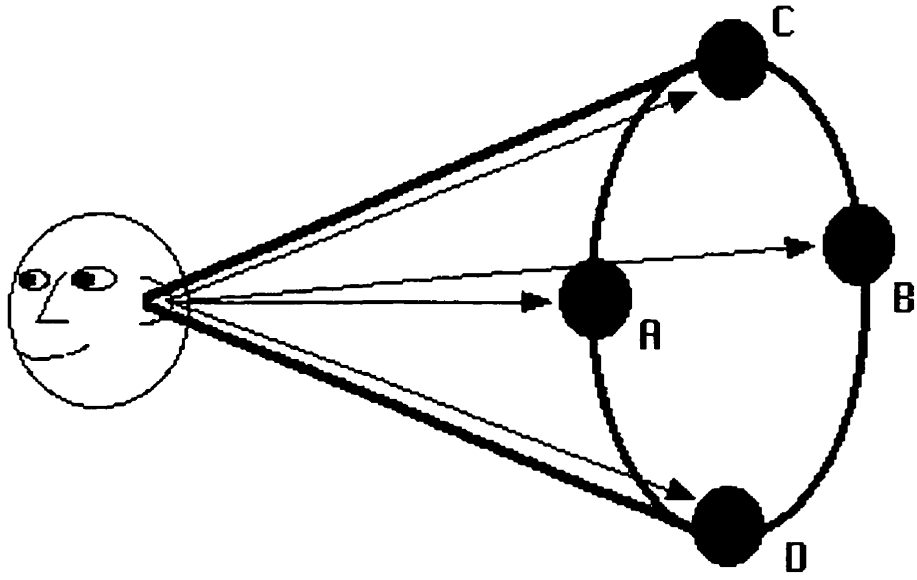
The Doppler effect is an important localization cue because it helps us detect moving and accelerating sound sources. However, although it has been studied and understood as a perceptual phenomenon, further investigation is needed to examine the interaction of the Doppler effect with other localization cues, including cognitive processes such as experience and familiarity.

3.1.3.5 Spectral Cues

Although IIDs and ITDs are probably the most important cues for localization of sound sources on the horizontal plane, they provide rather ambiguous cues for sources located on the median plane. Although IID and ITD values won't be exactly the same due to the asymmetrical construction of our head and pinnae, interaural difference values will be minimal along the median plane. This would lead to confusion when trying to determine whether a source is directly in front (0 degrees azimuth) or directly in back

(180 degrees azimuth) solely based on interaural difference cues. The Cone of Confusion (Moore 1982, 203) illustrates how for any two points on a conical surface extending outwards from a listener's ear, identical (hence ambiguous) IID and ITD values may be calculated (points A & B, and C & D in figure 3-3). It is in these situations that spectral cues provide further aids for localization. They help the listener disambiguate sound source positions between front and back, and above and below.

figure 3-3 - Cone of Confusion



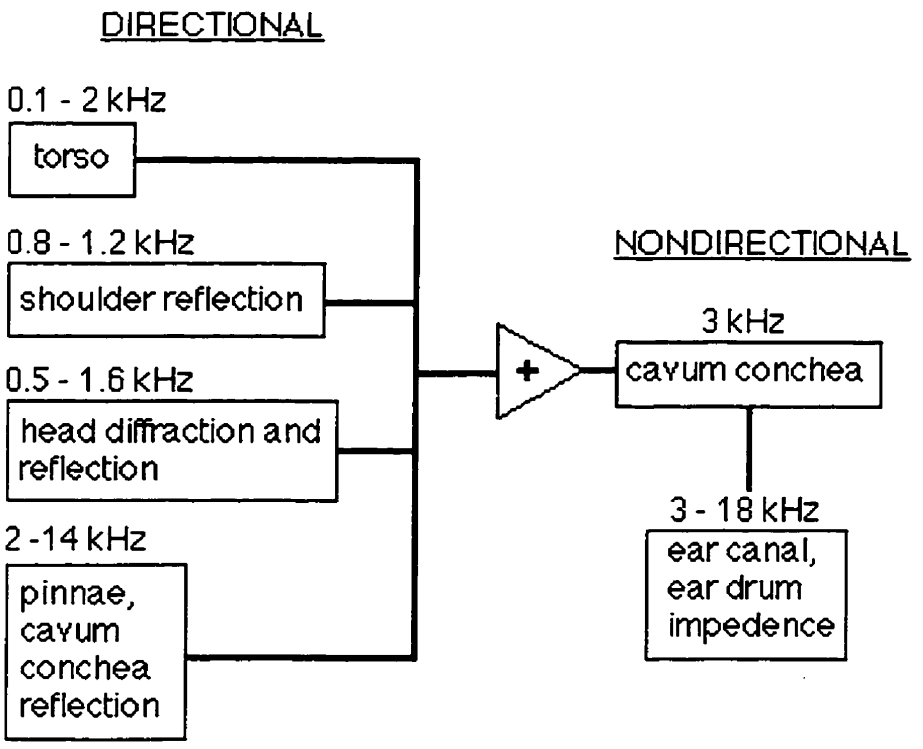
The pinnae are responsible for the spectral alterations of incoming soundwaves. They acts as directional filters, imposing amplitude and phase changes as a function of sound source location. Most of these spectral alterations are caused by time delays (0 - 300 μsec.) due to the complex folds of the pinnae (Begault 1994, 52). Because of the asymmetrical construction of the pinnae, sound coming from different locations will have different spectral changes imposed on it. The listener recognizes these modifications as spatial cues. The result of the filtering process of the outer ears is most often called

the Head Related Transfer Function (HRTF). Other terms include Head Transfer Function (HTF), pinnae transform, Outer Ear Transfer Function (OETF), and Directional Transfer Function (DTF) (Begault 1994, 53).

Modern experiments and studies record, analyze, and simulate HRTFs in order to gain a better understanding of the process of using spectral cues for localization. In general, studies have shown that although different people exhibit different ear impulse responses, or HRTFs, most measurements share similar spectral patterns (Hiranaki and Yamasaki 1982, and Asano, Suzuki, and Sone 1990). Although people do better in localization tests when using their own HRTFs, they are able to make quite accurate localization judgments when using HRTFs of others. These studies have also shown that some people even do better with "foreign" HRTFs than with their own, implying perhaps, that some people have more suitably shaped ears for localization.

Besides the pinnae, other parts of the body can influence the spectrum of an incoming soundwave. These can be broken down into directional and non-directional spectral modifications. For example, the upper body will cause directionally-dependent alterations to the spectrum in the 100Hz - 2kHz range (Genuit 1984, adapted from Begault 1994). The ear canal on the other hand, is a non-directional influence due to its natural resonance between 2kHz - 5kHz. Figure 3-4 lists the various directional and nondirectional influences on HRTFs (adapted from Begault 1994).

figure 3-4 - directional and nondirectional HRTF components



3.1.3.6 Spatialized Sound and Auditory Displays

Auditory displays may become more realistic, and immersive when spatialized sound is incorporated. These enhancements are similar to those provided by three dimensional graphic systems: objects look more realistic and navigation through virtual worlds is more intuitive. Three dimensional sound engages our auditory system in similar ways, and can enhance a user's auditory display experience.

A summary of the benefits spatialized sound brings to auditory displays is provided by Wenzel (1994). She describes the two most important performance advantages of spatialized auditory displays as enhanced

situational awareness and enhanced comprehension of multiple, simultaneous auditory streams. The roles that spatialized sound plays in supporting these two performance advantages include: direct representation of spatial information, spatial metaphors for displaying nonspatial information, enhanced stream segregation, and an enhanced sense of realism. Wenzel goes on to describe how applications in various fields may benefit from the use of spatialized sound including architectural acoustics, large-scale database and information systems, data visualisation, aeronautics, and telerobotic control. Other benefits of spatialized auditory displays include a decrease in visual search times (Perrot *et al.* 1991, and Begault 1993) and a reduction of noise through spatial and spectral release from masking (Doll and Hanna 1995).

Another impetus for the incorporation of 3D sound in auditory displays, is that enabling technologies are now becoming available to the general public. Sound spatialization was once the sole domain of researchers with access to expensive computer workstations. 3D sound capabilities are now being offered to the general public in the form of desktop computer hardware, software, and consumer audio gear.

3.2 Elements of Auditory Display Design

The successful use of audio in human-computer interfaces and interactions requires system designers to consider sound as an integrated and functional element of the system. Too many systems and applications incorporate audio as an afterthought and assign sounds to various system elements in an ad hoc manner. This section will discuss the important issues that need to be addressed in a successful design of auditory displays: the function of sound, the mapping of sound to data, sound design, and sound generation mechanisms.

3.2.1 The Function of Sound

The fundamental question of auditory display design is: what function will audio serve in this application or system? Buxton (1989) gives an overview of three general types of information capable of being conveyed through non-speech audio messages (Buxton 1989):

1 - alarms and warnings

2 - status and monitoring indicators

3 - encoded messages

Alarms and warning messages are discrete events and convey urgency. They are meant to disrupt any ongoing tasks and bring to the user's attention some vital aspect of system performance or state. These types of acoustic signals have been in use in computing systems and various other sorts of machinery for some time. However, not much attention has been paid to important aspects of their sound design. For example, alarms play a vital role in high stress environment such as medical centers and airline cockpits. Due to their loud and jarring characteristics however, alarm sounds often becoming annoying and can even induce more stress (Begault 1994). Research addressing the specifics of auditory alarms has shown that important aspects of alarm design include overall level, temporal characteristics, and spectral characteristics (Patterson 1982, adapted from Kramer 1994).

Status and monitoring indicators provide feedback on current system states and on-going processes. These can take the form of discrete auditory feedback signals, or continuous sounds. As an example of the former, Buxton (1989) describes how the changing pitch of key click sounds can inform the user of a text processor whether the application is in edit mode or view mode. Continuous sounds on the other hand, can represent on-going internal system processes, and last for as long as the process is in progress. If appropriately designed, these sounds can take advantage of the user's ability to monitor background auditory streams. Changes in these background streams will be detected by the listener and will point to changes in the underlying processes. As Buxton (1989) states however, auditory display design needs to take into consideration the fact that "although we can recognize and simultaneously monitor a number of different audio cues, we can normally

only respond to one or two at a time."

Encoded messages are used to convey quantitative or numerical data in patterns of sound. These are designed to take advantage of a listener's auditory pattern matching and recognition skills. An example of these types of messages would be the mapping of sounds to multivariate data sets such as statistical and financial information. The user would then listen for sound patterns which may either be familiar and expected, or completely new leading to novel conclusions. These techniques are similar to those found in data visualization systems.

The functionality that audio can provide, points to two broad categories of auditory display applications: monitoring systems, and data analysis systems. Each requires different approaches towards design and user training (Kramer 1994b). It should be noted however, that many tasks involve an overlap between these two categories.

Monitoring applications allow the user to monitor various on-going processes and examine system states. Although this implies a certain level of data analysis, the purpose of the analysis in this case is to recognize and attend to familiar patterns representing normal and problematic system states.

In terms of design, an auditory display used for monitoring events needs to unambiguously provide the user with process and system state information. User training involves learning a limited, but not necessarily small, set of auditory messages corresponding to various system elements and processes. Examples of monitoring applications include medical data monitoring systems, air traffic control equipment, and general computer interfaces.

In contrast, data analysis systems are used for data exploration where the data is often time-varying and multidimensional in nature. Although the user may recognize certain system states, the goal of data exploration is to examine the interaction of various data variables by listening for and exploring unfamiliar sound patterns. Due to the nature of the task, auditory displays used in data analysis systems need to provide far more interactivity than monitoring systems. Users need to be able to tweak and tune various parameters as they explore unfamiliar data sets. This requires a higher level of

end user training, which in some ways is similar to that of musicians. Users need to learn to "listen" to the system with an analytical ear, and be able to draw conclusions from what they hear. This also implies that users need to be able to hear data in multiple ways (Kramer 1994b). Auditory data exploration can prove to be useful in some of the same areas as data visualisation systems. These include scientific studies dealing with environmental, chemical, and physical modeling, as well as more commercial applications such as statistical and financial analysis.

3.2.2 Mapping Sound To Data

The success of using audio in human-computer interactions is dependent on the user's capability of extracting and understanding relevant information from the sounds produced by the system. The well implemented mapping of sound to data is therefore a crucial element in the design of auditory displays. This section will present several of the most common mapping techniques used in current auditory displays.

Buxton (1989) and Gaver (1986) state that most auditory display research has been based on the traditional understanding of sound and our auditory system. Mapping schemes based on this approach map data to the psychophysical attributes of sound: pitch, timbre, and amplitude. Bly (1982, 1987, 1992) has shown that this type of mapping can be successfully used in representing time-varying multivariate data. Similarly, Kramer (1991, 1994a) uses a technique he calls parameter nesting. Essentially, he maps data to various levels of the psychophysical attributes of sound. For example, there are five levels of loudness nesting: pulse speed, duration, envelope, cluster speed, and master volume, each of which can represent a unique data variable. Pitch can be controlled at various levels as well, including overall pitch, modulation frequency, and musical scale.

Earcons are also based on mapping data to the psychophysical attributes of sound. They are abstract, synthetic tones that can be used in structured combinations (Brewster *et al.* 1994). In many ways earcons resemble motives in music compositions in that they both take the form of short, malleable rhythmic and melodic fragments.

Blattner *et al.* (1989) discuss a hierarchical approach to earcon design which lead to families of related messages. They give the following example of how an error-message family may be built: At the top level of the hierarchy is a unique rhythm which identifies the error-message family. The next level down would then assign various pitch collections to the different types of errors. For example, program execution errors can be identified by a descending minor arpeggio coupled with the distinct rhythmic pattern of the error message family. Using the same rhythmic pattern, a descending diminished seventh chord may signal an operating system error. Various types of error messages within these categories can then be distinguished by unique timbres. For example, a descending diminished arpeggio with a trumpet sound can signal a write to disk error, while the same pattern with a flute sound can signal a read from disk error. Similarly, register and dynamics can be applied to earcons to further categorize related system events.

In contrast to these mapping techniques, are those which use real-world sounds or auditory icons, to present information. Gaver (1986, 1989, 1994) has proposed a mapping scheme based on a theory of sources rather than one based on the proximal or psychophysical stimuli of sound discussed above (Buxton 1989). His argument is that our normal mode of hearing involves listening to sounds in order to identify their sources. When someone hears footsteps behind them in a dark alley, they are more likely to hear that they are being followed than to focus on the timbre, rhythmic regularity, and pitch of those footsteps. Real-world sounds carry information describing the physics of the events that caused them. This includes the material and size of objects, as well as the type of event (e.g. glass breaking, door slamming, sandpaper scraping, etc). Auditory icons apply these concepts to elements of the computer system where text files sound like shuffling paper, large applications sound like large metallic objects, and background processes sound like machines at work. There has been a great deal of work with auditory icons based on Gaver's approach, including that of Jonathan Cohen (1994) and Elizabeth Mynett (1994).

Auditory icons may provide an intuitive approach to understanding what is being represented by a given sound (in terms of both object and

event). Earcons on the other hand, are inherently more abstract, making a particular mapping useful only after the user is given a certain amount of training. While interpreting auditory icons is based on existing real-world listening skills, interpreting earcons requires listening skills similar to those employed when listening to music and may be more difficult to develop for some people.

Although auditory icons convey the notion of sources and interacting objects, much research is needed to investigate what aspects of sound give us this information. On the other hand, the psychophysical attributes of sound have been well studied and may offer a higher dimensionality in terms of data representation: data variables may be mapped to the many parameters of sound (pitch, timbre, amplitude, frequency modulation rate, etc.). The selection of an appropriate mapping scheme for a given auditory display system depends on what kind of data needs to be represented and who the end-users of the system will be.

3.2.3 Sound Design

Computer users often turn off the sound features of their systems because they find the sounds annoying, distracting, and uninformative. Clearly, the choice of sounds and their design have a great impact on the success of a given auditory display (similar to the impact of music and sound effects in movies, video games, etc.). This section will discuss some general issues that sound designers need to be concerned with.

The fidelity of sound is of great importance. Most sounds used in computer systems are of low fidelity. In the past this was due to hardware restrictions: high fidelity sound playback was not available and the storage of high fidelity sound files was not practical. The trend to use low fidelity sound has continued even today where systems with greatly improved sound playing capabilities and storage space, as well as greater processing power, are widely available to the general public. This reflects the consideration of audio as an afterthought or "bonus" feature in many system designs.

Besides offering a more enjoyable user experience, high fidelity audio provides for greater functionality in auditory display systems. In describing

how sound fidelity affects user interpretability of audio messages, Bargar (1992) states that high fidelity audio allows the differentiation of similar sounds used in complex representations. He goes on to say that low fidelity sound changes the listener's task from differentiation to categorization. This is similar to comparing high and low quality photos: more detail is present in the higher quality picture, hence conveying more information to the viewer.

Patterson (adapted from Kramer 1994, 44) suggests some other guidelines for auditory display sound design. Although he worked on auditory warning systems used on civil aircrafts, his findings may influence sound design in general. Patterson states that the main design issues include overall sound level, temporal characteristics, and spectral characteristics. For audio messages to be informative, they need to be detected by the user. Loud sounds are annoying, distracting, and in Patterson's study, tend to incapacitate users. He found that auditory messages need to be at least 15 dB above the noise floor or masked threshold, but should not exceed 30 dB. As far as a alarm sound's temporal characteristics are concerned, onset and offset times should not be too abrupt so as not to startle the user. Patterson found that onset times between 20 and 30 milliseconds were preferable. He also found that similar temporal patterns tend to lead to confusion between messages. Similarly, spectral patterns of individual sounds need to be diverse as well. When faced with the decision on how much control to give to the user when making preference adjustments, auditory display designers may take Patterson's findings as base values.

Other important, yet poorly understood, aspects of sound design are the affective and emotional responses that sounds elicit in listeners (Kramer 1994a). One only needs to examine the various effects that music has on people: it has the ability to convey a whole palette of emotions including pleasant and unpleasant feelings. In terms of auditory displays, these aspects of sound can influence how information is interpreted (Kramer 1994a). Kramer (1994a, 214-215) gives a partial list of affective associations with sound which includes ugliness, richness, hollowness, and unsettling. Although music provides us with examples of how sound is used to convey these feelings, more scientific study is needed for a better understanding of how

these factors may be controlled and put to use in auditory displays.

3.2.4 Sound Generation

Auditory display designers have three broad categories of sound generating techniques at their disposal: sample playback, real-time synthesis, and MIDI.

Sample playback is currently the most widely used playback mechanism. Practically all computer systems today offer the user some form of capturing (sampling) a sound, storing it, and playing it back. In terms of auditory display, this situation is deceptively simple. Gaver (1994) outlines some difficulties related to using sampled sounds. The first is that finding a real-world sound that best represents a certain auditory display element may be difficult: while emptying the computer desktop trash has a real-world counterpart, copying a file from one directory to another does not. Second, shaping and real-time modification of sampled sounds in terms of auditory display parameters is difficult. Most sound design software is designed for musical purposes and not for addressing auditory display issues. Lastly, high-fidelity sampled sounds require large amounts of storage memory.

Real-time synthesis addresses some of the problems of using sampled sounds. Sound designers are able to create their own sound generating algorithms and generate tones best suited to the requirements of a given auditory display. This provides a rich opportunity for sound exploration and manipulation. However, most synthesis programs are standalone applications and are not easily integrated with other system functionality. Furthermore, direct synthesis requires auditory displays to maintain their own synthesis engines in either software or hardware, increasing the size and complexity of the auditory display system.

A middle ground between the above two methods of sound generation is MIDI. MIDI provides the communication protocol to control devices capable of both real-time synthesis and sample playback. MIDI devices have become affordable and widely accessible to the general public in the form of musical instruments, computer sound cards, and software. Auditory display systems may easily incorporate outboard MIDI gear for sound generation,

freeing up valuable processing resources.

The problems with MIDI are similar to those of sound design software. These devices are primarily designed for applications in music and not for dealing with the specifics of auditory displays. In addition, the low resolution of MIDI data (128 possible values for most MIDI messages) and its maximum serial transfer rate of 31.25 Kb may be insufficient for complex auditory displays.

4. The Use of Sound on the WWW

Although the Internet originated in the nineteen sixties, it wasn't until the development of the World Wide Web (WWW) in the early nineteen nineties that the Information Superhighway was embraced by the general public. The WWW has provided easier access to the Internet in the form of a graphic user interface (GUI) - essentially a window into a vast pool of information. It has become a powerful communication tool, providing connectivity across the globe.

As with most computer system developments, the visual component of WWW applications is far ahead of audio support. WWW research and development is mainly concerned with extending the GUI paradigm to accessing the Internet. For the WWW to be a true multimedia interface to the Internet, the communicative power of sound cannot be ignored.

4.1 General Aspects of the WWW

Fluckinger (1995, 274) describes the WWW as follows:. First, it is the name of a project started at CERN, the European Laboratory for Particle Physics, in 1989. The impetus for this project was the need for researchers to manage and share large amounts of information with fellow collaborators who were spread around the world. Second, it is a collection of specifications and protocols designed to address how this body of information may be managed and shared. These specifications include the structure of WWW documents, how to access these documents, and how they may be transferred over computer networks. The development of these protocols was quickly adopted by the general Internet community. This in turn lead to the establishment of a world wide organization concerned with the continued development of WWW standards in 1995, led by MIT in the USA and INRIA in Europe. Lastly, Fluckinger describes the WWW as "the space of digitized information, the hyperspace, available over the Internet and supported by a set of interlinked information servers".

Fluckinger (1995) offers two fundamental reasons for the wide spread popularity of the WWW:

1- the lack of a central authority

2- the universality of the WWW

A lack of a central authority provides the opportunity for anyone to publish and integrate information with preexisting WWW documents. World wide connectivity, the ability to exchange information over heterogeneous computer networks, and the standardization of communication protocols all contribute in making the WWW a universally accessible information resource. For example, the WWW can provide students across the globe with easier access to research material which would otherwise be buried in the corners of some distant library.

These defining characteristics of the WWW also point to some inherent problems. Although the ease of WWW publishing facilitates the dissemination of useful information, it also allows for "hyperspace pollution": just because something is published does not mean it is important or useful, and may waste valuable resources like storage space and network bandwidth. As in the real world, the Internet community is very much concerned with human right issues like freedom of speech and censorship.

The main problem with the universality of the WWW, is that the development of tools to be used over heterogeneous networks must be designed to deal with the lowest common denominator. This often results in sacrificing the use of higher-end system functionality to be compatible with lower-end systems.

Over the past couple of years, the WWW has experienced an explosive growth in both the development of related technologies and the number of people accessing online resources. This has increased the complexity of the WWW in terms of the volume of available information, how information is accessed, and the very nature of the information itself. It is no longer simply a playback mechanism for static hyperlinked documents, but has developed into an interactive medium full of dynamic content.

The use of the WWW has quickly evolved from the exchange of knowledge through simple documents, to general, multimedia

communication. Today's interactive WWW applications include commercial transactions, entertainment and "edutainment" applications, and computer-supported cooperative work. WWW technologies will continue to develop as computer companies and software developers look to integrate Internet accessibility and services with more traditional desktop computer systems.

4.2 Current Uses of Audio on the WWW

The use of audio on the WWW falls under three main categories: audio file download, streaming audio, and embedded audio. The following section will discuss each of these categories and how they relate to the demands of interactive WWW auditory displays.

4.2.1 Audio File Download

The most conventional use of audio on the WWW is that of the distribution of audio files. Users first select which sound file(s) they wish to hear and initiate a download. Web sites usually offer sound files in several formats (e.g. AIFF, WAVE, MPEG, etc.) and leave it up to the user to select which one is the most suitable for their particular system. Once downloaded, the audio file is played with an appropriate sound playing application set up by the user. Very often this consists of configuring the WWW browser to launch an appropriate application responsible for playing the sound file. However, it is becoming more common for WWW browsers to handle audio (as well as other media) playback themselves as an integrated feature of the browser application.

Audio file transfers have become quite popular with musicians. For example, composers are able to set up web sites containing archives of "sound bytes" from recent compositions, making examples of their material available to anyone who has access to the WWW. Similarly, record companies are posting sound files in an attempt to promote the sales of recordings.

Audio file downloads however, do not meet the demands of auditory display systems. In contrast to the real-time interactive aspects of auditory displays, the nature of conventional audio file downloads is that of a selection

process and information request. It is up to the user to select a file, wait for it to download, and make sure that the playback application is properly installed. In addition, the latencies inherent in transferring files over the Internet dispel the notion of real-time system response.

4.2.2 Streaming Audio

Streaming sound files enable the real-time transmission of audio on the WWW. Instead of having to wait for the whole file to be downloaded in order to hear it, audio is played back on the user's system as it is being transferred over the network.

Fluckinger (1995) describes four critical performance criteria related to the real-time transmission of time-dependent media:

1- The throughput of the network, usually expressed as the number of bits the network is capable of accepting and delivering per unit time.

2- The transit delay. This is the time elapsing between the transmission of the first bit of a data block by the transmitting system and its reception by the receiving end-system.

3- The delay variation is the variation over time of the transit delay.

4- The error rate measures the behavior of the network with respect to alteration, loss, duplication, and out of order delivery of data.

Fluckinger (1995) also discusses the demands placed on underlying networks by real-time audio transmissions . He states that for uncompressed audio streams, telephone quality audio requires a bit rate of 64 kilo-bits per second (Kbps) while CD quality audio requires a throughput of 1.4 mega-bits per second (Mbps). On the other hand, compressed audio streams require lower bit rates: telephone quality can require 32, 16, or 4 Kbps (depending on compression algorithm used), and CD quality can vary from 384 down to 192 Kbps (again, dependent on compression algorithm). Some compression algorithms, such as MPEG-Audio Layer 3, even offer near CD quality with bit

rates as low as 64 Kbps. If other media (e.g. video) accompany the audio, streaming technologies also need to address the issues of synchronization between the various elements of the given real-time transmission.

Numerous companies are involved in developing streaming technologies, and have made great strides in a rather short amount of time. The three major types of Internet applications which benefit from this technology are telephony, audio/video broadcasting, and teleconferencing (Lombardi, 1995). Internet telephony is mainly used for bypassing the long distance costs of regular telephone service. Whether or not this is a waste of network resources is open for debate. Internet-based audio/video broadcasting and teleconferencing are perhaps more useful applications. They extend the capabilities of traditional information media like news casts and educational seminars, by offering new far-reaching broadcasting opportunities. The viewer/participant base is no longer confined to a small local area, and can now include people across the world.

Both audio streaming and auditory displays are concerned with time critical issues in audio delivery. While the former is mainly concerned with maintaining the integrity of a given audio stream, auditory display systems have the additional task of performing with minimal response times. The main purpose of auditory displays is to provide immediate feedback to the user regarding the state of the system and user actions. An Internet broadcast of radio news, for example, does not need start at the moment the radio transmission begins, unlike an auditory display which needs to inform the user immediately about the state of some mission critical process. Due to inherent network latencies, auditory displays can not (yet) rely on sending immediate informative auditory messages and cues across a network.

4.2.3 Embedded Audio

This third category of audio use on the Internet is more directly related to auditory displays. Embedded audio involves hiding the complexities of audio file download and streamed audio: an audio or MIDI file is automatically downloaded and begins playing when a given web page is visited.

Applications of embedded audio range from simply setting a mood to enhancing elaborate interactive content. Technologies like the Virtual Reality Modeling Language (VRML), Macromedia's Shockwave, Sun's Java, and Netscape's ONE programming suite provide the tools necessary for developing sophisticated, WWW multimedia environments. In turn, these technologies afford the opportunity to implement various auditory display techniques as part of the embedded interactive content.

4.3 Auditory Displays for The WWW

The WWW is a relatively young area of application development. By incorporating auditory display techniques early on, sound can become an integral and expected component of the WWW user experience rather than a "special" feature. Audio can complement visually presented material, heighten user engagement, and increase the perceived quality of the application. This section will discuss how a variety of WWW applications stand to benefit from various sonification techniques.

4.3.1 WWW Browsers

WWW browsing has brought with it a whole new set of user interface issues. Much like the rest of the computing world, most information is displayed visually. The sole sound these applications make is a system beep accompanying alert boxes. Sound can not only enliven the WWW experience, but provide services and functionality which may be difficult, or impossible, to present visually.

Many of the benefits of conveying information through sound can be applied to WWW browsers. At the most basic sonification level, audio can provide assurances and feedback for user actions. These include basic operations shared by all applications such as menu selections, button presses, and the success or failure of various commands such as file saves. Auditory messages can also be applied to elements and operations specific to WWW browsing. Our ability to a monitor a number of simultaneous background

auditory streams, can be used to indicate connectivity status and the state of downloads in progress. The telephone provides some examples of these types of auditory cues. Dial tones, busy signals, and noisy lines all convey the current status of telephone connectivity to the user. In WWW browsing environments, the user is not always aware of whether the state of their online connection(s) unless they look for visual indications in the form of windows and progress bars (which may be hidden from view). Connectivity status can be represented aurally with the use background ambient sounds. Any changes in connectivity (e.g. interruption, stall, loss of connection, etc.) can be indicated by changes in the background sounds. The user is able to notice these changes without being distracted from whatever task he/she may be involved with such as reading an onscreen document.

Hypertext navigation can also be made more productive through auditory cues. Very often the user does not know what's "on the other side" of a particular hypertext link: a text file, an application, a sound file, etc. One solution might be to play identifying sounds as a user passes the mouse over a particular link. For example, links to text files can sound like paper being shuffled, while links to downloadable applications may sound like metallic containers. In addition, the approximate size of linked objects may be conveyed through sound: large text files may sound like large books and large applications may sound like large, full metallic containers. This "mouse-over" technique has the benefit of providing file information without having the user activate a particular link.

Other elements which may be sonified can include the user's bookmark file. Sounds can be used to indicate not only the file type and size, but age as well. For example, the older a bookmarked document is, the more muffled it sounds.

4.3.2 Network Administration Tools

Network monitoring and WWW server applications can be enhanced by incorporating status and monitoring sonification techniques. For example, network administrators can listen to a WWW server in order to hear how busy it is: the greater the load, the busier it sounds (these cues can be similar to

what an overloaded car engine may sound like in the real world). The server's auditory display can also offer the capability to be tuned into specific processes to determine their particular states. For example, the administrator might hear that mail deamon is still running (e.g. an identifiable continuous background sound) but that the FTP service is down (e.g. the absence of the "FTP deamon" sound). Similarly, the status of individual nodes of a network may checked. For example, a network ping (sending out and getting back a test packet over the network to a particular machine) can have an auditory counterpart.

There are a growing number of data analysis applications written for analyzing WWW traffic. These applications are used by web site managers to analyze various server system states such as incoming network traffic and web server log files. These applications employ techniques quite similar in nature to data visualization techniques used by the financial and scientific communities. Variables such as how often requests are made for various services, from where the WWW server is being hit, and the times of greatest load on the server are presented visually in the forms of graphs and customized reports. These multivariate data sets can be sonified using auditory data analysis techniques. Various data parameters may be mapped to auditory variables such as pitch, amplitude, and timbre. Consequently, various trends in the data may be heard which might otherwise be missed when attempting to correlate a number of data values visually.

4.3.3 Communication Tools

The use of the WWW has gone from the simple exchange of documents to providing novel communication tools. Examples include computer telephony and teleconferencing tools, and Computer-Supported Cooperative Work (CSCW) applications are examples of other emerging WWW communication tools. Although text-based CSCW applications have been around for quite some time on the Internet (e.g. e-mail, MUDs, chat lines, etc), these have traditionally consisted of Unix applications with command line interfaces. The WWW has made the use of these applications more intuitive and accessible to the general public through the use of GUIs.

Instead of remembering command line options and arguments, users can now make menu selections and operate graphic buttons.

Some examples of CSCW WWW applications include chat-rooms and message-posting areas. These take the form of simple windows with various menu options and buttons for entering and using chat-rooms or posting areas. Auditory displays can provide sonic assurances. For example, auditory cues can be used for hearing when a new user has joined (or left) a chat session, the success or failure of posting new messages, and connectivity status. Chat rooms may also take the form of more complex VRML worlds, complete with avatars (graphic representation of logged on users) and landscapes. Spatialized sound can make these worlds even more complete where objects and user actions not only look real, but sound real as well.

Other popular uses of CSCW include shared whiteboard tools and and shared application tools (Fluckinger 1995). Shared whiteboards allow multiple participants to view a common virtual whiteboard from their respective computer monitors. This shared space can then either be used as a common sketching pad, or a container for imported documents where multiple participants can annotate, highlight, or markup a given document without actually altering its original contents. Shared application tools extend the shared whiteboard paradigm, allowing multiple users to work on a single document or share a single application. In this case, various users alter the actual contents of a shared document. Examples include collaborative graphic design tools, multimedia authoring suites, and programming environments.

A common issue shared by these applications is conveying to all participants information concerning who is the active speaker, who is making the current annotations, and who is actually present at the session. This information, called floor control (Fluckinger 1995, 143), is usually presented visually in one form or another. For example, a given user types text in a window to notify the other participants that they wish to perform some action (e.g. speak, highlite, draw, etc.). Each participant is also assigned a color so that everyone can identify who marked what on the whiteboard. Finally, the list of people present at the given session is displayed in a window.

While audio/video teleconferencing tools are often used to establish

more elaborate communication channels, they place heavy demands on the underlying network. These may result in long latencies, thereby reducing the productivity of a cooperative session.

Spatialized auditory displays can complement the graphic nature of these applications. For example, auditory displays may more conveniently address the issues related to floor control. Instead of assigning colors to individual participants, they can each be assigned a position in 3D space. Then, when someone makes changes to a given shared document, perhaps drawing a figure or typing some text, appropriate sounds would come from assigned "seating" positions. These sounds can consist of typical drawing and typing sounds like paint brush strokes and typewriter keys. Auditory cues would reduce the visual load placed on users and allow for less distractions while working on the primarily visual tasks of document and application sharing.

4.3.4 General WWW Applications and Documents

More general WWW and Internet applications may also incorporate auditory display techniques. Among these are tools used to store and retrieve data on remote computers (FTP applications), connect to remote computers (telnet clients), and perform WWW information searches (search engines and web robots). All of these need to relay to the user the status of data transfers and connectivity, and may integrate many of the auditory cues described above.

While the applications discussed thus far are means of sonifying the tools for accessing the WWW and the Internet, WWW users and developers may contribute to the sonification of the web by creating sonified web documents. HTML documents have gone from being static documents to interactive content full of graphic elements like menu bars, image maps, animations, and games. More and more web-site developers are now incorporating embedded background music to provide ambience and set a mood while visiting their sites. Embedded sounds may also be used to identify the type of web site (e.g. education, commercial, organization, etc) as well as web site areas (e.g. recent events, press releases, product info, etc.).

5. THESIS SOFTWARE

The software developed as part of this thesis consists of the Netscape plug-in for Macintosh PowerPC called SprocketPlug, and a companion Macintosh utility program Theme Editor. SprocketPlug offers WWW developers the opportunity to incorporate interactive spatialized sound in numerous WWW project settings. The Theme Editor utility is used to create collections of sounds known as SprocketPlug Themes, which contain the actual sounds to be played by SprocketPlug. Theme Editor may also be used by end-users to customize these theme files.

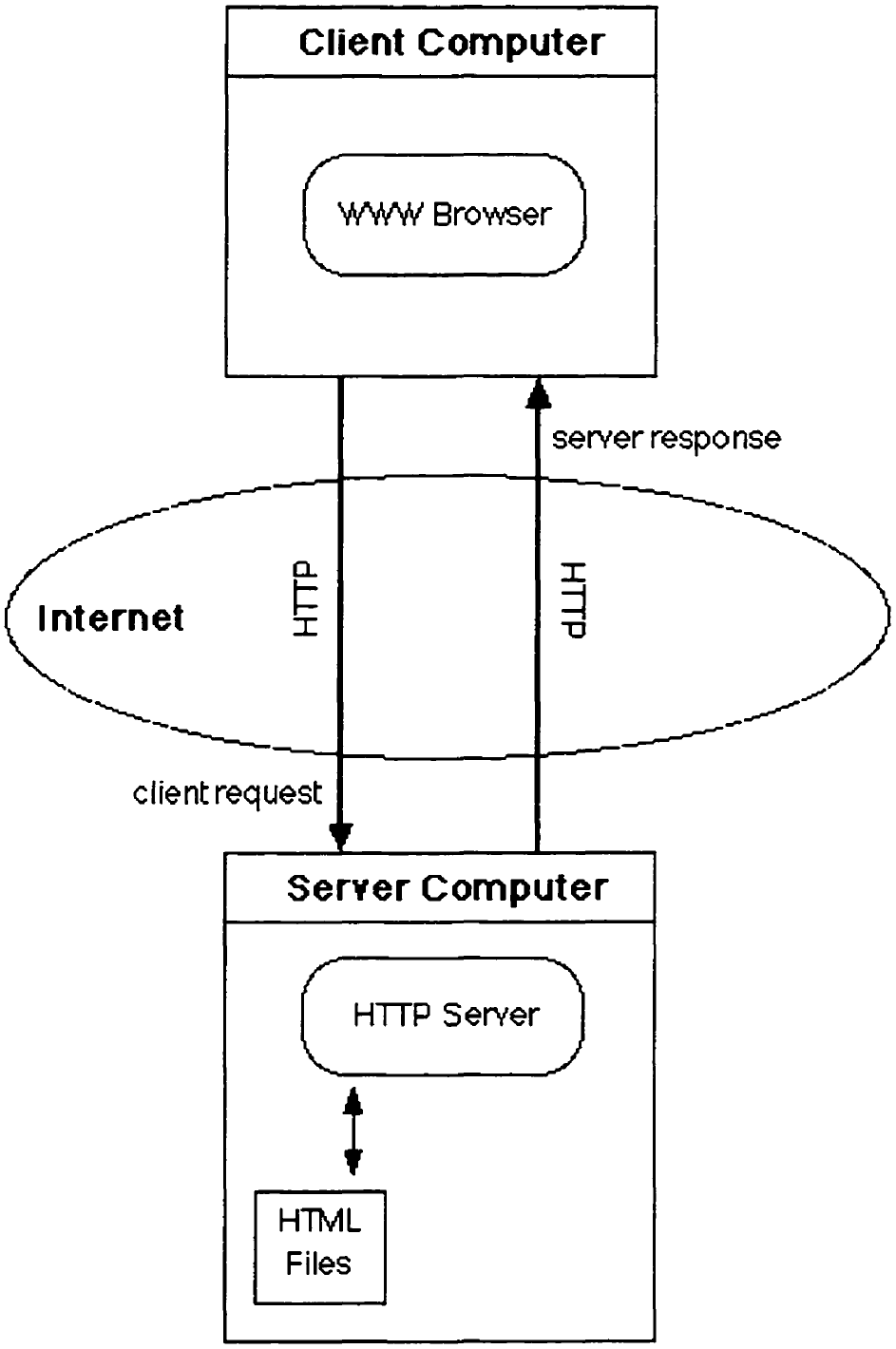
This chapter will begin by discussing how extending client-side functionality has enabled elaborate interactive content to be delivered on the WWW. Next, the technologies specific to the development of the SprocketPlug plug-in will be presented. The remainder of the chapter will describe the thesis software and how programmers can incorporate SprocketPlug functionality at various levels of WWW document/application development (namely HTML, Javascript, and Java).

5.1 Interactive Content On The WWW

5.1.1 Traditional WWW Communication

The WWW is based on a client-server model of computer networking. A WWW client is a web browser running on the end-user's machine which communicates with web servers via the Hypertext Transfer Protocol (HTTP). The communication consists of client requests and server responses. Figure 5-1 illustrates this process.

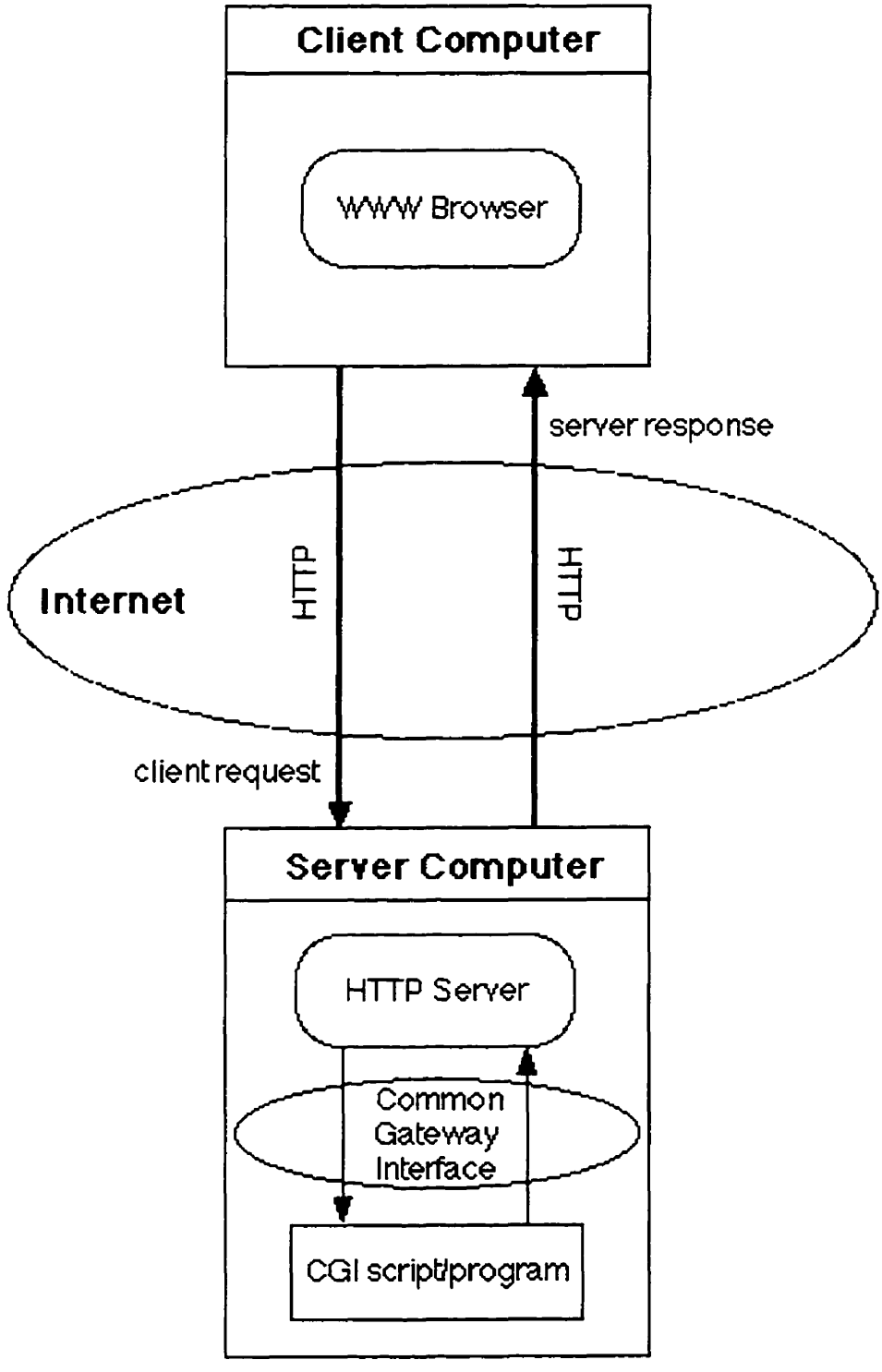
figure 5-1 - client server communication



Communication begins with a client requesting a file from a particular HTTP server. The server then looks for the file and, if available, sends the file to the client. If the server can not locate the file or the client is denied access to the file, the server sends an appropriate error message indicating the reason the file was not delivered. Immediately preceding the delivery of the file however, the server informs the client of the file's type. The server sends the client a message (or header) describing the file's contents using a subset of the Multi-purpose Internet Mail Extensions (MIME) specification. This message is typically "Content-type: text/html", which informs the client it is about to receive an HTML document. Other MIME type messages are used to describe various other types of documents including AIFF sound files (Content-type: audio/aiff), JPEG images (Content-type: image/jpeg), and MPEG encoded videos (Content-type: video/mpeg).

The Common Gateway Interface (CGI) describes the mechanism used by servers to process more complex client requests such as form processing and database searching. Instead of processing these requests themselves, servers invoke external programs (CGI scripts) to do the work for them. The results of the CGI script are then sent back to the client. Figure 5-2 depicts this process.

figure 5-2 - client server communication incorporating CGI



The CGI specification is essentially an interface through which servers pass parameters to external programs and receive results from them. Although CGI programs allow for some interactive client-server communication, due to network latencies and potential server load, they do not meet the demands of real-time, interactive multimedia content delivery.

5.1.2 Extending Client-side Functionality

Several technologies exist to address the issue of more efficient multimedia delivery over the WWW. These include Netscape's plug-in architecture and Sun's Java language. Instead of placing all the load on the server side of WWW communication, these technologies give the client computer the ability to process information locally. Client-side processing allows for more complex and efficient behavior to be integrated into a web page/site than was previously available through CGI programming.

Netscape plug-ins are software modules which extend the functionality of the Netscape Navigator WWW browser, and hence, client-side processing. A plug-in may be created to display various custom data types without having to update or modify the browser application itself. A simple example might involve the display of 3D graph data. The client computer would retrieve the data from a company server, and then proceed to interpret and display the data itself. Previously, server-side CGI programs or external applications were required to execute these types of tasks.

Plug-ins can not only display data within a WWW document, but can also behave like any other application. For example, Macromedia's Shockwave plug-in allows multimedia presentations (prepared with the Director authoring program) to run from within a standard WWW document. Another popular plug-in is Netscape's Live3D, which processes user-navigable three dimensional graphic environments coded in VRML. However, plug-ins consist of compiled native code. Various versions of a given plug-in must be created to accommodate different computer platforms.

Java (Sun Corp. 1996), is a platform-independent, interpreted object-oriented programming language. Although Java may be used to write standalone applications, it's real power is the ability to write Java applets,

which are applications and data embedded in standard WWW documents. In contrast to plug-ins, a Java applet's code is downloaded with the WWW document. This code is then executed by a Java interpreter running on the client machine in conjunction with the WWW browser. Some examples of Java applets include educational programs, games, and utilities.

Netscape and Sun have also developed the Javascript language. This is is a less complex interpretive language, and consists of lines of code written directly into HTML documents. Javascript can communicate directly with other elements on the HTML page, allowing more elaborate processing than is possible with standard HTML. Many of the tasks previously requiring CGI programs, can now be accomplished using Javascript. These include dynamic web page construction based on user input, and user event-processing. Javascript may also be used to lighten the load of server-side processing.

5.1.3 Integrating Local and Remote Resources

A growing number of applications are being developed which integrate local and remote resources. These "Internet savvy" or "hybrid" applications combine the dynamic nature of the WWW (remote resources) with assets stored on client computers (local resources).

There are two approaches to developing applications which integrate local and remote resources. The first approach involves an application running on the client machine reaching over the WWW for content stored on a server. An example might be an educational CD-ROM on space exploration with WWW links to the most recent snapshots of space from NASA's web site.

The second approach in integrating local and remote resources consists of remote applications, or applications running in a WWW browser (e.g. Java applets), accessing locally stored content. By using locally stored content, download and startup times of remote applications can be greatly reduced. Examples include multi-user WWW educational programs and games which use locally stored images and sounds. Locally stored media can then be updated, edited, or even replaced by the user without making any changes to the parent application (found somewhere on the WWW).

The SprocketPlug plug-in developed as part of this thesis, extends client-side functionality by providing a mechanism for interactive spatialized sound handling. In particular, SprocketPlug addresses the issues of limited audio handling in Java and the non-interactive nature of current Netscape Navigator audio plug-ins. Although Java provides basic sound playing capabilities (play, stop, loop), it limits the use of sound to 8 kHz/8 bit monophonic μLaw encoded audio files. In contrast, SprocketPlug can play CD-quality, full bandwidth (44 kHz/16 bit) audio, and provides real-time 3D sound processing (while many audio plug-ins also support full bandwidth audio, they act as standalone players). The sounds used by SprocketPlug are stored locally in sound collections called SprocketPlug Themes. These themes are created and customized using the Theme Editor utility without having to change any source code in the application using SprocketPlug.

5.2 SprocketPlug Enabling Technologies

This section will present the technologies directly related to the development of the SprocketPlug plug-in. Specifically, SprocketPlug incorporates Apple's SoundSprocket technology for sound spatialization, and provides a LiveConnect interface for accessing SprocketPlug functionality from HTML, Javascript, and Java applets.

5.2.1 Apple SoundSprocket

The Apple SoundSprocket API is part of the larger Apple Game Sprockets software developer's kit aimed at game and multimedia development for macintosh PowerPCs (there is no version available for Macintoshes based on the Motorola 68000 family). In addition to the SoundSprocket, this kit includes APIs for input device handling (InputSprocket), graphics (DrawSprocket), and networking (NetSprocket). SoundSprocket works on top of Apple's Sound Manager to provide real-time three dimensional audio filtering (position and distance in 3D space), reverberation, and Doppler effects. It consists of the SoundSprocket Filter

system extension, the SoundSprocketLib shared library, and requires Sound Manager version 3.2 or higher. SoundSprocket can be configured to work with headphones, stereo speakers, or a mono output. The difference between headphone and loudspeaker playback is that crosstalk cancellation filtering is active only for the latter. Directional cues are obviously lost during mono playback, but reverberation and Doppler effects are still present to some degree.

5.2.2 Netscape's LiveConnect SDK

Netscape's LiveConnect architecture allows the integration and interaction of Javascript, Java, and plug-ins within the Netscape Navigator environment (version 3.0 and higher). It gives programmers the following possibilities:

1 - call a Java applet's public variables and methods from Javascript

2 - call Javascript functions and objects from a Java applet

3 - call Java methods from plug-ins

4 - call plug-in methods from Java applets.

5 - call plug-in methods from Javascript

Although points 1, 2, and 3 are useful, points 4 and 5 are perhaps the most interesting. Point 4 describes the ability of Java applets to call native methods implemented by a plug-in. The way this works is that a Java wrapper class is defined for the plug-in which defines the methods the plug-in wishes to export. This class extends Netscape's own netscape.plug-in.plug-in class and appears to Java applets as another available class (the plug-in's Java wrapper class must be in Netscape's plug-ins folder along with its associated plug-in). The ability to implement and call native methods was previously only available to standalone Java applications.

Regarding point 5, an embedded plug-in is accessed from Javascript as an element of the HTML page, much like various other standard HTML elements like form input fields and buttons. Similarly, a given plug-in's methods are called using standard Javascript syntax.

5.3 Thesis Software Overview

5.3.1 The SprocketPlug Plug-in

The SprocketPlug plug-in is a Netscape plug-in for Macintosh PowerPC which implements a virtual audio environment. This environment consists of a listener, one or more sound sources, and a set of room characteristics. These elements have various parameters associated with them which affect audio processing. This section will give an overview of the plug-in's functionality, while section 4.4 will describe the specifics of using SprocketPlug in various settings.

The listener and sound source(s) are located in a three-dimensional space defined by azimuth, elevation, and distance. Each of these may also have associated velocities, giving rise to Doppler effects as the listener and sound sources move and pass each other. In addition, a sound source has several other characteristics including a reference distance and an angular attenuation cone. The reference distance is the distance at which the given sound source was recorded. When the source is exactly at this distance from the listener, no attenuation of the source's sound occurs. For distances further away, the sound source is attenuated, while at distances closer than the reference distance, the sound is amplified. A sound source's angular attenuation cone determines the direction of maximum sound intensity and the amount of attenuation applied relative to the angle between the listener and the sound source. A sound source is always the loudest when the listener is directly facing it. As the angle between the source and listener increases, the sound is gradually attenuated, as defined by the source's attenuation cone. The audio environment also has parameters related to room reverberation including room size, room reflectivity, and the amount of reverberated signal in the final output.

Associated with the SprocketPlug plug-in are SprocketPlug themes. These are Macintosh *resource* files containing a collection of sounds to be played by the plug-in. For example, a Java applet game developer using SprocketPlug for the game's sound handling, puts all the required sound

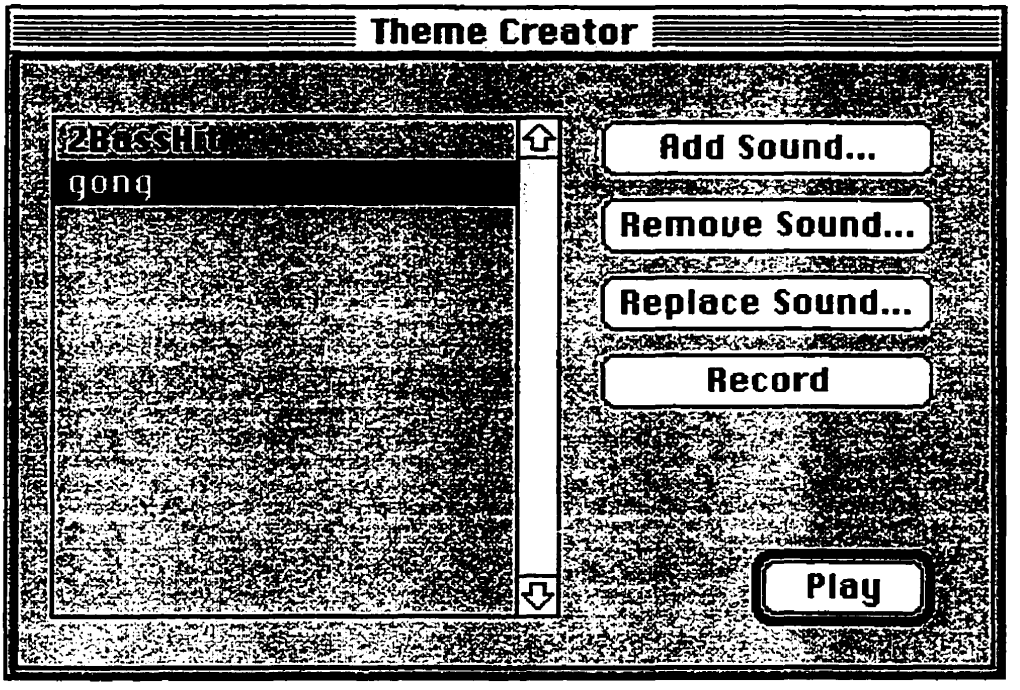
effects and theme music for the game in one SprocketPlug theme. When SprocketPlug is initialized by the game applet, the given theme file is opened and its sound resources are made available for the duration of the game. SprocketPlug theme files are a convenient method of organizing, distributing, and storing the various sounds required by a project.

5.3.2 The SprocketPlug Theme Editor Utility

The *Theme Editor* is a small Macintosh utility program which allows the creation and editing of SprocketPlug themes. SprocketPlug themes are resource files containing three types of resources: a custom resource 'spkr', a template resource of type 'TMPL', and the 'snd ' resource. The 'spkr' resource specifies the number of sounds in the file. The 'TMPL' resource is used for editing the 'spkr' resource when working with the ResEdit resource editing program. The 'snd ' resource contains all the sound resources for a given SprocketPlug theme. Some, or even all, of these sound resources may be compressed, thereby reducing the storage space required for the given SprocketPlug theme.

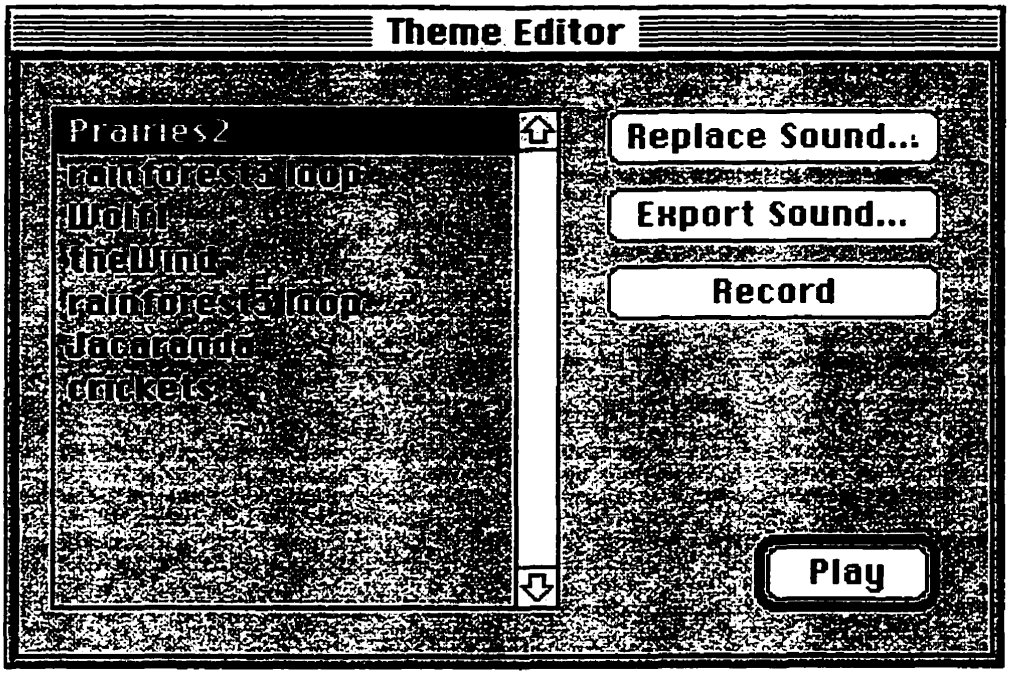
A new theme file is created by choosing "**New...**" under the **File** menu. The *Theme Creator* window pop-ups up (figure 5-3), displaying the theme's sound list (initially empty) and a set of buttons. These allow the user to add, remove, replace, record, and play sounds. The user does not need to explicitly save any operations because *Theme Editor* automatically updates the theme file it is working on.

figure 5-3 - the Theme Creator window



An existing SprocketPlug theme file may be edited by choosing "**Open...**" in the **File** menu. After selecting which theme file to open, the user is presented with the *Theme Editor* window. This window displays the theme's sound list and buttons for replacing, exporting, and recording sound files (figure 5-4). As when creating new themes, any changes to the file are automatically saved by *Theme Editor*. When editing a theme, the user is able to replace, export, and record sound.

figure 5-4 - the Theme Editor window



Theme Editor imports (i.e. adding and replacing sounds), exports and records System 7 sound files. As mentioned above, imported sound files may be compressed files, stored in either standard Apple compression formats (MACE 3:1 and MACE 6:1) or in IMA 4:1(Interactive Media Association) format.

5.4 Using SprocketPlug

The SprocketPlug plug-in provides functionality on three levels of WWW development: HTML, Javascript, and Java programming. This section will discuss how the plug-in's functionality may be accessed from each of these three modes.

5.4.1 HTML Authoring

At the HTML level, using the SprocketPlug plug-in is similar to using most other plug-ins. The HTML author uses an <EMBED> tag to initialize an instance of the plug-in and set various parameters. When embedding a SprocketPlug instance, the embed tag's type parameter must be set to "audio/SprocketPlug", which is SprocketPlug's MIME type. Note that this MIME type does not need to be known by the server since all the data (i.e. the SprocketPlug themes) is on the client machine.

The following line of HTML embeds the SprocketPlug and sets some of its parameters:

```
<EMBED TYPE="audio/SprocketPlug" WIDTH=16 HEIGHT=16
       NUMCHANNELS="5"
       SETTHEME="Nature"
       PLAYLOOP="1 1">
```

The above tag initializes a SprocketPlug instance with five sound channels, opens the Nature SprocketPlug Theme file, and starts playing a looped sound on channel 1. The sound played is the first sound in the theme file.

The following are descriptions of all available SprocketPlug embed tag parameters:

1 - *NUMCHANNELS numChannels*
- sets the number of channels to be allocated where numChannels is the number channels
- this should only be set once for the whole sonified site
- i.e. only the site's main/home page

2 - *PLAYLOOP* `theChannel soundID`

- starts playing a looped sound on the given channel
- `soundID` is an index into the open SprocketTheme file, specifying which sound to play
- `theChannel` specifies which channel to use for playback

3 - *PLAY* `theChannel soundID`

- starts playing a sound on the given channel
- `soundID` is an index into the open SprocketTheme file, specifying which sound to play
- `theChannel` specifies which channel to use for playback

4 - *SETVOLUME* `theChannel theVolume`

- sets the volume on the given channel
- `theChannel` specifies on which channel to set the volume
- `theVolume` specifies a volume level (integer in the range 0-255)

5 - *SETDISTANCE* `theChannel theDistance`

- sets the distance of the sound source, affecting it's attenuation
- `theChannel` specifies on which channel to set the distance
- `theDistance` is a floating point number representing the distance between the listener and sound source in meters

6 - *SETPOSITION* `theChannel theAzimuth theElevation`

- sets the azimuth and elevation for the sound playing or about to be played on the channel specified by `theChannel`
- `theAzimuth` and `theElevation` are floating point numbers specifying the azimuth and elevation in degrees of the sound source
- positive azimuth values are positions to the right of the listener while negative values are to the left

- positive elevation values are positions above the listener and negative values are below

7 - *SETROOM* `roomSize roomReflectivity reverbAttenuation`
- sets the global reverb parameters
- `roomSize` is a floating point number specifying the distance in meters between reverberant walls
- `roomReflectivity` is a floating point number specifying the amount of attenuation in dB (less than or equal to 0.0) that occurs each time a sound bounces of a wall
- `reverbAttenuation` is a floating point number specifying the amount of attenuation applied to the reverberated signal in the final output signal in dB (less than or equal to 0.0)

8 - *SETTHEME* `themeName`
- specifies the name of the Sound Theme file to use
- if this tag is not used, then the Default theme file will be used
- the theme files are located in Netscape's Plug-ins folder inside the SprocketPlug Themes folder

While most of these parameters are optional, the HTML author *must* specify the number of channels to be opened for sound playback. These channels are global. Channels only need to be initialized for the very first instance of SprocketPlug. All subsequent instances of the plug-in will share these channels. For example, when creating a multi-page web site, the first SprocketPlug instance will probably be found on the site's main page. In the case of a site which uses frames, the first SprocketPlug instance might be loaded into the frame used as the site's index. Once the channels have been initialized, any other SprocketPlug instances can send various playback and processing commands to these channels.

If the HTML author does not specify a theme name (using the

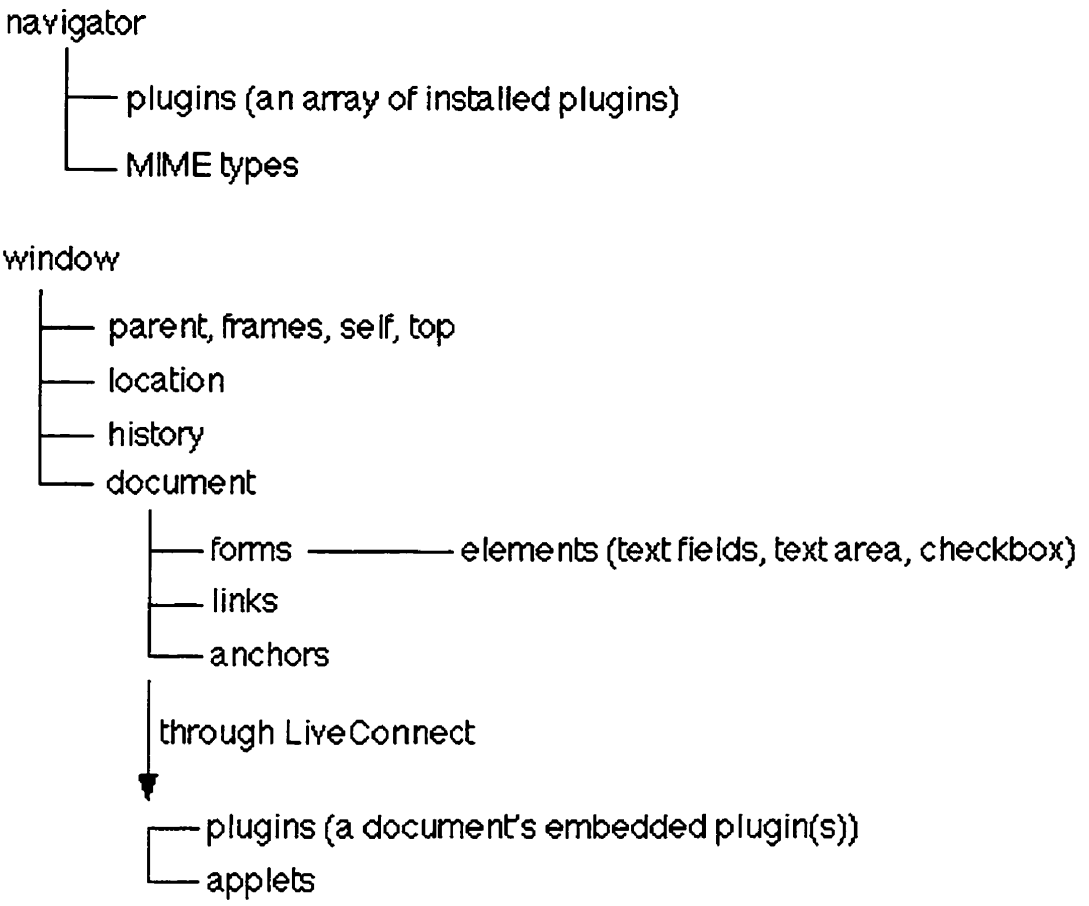
SETTHEME parameter), a default theme will be opened. While setting the theme is not required for SprocketPlug to operate, the default sounds will most likely not correspond with the nature of the given web page/site. Unlike SprocketPlug channels, themes are not global. Meaning, that unless a given instance specifies the same theme as a previous instance, the sounds used by the new instance will not come from the same theme file. Although this may seem an inconvenience, it allows individual SprocketPlug instances to play sounds from different theme files.

If SprocketPlug is not a hidden plug-in (i.e. the <EMBED> tag parameter HIDDEN is set to FALSE), the SprocketPlug icon will appear. When the user clicks on the icon, a pop-up menu will appear, allowing the user to adjust the volume (including muting) of the sounds embedded in the given page.

5.4.2 Javascript Programming

Javascript allows simple logic and behavior to be embedded within an HTML document. It is a scripting language whose capabilities lie somewhere between HTML authoring and Java programming. Netscape's LiveConnect extends the available predefined Javascript object hierarchy to include Java applets and plug-ins. Figure 5-5 shows the Javascript hierarchy.

figure 5-5 - the Javascript object hierarchy



To call on SprocketPlug's functionality from Javascript, the plug-in needs to be embedded in an HTML document using HTML's EMBED tag. All SprocketPlug functionality (including channel initialization and SprocketPlug theme setting) is then accessed via LiveConnect by calling SprocketPlug methods. The following is an example of embedding a SprocketPlug instance for Javascript use:

```
<EMBED TYPE="audio/SprocketPlug" NAME="sprockets"
       MAYSCRIPT=true HIDDEN=true>
```

In order for Javascript to be able to communicate with embedded plug-in, the embed tag's MAYSCRIPT parameter must be set to true. In addition, the embedded instance is given a name for easier Javascript referencing, and the HIDDEN parameter is set to true so that the SprocketPlug icon is not displayed. Although the other SprocketPlug embed parameters (described in section 4.4.1) may be used to set the initial state of the SprocketPlug instance, calling SprocketPlug methods from Javascript allows the WWW developer to incorporate interactive spatialized sound within an HTML document.

Once a SprocketPlug instance has been embedded, its methods are accessed using standard Javascript syntax. Using the above embed tag declaration, the following lines of Javascript code initialize the SprocketPlug instance, set the 3D position of the sound to be played on channel one, and plays a looped sound on that channel. The sound played is the third sound in the Space SprocketPlug theme:

```
document.sprockets.SprInit(5, "Space", 1);
document.sprockets.Spr_LoLvlSetPosition(1, -90.0,
0,0);
document.sprockets.SprPlayLoop(1, 3);
```

5.4.3 Java Applet Programming

LiveConnect gives Java applets access to the full Javascript object hierarchy including plug-ins and other Java applets. In order to do so, the applet must first import the netscape.javascript Java package provided by Netscape in its "java_30.zip" file (found in Navigator's Java folder). The applet then has access to the JSObject class through which the Javascript window object can be retrieved. In turn, this window object then enables the Java applet to retrieve the document object which contains the embedded plug-in instance. A local instance of the plug-in's class can then obtained through the document object. The following is a snippet of Java code illustrating this chain of calls:

the import statements:

```
import netscape.javascript.*;
import SprocketPlug;
```

the code to get a local instance of the plug-in:

```
JSObject          win, doc;
SprocketPlug      sprockPlug;
// The SprocketPlug class must be with
// the SprocketPlug plug-in in Navigator's Plug-ins
// folder
win = JSObject.getWindow(this);
doc = (JSObject) win.getMember("document");
sprockPlug = (SprocketPlug);
doc.getMember("instanceName");
```

In addition to importing the `netscape.javascript` package, the applet needs to import the class which defines the plug-in's Java class to be used for LiveConnect communication between the applet and the plug-in. This class needs to reside in Netscape's plug-ins folder, along side the plug-in for which it provides the interface. Once the applet has a local instance of the plug-in, all of the plug-in's associated methods are available. The following code example initializes the SprocketPlug instance, positions a sound in three dimensional space, and plays a non-looped sound:

```
sprockPlug.SprInit(2, "Space", 1);
sprockPlug.Spr_LoLvlSetPosition(1, 0.0f, 0.0f);
sprockPlug.SprPlayOneShot(1, 3);
```

5.4.4 The SprocketPlug LiveConnect Interface

To export LiveConnect functionality, a plug-in needs to have an associated Java class. This class is essentially a wrapper class for a plug-in's exported functions, and allows the plug-in to appear as a predefined Java object. A plug-in's Java class extends the `netscape.plugin.Plugin` class provided by Netscape, and may consist of standard Java methods and native methods. The native methods are the methods implemented by the plug-in.

The public methods of this class constitute the plug-in's LiveConnect interface: the methods that a programmer may call from Javascript and/or Java.

The compiled class needs to reside in Netscape Navigator's Plug-ins folder along with its associated plug-in, in order to be accessible to Javascript and Java via LiveConnect.

The SprocketPlug LiveConnect interface (i.e. SprocketPlug's associated Java class) is divided into four groups of methods: setting up and freeing SprocketPlug, general controls, low-level calls, and high-level calls. The first group contains the methods for initialization, freeing up resources, and configuring output settings. They are as follows:

public native void SprInit(int numChan, String themeName, int modeFlag)

Initializes the SprocketPlug plug-in, allocates the number of channels specified by numChan, sets the current SprocketPlug theme to the file specified by themeName, and sets the SprocketPlug to use either the low-level or the high-level SoundSprocket API as specified by modeFlag, where a value of 1 sets SprocketPlug to use the low-level API, while a value of 2 sets it to use the high-level API. (The difference between using the low-level and high-level calls will be discussed shortly.)

public native void SprConfigureOutput()

This method pop-ups the SoundSprocket configuration dialog. The user is able to select headphone, stereo speaker, or mono playback. The difference between headphone and stereo speaker playback is that SoundSprocket's cross cancellation filters are active in the latter case. In addition, the user is able to set the angle of the stereo speakers. Although 3D positional data is lost during mono playback, reverberation, Doppler, and

distance cues are still present to some degree.

public native void SprCleanUp()

This method is responsible for releasing any resources used by the current SprocketPlug instance (e.g. the listener and sound sources).

The next group of methods provides general SprocketPlug controls which are as follows:

public native void SprPlayLoop(int chanID, int soundID)

Plays a looped sound on the channel specified by `chanID`. The sound played is determined by `soundID`, which is an index into the current SprocketPlug theme file.

public native void SprStopLoop(int chanID, int nowFlag)

Stops playback of a looped sound on the channel specified by `chanID`. The `nowFlag` determines whether to stop the sound immediately (a value of 1), or to stop the sound when the current loop finishes (a value of 0).

public native void SprPlayOneShot(int chanID, int soundID)

Plays a sound on the channel specified by `chanID`. The sound played is determined by `soundID`, which is an index into the current SprocketPlug theme file.

public native void SprStop(int chanID)

Immediately stops the playback of a sound on the channel specified by `chanID`.

public native void SprSetVolume(int chanID, int volume)

Sets the playback volume of the channel specified by chanID to the value of `volume` (must be in the range of 0 - 255).

public native void SprSetRateMult(int chanID, float rateMult)

Multiplies the rate of playback on the channel specified by chanID by a factor specified by rateMult. This affects both the time and pitch of the currently playing sound. Normal playback speed is represented by the value of 1. Higher values increase the playback speed, while lower values slow it down. For example, a value of 2 doubles the playback speed, a value of 0.5 slows it down in half, and a value of 0 essentially pauses playback. Legal rateMult values are in the range of 0 - 200.

public native void SprSetTheme(String themeName)

Closes the currently open SprocketPlug theme (if any) and opens the theme specified by themeName. If the new theme is not found, SprocketPlug's default theme is used instead.

public native void SprSetRoom(int chanID, float roomSize, float roomReflectivity, float reverbAttenuation)

Sets the room reverberation parameters of the virtual environment maintained by SprocketPlug. roomSize specifies the distance in meters between reverberant walls of the environment, roomReflectivity specifies the amount of attenuation (in dB) each time a sound bounces of a reverberant wall, and reverbAttenuation specifies the amount of attenuation (in dB) of the reverberated signal in the final output. roomReflectivity and reverbAttenuation must be equal to or less than 0.0. Although reverberation parameters affect all channels, a channel number needs to be specified to maintain consistency with the Apple's low-level SoundSprocket API.

SprocketPlug can be set to operate in one of two modes: low-level and high-level. These make use of SoundSprocket's own low-level and high-level API. When using the low-level methods, all sound source positions are specified using listener-relative polar coordinates (i.e. azimuth and elevation angles). Distances are specified in meters. If SprocketPlug is set to use the

high-level calls, all locations are specified using Cartesian coordinates. Besides these differences, the high-level mode of operation allows the user to specify the size of a sound source in terms of length, width, and height. In addition, the relative distances and velocities between sound sources and the listener are calculated by SprocketPlug with a single call to the high-level method `Spr_HiLvlSourceCalcLocalization`. In contrast, this information needs to be set explicitly if SprocketPlug is set to use low-level methods. Note that low-level methods can not be called if SprocketPlug is set to high-level mode and vice versa. The following are the low-level methods:

`public native void Spr_LoLvlSetPosition``(int chanID, float azimuth, float elevation)`

> `azimuth` and `elevation` are the azimuth and elevation angles in degrees used to position the sound playing or to be played on the channel specified by `chanID`. Positive azimuth values specify positions to the right of the listener, while negative values specify positions on the left.Positive elevation angles specify positions above the listener, while negative values specify positions below the listener. The default position of a sound source is at 0 degrees azimuth and 0 degrees elevation (i.e. straight ahead of the listener).

`public native void Spr_LoLvlSetDistance``(int chanID, float distance)`

> This method sets the distance (in meters) specified by `distance`, of the sound source playing or to be played on the channel specified by `chanID`. The default distance value is 1 meter.

`public native void Spr_LoLvlSetReferenceDistance``(int chanID, float refDistance)`

> The reference distance specified by `refDistance` is the distance from the listener at which the sound source was recorded. This method sets the reference distance of the sound playing or to be played on the channel specified by `chanID`. The value for the reference distance must be greater than 0.0. The default reference

distance value is 1 meter.

public native void Spr_LoLvlSetProjectionAngle(int chanID, float projectionAngle)

This method sets the cosine of the angle between the sound source's attenuation cone axis and the vector from the source to the listener. A value of 1.0 indicates that the attenuation cone points directly at the listener, since the cosine of 0 equals 1. The value specified by `projectionAngle` sets the projection angle of the sound playing or to be played on the channel specified by `chanID`. The default value is 1.

public native void Spr_LoLvlSetConeAngleCosine(int chanID, float coneAngleCosine)

`coneAngleCosine` specifies half of the cosine of the angle at the apex of the sound source's attenuation cone. This value is set for the sound playing or to be played on the channel specified by `chanID`. The default value is 1.

public native void Spr_LoLvlSetConeAttenuation(int chanID, float coneAttenuation)

This method sets the amount of attenuation (in dB), specified by `coneAttenuation`, occurring outside of the angular attenuation cone of the sound playing or to be played on the channel specified by `chanID`. The default value is 0.0 attenuation.

public native void Spr_LoLvlSetSourceVelocity(int chanID, float sourceVelocity)

This method sets the velocity in meters per second, specified by `sourceVelocity`, of the sound source playing or to be played on the channel specified by `chanID`. The default value is 0.0.

public native void Spr_LoLvlSetListenerVelocity(int chanID, float listenerVelocity)

This method sets the velocity in meters per second, specified by `listenerVelocity`, of the listener along the vector from the listener to the sound source on the channel specified by `chanID`. If there is more than one sound source in the virtual audio environment, this method needs to be called for each sound (i.e. each channel). The default value is 0.0.

The following code example pans a sound source from the listener's right to left in increments of 5 degrees using SprocketPlug's low-level method calls.

```
localSprocketPlug.SprPlayLoop(1, 1);
(for newAzimuth = 90.0f; newAzimuth >= -90.0f; newAzimuth -= 5.0f)
{
     localSprocketPlug.Spr_LoLvlSetPosition(1, newAzimuth,
                                            0.0f);
}
```

In this example, the sound source moves in arc in front of the listener. To incorporate a Doppler effect, distance and velocity values need to be set using the `Spr_LoLvlSetDistance()` and `Spr_LoLvlSetSourceVelocity()` methods.

As mentioned previously, the main difference between the low-level API and the high-level API is that the high-level methods take Cartesian coordinates while the low-level methods take polar coordinates to specify positions in three-dimensional space. In addition, while low-level methods directly set the velocity and positional data of sound sources relative to the listener, when using the high-level routines, `Spr_HiLvlSourceCalcLocalization()` must be called to have SprocketPlug calculate these relative values. `Spr_HiLvlSourceSetInfo()` then needs to be called to send these calculations to the specified channel. All positional values of the listener and sound sources are in listener units set

with the `Spr_HiLvlListenerSetMetersPerUnit()` method.

The following are the methods available when SprocketPlug is set to high-level mode:

public native void Spr_HiLvlListenerSetPosition`(float x, float y, float z)`

This method sets the position of the listener, specified by the values of `x, y,` and `z`. The default position of the listener is at the origin (0, 0, 0), looking straight down the x axis.

public native void Spr_HiLvlListenerSetOrientation`(float x, float y, float z)`

This method sets the orientation vector of the listener specified by the values of `x, y,` and `z`. This is the unit vector which points forward from the listener's position. The default orientation vector is the unit x vector (1, 0, 0).

public native void Spr_HiLvlListenerSetUpVector`(float x, float y, float z)`

This method sets the up vector of the listener specified by the values of `x, y,` and `z`. This is the unit vector which points straight up from the listener's position. The default up vector is the unit y vector (0, 1, 0).

public native void Spr_HiLvlListenerSetVelocity`(float x, float y, float z)`

This method sets the velocity vector of the listener specified by the values of `x, y,` and `z`. The actual velocity of the listener is calculated when the `Spr_HiLvlSourceCalcLocalization()` method is called. Velocity is specified in listener units per second where the listener units are set by the `Spr_HiLvlListenerSetMetersPerUnit()` method. If this method is not called, SprocketPlug automatically computes the relative velocity values of a moving listener between successive calls to

`Spr_HiLvlSourceCalcLocalization()`. The default velocity vector is (0, 0, 0).

public native void Spr_HiLvlListenerSetMetersPerUnit(float metersPerUnit)

`metersPerUnit` specifies the number of meters per listener unit to use for subsequent calculations. For example, calling this method with a `metersPerUnit` of 0.3408 would set one listener unit to be equal to foot. The default listener unit is 1 meter.

public native void Spr_HiLvlSourceSetPosition(int chanID, float x, float y, float z)

This method sets the position, specified by the values of `x`, `y`, and `z`, of the sound source playing or to be played on the channel specified by `chanID`. The default position of a sound source is at the origin (0, 0, 0) looking down the x axis.

public native void Spr_HiLvlSourceSetReferenceDistance(int chanID, float refDistance);

The reference distance specified by `refDistance` is the distance (in listener units) from the listener at which the sound source was recorded. This method sets the reference distance of the sound playing or to be played on the channel specified by `chanID`. The value for the reference distance must be equal to or greater than 0.0. The default reference distance is one listener unit.

public native void Spr_HiLvlSourceSetOrientation(int chanID, float x, float y, float z)

This method sets the orientation vector, specified by the values of `x`, `y`, and `z`, of the sound source playing or to be played on the channel specified by `chanID`. This is the unit vector which points forward from the sound source's position. The default orientation vector is the unit x vector (1, 0, 0).

public native void Spr_HiLvlSourceSetUpVector(int chanID, float x, float y, float z)

This method sets the up vector, specified by the values of x, y, and z, of the sound source playing or to be played on the channel specified by chanID. This is the unit vector which points straight up from the sound source's position. The default up vector is the unit y vector (0, 1, 0).

public native void Spr_HiLvlSourceSetVelocity(int chanID, float x, float y, float z);

This method sets the velocity vector, specified by the values of x, y, z, of the sound source playing or to be played on the channel specified by chanID. The actual velocity of the sound source is calculated when the Spr_HiLvlSourceCalcLocalization() method is called. Velocity is specified in listener units per second. If this method is not called, SprocketPlug automatically computes the relative velocity values of a moving sound source between successive calls to Spr_HiLvlSourceCalcLocalization(). The default velocity vector is (0, 0, 0).

public native void Spr_HiLvlSourceSetSize(int chanID, float length, float width, float height);

The length, width, and height values are used to set the size of the sound source playing or to be played on the channel specified by chanID. The default size values are (0, 0, 0)

public native void Spr_HiLvlSourceSetAngularAttenuation(int chanID, float coneAngle, float coneAttenuation);

The coneAngle and coneAttenuation values are used to set the angle of the apex of the angular attenuation cone and the amount of attenuation for the sound source playing or to be played on the channel specified by chanID. Angles are specified in radians and should be between 0 and 2π. Attenuation is specified in dB. The default angle is 2π, while the default attenuation is 0 dB (i.e. no

attenuation in any direction).

public native void Spr_HiLvlSourceCalcLocalization(int chanID)

This method is used to tell SprocketPlug to calculate the relative positions and velocities of the listener and the sound source playing or to be played on the channel specified by `chanID`. Specifically, this method should be called after setting the position, up vector, orientation vector, and/or velocities of the listener and/or sound source(s).

public native void Spr_HiLvlSourceSetInfo(int chanID)

Although the `Spr_HiLvlSourceCalcLocalization()` performs the necessary calculations of any recently made positional and velocity changes, the `Spr_HiLvlSourceSetInfo()` method must be called for any of these changes to be sent to the channel for processing.

The following code example uses high-level SprocketPlug calls to move a sound from the listener's right to left in increments of 0.1 listener units:

```
localSprocketPlug.SprPlayLoop(1, 1);
(for newZ = 5.0; newZ >= -5.0;  newZ -= 0.1)
{
     localSprocketPlug.Spr_HiLvlSourceSetPosition(1, 1.0f,
          0.0f, newZ);
     localSprocketPlug.Spr_HiLvlSourceCalcLocalization(1);
     localSprocketPlug.Spr_HiLvlSourceSetInfo(1);
}
```

In the above example, SprocketPlug processing includes positional filtering, distance attenuation, and pitch modulation due to the Doppler effect. To cancel the Doppler effect, sound source velocity should be set to 0 by placing the call `Spr_HiLvlSourceSetVelocity`(1, 0.0f, 0.0f, 0.0f) right before the call

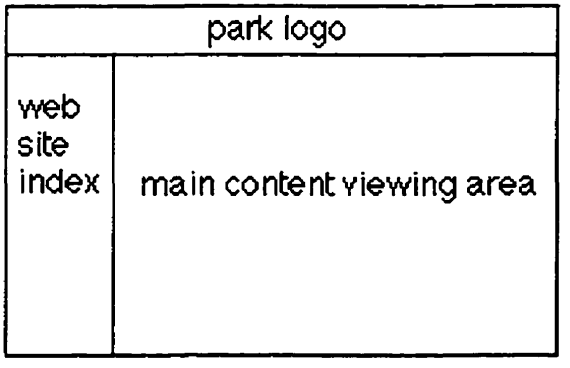
to `Spr_HiLvlSourceCalcLocalization()`.
Note that `Spr_HiLvlSourceSetInfo(1)` must be called to actually process the sound playing on channel 1 with the most recent SprocketPlug calculations.

5.5 SprocketPlug Demonstrations

5.5.1 Enhancement of A Standard Web Site

The first demonstration involves the use of SprocketPlug as a background audio player in a conventional HTML document setting (i.e. no Javascript or Java applets). A WWW site was designed for a fictitious provincial park where online users can obtain general information about the park, and the various services and activities offered to visitors. The WWW site layout consists of three frames contained in a single browser window. The top window was used to display the park's logo. The left frame provides an index (hypertext links) to various areas of the park's WWW site. The frame on the right is the main content viewing area. This layout is depicted in figure 5-6.

figure 5-6 - window layout



An instance of the SprocketPlug plug-in is embedded in the index frame as well as in each HTML page loaded into the main content viewing area.

When the site is first visited, the SprocketPlug instance in the index frame begins playing a looped background sound. The sound is that of a typical nature scene made up of chirping birds, a quietly flowing river, and the soft rustling of trees. A short non-looping melodic fragment is played as well. These sounds are meant to set an appropriate mood and invoke a sense of being out in nature. As each area of the site is visited, various sounds are played to enhance the information displayed in the main content area. For example, when the user visits the page describing the park's wolf howling expeditions, wolf howls are heard in the background. Similarly, when the page describing canoe trips is accessed, paddling sounds are played. Another area serves as the park's special bulletin area and has a news-theme type sound associated with it. All these sounds are spatialized, creating an enveloping sound field around the listener.

This SprocketPlug demonstration is similar to the common use of embedded MIDI files as background music for WWW pages. Although MIDI music may be appropriate in some cases (e.g. used as a jingles for WWW advertising, school theme songs, etc.), this demonstration attempts to illustrate how SprocketPlug may be used in creating more immersive environments on the WWW by incorporating real-world sounds.

5.5.2 Sonification of User Events With SprocketPlug and Javascript

A second SprocketPlug demonstration was built to demonstrate how various user interface elements common in WWW pages may be sonified. This demonstration uses the same page layout as the the first demonstration in creating a fictitious home page: the top frame contains the site's title, the left frame contains a graphic navigation bar (made up of a vertical row of buttons), and the third frame is used as the main content area.

A key feature of this demonstration is the sonification of user events. When the user passes the mouse over various areas of the graphic navigation bar, the buttons change their visual appearance and short thumping sounds are heard coming from the listener's left side. By clicking on a given button, a

sound is heard panning from left to right as a page is loaded into the main viewing area. Passing the mouse over hypertext links in the main document viewing area is also accompanied by sounds. Different types of links are accompanied by various sounds indicating the linked document's type and relative size. For example, links to text files sound like sheets of paper being shuffled, while links to large text documents sound like pages flipping in a big book. Other sounds used include camera snapshots for picture files, and metallic containers for files available for download (e.g. software).

5.5.2 An Auditory Display Component for a Java Applet

Work undertaken by Pennycook, Breder, and Dawkins (1996) involved the development of an auditory display component for the Java application called Merz. The following is a description of the Merz project:

> Merz is an environment written in Java for the WWW which supports personal information management and knowledge work. Merz information visualization emphasizes personalization of views, the rapid generation of multiple views of information through zooming and filtering, and smooth transition between views under user control. Moreover, Merz aims at integrating automatic processes, "agents," into the visualization environment in order to aid tasks like querying and the monitoring of information.
> (online document at
> http://www.merzcom.com/eng/products/products.html)

The auditory display component, called the Merz Soundscape, developed by Pennycook *et al.* consisted of native code libraries (for Macintosh PowerPC and Pentium Windows 95 systems) integrated with the Merz Java application. These libraries were responsible for handling the sonification of various elements of the Merz visual display including onscreen information representation, the state of background automatic processes, and user interaction. At the time however, there was no solution to provide this kind of functionality for the version of Merz running as a Java applet; Java applets (as specified in the original Java specifications from Sun Corp.) could not be integrated with native methods.

As part this thesis, a third demonstration was created to show how SprocketPlug can provide the Merz Java applet with the missing auditory display functionality. All the original Merz Soundscape functionality described by Pennycook, Breder, and Dawkins (1996) is now accessible to the Merz Java applet including real-time auditory spatialization of simultaneously occurring sound streams (e.g. background looping sounds and foreground user events), and the use of specially designed Soundscape theme files (i.e. SprocketPlug themes). While the functionality is there, SprocketPlug has not been integrated and tested with the actual Merz Java applet.

5.6 Critical Assessment

Auditory display research reviewed in this thesis has shown that human-computer interactions can be enhanced with sound. This involves the design and implementation of auditory display components which are integrated with the systems they serve. SprocketPlug was designed with these considerations in mind. The demonstrations of the thesis software have shown that SprocketPlug can be easily incorporated with current WWW development environments.

However, SprocketPlug is not a complete solution. In particular, sounds are manipulated by SprocketPlug in terms of their psychophysical attributes such as pitch (or rate of playback) and amplitude, rather than with more general auditory display parameters such as object material, object size, and surface impact (Gaver 1994).

Formal studies on the effectiveness of auditory displays for the WWW have not been undertaken largely due to a lack of supporting tools. SprocketPlug addresses this issue by providing the means to incorporate auditory display techniques, if only for testing purposes. The demonstrations described above are informal assessments, and while interesting, more research is required to evaluate how useful sonically enhanced WWW applications are from the standpoint of end-users.

Another limiting factor of SprocketPlug as a general auditory display component for the WWW is its platform-dependance. While a great part of

the success and popularity of the WWW is the fact that it provides communication across a multi-platform network, SprocketPlug is based on native Apple Sound Sprockets technology, available only on the Macintosh PowerPC platform. This requires WWW developers to include "platform checking" routines in their otherwise platform-independent code in order to take advantage of the SprocketPlug plug-in.

Related to the issue of platform-dependence is the fact that SprocketPlug functionality is only available to Java applets running under Netscape's Java Runtime Interface environment (JRI). While Netscape has made its technology available to most platforms, not all WWW clients are running Netscape software. Unless other WWW client software is made to be Netscape compliant, JRI and LiveConnect technologies (and hence SprocketPlug) will be unavailable to clients running non-Netscape software.

Finally, SprocketPlug was designed to interface with Java, and it is yet to be determined to what degree Java will play a role in WWW application development. Java is a robust language, but due to its abstraction layers and interpreted nature, Java code tends to execute much slower than compiled native code. However, the advent of Just-In Time compilers and specialized Java chips (e.g. Pico chips from Sun Corp.) are likely to bring Java performance up to the speeds of applications written in C/C++. While Java has become widely recognized and adopted by many developers, there are several competing technologies, most notably Microsoft's ActiveX, which may adversely impact Java acceptance by WWW developers.

Programming efforts are moving away from hardware platforms to "software platforms" distributed via the WWW. Which "software platform" will become the dominant one, as well as the most viable solution for WWW auditory displays, remains to be seen.

6. CONCLUSION

Based on auditory display research, this thesis has suggested that auditory display systems in conjunction with WWW browsers could significantly enhance information space and pleasure of using an essentially visual medium. Due to a lack of tools however, sound has not yet become an integrated component of the user experience.

The software developed as part of this thesis, SprocketPlug, incorporates many of the currently available technologies, to provide elaborate interactive sound services to WWW developers. Although not a final solution, SprocketPlug was demonstrated as effective in implementing various auditory display techniques at the three most popular levels of WWW development: HTML, Javascript, and Java.

BIBLIOGRAPHY

Albers, M., and E. Bergman 1994. "The Audible Web: Auditory Enhancements for WWW Browsers."http://www.isye.gatech.edu/chmsr/Mike_Albers/papers/www/www-AW.html

Apple Computer, Inc. 1996. *Game Sprockets Guide.* Reading, MA: Addison-Wesley.

Apple Computer, Inc. 1994. *Inside Macintosh : Sound.* Reading, MA: Addison-Wesley.

Asano, F., Y. Suzuki, , and T. Sone 1990. "Role of Spectral Cues in Median Plane Localization." *Journal of the Acoustical Society of America,* 88(1), p.159-168.

Ballas, J. A. 1994. "Delivery of Information Through Sound." *Auditory Display: Sonification, Audification, and Auditory Interfaces.* Reading, MA: Addison-Wesley.

Bargar, R. 1994. "Pattern and Reference in Auditory Display." *Auditory Display: Sonification, Audification, and Auditory Interfaces.* Reading, MA: Addison-Wesley.

Begault, D. R. 1991. "Challenges to the Successful Implementations of 3-D Sound."*Journal of the Audio Engineering Society,* 39(1), 864-870.

Begault, D. R. 1993. "Head-Up Auditory Display for Traffic Collision Avoidance System Advisors: A Preliminary Investigation." *Human Factors,* 35(4), p. 707-717.

Begault, D. R. 1994. *3-D Sound for Virtual Reality and Multimedia.* San Diego, CA: Academic Press.

Blattner, M. M.,D. A. Sumikawa, and R. M. Greenberg 1989. "Earcons and Icons: Their Structure and Common Design Principles." *Human-Computer Ineraction,* 4(1), p. 11-44.

Blattner, M. M., A. L. Papp, and E. P. Glinert 1994. "Sonic Enhancement of Two Dimensional Graphic Displays." *Auditory Display: Sonification, Audification, and Auditory Interfaces.* Reading, MA: Addison-Wesley.

Bly, S. 1994. "Multivariate Data Mappings." *Auditory Display: Sonification, Audification, and Auditory Interfaces.* Reading, MA: Addison-Wesley.

Bregman, A. S. 1990. *Auditory Scene Analysis.* Cambridge, MA: MIT Press.

Brewster S. A., P. C. Wright, A. D. N. Edwards 1994. "A Detailed Investigation into the Effectiveness of Earcons." *Auditory Display: Sonification, Audification, and Auditory Interfaces.* Reading, MA: Addison-Wesley.

Butler, R. and K. Belendiuk 1976. "Spectral Cues Utilized in the Localization of sound in the Median Sagittal Plane." *Journal of the Acoustical Society of America,* 61(5), p.1264-1269.

Buxton, W. 1989. "Introduction to This Special Issue on Nonspeech Audio." *Human-Computer Ineraction,* 4, p. 1-9.

Cherry, E. C. 1953. "Some Experiments on the recognition of Speech with One and Two Ears." *Journal of the Acoustical Society of America,* 25 (1953), p. 975-979.

Cohen, J. 1994. "Monitoring Background Activities." *Auditory Display: Sonification, Audification, and Auditory Interfaces.* Reading, MA: Addison-Wesley.

Doll, T. J., and T. E. Hanna 1995. "Spatial and Spectral Release from Masking in Three-Dimensional Auditory Displays." *Human Factors,* 37(2), p. 341-355.

Eysenck, M. W., and M. T. Keane 1990. *Cognitive Psyochology: A Student's Handbook.* (second edition) London, England: Lawrence Erlbaum Associates.

Fowler, C. A. 1990. "Auditory perception is not special: We see the world, we feel the world, we hear the world." *Journal of the Acoustical Society of America,* 89(6), p. 28910-28915.

Fitch, W. T., and G. Kramer 1994. "Sonifying the Body Electric: Superiority of an Auditory over Visual Display in a Complex, Multivariate System." *Auditory Display: Sonification, Audification, and Auditory Interfaces.* Reading, MA: Addison-Wesley.

Fluckinger, F. 1995. *Understanding Networked Multimedia Applications and Technology.* Englewood Cliff, New Jersey: Prentice Hall. Gardner, M. and R.

Gardner, M. 1972. "Problem of Localization in the Median Plane: Effect of Pinnae Cavity Occlusion." *Journal of the Acoustical Society of America*, 53(2), p.400-480.

Gardner, M. 1973. "Some Monaural and Binaural Facets of Median Plane Localization." *Journal of the Acoustical Society of America*, 54(6), p.1489-1495.

Gaver, W. 1986. "Auditory Icons: Using Sound in Computer Interfaces." *Human-Computer Ineraction*, 2, p. 167-177.

Gaver, W. 1989. "The Sonic Finder, An Interface That Uses Auditory Icons." *Human-Ineraction*, 4, p. 67-94.

Gaver, W. 1994. "Using and Creating Auditory icons." *Auditory Display: Sonification, Audification, and Auditory Interfaces*. Reading, MA: Addison-Wesley.

Hapeshi, K., and D. Jones 1992. "Interactive Multimedia for Instruction: A Cognitive Analysis of the Role of Audition and Vision." *International Journal of Human-Computer Interaction*, 4(1) p. 79-99.

Hiranaka, Y. and H. Yamasaki 1982. "Envelope Representations of Pinnae Impulse Responses Relating To Three-Dimensional Localization of Sound." *Journal of the Acoustical Society of America*, 73(1), p.291-296.

James, F. 1995. "Presenting HTML in Audio: User Satisfaction with Audio Hypertext (Pilot Experiment)." http://www-pcd.stanford.edu /~fjames/pilot/

Jameson, D. 1994. "Sonnet: Audio-Enhanced Monitoring and Debugging." *Auditory Display: Sonification, Audification, and Auditory Interfaces*. Reading, MA: Addison-Wesley.

Kendall, G. S. 1996. "A 3D Sound Primer." *Computer Music Journal*, 19(4), p. 23-46.

Kendall, G. S. and W. Martens. 1984. "Simulating the Cues of Spatial Hearing in Natural Environments." *Proceedings of the 1984 International Computer Music Conference*. San Fransisco: International Computer Music Association.

Kramer, G. 1994a. "Some Organizing Principles for Representing data with Sound." *Auditory Display: Sonification, Audification, and Auditory Interfaces*. Reading, MA: Addison-Wesley.

Kramer, G. 1994b. "An Introduction to Auditory Display." *Auditory Display: Sonification, Audification, and Auditory Interfaces*. Reading, MA: Addison-Wesley.

Kramer, G., ed. 1994c. *Auditory Display: Sonification, Audification, and Auditory Interfaces*. Reading, MA: Addison-Wesley.

Lombardi, V. 1995. *Audio and the Internet*. http://www.nyu.edu/ncm/library/papers/internet.audio

Madhyastha, T. M., and D. Reed 1994. "A Framework for Sonification Design." *Auditory Display: Sonification, Audification, and Auditory Interfaces*. Reading, MA: Addison-Wesley.

McCabe, K., and A. Rangwalla 1994. "Auditory Display of Computational Fluid Dynamics Data." *Auditory Display: Sonification, Audification, and Auditory Interfaces*. Reading, MA: Addison-Wesley.

MerzCom, Inc. 1996. "Products: NetScope." http://www.merzcom.com/eng/products/products.html

Moore, B. 1982. *An Introduction to the Psychology of Hearing*. (second edition) London, England: Academic Press.

Moore, F. Richard 1990. *Elements of Computer Music*. Englewood, New Jersey: Prentice Hall.

Neilsen, J. 1996. "Relationships on the Web." http://www.useit.com/alertbox/9601.html

Neilsen, J. 1996. "The Internet desktop." http://www.useit.com/alertbox/9603.html

Neilsen, J. 1996. "Features for the Next Generation of Web Browsers." http://www.sun.com/950701/columns/alertbox/

Pennycook, B. "The Merz Soundscape." http://www.music.mcgill.ca:80/~brp/merzscape.html.

Pennycook, B., E. Breder, and K. Dawkins 1996. "The Merz Soundscape Project." Final report submitted to Michael Century at the Center for Information Technology Innovation. Laval, Canada.

Perrot, D. R., K. Saberi, and T. Z. Strybel 1990. "Aurally Aided Visusal Search in the Central Visual Field: Effects of Visual Loads and Visual Enhancement of the Target." *Human Factors*, 33, p. 389-400.

Plenge, G. 1972. "On the Differences Between Localization and Lateralization." *Journal of the Acoustical Society of America*, 56(3), p. 944-951.

Rossing, D. 1990. *The Science of Sound*.. (second edition) Reading, MA: Addison-Wesley.

Scaletti, C. 1994. "Sound Synthesis Algorithms for Auditory Data Representation." *Auditory Display: Sonification, Audification, and Auditory Interfaces*. Reading, MA: Addison-Wesley.

Sekuler, R., and R. Blake 1990. *Perception*. (second edition) New York, NY: McGraw-Hill.

Seligman, D., R. Mercuri,and J. T. Edmark 1995. "Providing Assurances in a Multimedia Interactive Environment." *Computer-Human Interactions '95 Proceedings*.

Schmandt, C., and A. Mullins 1995. "AudioStreamer: Exploiting Simultaneity for Listening." *Computer-Human Interactions '95 Proceedings*.

Sikora, C. A., L. Roberts, and L. T. Murray 1995. "Musical vs. Real World Feedback Signals." *Computer-Human Interactions '95 Proceedings*.

Smith, S., R. M. Pickett, M. G. Williams 1994. "Environments for Exploring Auditory Representations of Multidimnesional Data." *Auditory Display: Sonification, Audification, and Auditory Interfaces*. Reading, MA: Addison-Wesley.

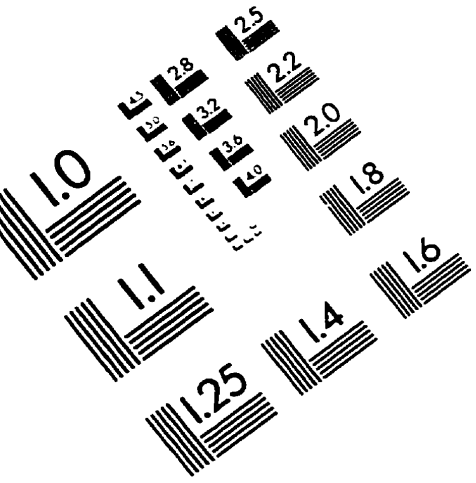
Sorkin, R. D., D. S. Kistler, and G. C. Elvers 1989. "An Exploratory Study of the Use of Movement-Correlated Cues in an Auditory Head-Up Display." *Human Factors*, 31(2), p.161-166.

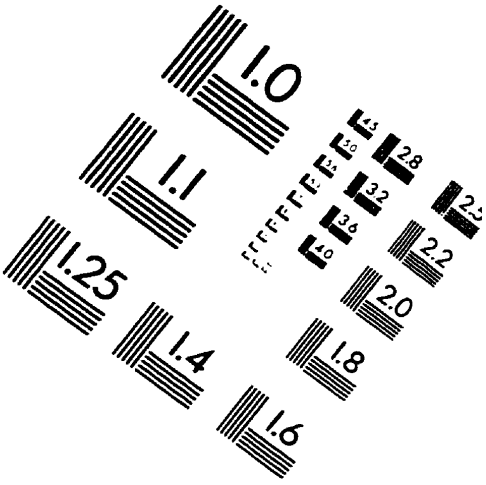
Wenzel, E. 1994. "Spatial Sound and Sonification." *Auditory Display: Sonification, Audification, and Auditory Interfaces*. Reading, MA: Addison-Wesley.

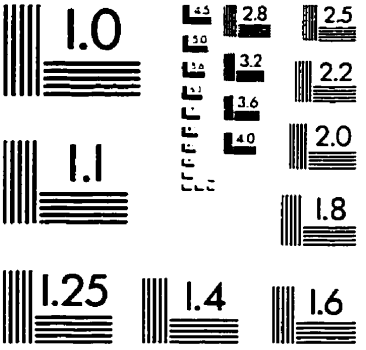
Williams, S. M. 1994. "Perceptual Principles in Sound Grouping." *Auditory Display: Sonification, Audification, and Auditory Interfaces*. Reading, MA: Addison-Wesley.

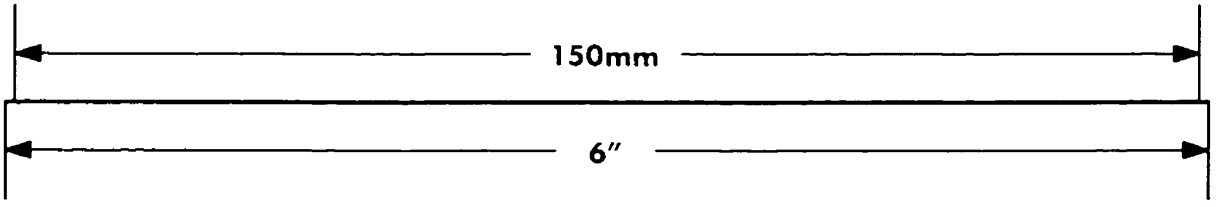
Young, D. A. 1997. *Netscape Developer's Guide To Plug-ins*. Upper Saddle River, NJ: Prentice Hall.

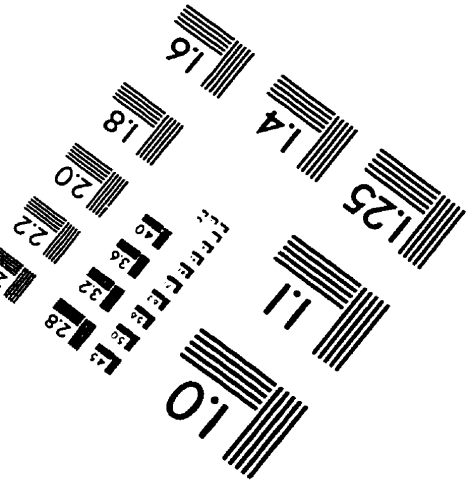
IMAGE EVALUATION TEST TARGET (QA-3)

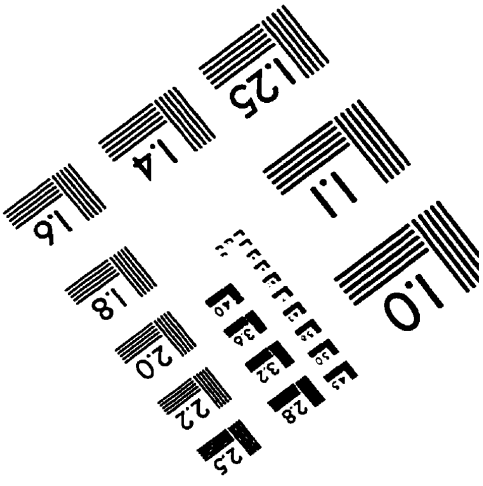












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989