Using sound to extract meaning from complex data

Carla Scaletti

University of Illinois, CERL
Urbana IL 61801-2977 USA
scaletti@cerl.uiuc.edu
(217) 333-0766

Alan B. Craig

University of Illinois, NCSA
Champaign IL 61820 USA
acraig@ncsa.uiuc.edu
(217) 244-1988

ABSTRACT

In analyzing abstract data sets, it is useful to represent them in several alternative formats, each format bringing out different aspects of the data. While most work in data mapping has focused on visual representations, we have found that sonic representations can also be effective aids in interpreting complex data, especially when sonification is used in conjunction with visualization.

We have developed prototypes for several high-level sonification tools that can be applied to a wide variety of data. While we used programmable multi-processor digital signal processing hardware to develop and experiment with these prototypes, each of these tools could be implemented as special-purpose hardware or software for use by scientists in specific applications. Our prototype tools include: Mapper (maps data to various sonic parameters), Comparator (feeds a different mapping into each speaker channel), Sonic Histogram (maps the magnitude of each category onto the amplitude of its associated sound), Shifter (shifts signals into the audible range), Marker (a sonic alarm that marks a specific condition).

We tested these tools by using them to generate data-driven sound tracks for video animations generated from the same data. The resulting videos provide an increased data bandwidth and an increased sense of virtual reality to the viewer.

2. INTRODUCTION

2.1. Defining the problem

Our goal is to transfer information between two sophisticated, specialized processors; one is electronic and of recent design, and the other is biological and has been undergoing strenuous testing and refinement for several million years. Modelling and analysis programs on the electronic computer are capable of generating vast amounts of data, though not necessarily in an optimal input form for the biological computer. Our job is to structure this data in ways that will improve the performance of the human observer in interpretation and analysis.

There are five input ports to the biological processor: the eyes, the ears, the nose, the tongue and the skin. The first of these perceptual modes, vision, has received the most attention. We decided to investigate the auditory system and its suitability as an additional input channel.

2.2. Why use sound?

There are several reasons for investigating the use of sound in data interpretation and analysis:

- It adds additional bandwidth to a purely visual interface
- It increases the user's sense of engagement with the model world
- People have come to expect sound to accompany animated images
- The auditory system excels at certain tasks

2.2.1. Bandwidth, engagement, and expectation

The auditory system can provide an alternative input channel alleviating visual overload or clutter.[1,2] Visual and auditory representations can be used together to reinforce information and provide a greater sense of engagement with the model world.[2,3]

Our initial project was to add data-driven sound to videos of scientific visualizations generated at NCSA. In the context of watching a video, people have come to expect a sound track and may even feel uncomfortable in its absence. Rather than

adding an arbitrary musical sound track (which may add no information or even contradict the visual information) we decided to add *data-driven* sound—sound that had some chance of enhancing, reinforcing, or even adding to the information presented in the visual portion.

2.2.2. Unique attributes of the auditory system

Unlike the eyes, which have closeable lids, the ears are always open channels. As such they can serve as an early warning system, operating in the background without requiring our full attention. (Consider the fact that you can be awakened from sleep by an unusual sound or even an unusual absence of sound.) Gaver exploits this attribute of the auditory system by constructing auditory icons that do not require the direct visual attention of the user.[2] Wenzel points out that hearing can serve an alerting or orienting function.[3,4] As she puts it, "The ears tell eyes where to look."

Bly maintains that time-varying and logarithmic data are probably better understood through the ears than through the eyes.[5] Both Bly and Gaver suggest that multidimensional data might be better represented sonically than visually.[2,5] Mezrich et al showed that a dynamic sonic and visual representation could assist people in analyzing an eight-variable economic indicator.[6]

The auditory system possesses a remarkable ability to detect small interaural timing differences, registered subjectively as changes in the apparent location of a sound source. More generally, the ability of auditory system to *compare* two streams of data presented to the left and right ears through headphones is one that could be exploited in data sonification.

We speculate, although we have no objective evidence for these ideas, that a sonic representation may be a more effective aid than a visual representation for identifying and remembering periodic patterns, and that the strong associative memories evoked by music may indicate that the way in which we remember sonic patterns differs from the way we remember visual patterns.

2.3. Previous work

There has been an increasing interest in the possibilities of representing data as sound over the past ten years[5], and this interest has grown as improvements in the technology of sound synthesis have made it possible to generate digital audio signals in real time. Work in this area falls roughly into two categories: the use of sound to augment the general computer/human interface, and the use of sound as a tool for the specific task of interpreting and analyzing data.

2.3.1. Computer human interface

Gaver's well-known Sonic Finder adds auditory icons, created primarily from digital audio recordings, to the Macintosh Finder.[2] The "earcons" described by Blattner are created by hierarchically combining elemental "motives" to create new meanings.[1] Edwards[7], Mansur[5], and Morrison[5] describe the development of auditory interfaces for visually impaired computer users.

2.3.2. Data sonification

More closely related to the work described in this paper are those projects addressing the general problem of how sound can be used to assist the human analyst in the interpretation of a wide variety of data.

Bly mapped n-tuples of data to the frequency, amplitude, duration, envelope, and waveshape parameters of a single, discrete sound or note and tested this mapping for its effectiveness in conveying information to 75 subjects. She demonstrated that subjects were better able to correctly categorize n-tuples into one of two categories when they were presented with *both* sonic and graphic representations than when they were presented with either one separately.[8]

Wenzel's work centers on the creation of virtual acoustic environments for visualizing multidimensional data. She uses special-purpose hardware to implement FIR filters simulating the effects of the pinna on data sources at 144 different locations. The virtual acoustic environment is part of the NASA-Ames VIEW system, a multisensory environment that allows users to explore and interact with a three-dimensional synthesized or remotely sensed world. In this system, an auditory cue or icon is designed offline by assigning it a synthesizer patch number and a localization method; during the simulation, cues can be triggered and parameters updated via MIDI continuous controllers.[3,4]

In the Exvis project at the University of Lowell, Smith uses sounds as one-dimensional probes of a two-dimensional image, addressing the problem of how to represent fields or multidimensional data using an inherently one-dimensional signal. In the Lowell perceptualization workstation, each visual icon has a discrete auditory component realized using MIDI synthesizers. Using the mouse, a user can probe the two-dimensional field of visual icons (as if using a metal detector or a Geiger counter), hearing the corresponding auditory icons as the visual icon is scanned.[9]

2.4. Problems facing sonification

In an insightful critique based on his years of experience and experimentation in the field, Stuart Smith has identified three impediments to further progress in the area of sonification research:[10]

- Arbitrary mappings of data to sound parameters without regard for any "natural" connection to the data represented
- Lack of general purpose real-time sound synthesis hardware
- An absence of models that would allow sonification systems to run on several different hardware platforms, and a lack of a theory of timbre perception and generation

We have tried, with varying degrees of success, to address these problems in our sonification project. We experimented with both abstract and data-related mappings, we chose to do software sound synthesis on general-purpose digital signal processing (DSP) hardware rather than doing hardware sound synthesis on MIDI synthesizers, and we attempted to address the issue of portability and a theory of timbre through the use of a stream-based sound specification language and the development of a set of high-level sonification tools.

3. DATA SONIFICATION PROJECT

3.1. Goals

This was a preliminary investigation, aimed at demonstrating to ourselves and to the NCSA Visualization Group that data sonification could be an effective means for assisting researchers in analyzing and interpreting complex data. Our goal was to come up with a set of standard, high-level sonification tools that could be applied in a variety of contexts; these are the sonic analogs of such standard visualization tools as split-screen comparisons, arrows pointing out areas of importance, histograms, and x-y graphs.

We developed and tested these tools by generating data-driven sound tracks for some of the visualization videos produced by NCSA. Our purpose in generating data-driven sound was to elucidate the data, to convey information as clearly and unambiguously as possible. This sound is not intended to be music, nor is the process of sonification the same as that of algorithmic music composition. Our goal was to evaluate the sonifications primarily on the basis of how well they expressed the data.

Brian Evans has also done some work at NCSA in generating both animation and sound from the same algorithm.[11] While the goal of this work—developing a new, integrated art from—differs from our goal of assisting in data interpretation, it does employ a similar paradigm: the generation of multiple representations from a single, abstract data set.

3.2. Underlying model

Our model is one of a single data set with *multiple* visual and audio representations; each representation brings out different features of the data, and the use of multiple representations helps to compensate for loss of information in any single mapping.

We think of the NCSA videos as live footage of some model world. Just as a video camera filters the *physical* world through a camera and a microphone and stores visual and sonic representations of that world on tape, so the NCSA videos filter some abstract data set to create visual and sonic representations of a *model* world on tape (Figures 1 and 2).

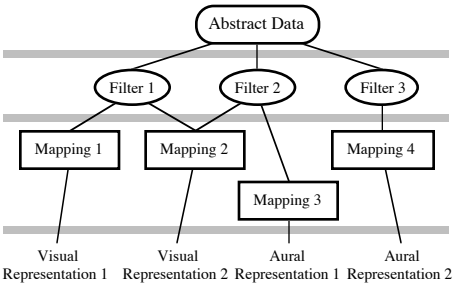

Fig. 1. Multiple representations of a model world.

Sound is a change in air pressure over time. We cannot hear a single air pressure value; we can only detect *changes* in air pressure over time. Since sound is inherently time-varying, we concentrated on representations for *time-varying data* mapped to animated graphics and sound.

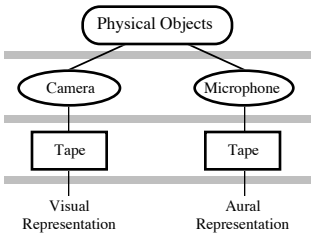

Fig. 2. Multiple representations of the physical world.

3.3. Using sound to convey information

C. S. Peirce defines three ways in which a representation may relate to an object:[12]

- An **icon** is typically mimetic and relates to an object which may or may not exist
- An **index** is causal and is affected by the object which it represents
- A **symbol** has an association with a general *class* of objects, although it appears as a specific *instance* of that class.

Representations can also be some combination of icon, index, and symbol.

Sonic **icons** can range from the literal (as in the digital recordings used by Gaver in his auditory icons[2]) to the abstract (as in the motivic earcons described by Blattner.[1]) While an icon is often mimetic or suggestive of the object it represents, it need not be; it is enough merely to define it as representing that object.

A digital recording would be classified as an **index** in that it is a mapping from a variation in air pressure to numbers encoded as voltage levels on magnetic or optical media. If we define an analogy as a mapping from a domain (the source) into a range (the target) in which the same system of relations that holds between objects in the source is maintained between objects in the target[13] , then we can say that an analogy is another example of an index.

The bell at a railroad crossing is a sonic **symbol** in that it invokes an generally understood association or social convention; likewise, the playing of Taps invokes associations of endings, death and burial. A downwards slide whistle associated with a falling cartoon character is symbolic in that it is a social convention, yet it is also an index in that it maps a *falling* character to a *falling* frequency.

We concentrated on the use of sonic *indices* or analogies, that is mappings from streams of time-varying data to one or more parameters of sound. The steps in designing a sonic index are:

- Design the basic sound
- Decide which parameters are to be controlled by the data stream
- Define a mapping between the data stream and the time-varying sound parameter

There are two kinds of analogy that can be made when designing sonic indices. One is to make the basic sound suggestive of the process you are trying to represent. This is an object analogy. The other is by mapping a process or a relationship between objects to a sonic parameter or relation between parameters. This is a relation analogy.

The advantage of using object analogies is that you can use them to convey additional information about the data being examined; for example, air speed could be mapped to white noise which actually sounds like rushing air. By using the same abstract mapping on a variety of data, however, you can sometimes discover similar processes taking place in totally different contexts. For example, when we mapped the displacement of a damped pendulum and the vertical position of a bouncing ball to the frequency parameter of a simple sine wave, we stumbled upon the fact that the two functions are extremely similar.

In her studies of how human beings can develop understanding through analogy, Gentner found that that adults tend to use analogous *relations*, and that children tend to use analogous *objects*. In time trials, subjects are able to pick up on object analogies faster than relational analogies, and long-term memories were more likely to be retrieved via object similarities than by relational similarities; nevertheless, most of her subjects rated relational analogies as “better” analogies than object analogies.[14] In our data-driven sound tracks, we tried to incorporate both types of analogy at various times.

4. DEVELOPMENT SYSTEM

4.1. Digital sound synthesis

Max Mathews' group at Bell Laboratories was the first to do sound synthesis using computers, developing a series of "acoustic compilers", languages similar to the simulation languages of the time.[15] In essence, these languages were simulations of analog circuits—like oscillators and filters—implemented in software. While these software synthesis languages could not produce sound in real time, their modularity and the fact that all sample generation was done in software made them extremely flexible. Since many of these music languages and their descendents carry names like Music4 and Music5, they are often referred to as a group as the Music N languages.[16]

In MIDI synthesizers, the sound synthesis algorithms are implemented in hardware. Unlike the software synthesis languages, MIDI synthesizers can generate digital audio signals in real time; however, hardware synthesis is not as flexible or customizable as software synthesis.

In the development system we used for the data sonification project, the sound synthesis algorithms are implemented in *software* that runs on several digital signal processors (DSPs) in parallel. DSPs are microprocessors designed specifically for high-speed digital signal processing operations. These DSPs make it possible to have the flexibility of software synthesis *and* the real-time response of MIDI synthesizers.

4.2. Kyma System

To develop our set of prototype tools, we used a general-purpose sound specification system called the Kyma System, developed by the Symbolic Sound Corporation. The hardware component of the Kyma System is the Capybara, a multiprocessor based on nine Motorola 56001 digital signal processors (DSPs) operating in parallel; this hardware allows us to do software sound synthesis in real time.

4.2.1. Data representation

Kyma, the software component of the Kyma System, is a functional, data stream language used to control the Capybara hardware. Kyma was written in ParcPlace Systems Smalltalk-80[17] and provides a graphic interface for interactively designing and testing audio signals.[18,19,20,21,22]

In Kyma, an audio signal is represented as a graphic object called a Sound. Each Sound represents a stream of samples (i.e. instantaneous amplitude values). A Sound may be atomic or it may be a function of one or of several other Sounds. Some Sounds are never heard directly but control the parameters of a Sound that *is* heard.

Each Sound is represented by an icon; Sounds that are functions of other Sounds are represented as graphs. Figure 3, for example, represents the function, **product(aForcedDuration (?aVariableSound), aThreshold (dataStream))**. Data "flows" from the bottom to the top, so for instance each sample from **dataStream** flows through **aThreshold** before reaching the **product** at the top of the graph.

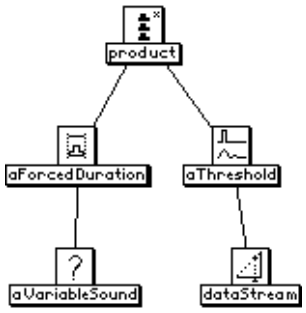

Fig. 3. A Sound as represented in Kyma.

Sounds are uniform; thus any Sound may replace any other in a function. Some Sounds represent synthesis algorithms, others represent the analog-to-digital input, some represent digital recordings stored on the hard disk, some represent filters and other signal processing algorithms, and still others represent external MIDI devices (as streams of MIDI events). Sounds that represent time offsets make Kyma more than a synthesizer "patch designer"; an entire musical composition can be constructed by introducing time offset Sounds.

Since every Sound represents a stream of samples, external data streams can also be treated as Sounds. A Sound representing *external* data is just like any other Sound object and can appear anywhere in a function.

4.2.2. Editing

An editor like the one depicted in Figure 4 is used to change the graph or the parameters of a Sound. The graph of the Sound is displayed on the left, and its parameter values are displayed on the right. Various controllers can be used to alter parameters: text fields, sliders, lists, etc.

4.2.3. Variables

There are two kinds of variables in Kyma: variable parameters within Sounds and variable Sounds. A variable Sound appears as a question mark icon on the left side of the Sound editor; a variable parameter appears as a name preceded by a question mark in one of the fields in the right half of the editor.

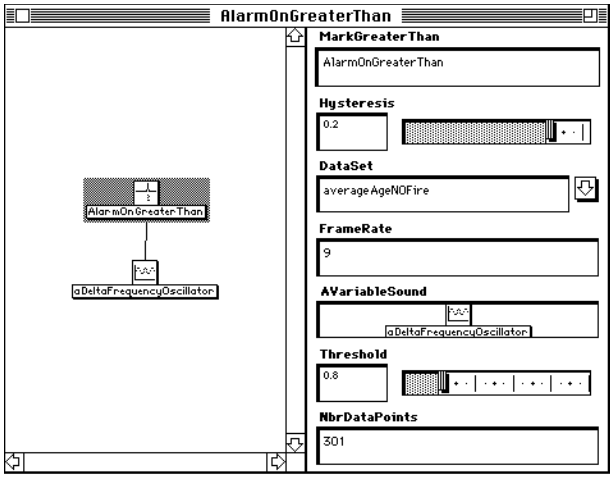

Fig. 4. A Sound editor in Kyma.

5. TOOLS

We used the Kyma System to develop and test prototypes of high-level tools that could be used by researchers to interpret and analyze a variety of data. Elemental and transformational Sound objects can be combined to form new Sound objects; these new Sound objects can then be encapsulated as new *classes* of Sounds. New Sounds can be systematically constructed out of the previously defined Sounds, allowing the user to continue this process of customizing and expanding upon the existing library of Sounds.

5.1. Building blocks

First, we developed a set of building blocks out of which we could then construct higher level tools. In order to treat the external data set as a stream of audio samples, we first scaled it to the integer word length of the signal processor. Then we used a linearly interpolating table-lookup Sound in Kyma to read up these samples at the frame rate of the corresponding animation. This mapping from the data stream to a stream of samples forms the basis for all of the other sonification tools.

5.1.1. Shifter

What we called **Shifter** is the most primitive of the tools; it is simply an interpretation of the data stream as a waveform. This is unlikely to be of much interest except in those cases where the data are quasi-periodic and change slightly over time.

Shifter can be used as a sort of sonic version of night-vision goggles; it can shift signals from the sub or ultra sonic range into the range of human hearing, in effect allowing you to listen to the data stream directly. In fact, when used in conjunction with other Sounds, this tool is simply called **DataStream**.

5.1.2. Mappers

In most cases the **DataStream** is used as a control signal, controlling one or more parameters of another Sound object. We constructed several tools in which the **DataStream** controls various physical parameters of other Sound objects: frequency, change in frequency, amplitude, the cutoff frequency of a low-pass filter, the density of discrete events in time, and stereo position.

Figure 5 is a screen image from Kyma showing the structure of the simplest of these mappers: one in which the amplitude of a Sound is controlled by the **DataStream**.

The DataToAmplitude mapper is the product of the **DataStream** and a variable Sound called ?scaledSound. Once a Sound has been defined it can be encapsulated and abstracted as a new *class* of Sounds. The class editor, shown in Figure 6, allows you to

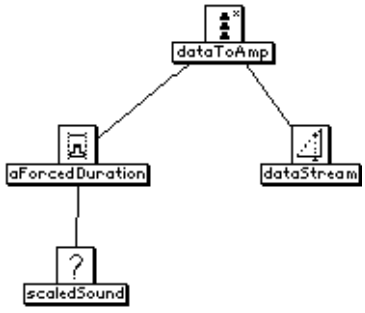

Fig. 5. Data to amplitude mapping.

design a new icon for the class and to define the types and the controllers for its parameter values. Once the class has been defined, all of its internal details are encapsulated, and it appears on the display as a single, atomic icon.

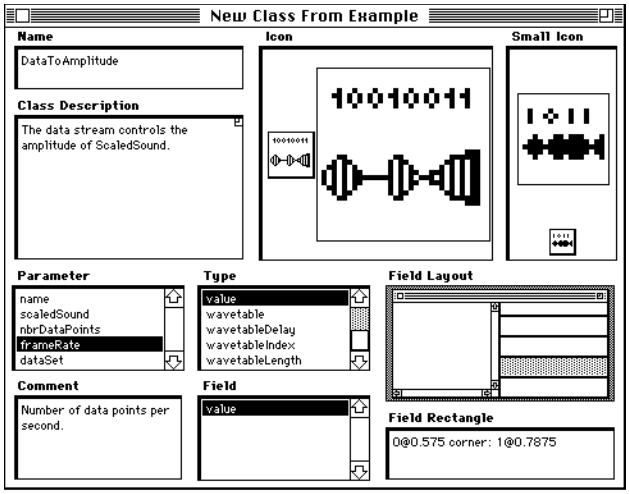

Fig. 6. The class editor in Kyma.

Each of the tools in our collection was constructed in the same manner: first by constructing it out of other Sounds, and then by encapsulating it and defining it as a new class of Sounds having its own icon and graphic editor.

5.1.3. Analyzers

Analyzer Sounds are never heard directly; they extract information from other signals and, directly or indirectly, control the parameters of Sounds that *are* actually heard. Tools in the analyzer category include: **Sign**, **Slope**, and **ZeroCrossing**.

A **ZeroCrossing**, for example, compares each sample in the stream of its argument Sound against each sample of a **Constant** Sound whose value is always zero. Whenever a sample from its argument is zero, the **ZeroCrossing** outputs a one; otherwise its output is zero.

5.1.4. Combiners

Sounds in the combiners category are arithmetic expressions involving two or more sample streams, for example **Sum**, **Difference**, and **Product**. These Sounds are used to construct combinations of other Sounds. In order to specify that two Sounds should occur simultaneously, for example, you would make both of them arguments of a **Sum**; a **Sum** adds its arguments' sample streams, functioning as an audio mixer.

5.2. High-level tools

Any division between the "building blocks" and the "high-level" tools is somewhat arbitrary. All tools were constructed out of previously defined tools, and even the "high-level" tools can be combined to construct even higher-level tools.

5.2.1. Comparator

In order to exploit the auditory system's remarkable ability to detect small interaural differences, we constructed the **Comparator**. The **Comparator** is simply the **Sum** of two other Sounds, one in each stereo channel. By listening to two Sounds simultaneously it is possible to detect the points at which they diverge and to get a sense for the magnitude of the divergence. Figure 7 shows the **Comparator** (encapsulated and with its own icon) being used to compare two **DataToFrequency** mappers.

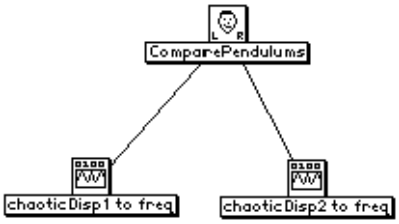

Fig. 7. Comparator.

5.2.2. Marker

Markers exploit the "open channel" nature of the auditory system, the fact that we can hear events or alarms when not actively concentrating on the perceptualization A **Marker** allows you to trigger one Sound when a particular condition, for example a zero-crossing or an exceeded threshold, occurs in another Sound. For example, the **MarkGreaterThan** tool triggers its argument Sound whenever a sample in the data stream exceeds a specified threshold value (Figure 8).

5.2.3. Histogram

One common way to represent multidimensional data graphically is as a histogram. The sonic histogram, like a graphic histogram, represents the relative magnitudes of several individual elements. In the sonic histogram, each element is represented by a unique frequency, and the time-varying magnitude or multiplicity of that element is mapped to the amplitude of the corresponding frequency. Depending upon the data and the choice of frequencies, the result can range from sounding like a single, fused, time-varying timbre to a collection of independent and identifiable simultaneous voices.

6. TEST CASES

We drew upon the collection of visualization videos produced by NCSA and added data-driven sound tracks to them in order to test the sonification tools. In most of the test cases we used several tools in combination, for example a **Comparator** of two **Histograms** or the **Sum** of a **Marker**, a **DataToFrequency**, and a **DataToDensity**, etc.

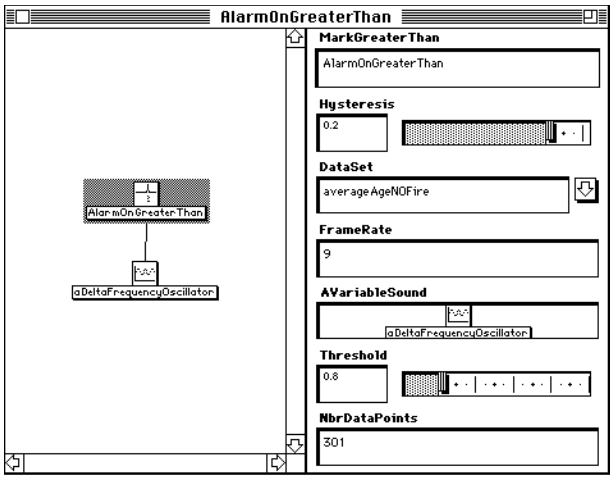

Fig. 7. This tool triggers a siren sound whenever a value in the data set exceeds the threshold.

6.1. Pendulum

Mark Bajuk was the first to approach us about adding a data-driven sound track to a scene from a visualization he did with David Hobill in which the movements of two pendulums illustrate the chaotic behavior of a Duffing oscillator. In this segment, two pendulums start out with nearly identical initial conditions, but end up wildly divergent.

We were able to make several informal observations after experimenting with pendulum mappings. None of these observations has been tested in any formal way; they are simply the reactions of colleagues and ourselves.

Mapping the *velocity* of the pendulums to frequency or amplitude only served to confuse the viewer. Velocities were at their maximum when the pendulums appeared on the video to be at zero; this apparent incoherence between the sound and the visuals was difficult for people to interpret, even when they were aware of the physics. Along these same lines, we found that representing pendulum displacement with *discrete* frequencies seemed to viewers to contradict the apparently *continuous* motions of the pendulums.

Other observations include:

- Viewers preferred a mapping of *displacement* rather than of absolute *position* to sound parameter.
- Attempts to simulate space with stereo channel placement were only effective through headphones or in close proximity to near-field monitors.
- In a mapping from displacement to frequency, beats between the two pendulum frequencies were apparent before they could be seen to to diverge visually
- At several points where the pendulums appear to be stationary in the animation, sonic **Marker**s of the inflection points (zero crossings of velocity) show that they are actually moving back and forth slightly. **Marker**s also make it clear that the initial conditions of the two pendulums are different even though they appear to be perfectly aligned at the beginning of the animation.

6.2. Silence

Everyone who worked on the Pendulum sonification remarked that they could no longer view the video silently without imagining the data-driven sound track.

6.3. Fire

This visualization was constructed out of data collected at Yellowstone Park; it shows a map of Yellowstone in which the age of the forest is represented by its shade of green—the darker the green, the older the forest. Forest fires appear as red areas. The map evolves over time, spanning the years from 1690 to 1990. The purpose of the video, realized by Alan Craig on the basis of data collected by David A. Kovacic et al, was to illustrate that the policy of complete fire suppression that was in effect from 1872 to 1972 resulted in diminishing diversity in the forest and contributed to the devastating fire of 1988.

We used a **Histogram** in which each age was represented by a different frequency; the amplitude of each frequency was determined by the area of the forest of that particular age. In the first attempt, we mapped greater ages to higher frequencies; however, viewers much preferred a mapping in which greater age was mapped to *lower* frequency. This may be a physical

association; larger objects tend to have lower resonant frequencies, and adults tend to have lower voices than children. During the fire suppression periods of the simulation, it was easy to hear the overall aging of the forest.

Forest fires were mapped to noise; the size of the area of forest that was on fire at each point in time controlled the amplitude of the noise. This mapping demonstrates Elizabeth Wenzel's observation that "The ears tell the eyes where to look"; the bursts of noise help to *underline* the flashes of red that are difficult to see in the silent version.

In this test, we also tried using sound to convey additional information not present in the animation; we constructed a **Comparator** of two sonic graphs: the *actual* average age of the forest over time compared to the *projected* average age of the forest over time assuming no fires at all; both were mapped to frequency. These sonic graphs are heard while the visual is a split screen of two maps of Yellowstone. Viewers remarked that the **Comparator** conveyed more information about the magnitude of the *difference* between the two sonic graphs than it did about absolute age values.

6.4. Smog

This visualization illustrates a numerical model of air pollution in the Los Angeles basin (visualized by Mike McNeill and Bill Sherman on the basis of research done by McRae and Russell). It shows the average and peak concentrations of ozone and carbon monoxide over time as x-y graphs at the top of the screen; below that is an isosurface representing the time-varying concentrations and dispersal of three kinds of airborne pollutants.

We used this video to experiment with the idea of object analogies, mapping the density of digital recordings of coughs to the ozone levels. When the background graphs are represented in sound, you can concentrate your visual attention on the complex isosurface while still keeping track of the ozone levels through the "open channel" of your ears.

6.5. Blood

In this case, the goal was not necessarily to arrive at the perceptually optimal mapping but to imitate a familiar sound in order to test the accuracy of a model. M. E. Clark et al, have developed a model of the human arterial system based on principles of fluid mechanics. When cardiologists were shown a video animation based on this model (visualized by Mike McNeill), they wanted to be able to "hear" the simulation as well as see it; their standard diagnostic tools include both visual and aural representations of blood velocity. Medical student Scott Bernstein supplied us with a cassette recording from one of these devices, an ultrasound instrument for measuring blood velocity. We emulated the sound produced by this device in order to provide a familiar, common element between the feedback obtained from actual patients and the feedback provided in this simulation.

After listening to a tape of the ultrasound device, we surmised that the sound was probably produced by feeding noise into a variable low pass filter and using the absolute value of the blood velocity as the cutoff frequency of the filter. We then used our **Comparator** tool to place mappings from measurements taken on the left and right sides of the body to the left and right channels of the audio representation. In both the video and the audio, a single simulated pulse is repeated several times.

For a normal heartbeat, there is no noticeable difference between the left and the right channels; the sonification serves only to outline the shape of the pulse.

When someone is taking the patient's pulse from the right arm, the flow of blood is restricted by the pressure being applied in order to measure the pulse; the flow of blood on the left side is unchanged. In this situation, it is easier to compare the sound of the left and the right sides sequentially rather than to detect the difference between the left and right channels when they are presented simultaneously; the amplitude of the left channel is so much greater that it overwhelms the right channel.

However, in a simulation of the subclavian steal syndrome, it is easier to detect the difference between the left and the right channels when they are presented simultaneously than when they presented sequentially. In this simulation, a blockage in the left subclavian artery introduces a delay in the pulse on the right side of the body with respect to that on the left side. This delay is more noticeable when both channels are played simultaneously than when they are played sequentially.

7. PROCEDURE

7.1. Data-driven animations

All of the video animations used in our tests were created by the NCSA Visualization Group. Since 1986, NCSA has been working with scientists to create animated, data-driven visualizations of scientific phenomena. Most of the visualizations are created by writing software to map data to geometric shapes and colors and then using Wavefront Technologies animation software to render the images and to choreograph camera movement and lighting. The images are then transferred to NCSA's post-production facility where they are stored on an Abekas A-64 digital disk recorder for animation. Titles and credits are added using an Abekas A-72 character generator. The animations are mastered to digital D-1 or Betacam videotape.

7.2. Data-driven sound

All sound development was done at the CERL Music Laboratory using a Kyma System on loan from the Symbolic Sound Corporation. The same data set that was used to generate the animation was used to generate the sound. Data sets were transferred from the Cray at NCSA to an Apple Macintosh II at CERL using the campus network. In Kyma, the data were then scaled to the integer word length of the signal processor so that they could be interpreted as streams of digital audio samples. The scaled data were then downloaded from the Macintosh to the memory of the Capybara signal processing hardware.

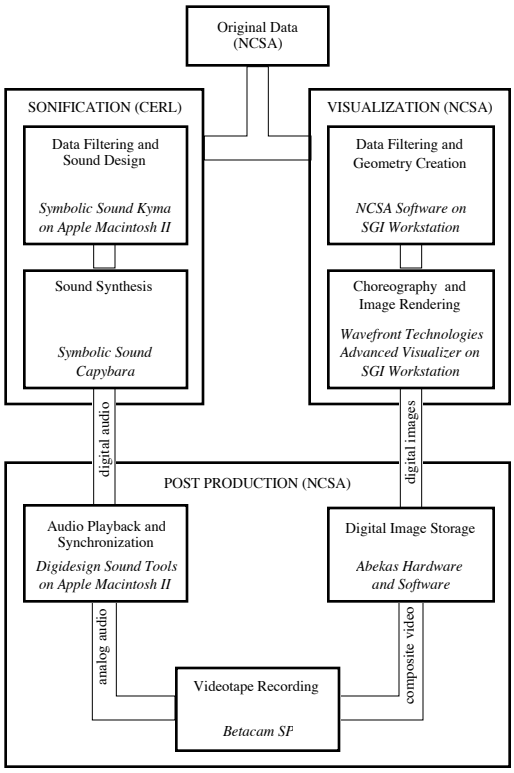

Fig. 8. The same data set was used to generate both the sound and the animation.

Once a mapping had been interactively developed and refined, it was stored as a sequence of samples on the hard disk of the Macintosh. This file could then be sent via the campus network to another Macintosh II in the Post Production facility at NCSA. At NCSA, the Digidesign Sound Tools program was used to play back the samples and synchronize them to the video through the use of a SMPTE time code trigger. Figure 8 shows the path of the data through various parts of NCSA and CERL.

8. DISCUSSION

8.1. Addressing the obstacles

We feel that we have been at least partially successful in addressing each of the obstacles to further progress in data sonification research as outlined by Smith.[10]

8.1.1. Arbitrary mappings

While all of our examples were relational analogies, we also tried a few object analogies—sounds that suggest the data they represent. It appears that object analogies contribute to the sense of engagement with the model world; however, the more abstract, structure analogies were better at bringing out similarities between processes.

8.1.2. General-purpose hardware

As Stuart Smith points out, "The ubiquitous MIDI devices are supposedly inexpensive and very flexible, but the benefits of MIDI equipment are mostly illusory. MIDI devices are designed primarily for the performance of rock and pop music, and they are very difficult to bend to other purposes. Furthermore, MIDI equipment is not really inexpensive. Although individual devices may be cheap, a complete system with which you can do experimental work is not."[10] By using general-purpose digital audio signal processing hardware, we were able to achieve greater flexibility and a greater degree of control over the sound than would have been possible using MIDI equipment.

8.1.3. Portability and a theory of timbre

There are two parts to the last obstacle: portability across platforms and the lack of a satisfactory theory of timbre. We chose to address this issue by developing a set of general tools that would not require that the user had expertise in audio signal processing. Our development environment was written in ParcPlace Systems Smalltalk-80, a programming language that runs on a "virtual machine" and is thus portable to a wide variety of personal computers and workstations.[15] Beyond the portability of Smalltalk-80, though, is the portability of the model of audio signal processing that underlies the Kyma language—the idea of a functional stream language. This basic paradigm has proven to be an intuitive and flexible one for designing audio signals, and it could be incorporated into visualization systems based on similar ideas of data flow or streams. By starting with basic units and providing an environment in which they can be combined in more complex ways, we have defined a sort of language of sonification with its own atoms and grammar. The use of a highly modular and uniform language allows the user to customize and expand upon the tools we have already provided.

Through a process of abstraction—of developing a general sound specification paradigm and a set of tools that can be applied across a wide variety of data—we have taken some first steps toward developing a theory of sound generation.

8.2. Observations

8.2.1. Sonifications

We have found that data-driven sound tracks can increase the bandwidth of scientific visualizations, providing supportive or additional information. Since sound can only exist in time, it lends itself well to the representation of time-varying data and concomitant animated graphics.

Informal reactions from listeners suggest that,while the data-driven sound tracks initially struck them as unusual or strange, they missed hearing the sound track when the video was replayed in silence. They reported that they found it harder to pick out and remember patterns in the silent animation, and that the animation seemed to last longer when it was silent than when it was sonified. Most reported that, after hearing the data-driven sound track, they still imagined it even when the video was played silently.

Other informal reactions include:

- The most effective mappings were achieved when the sound was coherent with the picture
- Mapping the data to frequency was the most effective way to represent small changes with accuracy
- **Marker**s were effective in pointing to small changes that were not evident in the visual representation
- The use of stereo channels only worked well through headphones
- High frequency sounds were more irritating than lower frequencies
- Complex timbres are easier to locate in space and are less tiring to the listener than are sine waves
- Simultaneous presentation of two data streams was effective for representing the *differences* between them more than at representing the absolute values of either stream.
- A **Comparator** is an effective way to represent two signals with small timing differences

8.2.2. Systems

Adding sound tracks to already existing videos while not being able to view the video is not the ideal way to work. Far too much time had to be spent tracking down data sets, retrieving data from mass storage, finding out what frame rates were used, requesting VHS copies, etc. These problems could be solved by working as part of an integrated visualization and sonification team where the data are readily available.

For tasks that would ordinarily be performed by a technician, it would make sense to reimplement one or more of our prototype tools in hardware or software, making the tool less expensive (and less flexible). For the kind of exploration done by *researchers*, however, the Kyma System provides the flexibility needed to quickly try out ideas and to adapt to new areas of inquiry.

To quote Blattner et al, "Regardless of which approach is taken for the design of and implementation of audio message in a sight-and-sound interface, *including an expert in sound relationships in the design team to determine the initial constants of a*

sight/sound computer/user interface is of utmost importance".[1] The musical background of Stuart Smith, Bill Buxton, and others working in sonification bears witness to the fact that the expertise of an electro-acoustic musician can be productively applied to problems in sonification.

9. FUTURE DIRECTIONS

Our initial emphasis was on the development of tools for sonification. We view this initial work as something of a "feasibility study", demonstrating to ourselves and to the NCSA Visualization Group that sonification is a viable option and can contribute to the elucidation of scientific data. For this initial study we drew upon the bank of completed videos at NCSA. In the next phase, we would recommend a closer relationship with individual researchers and the parallel development of the visual and the aural representations.

9.1. Tools

Since we were working with previously computed data sets, we used digital recordings as sources for the data streams. In the future, it might be useful to add a direct digital input to the Capybara hardware in order to allow for real-time data input at the sample rate (adjustable from 11.025 to 28 kHz). (Real-time input streams are already available for line-level signals through the analog-to-digital converter and via MIDI inputs.)

More work could be devoted to the development and testing of high-level sonification tools analogous to such standard visualization tools as tracer particles, isosurfaces, color mappings, etc. and to systematic testing of these tools by a perceptual psychologist. It would be interesting to determine which kinds of information are better presented to the auditory system than to the visual. For example, is the auditory system better adapted to detecting periodicity and small differences between two streams? Does data-driven sound assist people in remembering a visualization?

Throughout this paper we have hinted at the possibility of parallel development of visualization and sonification. We imagine an integrated, stream-based language for both sound and animation.

9.2. Applications

Finally, we believe that sonification and, for that matter, visualization can be more effective when not experienced passively (as is the case with the NCSA videos). We advocate placing more control in the hands of the observer (as in the Smith, Wenzel and Mezrich projects), and the development of systems for active rather than passive perception of the model world.

10. SUMMARY

Our model is one of a single data set with multiple visual and aural representations. We classify our sonic representations as sonic *indices* or *analogies*, that is causal representations affected by the objects they represent. All of our sonic representations are *relation* analogies (in that they preserve the relations between objects in the model world), and some of them are also *object* analogies (in that they mimic or suggest the actual sounds made by objects in the model world).

We use a general-purpose digital signal processor controlled by a visual programming language to generate and modify digital audio signals in real time. A general purpose signal processor (in which sound synthesis algorithms are implemented in software) affords greater control over the details of an audio signal and greater flexibility in designing the audio signal than is afforded by a MIDI synthesizer (in which sound synthesis algorithms are implemented in hardware).

We use a functional stream language for sound specification. In this language, each sound object represents a *stream* of samples; thus, *data streams* fit seamlessly into the environment. A sound object can be constructed using combinations of synthesized sample streams and streams of data from the outside world. By virtue of its modularity and uniformity, this language allows us to design more complex sounds by hierarchically combining simpler sounds; complex sounds can be encapsulated as new classes of sounds in order to hide some of the detail. This language was written in Smalltalk-80 and is thus portable to any of several platforms supported by ParcPlace Systems.

We have developed several high-level tools (or algorithms) for the sonification of time-varying data and demonstrated the application of these tools by producing data-driven sound tracks for several NCSA scientific visualization videos. These algorithms could be implemented as software or hardware tools that are independent of our software/hardware environment.

11. ACKNOWLEDGEMENTS

Many people have contributed to the NCSA videos and to the scientific research that generated the underlying data sets. We would like to thank the members of the Visualization Group at NCSA: Mark Bajuk, Mike McNeill, Bill Sherman, and Matthew Arrott. Robert Patterson and Jay Rosenstein of NCSA Media Services helped us in the edit suite. Colleen Bushell designed Figure 8. Thanks to our project leaders Dan Brady of NCSA Visualization Group and Lippold Haken of CERL Music Group for allowing us the time to pursue this interdisciplinary project. Kurt Hebel of Symbolic Sound provided much helpful feedback. Finally, our thanks to all of the researchers whose work is represented in the videos.

12. REFERENCES

1. M. M. Blattner, D.A. Sumikawa, and R.M. Greenberg, “Earcons and icons: Their structure and common design principles,” *Human-Computer Interaction*, Vol. 4, No. 1, pp. 11-44, 1989.
2. W. W. Gaver, "The SonicFinder: an interface that uses auditory icons," *Human-Computer Interaction*, Vol. 4, No. 1, pp. 67-94, 1989.
3. E. M. Wenzel, S. S. Fisher, P. K. Stone, and S. H. Foster, “A System for Three-Dimensional Acoustic ‘Visualization’ in a Virtual Environment Workstation,” *Proceedings of Visualization ‘90*, pp. 329-337, IEEE Computer Society Press, Los Alamitos, 1990.
4. E. M. Wenzel and S. H. Foster, “Realtime Digital Synthesis of Virtual Acoustic Environments,” *Computer Graphics*, Vol. 24, No. 2, pp. 139-140, March 1990.
5. S. A. Bly (Ed.), "Communicating with sound," *Proceedings of CHI '85 Conference on Human Factors in Computer Systems*, pp. 115-119, ACM, New York, 1982.
6. J. Mezrich, S. Frysinger, and R. Slivjanovski, "Dynamic representation of multivariate time series data," *Journal of the American Statistical Association*, vol. 79, pp. 34-40, 1984.
7. A. Edwards, "Soundtrack: an auditory interface for blind users," *Human-Computer Interaction*, Vol. 4, No. 1, 1989.
8. S. A. Bly, "Presenting information in sound," *Proceedings of CHI '82 Conference on Human Factors in Computer Systems*, pp. 371-375, ACM, New York, 1985.
9. S. Smith, “Stereophonic and surface sound generation for exploratory data analysis,” *Proceedings of CHI ‘90*, pp. 125-132, ACM, New York, 1985
10. L. H. Reuter (Ed.), “Human perception and visualization,” *Proceedings of Visualization ‘90*, pp. 401-406, IEEE Computer Society Press, Los Alamitos, 1990.
11. B. Evans, “Enhancing scientific animations with sonic maps,” *Proceedings of the 1989 International Computer Music Conference*, Computer Music Association, Columbus Ohio, 1989.
12. V. Tejera, *Semiotics from Peirce to Barthes: a conceptual introduction to the study of communication, interpretation, and expression*, pp. 2-26, E. J. Brill, Leiden, The Netherlands, 1988.
13. D. Gentner, “Mechanisms of analogical learning,” *Similarity and analogical reasoning*, S. Vosniadou and A. Ortony (Eds), p. 200, Cambridge University Press, London, 1989.
14. D. Gentner, “Mechanisms of analogical learning,” NCSA Representation Project Seminar Series, Champaign.
15. M. V. Mathews, *The Technology of Computer Music*, The M.I.T. Press, Cambridge, 1969.
16. G. Loy and C. Abbott, “Programming languages for computer music synthesis, performance, and composition,” *Computing Surveys,* Vol. 17, No. 2, pp. 235-265, ACM, New York, 1985.
17. A. Goldberg and D. Robson, *Smalltalk-80: the language and its implementation*, Addison-Wesley, Reading, Massachusetts, 1983.
18. C. A. Scaletti, “Kyma: an interactive graphic environment for object-oriented music composition and real-time software sound synthesis written in Smalltalk-80,” Technical Report No. UIUCDCS-R-89-1498, UILU-ENG-89-1717, University of Illinois Computer Science Department, Urbana, 1989.
19. C. A. Scaletti, "The Kyma/Platypus Computer Music Workstation," *Computer Music Journal*, Vol. 13, No. 2, pp. 23-38, 1989.
20. C. A. Scaletti, "Composing Sound Objects in Kyma," *Perspectives of New Music*, Vol. 27, No. 1, pp. 42-69, 1989.
21. C. A. Scaletti and R.E. Johnson, "An interactive graphic environment for object-oriented music composition and sound synthesis," *Proceedings of the 1988 Conference on Object-Oriented Programming Languages and Systems*, pp. 18-26, ACM, 1988
22. C. A. Scaletti, “Kyma: an object-oriented language for music composition,” *Proceedings of the 1987 International Computer Music Conference*, pp. 49-56, Computer Music Association, 1987.