SIFT: SONIFICATION INTEGRABLE FLEXIBLE TOOLKIT

J.W. Bruce and N.T. Palmer

Microsystems Prototyping Laboratory
Department of Electrical and Computer Engineering
Mississippi State University
Mississippi State, MS 39762-9571 USA

jwbruce@ece.msstate.edu

ABSTRACT

This paper describes work-in-progress on a platform-independent toolkit for sonification of scientific data. The data being displayed and the sonification control information can be provided in real-time and distributed over a wide area via Ethernet. The toolkit allows the designer to process, scale, and map data to a wide variety of sonification parameters and methods. Sonification processing and control commands are stored in standard XML syntax files and can be applied or modified in real-time. The toolkit described here is easily added to existing visualization applications and can be quickly expanded to use new data formats and sonification modalities. Early results of interactive auditory and visual analysis of an example domain are described, and extensive user tests are being planned.

1. INTRODUCTION

Data sonification is becoming a more widely acceptable way to overcome the limitations of data visualization. Many research fields have models and experiments that produce enormous amounts of data with great complexity. It is well known that auditory display of data, especially in conjunction with existing well-proven visualization techniques, can be effective, although sonification and related areas require much more study [1][2]. The sonification community has demonstrated promising results toward

- the interpretation of highly dimensional data;
- the interpretation of data when visualization is difficult to understand,
- revealing trends and finding patterns in long term dynamic temporal data.

With reporting of these successes, visualization researchers are increasingly looking toward auditory cues to enhance the efficacy of data visualization and interpretation. However, the installed hardware and software infrastructure to support the high performance computing of modeling and visualization is expensive and entrenched. For example, CAVE (Computer Automatic Virtual Environment) equipment can cost $100,000.00 for a simple setup. Also, ultra high resolution displays, which are gaining popularity, can cost $20,000.00 for the display and require non-standard video interface hardware. Visualization researchers need a flexible, preferably platform-independent, easy-to-use sonification toolkit that can be integrated into existing visualization applications with minimal modification.

This paper describes a toolkit that is being developed at Mississippi State University, USA, which supports sonification and auditory display of both simulated and experimental data in conjunction with visualization applications already in development or use. The toolkit is platform independent and has a wide variety of data input and sonification data output capabilities.

2. BACKGROUND

SIFT shares similar features and goals with other toolkits under development by the Sonification community. Lodha's work [3]-[5] with the Listen, MUSE, and MUSART toolkits provides an impressive foundation for what a toolkit should do. Many of the audio parameters utilized in MUSART are included as mappings in SIFT, such as pitch, timbre, loudness, and duration. SIFT attempts to utilize these mappings in a multi-platform package.

Previous work using platform-independent environments like Java greatly inspired our approach to SIFT. Upson's work on SoundGrid [6] uses the Java sound and graphics capabilities to create a usable (and fun) application for sonification experimentation. SoundGrid's educational objective during design resulted in an easy-to-use tool with real-time interactivity. However, SoundGrid is not suitable for sonification tasks required for research applications. Namely, it is difficult to import large or dynamic datasets using SoundGrid's current spreadsheet-style data entry. Walker and Cothran's Sonification Sandbox [7] is another excellent example of a multi-platform sonification application. The Sonification Sandbox targets research of auditory display for the blind, science and mathematics education, and data exploration. It also has sophisticated audio parameter mapping capabilities. The application's ability to import a static dataset via spreadsheet is extremely useful for experimenting with the effectiveness of various sonification techniques and parameters. However, because of this static data input functionality, the Sonification Sandbox cannot be integrated into existing applications, such as visualization or real-time models. The current version has no external interfacing capabilities.

Finally, SONART [8] is a very powerful sonification toolkit, especially for research into the mapping of scientific data onto synthesis models and algorithm parameters. However, its integration with applications not specifically designed for Sonification seems difficult. We are unable to locate a version of SONART for testing at the time of writting. Other projects, such as Pauletto and Hunt's work with PD [9] which support easily configurable audio mappings and real-time manipulation also influenced the design of SIFT. Like these tools, SIFT strives to allow users to rapidly experiment with configurations and mappings in real-time while sonifying data.

3. DESIGN

SIFT is designed to be platform-independent, flexible, modular, and easily integrated into existing visualization applications that require sonification abilities. Since many of the sponsors' existing visualization applications have widely varying architectures, are written in many different languages, and run on different hardware platforms, SIFT was must support a wide range of environments.

Much of the design effort is focused on flexibility. Researchers are more likely to utilize elements of sonification if there are tools available for the systems they already operate. Hence, it is our goal to be as platform and architecture independent as possible. Java was chosen for the core toolkit kernel because of its inherent platform independence. Java Virtual Machines are available for most major platforms used in industry and academia, including Windows, Mac OS X, and Linux. In addition, the SIFT project's source code will be made publicly available after alpha testing so that researchers can tailor the toolkit to their needs or add to its functionality.

Much effort is also taken to ensure the flexibility of the toolkits' components. At the heart of the toolkit is the sonification kernel. As shown in Figure 1, the kernel has four main components: a control interface, a data interface, a processing chain, and a synthesizer interface. Each of these components are independent from each other and can be changed or customized to meet specific user needs.

System control information arrives via the control interface. The control interface translates valid inputs into state control messages for the kernel. Inputs can come from a local interactive console, remote shell, or separate application (through TCP/IP, for example). System control information depends on the format and method of data retrieval and/or generation. For example, if the sonification kernel has access to a local copy of the data being sonified, control information can be sent to the SIFT kernel to access the appropriate data records. However, if the sonification data is being computed at runtime, the system control interface can receive model control parameters and independent variables to input into the computational models. Another option is that computational data is being computed at runtime by another system and sonification data is transmitted directly to the SIFT kernel for sonification. The control interface is very flexible and easily adjusted to support interactive, real-time or playback models.

The data interface abstracts dataset information away from a particular data format allowing researchers to use their current, possibly proprietary, data formats. The data interface receives the requests from the control interface and acquires the needed data to forward to the processing chain. Data is acquired by accessing data records from local data storage, computing data from models and algorithms, or simply passing external runtime data through. Extensions for custom dataset formats can easily be incorporated into the system allowing extreme flexibility in data access. For example, an extension for the data interface to acquire information from a database and combine it with simulation data and sensor data could be written easily.

The processing chain controls the mapping of stream data values to parameters that will be applied to sounds by the synthesizer. A set of virtual sound-objects are positioned in the dataset. When the listener queries the dataset, or "listens", the data at each virtual sound-object is mapped and synthesized. Each sound-object can be moved around independently in the dataset sending control commands from the application making use of SIFT. The mapping is performed by a chain of processing blocks that link a data element to a sound parameter of a virtual sound object. For example, the SIFT application may have a local data store of climate sensor readings. At regular intervals, system control information requests that the data interface retrieve hourly average wind speed data from the data store. The processing block applies several transformations to map the hourly average wind speed to sonification parameters, like pitch.

Processing blocks attempt to transform the data into a meaningful value before assigning it to a sound parameter. Mappings and processing chains are configurable at runtime using configuration files. Processing configuration files are stored in standard XML syntax. Processing blocks can perform mapping or transformation functions and simple pattern matching. The object structure of the processing blocks means that new processing functions are easily created. Some examples of currently existing processing block elements are

- transformation blocks (invert, scale, shift, normalize, constant, convert to scale)
- conditional blocks (pass-through, pass-on-repeat, pass-on-match)

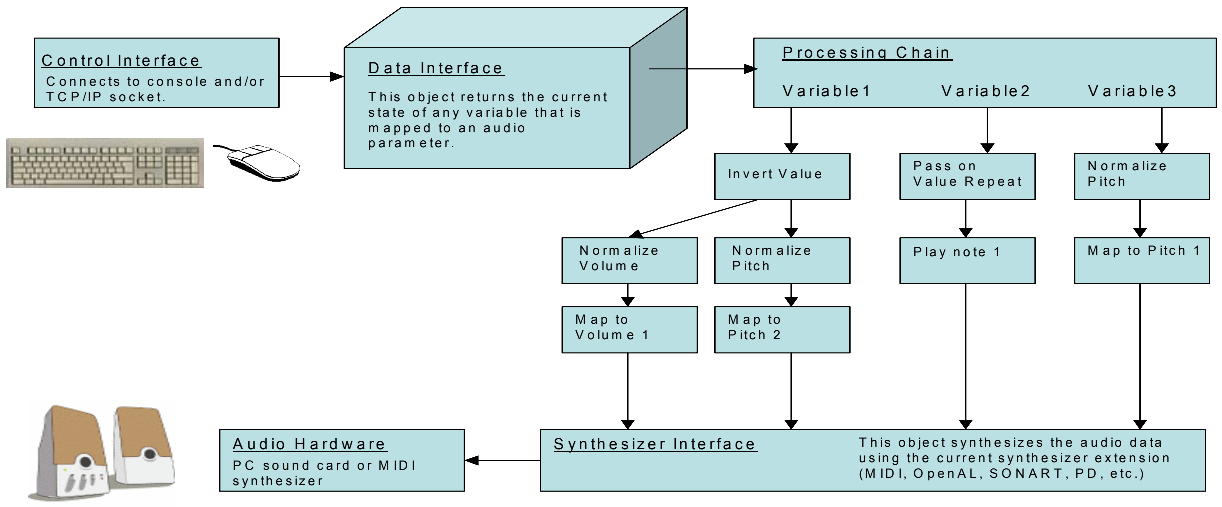
Control Interface
Connects to console and/or TCP/IP socket.
Data Interface
This object returns the current state of any variable that is mapped to an audio parameter.
Processing Chain
Variable1
Variable2
Variable3
Invert Value
Pass on Value Repeat
Normalize Pitch
Normalize Volume
Normalize Pitch
Play note 1
Map to Pitch 1
Map to Volume 1
Map to Pitch 2
Audio Hardware
PC sound card or MIDI synthesizer
Synthesizer Interface
This object synthesizes the audio data using the current synthesizer extension (MIDI, OpenAL, SONART, PD, etc.)

Figure 1. *SIFT kernel structure (data flow example)*

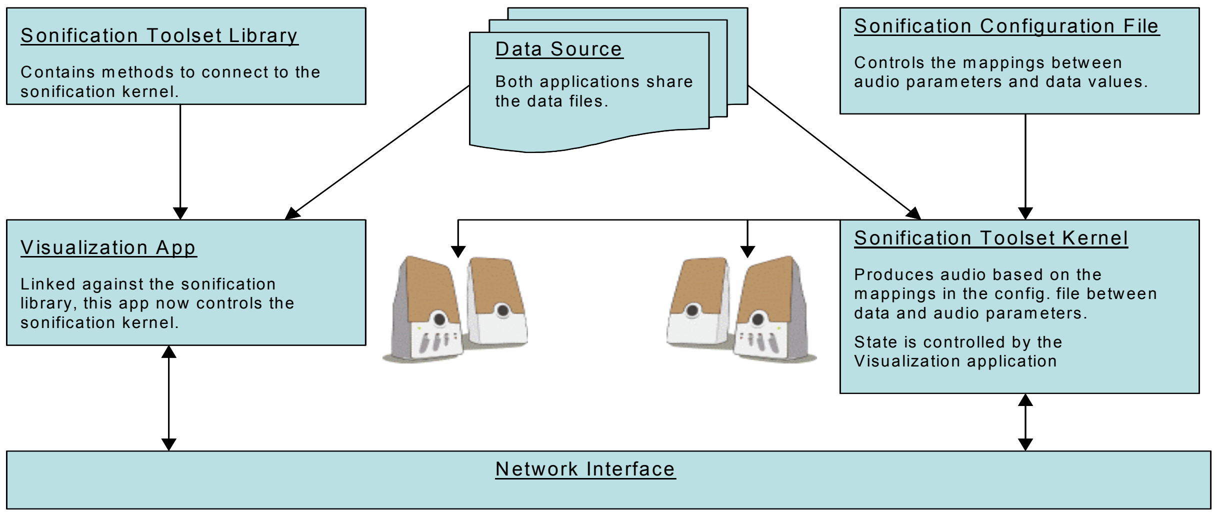
Sonification Toolset Library
Contains methods to connect to the sonification kernel.
Data Source
Both applications share the data files.
Sonification Configuration File
Controls the mappings between audio parameters and data values.
Visualization App
Linked against the sonification library, this app now controls the sonification kernel.
Sonification Toolset Kernel
Produces audio based on the mappings in the config. file between data and audio parameters.
State is controlled by the Visualization application
Network Interface

Figure 2. *Example configuration of SIFT with existing visualization application*

Finally, the synthesizer interface provides a flexible mechanism for producing the actual sounds heard by users. A synthesizer can be chosen to reflect the needs of a particular project and new synthesizers can be written if needed. Sound descriptions are generic with each synthesis module responsible for translation to specific sounds. Examples of some existing synthesis commands are

- audio parameters (pitch, volume, timbre, duration)
- control parameters (start sound, stop sound, sound duration, pitch change)

Currently, SIFT has implemented a relatively full-featured MIDI translation module. MIDI synthesis can be done by internal computer hardware, Java software synthesis, or transmitted over standard MIDI ports. The emphasis is on MIDI output for now because of the ease of parameter mapping and the low bandwidth. However, other interfaces such as OpenAL, PD, and a proprietary network format are being developed. Support for most existing synthesis toolkits in the sonification community can be easily added later.

3.1. Integration

The second factor driving the design, after the need for flexibility, is ease of integration. Currently, the sonification toolkit is being used to determine the effects of adding audio components to existing visualization applications. A library is available in C++ that allows researchers to control the sonification toolkit from within their applications. To connect with and control the sonification kernel, the application should be linked against this library and make calls to the functions made available through the library. Ultimately, the user interface of the original application remains familiar while new sonification features are easily added.

The toolkit library works by first establishing a socket connection to the sonification kernel, which acts as a server, over a TCP/IP network and then streaming control commands from the user. Commands such as “load dataset”, “move absolute”, and “listen” are available to the visualization application from the provided library. Sonification details, such as how data are mapped to audio parameters, are specified using the toolkit’s configuration files. Integration is simplified because only commands with intuitive visualization analogs, such as moving to a point in the dataset, need to be sent by the visualization application. Figure 2 shows the relationship between the sonification toolkit and the application utilizing it. The sonification kernel acts as a server responding to a client application’s commands.

3.2. Interactivity

A major goal of the sonification toolkit is to facilitate interactive sonification of data. Interactive sonification allows researchers to readily experiment with parameter mappings and transforms. This allows for immersion into the data being studied and provides an exploratory mechanism for the dataset. Researchers can maneuver through a data set, stopping to explore interesting sounding areas. Two particular aspects of the design encourage interactivity.

First, the data and control interfaces are self-contained modules that act as interfaces to the system. The data interface, for example, can fetch data from a static file, generate real-time data based on formulae, or use some combination of the two. Also, the control interface is in essence an event handler for real-time events from the application utilizing the toolkit.

Second, data is dynamically mapped to audio parameters by the processing chains. The mappings can be modified both before and during sonification, allowing for rapid experimentation during simulation.

3.3. Sonification Features

Several sonification features are available to researchers using the sonification toolkit. Data can be mapped to the pitch, volume, timbre, duration, or balance of a virtual sound-emitting object positioned around a virtual listener. There can be multiple virtual sound objects around the virtual listener. The synthesizer interface plays the sound as the virtual listener would hear it.

Data can be processed in many ways before becoming a sound parameter. Simple transforms such as scaling it by a constant factor, normalizing it to a range, or inverting its values are available. More complicated processing, such as vector magnitude calculation, is available for certain datasets. The data is processed much in the same way that audio remastering applications process audio streams. An input and output are specified along with 0 to n processing blocks in between. The input comes from a tuple, or collection of data values, of the dataset at the current index of the virtual sound-object. The output is then mapped to a parameter of that same virtual sound emitting object. Outputs can also be mapped to control parameters of the sound object, such as “start emitting sound”.

4. EXAMPLE

In collaboration between visualization group in Mississippi State University’s GeoResources Institute and researchers with the U.S. Army Corps of Engineers, the sonification toolkit is helping researchers study oceanographic data sets. Ocean current data from a 2001 study in the Gulf of Mexico is being concurrently visualized and sonified in an attempt to understand how the two data representation methods can work together to reveal information.

The visualization application was developed for previous research, so the main work was integrating the sonification toolkit. First the visualization application’s source code had to be modified to incorporate the SIFT initialization and communication routines. Basically, a socket connection is established at startup and a move command is sent whenever the mouse moves from one data point into another. To perform the experiment, the sonification kernel is started in listening mode on a Linux based test machine acting as a server. This sonification server also loads the dataset used in the visualization application that was modified above. Next, the visualization application is started on the same machine as the sonification kernel. The visualization application connects to the running sonification kernel and starts sending events as the mouse moves from data point to data point.

At this point, researchers are free to experiment with various mappings between these data points and audio parameters. An example configuration maps water current flow direction to pan (left and right) and triggers a “listen” event

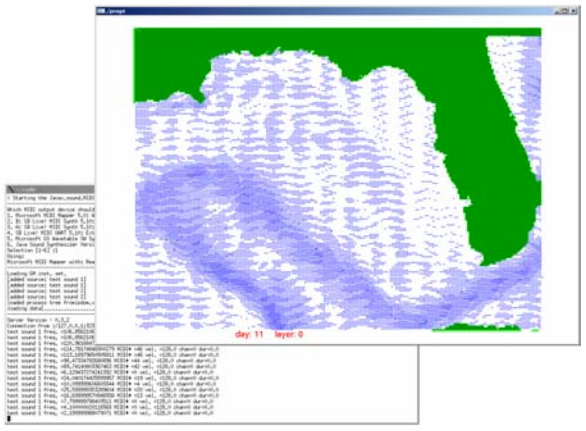

Figure 3. *Screenshot of the SIFT console and a visualization application*

every time the mouse moves over a new data point. This allows researchers to extract directional information about the data while viewing more general information such as water current speed distributions. Figure 3 shows a screenshot of the sonification kernel’s console output and the visualization application displaying a “wide view” of the data set.

5. CONCLUSIONS

This paper has described the first prototype of the Sonification Integrative Flexible Toolkit (SIFT) along with one of its initial uses in an oceanographic current visualization model. SIFT is easily incorporated into existing applications and supports local and remote data storage and real-time data calculation. SIFT implements an extendable synthesis interface at the backend, and is readily configured to use almost all sound synthesis methods available commercially and in the research community.

Much work remains to be done on the processing chain interface. Namely, a graphical interface for configuring audio parameter mapping is to be implemented.

Also, the synthesizer interface will be improved to provide a more robust set of mappings, such as register, thickness, and loudness. Another area to be improved is the data interface. Currently, data contains any number of values. However, the initial data access scheme assume that data is located by a four dimensional tuple (x, y, z, t). Most spatial and temporal data sets fit nicely into this indexing scheme but some datasets will not; therefore, it will be replaced with a more generalized indexing scheme in the future.

6. ACKNOWLEDGMENTS

The authors would like to acknowledge the support and guidance of Professor Robert Moorhead at Mississippi State University and Professor Cynthia Ford at Jackson State University. The authors are grateful for the assistance of Joel Martin during the visualization application integration process.

This work has been supported, in part, by the High-Performance Computing and Visualization Initiative program funded by the U.S. Army Corps of Engineers.

7. REFERENCES

[1] Kramer, G., *et al*, “Sonification Report: Status of the Field and Research Agenda,” National Science Foundation, 1997.

[2] Hunt, A. and Hermann, T., “The importance of interaction in sonification,” *Proc. ICAD*, Sydney, Australia, 2004.

[3] Wilson, C.M. and Lodha, S.K., “Listen: A data sonification toolkit,” *Proc. ICAD,* Palo Alto, USA, 1996.

[4] Lodha, S.K., Beahan, J., Heppe, T, Joseph, A.J., and Zne-Ulman, B., “MUSE: A musical data sonification toolkit,” *Proc. ICAD,* Palo Alto, USA, 1997.

[5] Joseph, A.J. and Lodha, S.K. "MUSART: Musical Audio Transfer Function Real-Time Toolkit,” Proc. *ICAD,* Kyoto, Japan 2002.

[6] Upson, R, “Educational sonification exercises: Pathways for mathematics and musical achievement,” *Proc. ICAD*, Kyoto, Japan, 2002.

[7] Walker, B. and J.T. Cothran, “Sonification Sandbox: A graphical toolkit for auditory graphs,” *Proc. ICAD*, Boston, USA, 2003.

[8] Ben-Tal, O, Berger, J., Cook, B., Daniels, M., Scavone, G. and Cook, P., “SONART: The Sonification Application Research Toolbox,” *Proc. ICAD*, Kyoto, Japan, 2002.

[9] S. Pauletto and A. Hunt, “A toolkit for interactive sonification,” *Proc. ICAD*, Sydney, Australia, July 2004.