SOUNDSERVER: DATA SONIFICATION ON-DEMAND FOR COMPUTATIONAL INSTANCES

Jorge Cardoso, José Carvalho, Luís Teixeira, Álvaro Barbosa

Catholic Portuguese University
Sound and Image Department
Rua Diogo Botelho 1327, 4169 - 005 Porto, Portugal
{jccardoso, jvcarvalho, lmt, abarbosa}@porto.ucp.pt

ABSTRACT

The rapid accumulation of large collections of data has created the need for efficient and intelligent schemes for knowledge extraction and results analysis. The resulting information is typically visualized, but it may also be presented through audio techniques such as sonification. Sonification techniques become especially interesting when the client application runs on graphically limited devices such as mobile phones or PDAs (Personal Digital Assistants). In this paper we present an architecture for a sonification server that will be used in the *Sound Data Mining* project. In this project sound will be used to increase perception and present information extracted by spatial data mining techniques. The server is based on an audio synthesis engine and will relieve clients with little audio synthesis capabilities from the burden of sound processing. By providing sonification modules, this server can potentially be used on a variety of applications where sonification techniques are required.

1. INTRODUCTION

The work described in this paper is part of a project that consists in the application of spatial data mining techniques to various fields: from real estate to hydrodynamic and water quality data in rivers. Sound will be used as one of the means for information presentation and data exploration. The project is called *Sound Data Mining*.

Spatial databases hold information on geo-referenced data, i.e., data regarding the location and shape of geographic features. Spatial data includes both topological and geometric data. As with other types of large databases, one of the most important, and difficult, aspects of spatial databases is the extraction of knowledge. Spatial databases typically have huge amounts of spatial data that render the human ability to analyze, useless, making it necessary for automatic methods of analysis and knowledge discovery, or extraction. As defined in [1], *spatial data mining* is the *extraction of implicit knowledge, spatial relations, or other patterns not explicitly stored in spatial databases*.

Spatial data mining techniques enable us to obtain information that would be difficult to get otherwise. A a simple example of the kind of information that could be obtained from a spatial data mining process is the correlation of a cholera outbreak in London in the 19th century, to a contaminated water pump located on Broad Street. Although this correlation was done "manually" in 1854 by Dr. John Snow [2], we might imagine an extrapolation to nation or world-wide disease analysis that would demand for automatic processing.

Sonification is *the use of non-speech audio to convey information* [3]. It is of special interest when there is a high data volume and number of variables; in these cases it may be useful to present part of the information visually and part through audio. Audio can be used to increase perception of the information that is being graphically displayed, or it may be used to present information that is not displayed visually. The output of a spatial data mining process can take many forms, e.g., clustering, classification, prediction, etc. As an example of the use of sonification in the case of a clustering technique, consider the case where we want to overlay clustering information on a standard map. Sound could be used to present clustering information without cluttering the visual interface with graphical cues. As the user moves the pointer around the map, a sound could inform him of to what cluster that point belongs. The use of acoustic information becomes more important as the graphical capacity of the user interface diminishes. This is especially true in the case of mobile devices where the graphical display is very limited, not only in terms of size, but also in colour depth and resolution.

In this paper we present an architecture for a sonification server based on an audio synthesis engine. This server will relieve clients with little audio synthesis capabilities from the burden of synthesizing the sounds.

The rest of this paper is structured as follows: in section 2 we argue for the need of a sonification server and present its design; in section 3 we show what we have done so far in implementing the server; finally, in section 4, we present our final remarks about this project and what we expect to do next.

2. DESIGNING THE SOUND SERVER

2.1. Why a Sound Server?

We have chosen to implement a sound server for the sonification process for several reasons. One of the main reasons for the use of a sound server is the fact that some devices have many limitations. Mobile phones, for instance, have very little audio synthesis capabilities but relatively good audio rendering capabilities. This means that implementing the acoustic interface in these devices would have to use very basic sound manipulation techniques if they were to be done at runtime. On the other hand, generating sound files at design time and playing them at runtime would reduce the flexibility of the sonification.

Another justification for a sound server is that we want to develop several applications, in different domains and for different platforms, from desktop computers to mobile devices. Program-

ming for these different platforms requires the use of different libraries. This means that without a server we would have to implement the sonification process for each application/platform. What we need is a platform independent and reusable solution.

The flexibility of the final system was also taken into consideration. Most applications will share some sonification processes and the use of a server enables us to easily change the implementation of that sonification without having to re-implement all the client applications.

One other advantage of using a sound server is that it can be used by developers of user interfaces that don't really care about the specific implementation of the sonification. In a sense, using a server provides encapsulation of the sonification techniques. This means that the programmer using a sonification available at the server need not know how that sonification is accomplished. He needs only to know what a certain sonification represents.

The idea of using a sound server is obviously not new. In *Public Sound Objects* [4] a processing engine is used as a *collaborative music creation* server for *geographically displaced communities*. This server synthesizes sound according to client parameters. In *Y-Windows* [5], Kaltenbrunner proposes a server that follows the philosophy of the X Window System [6], applied to an Auditory User Interface (AUI) environment where developers are relieved of the task of implementing basic auditory widgets.

2.2. Philosophy

The Sound Server design follows three premisses:

1. The server must accept requests from clients in different platforms.
2. Clients have different audio rendering and synthesis capabilities.
3. The server should provide a set of high level *sonification modules*.

The Sound Server design is based on the idea of *sonification modules*. A sonification module is an entity within the server that represents a particular sonification. The server can be configured with a multitude of sonification modules: a module that sonifies an *absolute value*, a module that implements a *parameter mapping* sonification, a module for the sonification of *density*, etc. A client may ask for the result of several modules in one request. In these cases the sonifications have to be combined together in one sound file. There are several ways in which this can be done. Figure 1 shows some possible combinations of modules. The sound stream from module C is put on both channels, module A only on the left channel at the beginning of the file, module B on the left channel at the end of the file and module D only on the right channel at the beginning. It is up to the client to specify how the modules should be combined together.

The sonification modules should be *self-describing*, i.e., there should be a way for the client to ask the module what tasks it performs and what parameters can be controlled. In its simpler form this description might simply be a text message informing the user of how the sound should be interpreted. The server should also provide a listing of the available modules.

The main output of the server will be a sound file with the sonification that the client requested. The client may ask for the sonification of several modules at the same time, and instruct how the several modules should be grouped together in the resulting sound file. If the client asks for module A and B, he may want the

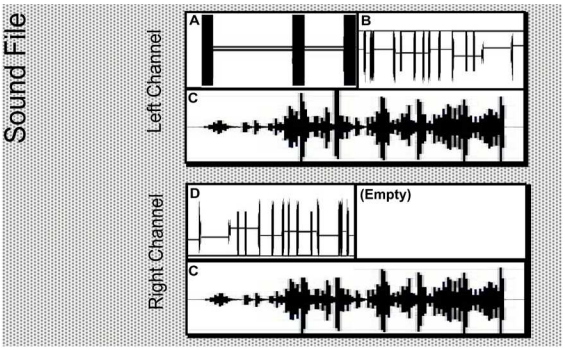

Figure 1: *Combining Sonification Modules*

respective sounds in the same channel, or in different channels, synchronized, or displaced in time. The client may also specify other parameters of the output, such as the sound quality (bitrate and samples per second), the output format (mp3, wave, aiff), etc.

The server will be accessed through HTTP enabling the clients to access it like a regular Web Server. This will facilitate the access by a wide variety of devices since the HTTP protocol is widely implemented.

Clients with greater synthesis capabilities should be able to process the sonification locally, needing to communicate with the server only to obtain the sonification module.

2.3. Architecture

Figure 2 shows the logical architecture of the Sound Server. The server is composed by two main units: *Web Server Unit* and *Synthesis Unit*.

The Web Server Unit is responsible for the communication management. This unit functions as a proxy to the Synthesis Unit adapting the requests from the clients to the correct syntax required by the Synthesis Unit and vice-versa. It will offer the client applications three interfacing modes: URL-encoded parameter-value pairs through the HTTP GET method; XML document encoded requests sent through the HTTP POST method; and SOAP-RPC requests through HTTP. Providing these different interface modes will allow for a wider range of supported client platforms and applications. It also allows for diferent levels of complexity in the messages exchanged between clients and the server, since messages structured in XML documents can be more complex than messages encoded in the URL, for example. For increased performance, this unit will cache the requests so that the same request is processed only once.

The Web Server Unit can itself be broken down into four subunits:

URLInterface Accepts requests in URL-encoded parameter-value pairs. This is the simplest interface to the server.

XMLInterface Accepts requests encoded in XML documents. This interface will allow for more complex messages to be exchanged between client and server.

SOAPInterface Will provide a SOAP-RPC interface to the server. This interface will allow clients to access the server as a Web Service [7].

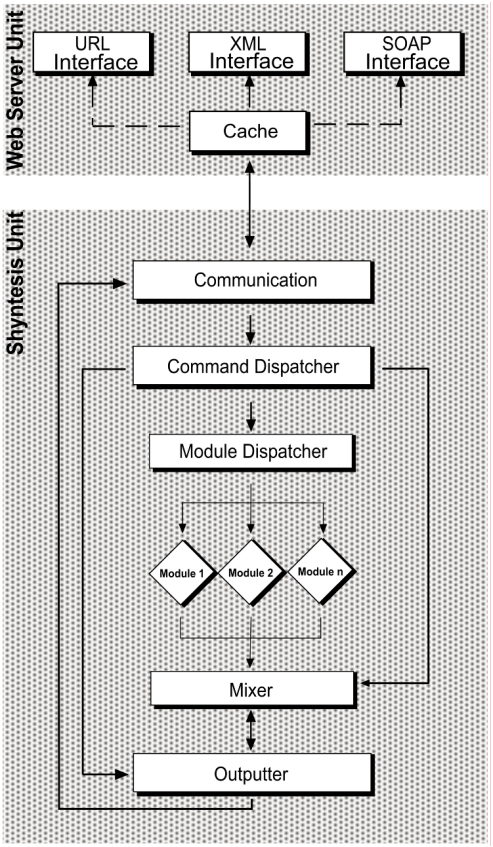

Figure 2: *Sound Server Architecture*

Cache Caches requests from all types of requests. The request is routed to the Synthesis Unit only if it cannot be found in the cache.

The Synthesis Unit constitutes the nucleus of the sound server and is where the actual sonification takes place. It is composed by the following subunits: Communication, Command Dispatcher, Module Dispatcher, Sonification Modules, Mixer and Outputter.

The Communication subunit manages the communication between the Web Server Unit and the Synthesis Unit. It is able to manage concurrent requests/responses.

The Command Dispatcher receives commands from the Communication subunit and routes them to the appropriate subunit (Module Dispatcher, Mixer or Outputter). This is needed because both the Mixer and the Outputter can be configured by the client.

The Module Dispatcher subunit has a similar function to the Command Dispatcher. It routes commands to the appropriate sonification module, enabling the client to pass parameters to them.

The Sonification Module is where the actual sonification takes place. These subunits are independent modules that can be added or removed from the server. We can think of them as a kind of *plugin* for the sonification server.

The Mixer is responsable for the final integration of the several sound streams into one file, according to the client specification. It gathers the audio signal from the sonification modules and combines them in a single audio stream.

The Outputter subunit takes the output from the Mixer and generates a sound file in the format, and according to the settings set by the client. This subunit can be configured to generate sound files in different formats and with different audio quality so that each client can adjust the sound to its capabilities.

These two units: the Web Server Unit and the Synthesis Unit can be loosely coupled, i.e., they could be run on different machines, allowing more complex topologies, if necessary.

3. IMPLEMENTING THE SOUND SERVER

The Web Server Unit prototype has been implemented in PHP [8] running on an Apache [9] HTTP server. At this stage we have only implemented the URLInterface subunit. The final version of this unit will probably be implemented in Java because it provides more structured code.

For the Synthesis Unit we have chosen Miller Puckette's Pure Data [10] since it is a free, open source platform for audio synthesis and it also provides a high level (graphical) object oriented programming environment.

At this stage, the server accepts only very simple commands. As an example, consider the URL: `http://serveraddress/sonifica.php?module=abs&value=500`

This request would return a sound file with the sonification of an "absolute value" of 500 units.

3.1. Sonification Modules

A sonification module is just a Pure Data patch with a specific interface (inputs and outputs). The architecture of the sonification modules is still under development, and the aim is to have modules that can be easily added to the server without too much effort.

Basically, each module will be able to define its own parameters that allow specific characteristics of the sonification to be changed, e.g., the sonification of an absolute value will accept the "value" parameter. The duration of the sound stream generated by the module will also be defined by the module itself. The module will signal to the server when the sonification is complete so that further processing on the request can be made.

3.2. Testing the Server

In order to test the server we have implemented a Java applet[1] (Figure 3) and two sonification modules used by the applet: one for the sonification of absolute values and one for the sonification of density. These modules are used by the test applet for the sonification of the area and population density, respectively, of the Portuguese districts in the map.

The absolute value module was implemented following the directions of [11] with some minor differences. Our implementation uses sounds made by simple sine waves while theirs used sampled grand piano sounds. In both cases the fundamental frequencies were nearly the same. For the density module we used granular synthesis, variating the pitch and number of grains according to density (higher pitch and number of grains represents a higher density).

In this applet, the user can select what data to sonify (area or population density) and, in the case of the area, whether the sonification uses time or time and pitch. When the user clicks inside a

[1]This applet is available at `http://artes.ucp.pt/citar/sdm/`.

district, a request is made to the sound server for the corresponding sonification. The requests to the sound server are regular HTTP requests with URL-encoded parameters, so that from the applet's point of view we are simply accessing a sound file on a web server.

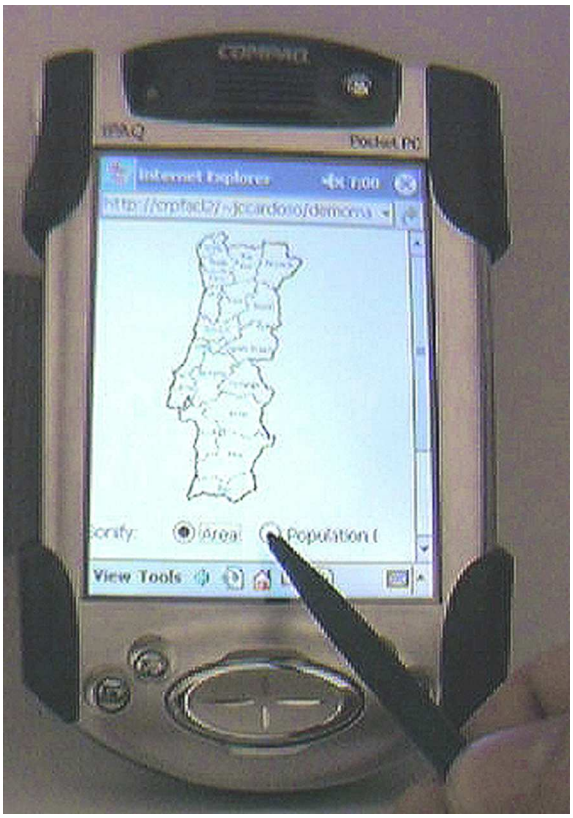

Figure 3: *Test Applet*

In our tests both the Web Server Unit and the Synthesis Unit were running on a single machine: a Pentium IV at 2.5GHz. The client application was running on a Compaq iPAQ Pocket PC connected to the PC via a USB cable.

3.3. Results

The test applet seems to indicate that we have a valid server design that can be further explored and implemented.

The two sonification modules used in the test were independently implemented in Pure Data and easily adapted to the server. This suggests that augmenting the server with other sonification modules will not be difficult. Implementing the Java test applet also shows that it is simple to use the sonifications available at the server. To obtain a particular sonification, all one has to do is to make an HTTP request to the web server.

In our tests, the delay time in receiving the sound file, is roughly equal to the time that the sonification modules take to generate the sound. The modules work in real-time therefore generating 3 seconds of audio will take exactly 3 seconds to complete. This delay time will depend only on the type of sonification being made and may or may not be acceptable, depending on the application.

4. CONCLUSIONS AND FUTURE WORK

We have presented here a design for a sonification server that will allow the use of sonification techniques in a wide range of applications and platforms.

Although the work described here is still in development, we believe that we already have a well designed and structured sonification system. In the following months, we will further develop this work: not only by continuing to enhance the implementation of the server but also by applying it to real case applications.

5. ACKNOWLEDGMENTS

This project is being supported by the *Agencia de Inovacao* (AdI)

6. REFERENCES

[1] Krzysztof Koperski and Jiawei Han, "Discovery of spatial association rules in geographic information databases," in *Proc. 4th Int. Symp. Advances in Spatial Databases, SSD*, M. J. Egenhofer and J. R. Herring, Eds. 6–9 1995, vol. 951, pp. 47–66, Springer-Verlag.

[2] John Snow, *On the Mode of Communication of Cholera*, John Churchill, New Burlington Street, London: England, second edition, 1855.

[3] G. Kramer, Bruce Walker, Terri Bonebright, Perry Cook, John Flowers, Nadine Miner, and John Neuhoff, "NSF Sonification Report: Status of the field and research agenda," 1999.

[4] A. Barbosa, M. Kaltenbrunner, and G. Geiger, "Interface decoupled applications for geographically displaced collaboration in music," in *Proceedings of the International Computer Music Conference - ICMC 2003*.

[5] Martin Kaltenbrunner, "Y-Windows: Proposal for a Standard AUI Environment," in *Proceedings of the International Conference on Auditory Display - ICAD 2002*.

[6] Open Group Inc., "X Window System," `http://www.x.org/X11.html`.

[7] World Wide Web Consortium, "Web Services Activity," `http://www.w3.org/2002/ws/`.

[8] PHP Group, "PHP: Hypertext Preprocessor," `http://www.php.net/`.

[9] Apache Software Foundation, "Apache HTTP Server Project," `http://httpd.apache.org/`.

[10] M. Puckette, "Pure data: another integrated computer music environment," in *Proc. the Second Intercollege Computer Music Concerts*, 1996, pp. 37–41.

[11] Anikó Sándor and David M. Lane, "Sonification of absolute values with single and multiple dimensions," in *Proceedings of the International Conference on Auditory Display - ICAD 2003*.