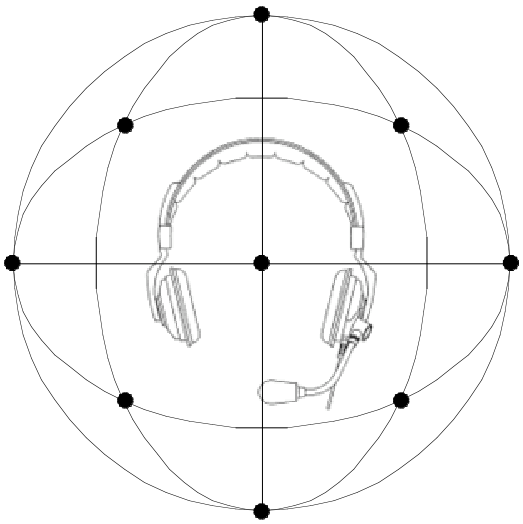
Treball de recerca - Program de doctorat en informàtica i comunicació digital de la Universitat Pompeu Fabra *(12 Crèdits)*

Realitzat por: KALTENBRUNNER, Martin

Dirigit per: SERRA CASALS, Francesc Xavier

Summary of Publications

Sound in Human Computer Interaction



Institut Universitari de L'Audiovisual – Grup de Tecnologia Musical
Estació de França – Paseig de Circumval•lació, 8 – 08003 Barcelona, España
http://www.iua.upf.es/mtg – mkalten@iua.upf.es

Abstract

This document presents an overview on my publications, which have been published so far during the last years. The main focus of my past research work has been around the topic of Human-Computer Interaction, especially concentrating on the particular role of sound. Additional research areas, which have been touched by this work, are Computer Supported Cooperative Work, Virtual Environments and Information Retrieval - as well mostly concentrating on the role of audio therein. The document additionally provides a rough introduction into the field of auditory user interfaces in order to allow the reader to position the commented publications within that context. I will also present some questions and concepts for my near future work leading towards my PhD dissertation.

Keywords

Human Computer Interaction, HCI, Auditory User Interfaces, AUI, Virtual Environments, VR, Computer Supported Cooperative Work, CSCW, Information Retrieval

Acknowledgements

At this place I'd like to thank all my co-authors of my articles which all contributed an important part of the work presented in this collection. First of all there is Dr. Avon Huxor, who introduced me at all to area of scientific publications and gave me the opportunity to have my first articles published already during my internship times.
Then there are Pedro Cano and Alvaro Barbosa, two current colleagues at my institute, who allowed me to contribute with my specific interests to their personal work. Finally I'd like to thank Dr. Xavier Serra for his supervision and patience during the first part of this PhD programme.

Contents

1. Introduction, Motivation & Personal Background

This brief introduction outlines the historical and personal context of my work and tries to define a common ground - sound in human computer interaction - for the various publications, which have been published so far within different backgrounds.

Since my career at the Polytechnic University of Hagenberg in Austria, where I studied "Media Technology and Media Design", my work was focused on Human Computer Interaction, at this time mainly on the development of new interaction methods within *virtual environments*. This was also when I first started to develop simple speech-controlled applications, which later led me to most of the questions, which concern me today about the design of auditory user interfaces.

By the end of my undergraduate career, I decided to pass an internship at a research institution, in order to gather more experience about scientific working methods, since by then I already had decided to continue with some post-graduate studies. Dr. Avon Huxor, Senior Researcher at the Centre for Electronic Arts at the Middlesex University in London was my supervisor during this research internship. Dr. Huxor at this time was also researching on *collaborative virtual environments*, which led us to the development of a joint project, the Marvin awareness robot. We developed an *acoustic monitoring application*, which was designed to increase the awareness of events within virtual environments. Again with this project, although generally planned to extend *computer supported cooperative work* (CSCW) applications, my personal interests shifted more towards the design of auditory interfaces. As a result of this internship project there were published two articles, which I will review in detail later in this document.

Encouraged by the results of the Marvin project, I decided to do a review on auditory user interfaces as topic for my Diplomarbeit (undergraduate thesis) with the title "Auditory User Interfaces for Desktop, Mobile and Embedded Applications". This review, which tried to capture the then state of the art auditory interface concepts, led me finally to the proposal of a general architecture for auditory interface design.

Based on this early work developed within my undergraduate thesis, I later published an article where I tried to further develop the rough ideas on this AUI architecture. The article on "Y-Windows" was then my first article that entirely focused on the development of *auditory user interfaces*.

Currently at the Music Technology Group at the University Pompeu Fabra I am involved in some cooperative projects with some of my colleagues, which resulted in the publication of two further articles, which I also will review here because of the close relation to my general work on human computer interaction. The "Public Sound Objects" project, which is currently under development together with Alvaro Barbosa, defines a framework for *Computer Supported Cooperative Music*. The article "On the Use of Fastmap", written with Pedro Cano et al describes a *sonic browsing* tool, which we developed for the browsing of large collections of music.

Part I

Publication Context

2. A Glossary of Auditory User Interfaces

This section provides a quick introduction into some key concepts of auditory user interfaces, which might be yet unknown to the reader. In general this document will refer to auditory interfaces incorporating both speech and non-speech audio.

Some central questions for the design of auditory interfaces are, how many different dimensions and variables are available and usable for display in an auditory universe? What are the major differences from the way we visually perceive our surroundings and is the audio channel really secondary compared to the visual system? When we look, we only can see those objects which are in our visual field, while we can hear everything which surrounds us, we even can hear things we can't see. Hearing is a passive process, we can easily listen to the radio while being occupied with other tasks. But when we read a book, watch television or sit in front of the computer monitor, all our visual attention is required for this task. On the other hand environmental noise might be distracting, we even can't close our ears like we can do with our eyes. Auditory perception is linear in time, therefore sound objects exist in a certain time, but over space. Whereas visual perception is linear in space, so visual objects exist in a certain space, but over time [2]. These are some of the main thoughts, which should be considered for the development of an acoustic interface.

2.1 Sonification, Auralisation & Audification

Where visualization tries to make abstract data visible to the user, Sonification intends to present this abstract data acoustically. While basic sonification uses arbitrary data and tries to define a model for mapping that data to the synthesis parameters of sound, audification transforms existing non-audio signal data into the audible domain.

Recent examples of audification are the transformation of earthquake records to an audible domain by F. Dombois [3] or the transformation ocean wave spectral data by B. Sturm [4]. Sonification Examples include the sonification of EEG data for analysis by Th. Hermann [5] and the initial work by G. Kramer.

Data sonification is especially suitable for time-varying data, since sound is a temporal medium [6], which allows to detect changes in the data patterns over time. The essence of good sonification is the convergence of the multidimensional data in a way that it is perceived as an integral whole. Only this careful design, mostly relying on psycho-acoustics, allows the perception of patterns in multidimensional data. *Interactive auralisation* enables the user to explore specific data regions by having direct influence on the data auralisation process. The tuning of a radio receiver to find the right channel is a good metaphor for the interactive exploration of patterns in auralised data streams.

2.2 Acoustic Monitoring

Many monitoring applications use simple auditory cues for event notification or for process sonification. One major advantage of audio is that it doesn't require any special attention for the task. As mentioned before, hearing is a passive process - we can easily listen to the radio while being occupied with other tasks.

An impressive example of an acoustic monitoring application is the Peep Network Auralizer [http://www.auralizer.com/], which auralizes the amount of network traffic by using water sounds and indicates additional events with auditory icons. The "Personal Webmelody" [7] by M. Barra et al uses simple MIDI music to allow the monitoring of a web-server. This constant display of a sonification allows the easy identification of irregular patterns, in this case faulty operation of the web-server.

2.3 Auditory Feedback

Auditory Icons are usually short samples of natural sounds, carrying symbolic meaning about the associated object or task. Parameterized Auditory Icons [10] are synthesized symbolic sounds, which allow the full control over their appearance through the modification of the synthesis parameters.

The simplest form of an auditory icon is an ordinary system beep to attract the user's attention. But while this may be sufficient within a graphical user interface, auditory icons for audio interfaces demand to be more complex. In modern everyday life there are already a huge set of auditory icons around wherever we are: The ringing of the phone, the announcement chime on a train platform or the hooting of a car. These everyday sounds can be digitised and used within auditory interfaces to indicate similar events and therefore they are easily recognisable for the user without any learning effort. Other real life sounds such as an opening door or a coughing person can be chosen for events which are easily to associate with the particular sound, although if badly chosen this already involves a short period of learning and recalling for the user. The third group of auditory icons are artificial sounds with no real world equivalent. The *Sonic Finder* [2], an auditory extension to the MacOS, which was never released commercially. is a good example for the concept of auditory icons within a mainstream operating system.

Earcons are short harmonic sequences with a dedicated characteristics, such as a common chord. This characteristics can be inherited by related Earcons in order to form hierarchical groups. The design allows the construction of complex hierarchies, such as the browsing through a tree structure [8]. A transposed common chord for instance indicates a higher hierarchical level, whereas a four note chord stands for a different branch of the tree. This supports the maintaining of a mental model of the tree structure as well as the reasonable localisation of the current position in the hierarchy. An advantage of the structure of earcons from the implementation point of view is that they can easily be created on the fly using MIDI hardware or software.

2.4 Multimodal Interfaces

In virtual reality applications, sound plays - together with additional senses like haptics - an important role in order to improve the sensation of immersion. The same is the case for common desktop applications: sound can significantly improve the interface experience although today's graphical interfaces still often behave like silent movies. Research in this area specially focuses on the creation of authentic soundscape by enhancing the perception of spatial audio and the accurate rendering of virtual auditory scenes by incorporating virtual room impulse responses etc.
To achieve plausible perception of spatial audio such as accurate identification of distance and direction of the virtual sound sources, several aspects have to be added to the dry audio signal. In short the signal for both ears is slightly different: There are the *interaural time delay* (ITD) and the level difference as a result of the shading of the further ear, the *interaural intensity difference* (IID). A Head Related Transfer Function (HRTF) covers the physiological filter characteristics of a particular listener (head, shoulders), which is convolved with the signal. Finally early reverberations, and the room impulse resonse help for the better localisation of virtual sound sources.

2.5 Voice Applications

Voice applications are already quite common in telephone inquiry systems. Standards such as VoiceXML together with powerful products such as IBM WebShere Server have been boosting the development in this sector recently. This voice mark-up language allows the development of web-based client-server applications, where the actual interface dialog is executed within a local voice browser providing basic speech recognition and synthesis functionality. The success of the World Wide Web has shown that networked applications which separate computing logic and the actual display on the client's browser software is a versatile concept which will also suit voice only applications. On the desktop speech command & control applications remain niche products, while continuous speech recognition products for text dictation have become more useful and popular.

2.6 Interfaces for the Blind

Typical auditory interfaces for people who actually fully rely on them are accessibility applications for the visually impaired. This includes screen-readers such as Jaws [http://www.freedomscientific.com/], which help to explore graphical interfaces by reading them loud using a speech-synthesizer. There even exist complete audio only desktop environments for the blind such as EmacsSpeak by T.V. Raman. [9]. Finally there exist a large number of experimental systems, which try to make the visual domain hearable, by the sonification of images. [http://www.seeingwithsound.com/]

Emacspeak is currently one of the most mature speech-enabled applications, which provides complete eyes-free access to daily computing tasks. Although it is an extension of the popular Unix editor Emacs it should not be considered simple screen reader software, but rather as an independent auditory desktop application. Emacspeak doesn't speak the visual screen output of the incorporated applications but provides a set of extensions to these applications, which render the basic data output to a speakable format. With Emacspeak blind users can already perform almost any computing task, such as text processing, web browsing, or file management. Other applications include e-mail and newsreaders as well as complete integrated development environments. Further applications can be speech-enabled by implementing an appropriate extension.

3. Commented Publications

3.1 Y-Windows: Proposal for a Standard AUI Environment

This article was presented in July 2002 at the 8th International Conference on Auditory Display [http://www.icad.org/] in Kyoto, Japan. The ICAD is one of the most important conferences covering auditory display and its related fields.

The paper outlines a model for the integration of basic auditory interface elements into modern operating systems. The main intention was to identify those basic components, which are necessary to construct a standard auditory interface allowing developers to benefit from the provided infrastructure. GUI shells of today's operating systems allow the coherent co-existence of various applications at the same time by providing a common interface metaphor, the desktop. For the creation of auditory applications there exist a variety of libraries and toolkits [12], but there is no real coherent operating system integration available.

While working on the *Marvin* project we stated a lack of such an infrastructure for the creation of applications with auditory interfaces. In an analogy to the architecture of the X-Windows GUI architecture the Y-Windows concept tries to define this missing infrastructure for a Unix-style operating system. This includes the general idea of supplying a central server instance, which manages the whole auditory display. On top of this engine, providing the basic auditory display functionality, resides an AUI toolkit library, implementing a collection of modular user interface elements – the auditory widgets.

The main reason for the development of the Y-Windows concept was the need for such a re-usable component library, which should bring advantages for both the developers and the users of auditory applications. The developers do not need to fully implement any new interface concept by themselves, and the user or not forced to learn any new interface they are confronted with, as soon as they understand the basic principle of a single Y-Windows application, they should be able to handle any further application implemented with the toolkit.

The key motivation for the concept can be summed up with four central objectives:

- The creation of an adequate development and operating environment for primarily pure acoustic user interfaces. This adds an additional pure auditory mode to the existing text and graphical modes of current operating systems.

- The system must provide a shared environment where multiple auditory applications can share simultaneously the audio hardware resources and also can exchange audio data with each other. It also manages the final rendering and display of the complete auditory scene, creating a complete and coherent auditory interface.

- A central engine performs basic audio signal processing functions, without the need of redundant re-implementation of known concepts. This rendering engine should provide optimal implementations of basic sound generators and filters and other similar components equivalent to generic graphical operations for visual interfaces.

- The environment has to create a common hear-and-feel by providing common acoustic user interface elements, which stay consistent over all applications using the provided AUI libraries. Those libraries and interfaces not only have advantages for the developers of auditory applications, they are also crucial for the actual users of the framework which are not forced to get used to various interface metaphors while using different applications.

While still being interested in a real implementation of this concept, I am aware that this task would exceed the work force of a single person. While continuing on the further refinement of the concept itself, my personal focus will be concentrated on the interface metaphors and the implementation of some auditory widgets, as stated below in the section on the possible future work for my PhD dissertation.

3.2 Marvin: Supporting Awareness through Audio in Collaborative Virtual Environments

This is the first of two articles on the Marvin project, which was developed during my internship at the *Lansdown Centre for Electronic Arts*, Middlesex University in London. It was presented in April 2000 at the conference on "Digital Content Creation" in Bradford, England [http://www.inf.brad.ac.uk/conf2000.html] and later in 2001 it was included as a chapter in a book edited by Rae Earnshaw and John Vince, published by Springer under the same title "Digital Content Creation".

The project leader and co-author Dr. Avon Huxor had been working on several projects in the field of Collaborative Virtual Environments and Computer Supported Cooperative Work. He developed a virtual office within *ActiveWorlds* [http://www.activeworlds.com/] - an online community system with a VR-style graphical 3D interface. This virtual office environment should help to maintain social links and chance encounters for tele-workers and collaborators, who normally work isolated in their separated environments, using common CSCW tools such as BSCW [http://bscw.gmd.de/] to share their work.

The Marvin system intended to overcome a problem, which arose by the use of those systems: the lacking instant awareness of events happening in the virtual worlds. Therefore we designed and implemented an agent program, which should log into various information systems, monitor events with those systems and finally report those events to the user by auditory cues. In the system we implemented, the Marvin robot only was operating within the *ActiveWorlds* space, but due to a sophisticated plug-in architecture, the system can be easily extended to report events from virtually any internet-based information system.

So, besides it functionality in supporting the CSCW applications, Marvin primarily was designed as an acoustic monitoring application. An outstanding advantage of acoustic monitoring is, that it doesn't require the permanent attention of the user. While performing other tasks, which require high attention such as reading, the user's attention can be instantly drawn to a desired event by the playback of audio. In the case of the Marvin system, this audio cues have been designed using basic Auditory Icons, which transported some corresponding symbolic meaning about the events happening, such as a door sound for a person entering a virtual room. Events requiring a higher detail of information have been reported by speech, using a standard text-to-speech system.

In order to maintain and support the advantage of paying low attention to the application itself, we also implemented basic voice command functions, which allowed the control of the application and the change of a few parameters. The only visual component of the system was a single window displaying a log of all events occurred.
An initial configuration of log-levels which defined the importance of the various event, if they should be reported by using a specific auditory icon or speech, had to be done via a flat configuration file.

Although the project is not actively maintained anymore it was published as open source and still can be downloaded from the project page. [http://yuri.at/marvin/].

3.3 Multiple Presence through Auditory Bots in Virtual Environments

This second article published as a result of the Marvin project was presented at the "Presence 2000" workshop [http://www.presence-research.org/] in Delft, Netherlands. Since during the work with the Marvin system some interesting results around the sense of presence within virtual environments arose, we decided to present the project as well at this event, dealing as its name suggests, with the creation and evaluation of the sensation of presence in virtual environments.

Thanks to its multi-threaded plug-in architecture, the Marvin robot is able to log into various information systems at the same time, and even to log simultaneously into the same system with various instances at different locations. In the case of the *ActiveWorlds* plug-in, this placed the robot avatar into different locations of the virtual world, monitoring and reporting the events within its closest environment. This simple property, of being acoustically aware of the events of several places at the same time, created the impression of being at all of these place at the same time, a notion of social presence. Presenting the same spaces visually on a screen, looking at several windows showing the scenery of different locations did not at all create that impression.

It is a special feature of audio that even sounds from virtual sources can become, when displayed via speakers, an equal and natural part of the overall soundscape. A telephone ringing on the desk and its virtual counterpart ringing in a virtual environment and being played back by a speaker on a desk create a similar sound, none of them is more or less "real" than the other, even when using low quality hardware. In the case of the Marvin system, therefore all monitored spaces acoustically become part of the natural soundscape, thus extending their virtual space to the real world. In the visual domain this sensation is hardly to achieve, even when high-end VR environments such as the CAVE [http://www.fakespacesystems.com/] already provide an outstanding immersion, the displayed objects never become real matter. But virtual sound - even of bad quality – can achieve this maximum reality. It is just like sound from real sources is changing air-pressure around the listener, therefore it is an optimal form of augmented reality. This finally allows for the perception of a higher sense of presence, supporting the sensation of being there - in the virtual environment.

Another aspect we noted while working with the system was the often-cited cocktail-party effect [1], the ability to "tune into" and segregate one of many audio streams produced by various sound sources, such as listening to a certain speaker at a crowded cocktail-party. In the case of the Marvin the various audio streams are produced by each of the proxy instances. In order to support this ability we applied to each audio stream a different sound effect, such as an echo or other distortion effect. This allowed to easily distinguishing between the various event sources, while we still have been able to use the same auditory icons for the same events. So the type of sound indicated the event type, while the sound effect provided an additional cue where the event occurred, even when many sounds have been played back at the same moment.

The ideas and results obtained by the Marvin experiment have been the basis for further ideas on the design of auditory interfaces, as later presented with the Y-Windows concept.

3.4 Public Sound Objects: A Shared Musical Space on the Web

The Public Sound Objects project is currently in development at the Music Technology Group. The first author of this article, Alvaro Barbosa, is developing concepts for Computer Supported Cooperative Music towards his PhD thesis, and this project should provide a prototype implementation of the ideas developed together. My personal focus within the project is around the development of new interfaces incorporating sound. The article along with a basic prototype will be presented at the 2002 Conference on "Web Delivery of Music" [http://www.wedelmusic.org/] in Darmstadt, Germany.

The Public Sound Objects define a prototype implementation of a framework for computer supported cooperative music on the web. The system should allow a large number of connected Internet users to take part and collaborate in the creation of musical piece. Using the classic client/server architecture users can control a server based synthesis engine with their web based client software. The resulting piece is then played back on the installation site, where local users also can participate, and it is also streamed back to the individual Internet users.

The project raises a series of interesting questions about human computer interaction and cooperative work in general. Especially two aspects, the design of the client-interface - the actual instrument – and the constraint control over the aesthetics of the resulting cooperative piece are in particular challenging. We also try to overcome imminent limitations of the network itself, the network-delay for example, by integrating it consciously into the system's architecture. An additional important feature is the skin-concept, which will allow the creation of individual instrument interfaces.

3.5 On the Use of FastMap for Audio Retrieval and Browsing

In cooperation with Pedro Cano, a researcher at the MTG who focuses on Audio Information Retrieval, we developed a simple tool for browsing and comparing large music or sound repositories. This article and the tool will be presented this October at the 3rd International Conference on Music Information Retrieval [http://ismir2002.ircam.fr/] in Paris, France.

Using feature vectors obtained by technology developed by the AIDA project [http://www.iua.upf.es/mtg/] we have been able to calculate distances of similarity of songs. In order to project this multi-dimensional distance matrix onto a two- or three-dimensional space, which is easier to understand and to analyse, we applied the *multidimensional scaling* (MDS) method FastMap to our dataset.

We also developed a simple tool, which allowed us to display the obtained map on a two-dimensional plane, and experimentally also within a 3D-environment. The objective was a tool, which allows for fast comparison and evaluation of the obtained distance results. Currently only ordinary playback of the involved sounds is possible, but we plan to experiment with advanced auditory display techniques, such as spatial audio, which we think might add additional value for the fast comparison and evaluation of these large audio datasets.

4. Possible Future Work

4.1 Auditory Interface Metaphors

One major task in the near future will be an experimental design and evaluation of auditory interface metaphors. While with the Desktop and WIMP (Windows, Icons, Menus, Pointers) there exist well established and easily understandable metaphors for graphical interfaces, which make it easier for the average user to instantly understand and remember the meaning of most user interface elements and interaction principles, there apparently do not exist such strong common metaphors for acoustic interfaces yet.

Of course, elementary concepts such as Auditory Icons try to map the same symbolic value transported by visual icons to the acoustic world, but simply transferring all the approved concepts of GUIs to the auditory space often doesn't seem to be a valid solution, since yet basic interaction models such as direct manipulation with a mouse pointer for example don't work that easily in a "blind" space.

Providing a suitable interface metaphor makes it easier for the user to establish a mental model of the system he is controlling, and of course maintaining such a mental model is even more important for a pure acoustic user interface.

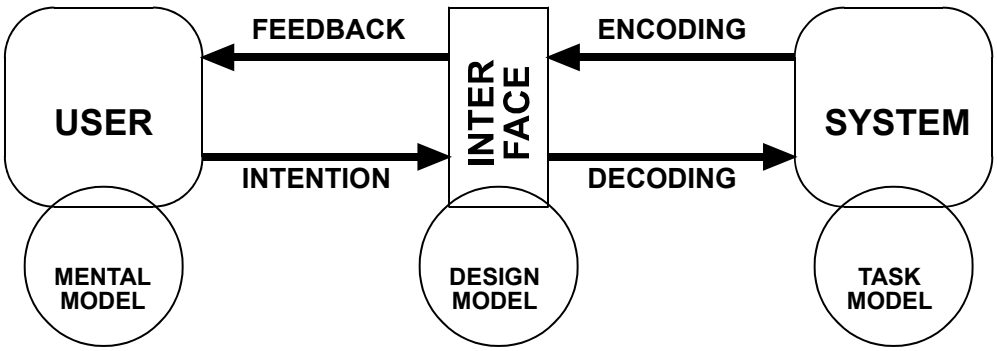


Fig. 1: Basic HCI model

In an experiment together with some undergraduate students we will create various binaural acoustic spaces – without any direct visual representation, basically composed of some audio sources placed and moving in a virtual 3D space around the head. We will then first try to develop different working interaction models for the direct manipulation and control of those objects, with voice control or standard hardware control devices for example. We will try to organize the acoustic space in a way, which allows common users to explore and understand the space with little or no additional instructions. Finally we will evaluate these installations in user experiments in order to identify the best working models. The results of this joint effort will then be published at an appropriate event.

We will develop these models by using existing toolkits for binaural audio display, such as sLab [http://human-factors.arc.nasa.gov/SLAB/] and the ViaVoice SDK [http://www-3.ibm.com/software/speech/dev/] for the speech control interfaces. If the necessary infrastructure is made available we will also work with head-tracking in order to increase the acoustic immersion.

4.2 Auditory Interface Aesthetics, Hear & Feel

Hear & Feel is a term, which I derived from the Look & Feel in GUI Environments. While for GUIs this basically refers to the coherent aesthetic appearance of the graphical design and presentation of all user interface elements (see the different appearance of the various OS shells, such as MacOS X or Windows XP for example), in the acoustic space this seems to be even more important: where a visually badly designed interface environment at the worst might not have a pleasing look, badly designed sound actually can be rather disturbing or even "hurt".

Therefore in areas such as sonification for example there has to be found a way to combine the maximum of information being transported by sound with an aesthetically not necessarily pleasing but at least bearable acoustic experience. Music in its widest interpretation of course could be a measure for such a definition of aesthetical boundaries.

The second aspect of the term Hear & Feel is the acoustic coherence of the complete interface experience, which might be composed of various independent parts. There is a need for a set of rules, which have to be met by all of these acoustic interface components in order to maintain a complete and coherent interface experience for the user. This also should allow the user the easier handling even of yet unknown interface components if they follow the concepts and metaphors of previously handled interfaces. This of course goes hand in hand with the yet articulated need for common interface metaphors.

Finally Hear & Feel should be modifiable by the user in order to meet his personal acoustic taste or needs. This basically is already implemented by changing the "themes" of most acoustic feedback sounds in any mainstream operating system, it has to be investigated though how this concept can be extended to all aspects of acoustic interface design.

4.3 Sonic Widget Definition & Design

A central idea of the model that I defined in [K1] was the need for an extensive definition, design and implementation of basic modular acoustic interface components, called Sonic or Auditory Widgets. While there already have been developed various research prototypes of such components, I am interested in the design of a complete library of these reusable acoustic interface components. Compared to the design of graphical user interfaces, where we are provided with a vast variety of different GUI libraries, the lack of those acoustic components slows down the possible development of advanced auditory user interfaces. While the complete implementation of such a framework appears to be enough work for a whole development team, I am personally interested in the conceptual design and prototype definition of a set of auditory widgets.

Concrete example of such sonic widgets is the "dynamic auditory icon" widget, based on Gaver's concept [10] of parameterized auditory icons. While Auditory Icons are commonly only simple sound samples of natural sounds, Gaver proposes the creation of physically modelled sound libraries, which allow the control over many characteristics of the Auditory Icon, by changing the various synthesis parameters. A library providing a collection of these synthesizable Auditory Icons can be considered as a basic Sonic Widget. Dynamic Auditory Icons could be based on natural sound samples, where a preliminary analysis process extracts the changeable parameters. The resulting data can

be used to re-synthesize many variations of the original natural sound by changing the obtained synthesis parameters. This allows a much larger flexibility in the use of the specific Auditory Icon, which can be squeezed to the needs of the current context, while providing a potentially huge collection of natural sounds.

4.4 Computer Supported Cooperative Music

While not directly related to the design of purely acoustic interfaces, the project on Computer Supported Cooperative Music *Public Sound Objects* raises several questions in Human Computer Interaction design in general. Especially the skin-interface will allow us as developers and the users the experimental design of new interface paradigms for virtual musical instrument design.

Another aspect, which will have to be evaluated, is the need of additional local auditory feedback for the instrument control, and how this could be made part of the local acoustic experience, without the being perceived as a dedicated part of the whole musical piece.

The project also involves some of the aesthetical sound design aspects as mentioned above, since there probably need to be introduced some constraints for the individual user in order to maintain a coherent appearance of the whole musical piece. The same questions could be applied to the concurrent display of various acoustic applications on a single display. Additionally, we hope that the server-side synthesis engine might be re-usable for further projects, such as a sonification engine or similar AUI components.

4.5 Sonic Browsing, Information retrieval

Continuing work on the *SoundSurfer* will focus on the implementation of those acoustic presentation models obtained by the experiments on interface metaphors described above. In this case actually we will focus on a multimodal interface, which concentrates on the visual and acoustic representation of the musical data to explore. This aims at the efficient exploration of large collections of sound and music data for easier browsing, retrieval and comparison.

A binaural 3D presentation of the audio repository might help to directly compare the similarities between songs, for example by the simultaneous playback of two songs - one displayed for each ear. 3D audio might also help to browse a graphical 3D representation of song distances, by being able to listen to audio clips, which are located in a near sphere around the head.

In general we plan to experimentally investigate further, yet unknown display methods, which allow the faster comparison of rather long sound clips. This for example could be achieved by re-sonification of the feature vectors, which uniquely represent each audio-clip, but which do not allow any reconstruction of the sound itself. Still acoustic comparisons of these sonified features could help in quickly identifying similarities between songs.

Part II

Appendix

5. Full Papers

[K1]

Kaltenbrunner, M. "*Y-Windows: Proposal for a Standard AUI Environment*", Proceedings of the 8th International Conference on Auditory Display 2002, Kyoto (Japan)

[KH1]

Kaltenbrunner, M. & Huxor, A. "*Marvin: Supporting Awareness through Audio in Collaborative Virtual Environments*", in: Earnshaw, R. & Vince, J. (eds.) "Digital Content Creation" p. 294ff, Springer Verlag, Hamburg 2001

[KH2]

Kaltenbrunner, M. & Huxor, A. "*Multiple Presence through Auditory Bots in Virtual Environments*", Proceedings of PRESENCE 2000, 3rd International Workshop on Presence, Delft (Netherlands)

[BK1]

Barbosa, A. & Kaltenbrunner, M. "*Public Sound Objects: A Shared Musical Space on the Web*", Proceedings of the 2nd International Conference on Web Delivering of Music 2002, Darmstadt (Germany)

[CK1]

Cano, P. & Kaltenbrunner, M. & Gouyon, F. & Battle, E. "*On the Use of FastMap for Audio Retrieval and Browsing*", Proceedings of the International Symposium on Music Information Retrieval 2002, Paris (France)

Y-Windows: Proposal for a Standard AUI Environment

Martin Kaltenbrunner
Music Technology Group, IUA – Universitat Pompeu Fabra
Passeig de Circumval·lacio 8 – 08003 Barcelona, España
mkalten@iua.upf.es

0. Abstract

This paper introduces a draft framework for a shared auditory user interface (AUI) environment. Y-Windows, similar to the approach of the X-Windows GUI framework [1] in the Unix world, aims to provide common functionality for the easier development and design of AUIs. This initial publication of these ideas, originally roughly developed within a diploma thesis [2], should encourage researchers and developers from the auditory interfaces community to contribute to the further development and possible future implementation of this concept.

1. Introduction

Today there exists a variety of libraries, APIs and applications ([3],[4],[5]) focused on the development of auditory interfaces, there even exist complete auditory desktop environments [6]. While each of these components provides its specific functionality it is often impossible to incorporate them together within a single application – because of their exclusive use of the sound hardware for example or simply their monolithic design. During his initial work involving the design of auditory user interfaces [7] the author noticed a lack for a common framework allowing these components to work together.

The Y-Windows concept therefore was designed to fill in this gap. Usually GUI application developers don't have to spend their time implementing basic interface design concepts, because they are already provided with a variety of libraries, which simplify the construction of an average GUI. Only few GUI-developers might still attempt to develop the code for their buttons or progress-bars "manually", but this is still the case for their equivalents in the AUI world. Therefore the implementation of some AUI widget libraries will be one of the tasks for the Y-Windows project.

Due to the known limitations in audio programming there is also a need for a central instance, which manages the resources for various applications requiring simultaneous access to the audio hardware. In the Unix world there already exist several so called sound servers, which have been designed exactly for that purpose. Such a sound server, extended and optimized for the special AUI requirements, will be the central component of Y-Windows.

While the examples in this document are mainly borrowed from the Linux platform context, Y-Windows ideally should be platform-independent and network–transparent. An application developed for Y-Windows should be easily portable to - or executable on - any platform implementing the framework. One possible approach to achieve this is the use of the Java programming language or easily portable code for the high-level interfaces, while using optimized native code for lower layers.

2. General Objectives

The general objective of this project is to create an adequate development and operating environment for primarily pure acoustic user interfaces. This adds an additional auditory mode to the existing text and graphical modes of current operating systems. Linux users might be familiar with the concept of run-levels, where the operating system boots directly into such an operating mode, therefore a pure auditory mode is easily to implement for those systems. Of course, the Y-Windows system should also be usable for multi-modal interfaces if needed.

Such a system must provide a shared environment where multiple auditory applications can share simultaneously the audio hardware resources and also can exchange audio data with each other. In the simplest case this should allow additional audio playback in the foreground while for example an acoustic background-monitoring task is running. On the other hand it should be possible to record audio from the microphone while a speech recognition process is listening to the same port. And finally several independent applications should be able to use the in- and outputs of others just like ordinary devices. This routing functionality also should allow the construction of application chains out of basic components.

A central engine should provide basic audio signal processing functions, which can be easily used by the connected applications, without the need of redundant re-implementation of known concepts. This rendering engine should provide optimal implementations of basic sound generators and filters and other similar components equivalent to generic graphical operations for visual interfaces. Additional rendering tasks such as advanced sound, music or speech synthesis and advanced signal processing should be realized as plug-ins and therefore as extendable or replaceable parts of this engine.

Graphical operating systems already come with the necessary libraries for the creation and execution of standard conformant GUI applications. This guarantees a uniform look-and-feel and handling of all different kinds of applications. The widely used classical desktop metaphor also allows inexperienced user the quick understanding of the interface principles. Therefore Y-Windows has to create a common hear-and-feel by providing common acoustic user interface elements, which stay consistent over all applications using the provided AUI libraries. Those libraries and interfaces not only have advantages for the developers of auditory applications, they are also crucial for the actual users of the framework which are not forced to get used to various interface metaphors while using different applications. The adequate metaphor for such an auditory “desktop” environment is still missing though.

3. Basic Design

According to these requirements the Y-Windows approach is roughly split into four major layers: A hardware layer should define an abstracted interface to the various parts of audio hardware. The central layer – the Y-Server – provides the shared acoustic workspace and performs the basic rendering of virtually all-acoustic interface elements. A third level allows abstract access to the rendering layer through a collection of libraries providing speech APIs, synthesis APIs and so on. Additionally it should provide libraries for higher-level AUI elements, such as parameterized auditory icons and common auditory widgets. Finally the actual auditory applications using the Y-Windows server and its libraries are part of the fourth layer. A vertical direct rendering interface (DRI) as shown in the diagram additionally allows direct access to all the lower levels for those applications, which rely on time-critical operations.

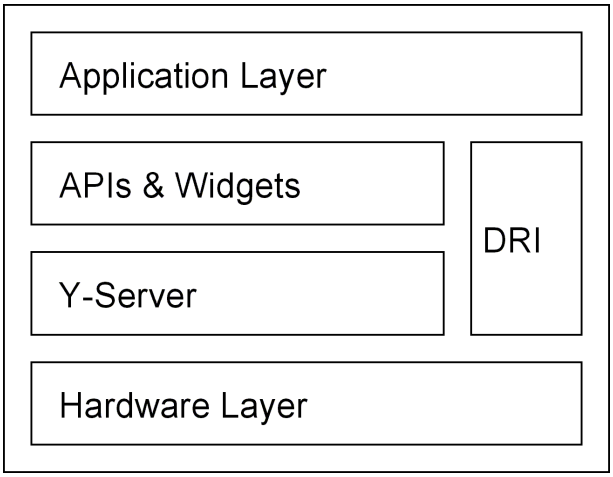


Figure 1. Basic Y-Windows Structure

3.1 Hardware Layer

This lowest-level layer mainly intends to create a common abstract interface to the various audio hardware. This not only should include the typical sound cards but also should consider all additional hardware relevant for auditory user interfaces:

- Internal, external sound cards
- MIDI equipment
- Hardware speech synthesizers
- 3D speaker systems
- Head trackers
- Additional multimedia hardware

Projects such as PortAudio [12] already provide such a common interface for the different audio programming interfaces such as OSS, ALSA, CoreAudio, DirectSound etc., which we know on the various operating system platforms. It has to be evaluated if such a system conceptually can be extended for additional hardware as listed above. Although the implementation of this lowest layer is not really crucial for the overall concept, it generally improves the portability of the higher layers, specially the server component.

3.2 The Y-Server

The Y-Server is the central instance of the Y-Windows architecture. It is in charge of the generation & rendering, composition & management and finally the actual display of the complete auditory scene.

As already mentioned above, the server process should already internally provide some basic built-in rendering operations, such as simple waveform generators and the most common filter operations, providing fast access to simple sound synthesis methods. A modular design and plug-in architecture should allow the versatile extension of this rendering layer with additional components. This modular approach permits the easy replacement of specific components, such as different brands of speech recognition or synthesis engines for example. It should also encourage the co-existence of both commercial and open-source components. A collection of standard plug-ins should include at least some of the following functionality:

- Advanced sound and music synthesis, such as spectral or physical modelling
- Speech synthesis
- Further filters and sound effects
- 3D sound spatialisation

The second major server task is the connection and management of the various applications, which are simultaneously using the sound server infrastructure. This basically includes the routing of the audio in- and outputs of all the connected components, which concatenates chains of internal components, external plug-ins and actual applications. All this components should be able to actuate equally as sound generators, filters and consumers within this system.

Finally, the server has to render the actual output of the complete auditory scene. While the output stream of a single task, which is simply played back via the speakers is no difficult issue, the rendering and presentation of multiple streams of independent applications could use a spatial display using headphones, where the various applications are placed at different positions within the virtual user space. Other possible final presentation methods to distinguish multiple tasks could be different volume levels or finally applied sound effects, which should be definitely part of this presentation layer.

The Y-Server as well manages the various user input by providing features such as continuous speech recognition, speech command and control, or DTMF decoding for the use in telephony systems. Additional features could add authentication methods such as speaker recognition.

3.3 Evaluation of existing sound servers

The implementation of the Y-Server most likely will follow the design of already existing sound-server solutions. Possible candidates are aRTs [8], a sound server and synthesis engine used by the KDE desktop, and Jack [9], a relatively new sound-server optimized for real-time performance with advanced routing features. aRTs also provides network transparent operation via MCOP, a CORBA-like interface adapted for multimedia. Although aRTs already provides more than the core features one would expect from a versatile sound server, the concept of Jack promises better performance: Instead of using a communication protocol, applications for Jack plug-in directly into

the server engine using a call-back interface. It has to be evaluated if the two approaches can be combined and extended with the most important features required for the Y-Server, such as speech recognition, advanced sound synthesis and transformation, or sound spatialisation. The lean and fast design of Jack is quite appealing, and offers better performance and expandability over the rather large and slow aRTs server.

There also exist several other sound server solutions for the Unix platform, such as the Enlightenment Sound Daemon [10] used by the GNOME desktop, or the Network Audio System [11], which focuses on network transparency for the use of audio in X-Terminals. It is remarkable though that there exist that many solutions for the same problem, which obviously all don't seem to satisfy the basic needs for such a system. Since too many sound-servers raise the same problems as concurrent audio applications, .it is necessary to find a common sound server solution, which is generally accepted.

3.4 Interface Layer: Widget Libraries & APIs

Based on the possibilities offered by the Y-Server, which provides the pool of basic technologies necessary for the rendering of an auditory display, the interface layer enables the access to this rendering layer through a series of libraries. Regarding the modular architecture of the Y-Server, there is also a need to define a set of common programming interfaces (APIs) in order to provide the same interface for the different possible modules. This then enables the replacement of the speech synthesis engine manufacturer for example, while maintaining the same interface. Existing APIs, such as the Linux Audio Plug-In Architecture LADSPA [13] should be included, where on the other hand still not existing APIs, such as a common Speech API need to be developed entirely from scratch.

Additionally, this layer should implement or adapt interface libraries of the various existing basic auditory design elements, such as parameterized auditory icons [14], Earcons [15], common auditory interface widgets (acoustic progress-bars [16], etc.) or further experimental designs. Using these basic components, higher level embeddable components such as sonification tools or voice browsers can be constructed. With a growing set of those tools brought together in a collection of AUI widget libraries - which then can be extended or modified - developers of auditory applications are not forced to invent the wheel again and again.

Basically such widget libraries need to provide suitable solutions and templates for the most common dialogs in interactive auditory systems. This should include basic menus, forms and even some direct manipulation interfaces. On the output side this should provide known sonification and data auralisation methods in order to facilitate auditory feedback design. This could also include some algorithmic composition methods to exploit music for auditory feedback as well.

Another crucial task is the choice or definition of the required communication protocols between Y-Server and applications, the required file-formats and interface construction languages. For the speech interface components, there already exist approaches such as VoiceXML [5], which might be also extended for the use in not purely speech based auditory interfaces. Similar XML based formats might be suitable for most of the other interfaces.

4. Implementations and Applications

An initial implementation of the Y-Windows framework should be realized on the Linux platform, mainly because of two complementary reasons: First of all Linux already offers a variety of open-source components, which can be modified for the use within this environment. On the other hand Linux still lacks of some common audio, multimedia and speech APIs, which could be developed as part of this project. Of course the framework should be ported to other mainstream operating systems such as MacOS or Windows as well, where it can adopt native interfaces such as QuickTime or DirectX for an initial implementation.

The first development steps need to focus on the refinement of the Y-Windows concept itself. Then the major requirements and tasks need to be defined, which includes the definition of the central API interfaces and communication protocols and a prototype implementation of the Y-Server.

The main application areas of the Y-Windows framework in its initial phase might be basically the research & development of auditory interfaces themselves, since it should allow the faster prototyping of test setups. From a user's point of view though, the main application areas will be auditory interfaces for mobile devices, acoustic monitoring applications, or auditory desktop applications for the sight-disabled. Especially mobile devices, such as PDAs or SmartPhones with their limited screens and keyboards would profit definitely of such a scalable system. The pure acoustic nature of the framework might also be interesting for various embedded applications within industrial or home devices, which increasingly are designed using the embedded Linux platform. Of course, the framework can also be used for the advanced acoustic enhancement of GUI desktop applications.

A possible future distribution of the Y-Windows framework should also include a set of demo applications such as a voice-browser, sonification tools, acoustic background monitors and an auditory "desktop" shell. While such applications initially will be probably mainly reference implementations for developers, the final goal should be the creation of a complete auditory environment based on the Y-Windows framework.

5. Conclusions

The Y-Windows concept is far from being completely thought through and there are obviously several odds and ends in some details of its design, but it is mainly meant to initiate a discussion in that direction. The advantage though of an implementation of such a framework would be the creation of a common pool of current knowledge in AUI research and design, allowing the easier access to and extension or improvement of this knowledge. The author therefore hopes that some researchers and developers from the auditory interfaces community will join this effort

Together with the presentation of this article there will be created an open-source project page in order to provide the necessary collaborative tools for this task. This includes the installation of a mailing-list, CVS code repository and additional web-based documentation including the further development of this concept paper.

6. References

[1] Open Group Inc.: X11R6 URL, http://www.x.org/last_release.htm

[2] Kaltenbrunner, M.: “Auditory User Interfaces for Desktop, Mobile and Embedded Applications”. Diploma Thesis, FH Hagenberg, 2000

[3] Cook, P. & Scavone, G.: “The Synthesis Toolkit”, URL, http://ccrma-www.stanford.edu/software/stk/

[4] Sun Microsystems: Java Speech API v1.0. URL, http://java.sun.com/products/java-media/speech/

[5] VoiceXML Forum, W3C: VoiceXML 2.0 Specification, URL, http://www.w3.org/TR/voicexml20/

[6] Raman, T.V.: “Emacspeak – A Speech Interface”, Proceedings of the Conference on Human Factors in Computing Systems. p66ff, ACM Press, 1996 URL, http://emacspeak.sourceforge.net/

[7] Kaltenbrunner, M. & Huxor, A.: “Marvin: Supporting Awareness through Audio in Collaborative Virtual Environments”, In: Earnshaw, R. & Vince J. (eds.): “Digital Content Creation” p294ff, Springer Verlag, Hamburg 2001

[8] aRTs, analog realtime synthesizer, URL, http://www.arts-project.org/

[9] JACK audio connection kit, URL, http://jackit.sourceforge.net/

[10] Enlightenment Sound Daemon, EsounD. URL, http//www.tux.org/~ricdude/EsounD.html

[11] Network Audio System, URL, http://radscan.com/nas.html

[12] PortAudio – An Open-Source Cross-Platform Audio API. URL, http://www.portaudio.com/

[13] Linux Audio Developer’s Simple Plug-in API, LADSPA, URL, http://www.ladspa.org/

[14] Gaver, W.: “Synthesising Auditory Icons”, Conference proceedings on Human Factors in Computing Systems, Amsterdam 1993

[15] Brewster, S.: “Using Non-speech Sounds to Provide Navigation Cues.” ACM Transactions on Computer-Human Interaction, p224ff, 1998

[16] Crease, M. & Brewster, S.: “Making Progress with Sounds. The Design Evaluation of an Audio Progress Bar”, Proceedings of the International Conference on Auditory Display, 1998

Marvin: Supporting Awareness through Audio in Collaborative Virtual Environments

Martin Kaltenbrunner
FH Hagenberg
Hauptstrasse 117
A-4232 Hagenberg
Austria
modin@yuri.at

Avon Huxor
Centre for Electronic Arts
Middlesex University
Cat Hill, Barnet
United Kingdom
a.huxor@mdx.ac.uk

0. Abstract

This paper describes Marvin, an awareness support agent written in Java. The system provides audio cues, text-to-voice and voice recognition, and currently operates as a bot in ActiveWorlds, an Internet-based, shared 3D virtual environment. The ActiveWorlds space was designed to facilitate chance encounters between members of distributed work groups, and Marvin was written to overcome the problems that arose in use. Audio allows us to free the user from both attending to the screen, and also from being present at only one virtual location within the world, drastically enhancing the chances of encounter. It also blurs the boundaries between the virtual and physical workplace.

1. Introduction

This paper builds on recent work that explored the use of AlphaWorld (AW), a simple multi-user Internet-based virtual environment to support chance encounters (Huxor 1999). That is, rather than being a virtual space in which scheduled meetings for distributed teams might occur, it supports the unplanned meetings that take place in the conventional workplace. These meetings, often occurring in corridors, or by the coffee machine, have been shown to be very important for the functioning of an organisation team (Backhouse & Drew 1992) and risk being lost in a distributed organisation. The informal flow of information, of contacts to maintain trust, are central, and are often not fulfilled by existing software tools such as email, which require an intent to communicate.

As part of a longer research program to develop a 'virtual Centre for Electronic Arts', a shared virtual space was build in AlphaWorld, the oldest and largest world that is available from the ActiveWorlds[1] client. This client supports easy navigation of the space and real-time text-chat between users. The virtual office 'space' within AlphaWorld was created with the aim of supporting new working forms in which people are working not only in the traditional office, but also at home, on the road, at customer premises or other venues. The authors use the space to access collaborative documents that are stored in an Internet-based server called BSCW[2], which allows for collaboration across institutions, and to informally meet both colleagues, and 'weak ties' (those people who pass through a space but are not immediate colleagues). These encounters were spatially managed, in that task-related content were placed in stable rooms in the virtual office, content which drew users relevant to these tasks. Encounter occurred visually, in that each user has an avatar that can be observed passing through

[1] http://www.activeworlds.com
[2] http://bscw.gmd.de

the space. This ActiveWorlds virtual office[3] has been active for some years now, and has proved itself useful in supporting both chance encounters and weak ties, as intended. However a number of problems arose, but the most important of these were those that prevented chance encounter taking place as often as possible.

1.1 Problems of the CVE

The space aimed to support encounter and awareness of other users, but awareness failed in many cases. These failures can be grouped into three types:

0. The user is at the machine, but the task they are undertaking employs the full screen, so that the virtual space window is hidden below that currently in use.
1. The user is nearby the machine, but not attending to it as they are reading a paper document, on the telephone or talking to colleagues, for example.
2. The user is away from the machine visiting another office, en route elsewhere, at lunch etc.

These problem of missed encounters are crucial, due the critical mass which is required to make such social media function. As I missed various visitors, they appear to visit less often, the chances for encounter drops further, and a vicious circle develops. This must be broken to allow for the communication process to be enabled.

It was recognised that the problem arose from a contradiction between the goals of the space, in terms of supporting mobility in the workplace, and its actual effects. Although the shared virtual space was employed to support a more flexible form of working, one in which a user is not tied a single office desk, it actually bound the user in two important ways. Firstly, they are tied to their computer monitors so as to see any passing avatars in the shared space, and secondly, they are further tied within the virtual space to a single location. The first problem calls for a means of indicating presence in the world that is non-visual, the second a reconsideration of the nature of presence in virtual spaces.

It retrospect, it seemed unnecessary that we adopt all the constraints of structuring action that are derived from the spatial metaphor. In the physical world users are adopting many other techniques, such as mobile phones, to overcome these, so why re-introduce them in virtual spaces? That is, can we separate the positive from the negative aspects of the spatial metaphor, to give users more benefit, a common problem in metaphorical interface design? It was concluded that it is the spatial management of tasks that remains central to collaborative spaces, and that we can be more flexible in our use of the term ‘avatar’. If each user has many avatar proxies, they can have a ‘foot in many camps’, depending on the range of tasks they are working on at the time.

[3] The office can be visited in AlphaWorld at co-ordinates 188S 34E

It was decided to address these concerns through two means:

1. Use of audio as an awareness mechanism, one that allows the user to not attend to either the screen or, if working with another application onscreen, to ActiveWorlds.
2. Allow users to have multiple presence in the virtual world, each with an associated set of audio cues. Just as we can listen to the physical office next to our own for cues, so audio facilitates attention to diverse spaces.

2. Audio Cues and Audio Spaces

Sound has been a neglected part of interface and systems design, but has recently seen a growth in interest. The requirements of an audio interface to a shared virtual environment can draw upon a wide range of work on audio use in computing. In one major area of research, sound is being used to support the standard GUI interface (Gaver 1989), especially for small screen PDAs, and mobile telephone access to information services (e.g. Brewster 1998), as such devices are seen as a major market segment in the future. Other work has investigated how real-time audio links between users and physical spaces can support distributed workgroups (Mynatt et al. 1998, Hindus, et al. 1996), and Sawhney & Schmandt (1999) have looked at how mobile access to services can be supported by a wearable headset/microphone device.

Our work on extending ActiveWorlds looks to all these. Many of the issues that apply to audio interfaces, namely the design of audio icons, sound effects and text-to-voice use, apply to supplementing the current AlphaWorld visual interface. Equally, as our concern is with facilitating the maintenance of a social sense throughout a distributed workgroup, lessons from audio spaces are relevant. For Hindus et al. (1996) found that audio only spaces can lead to 'social spaces', and although ActiveWorlds currently supports text-chat only, we are looking to augment this with voice over IP (see, for example, Onlive Traveler[4], a shared 3D world that employs voice rather than text-chat).

The audio supplement to ActiveWorlds (AW), the browser technology for AlphaWorld, is a system called Marvin, described in more detail below. Marvin creates a presence in various points in AW worlds, and gives audio cues representing various events in the world. In addition it uses text-to-speech to allow users to listen in on chat in the worlds, and we are investigating voice recognition technology to support hands-off navigation within them.

The detailed aspects of the sound design it employs are, where possible, drawn from published empirical results. For example, door knock and door opening worked well to indicate comings and goings (Cohen 1994), and such door sound cues are used to represent other users entering or leaving the area of interest. James (1996) notes that users preferred natural sounds to artificial sounds, even when poorly chosen, and typical sounds (e.g. the high-low tones of a standard doorbell) were identified quicker than atypical sounds (Ballas 1994), so these have also been used. However, we have also been aware that it was also discovered that, if too similar, it could confuse users between activities in the virtual and the physical spaces. With regard to text-to-voice, text-to-speech need 'prefacing' so users could be prepared to hear the main message properly (Cohen 1994). Also, James (1997) found that multiple voices were valuable

[4] http://www.onlive.com

when used to speak online documents, to mark macro-structures, such as headings, in a document. Although our application differs from James', we have used different voices to represent different users in the space, and we can easily add prefacing comments before the quoted text-chat.

This audio complement to the standard AW interface provides additional functionality over the 3D space, in that it allows users to be in more than one place at a time. Cohen (1994) found that priority attribution for notification seems task dependent: certain other users and/or files were important at different times. Exploiting our ability to attend to multiple audio channels, users can have agent bots present in various places in the virtual environment, depending on which tasks are relevant at the time. The *locales* (Fitzpatrick et al. 1996) that maintain task-related content and encounters can thus act as a simple mechanism for managing the setting of priorities within the audio awareness component. That is, we can avoid the situation where the user must modify notification priorities manually, by letting these be managed by a change of place, just as different physical spaces create new affordances. This approach follows Erickson's (1993) idea that increasingly, as we move to shared applications, the Interface can best be understood as an Interplace, a place or places that structure activity.

3. Marvin: An Auditory Awareness Bot

3.1 Implementation

The Marvin bot application is a simple programmable agent, which can enter multiple information servers (such as AlphaWorld) as a proxy and report noteworthy events via speech and audio to the user. Since with the help of the robot one is constantly aware what is going on, he can if desired then directly turn his attention to the appropriate application and enter the locale instead of the robot. The proxy itself isn't an independent intelligent agent; it is an extension to the user's senses for extended awareness of events in info-space.

Marvin is implemented in the Java programming language, version 1.1 or above. This decision was made mainly because of Java's platform independence and the availability of various multimedia features like Speech, Sound, and Media APIs. We use the IBM ViaVoice Runtime as the speech recognition and synthesis engine. The application runs and is developed on Windows NT, but it should also run on a Linux or Solaris machine since there is also Java compatible speech recognition software available for this platform, though we never tested it. If IBM also releases a Speech API package for ViaVoice on Apple Macintosh, Marvin should also run on this platform without any further modification.

The application consists of two layers:

1. The Marvin kernel, which provides the basic features like speech recognition and synthesis, sound output, logging and so on. Upon start-up this core loads all present plug-ins and starts them as independent threads. The plug-ins then can use Marvin's event processing interface.

2. The plug-in interface, which allows the easy addition of robots for any information service. At the moment we have only implemented an Active Worlds Robot. But due to this architecture and the fact that Marvin is released under the GPL (Gnu General Public License) it is easy for third party developers to add their own plug-ins for various other network information services or multi-user environments, such as BSCW/Nessie, IRC or ICQ.

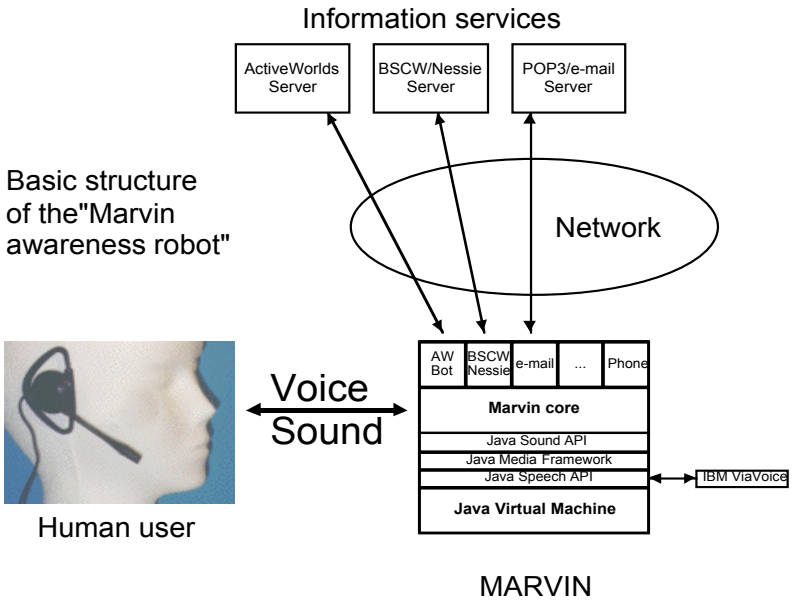


Fig. 1 Structure of Marvin system

3.2 Event processing functionality

As mentioned above, the Marvin kernel provides basic audio notification features for all available plug-ins. This means the simple playback of audio file or the output of synthesised speech, also provides an interface for the voice control of all components. The plug-in applications cannot directly access the sound and speech methods. These methods are combined in a central event-processing interface. Depending on their priority, events are either only logged to a file (no priority) or the user is informed with speech and audio (high priority). The table below shows the exact priority scheme we currently use. Higher priority level events implicitly include the processing of their lower priority levels.

no priority	logged to file
low priority	logged to screen
medium priority	audio clip
high priority	speech output
urgent priority	mobile notification (not yet implemented)

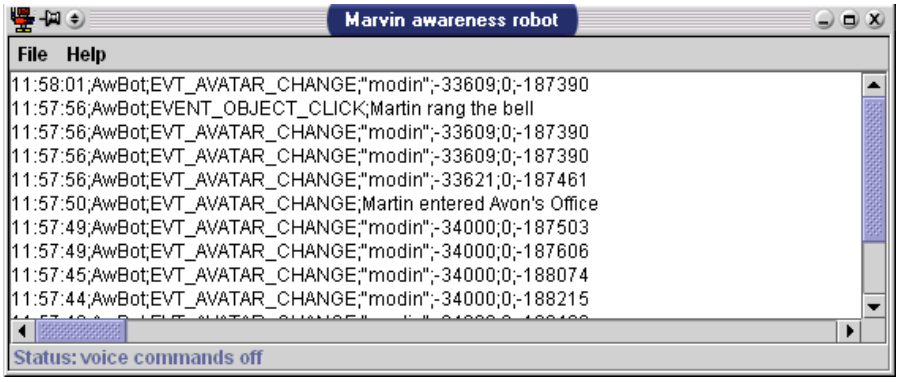


Fig. 2 Example Log of Events

3.3 The Active Worlds (AW) Plug-In[5]

The AWBot plug-in uses Thierry Nabeth's Java native interface for the Active Worlds Software Development Kit. Applications implementing this interface can place a remote-controlled avatar into the Active Worlds space, which can interact with other persons or events in this virtual environment. Once such a robot is placed in a certain predefined area of a world within the AW server, it will notice any event in its surroundings. These events are then caught be the robot application, analysed and then processed by the Marvin core according to their priority.

Since, as discussed above, a robot only can notice events in its nearest surroundings, multiple instances of the robot – called probes – were created and placed in various areas of the AW space. To ease the modification of the robots, all the variable parameters such as position, user names or sound files are stored in separate configuration files that are processed upon start-up.

3.4 Marvin in Use

Marvin is a separate application that can be started to supplement the standard Active Worlds browser. The user can then assign a number of proxy avatars of Marvin in places of interest. For example, in addition to one's own virtual office, one might place proxies in parts of the space that link to content relevant to ongoing projects. When another user enters the space, an audio cue of a door is heard, and Marvin greets them using both text-chat in the window, but also with text-to-voice. The former lets the visitor know they have been acknowledged, the text-to-voice allows any user with Marvin to hear the greeting (which includes the name of the guest), letting them know who has arrived. Similarly, users departing from the zone specified are giving a farewell in the text-chat box, and this is also spoken aloud. Clearly, I could be working away from my machine but still be aware of activity in the space, and respond accordingly. It is also significant that it allows persons sharing a physical office to be mutually aware of activity in the others spaces, adding to a sense of working community. Both authors share a physical office and this has been a noticeable result in adding to the overall sense of awareness within the workplace.

[5] Because the Active Worlds SDK is only available for the Windows operating systems, this plug-in is not platform independent.

Marvin also repeats (with text-to-voice) the words that have been 'spoken' (using AW text-chat) by other users, so that one can listen into a conversation while doing other tasks, and jump in when appropriate. This feature already proved interesting in the limited time that the authors were using Marvin. It allowed a conversation between two colleagues: one in a different part of the same building, and another across the city, to be followed (even though it was occurring on a machine across the room). The avatar, representing Marvin, assured that these other participants were aware of my proxy presence. For certain designated individuals, those that are particularly relevant to our collaboration, specific voice types (age and gender) have been assigned so that they can be recognised.

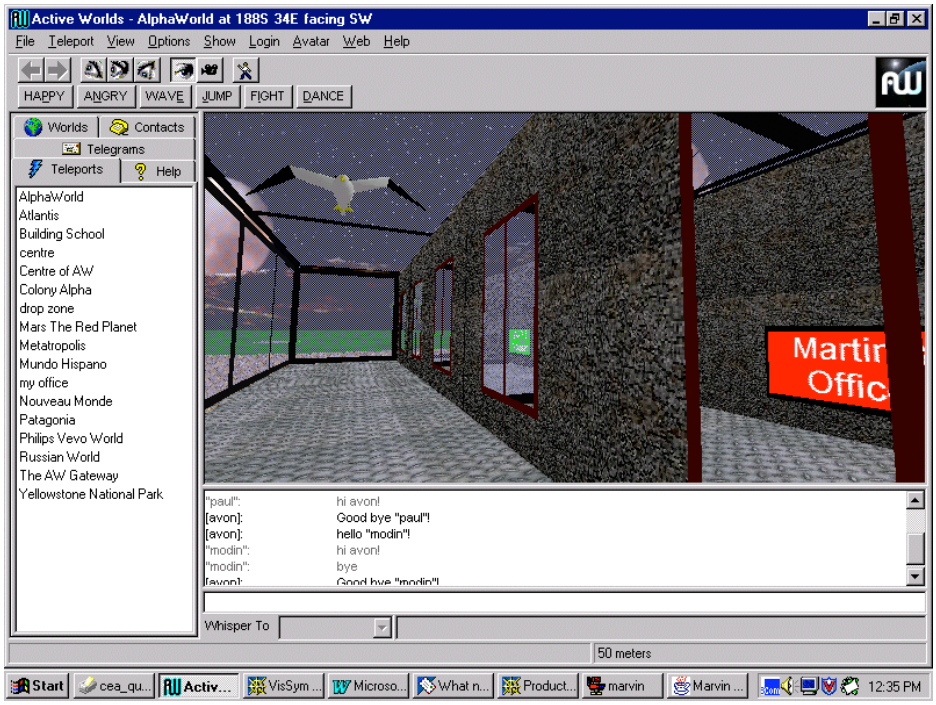


Fig. 3 Marvin in AlphaWorld

Other sound cues indicate if other users interact with objects that belong to me in the virtual space, such as those that hyperlink to web content. Thus, I can also be aware of task related work activity by other users in addition to supporting personal encounters.

The figure above shows Marvin, whose avatar is a bird, hovering in the virtual space. The text-chat box below the space captures the interaction between Marvin, who as my proxy uses the name [avon][6], and visitors.

[6] The square brackets indicate, in AW, that an avatar is a 'bot'

4. Issues Arising during Use

In addition to providing an auditory interface to overcome the problems identified in the original AW space, the implementation and use of Marvin has lead to a number of issues.

Firstly, it is easier to create 'mixed' spaces with an audio enhanced interface, as sound emanate from the both physical and virtual and are perceived in a similar manner. This differs from the visual aspect of the virtual space, which presently has a very distinctive appearance from physical space. It was already noticeable that during limited use, this similarity creates a different relationship to the virtual environment and events within it. The door sounds and phone ringing sounds feel similar to those of doors and phones from adjacent rooms. And it is noteworthy that, increasingly, the events in the physical world to which we respond have a virtual component. For example, an informal study of the soundscape of CEA was undertaken to assist in the design of Marvin's cues. It was found that the most important cues included phone rings (whether answered or not), and telephone answering machines. This hints at a view in which the conventional distinction between virtual spaces and physical ones break down, for sound has a 'physical' sense (Gaver 1993).

Secondly, audio also has problems that arise from its being so pervasive, namely annoyance of the user and others, and it particularly presents a problem for privacy. However, it was found that features from the virtual space could help in the management of these. The fact that the interactions are spatially set assists in managing the sense of what is public and what is private due to the ownership of spaces. Also, having avatar assists in preserving symmetry in an online interaction: if one has a proxy presence in another space, it is visible through the personification of the avatar. Not knowing who is listening in on an audio space is a frequent complaint. It would be possible for a user to have multiple presence within ActiveWorlds, but use an invisible avatar, but this would give an impression of surveillance. Therefore, we have personalised the various proxies of the user as one of the standard AW avatars. In the current version, it is a bird, as this conveys the sense of being aloof, of partial presence.

Finally, Gaver (1993) reports work in which blindfolded students could orient themselves by 'acoustic landmarks', resonance, echoes and near/far sounds. This suggests that it may be valuable for the sounds emanating from a virtual space to represent its visual spatial form to assist in identification. Thus we may have a large or a small virtual space within the world. One possibility that this suggests is that we can use different acoustic qualities to 'tie' sounds, which may be from a multiplicity of locations within the virtual space, to one in particular. For example, a large room in AlphaWorld would have reverberation, which would affect the sounds from it, and could be recognised as such by users directing their response to the appropriate place.

5. Conclusions and Future Work

This paper has argued that:

1. Audio can improve awareness in collaborative 3D virtual environments by freeing the user from being bound to the monitor.
2. Audio cues can also support multiple proxies for each user in the virtual space, allowing users to have access to many work contexts at any moment. Each place in the environment, however, maintains its spatial management, its sense of *locale* (Fitzpatrick et al. 1996).
3. This idea of using locales to manage different notification priorities illustrates the notion of Interplace replacing the conventional interface. Users do not modify a menu, they just move from one place to another.

In the introduction it was pointed out that the original AW space failed in supporting awareness in various ways. The work described above sought to address the problem for the user who is nearby to their computer, but future work seeks to extend the principle to support users who are away from any machine. It is inspired by work that has similar aims, such as the Nomadic Radio project of Sawhney & Schmandt (1999), but they use specialised hardware devices. We aim next to explore how awareness cues from the virtual space can be made available to users away from their physical office, on route to a colleague, in the coffee area, etc. As mobile phones have become so ubiquitous, it seems likely that they will become a natural portal to online resources. This can only contribute further to blurring of the distinction between the physical and the virtual, creating a unified 'space of work'.

Acknowledgements

The authors would like to acknowledge the efforts of Thierry Nabeth of the Department of Technology Management at the Centre for Advanced Learning Technologies, INSEAD, France. His Java port of the AlphaWorld SDK made the Marvin system possible, and his speedy updates from our comments were invaluable.

Bibliography

Backhouse, A. & P. Drew (1992) "The design implications of social interaction in a workplace setting." *Environment and Planning B: Planning and Design*, 19: 573-584.

Ballas, J. A. (1994) "Delivery of Information through Sound." In: Gregory Kramer (ed.) *Auditory Display*, SFI Studies in the Sciences of Complexity, Proc. Vol. XVIII, Addison-Wesley, pp. 79-94.

Brewster, S.A. (1998). "Using non-speech sounds to provide navigation cues." *ACM Transactions on Computer-Human Interaction*, 5(2), pp 224-259.

Cohen, J. (1994) “Monitoring Background Activities.” In: Gregory Kramer (ed.) *Auditory Display*, SFI Studies in the Sciences of Complexity, Proc. Vol. XVIII, Addison-Wesley, pp. 499-531.

Erickson, T. (1993) “From Interface to Interplace: The Spatial Environment as a Medium for Interaction.” *Proc. of Conf. on Spatial Information Theory*. Heidelberg: Springer-Verlag.

Fitzpatrick, G., Mansfield, T. & S. M. Kaplan (1996) “Locales Framework: Exploring foundations for collaboration support.” *IEEE Proc of the 6th Australian Conf on Computer-Human Interaction OZCHI'96*, Hamilton, NZ, pp. 34-41.

Gaver, W. W., (1989). “The SonicFinder, a prototype interface that uses auditory icons.” *Human Computer Interaction* (4): 67 - 94.

Gaver, W. W. (1993) “What in the world do we hear? An ecological approach to auditory event perception.” *Ecological Psychology*, 5(1): 1-29.

Hindus, D. et al. (1996) “Thunderwire: A Field Study of an Audio-Only Media Space.” *Proc. ACM Conf. on Computer Supported Cooperative Work (CSCW'96)*, pp. 238-247.

Huxor, A (1999) “The Role of 3D Shared Worlds in Support of Chance Encounters in CSCW.” In: Vince, J. & Earnshaw, R. (eds.) *Digital Convergence: The Information Revolution.* Springer-Verlag

James, F. (1996) “Presenting HTML Structure in Audio: User Satisfaction with Audio Hypertext.” *ICAD '96 Proceedings*. Xerox PARC, 4-6 November 1996, pp. 97-10.

James, F. (1997) “AHA: Audio HTML Access.” *Proceedings of The Sixth International World Wide Web Conference.* Santa Clara: CA, pp. 129-139.

Macaulay, C. and Crerar, A. (1998) Observing the Workplace Soundscape: Ethnography and Interface Design. *Proceedings of the International Conference on Auditory Display (ICAD ’98)*, Glasgow, November 2-5.

Mynatt, E. D., Back, M. & R. Want (1998) “Design Audio Aura.” *Proceedings of the Computer-Human Interaction (CHI) Conference*, Los Angeles, April 1998, pp. 566-573.

Sawhney, N. and C. Schmandt. (1999) “Nomadic Radio: Scaleable and Contextual Notification for Wearable Audio Messaging.” *ACM SIGCHI Conference on Human Factors in Computing Systems*, Pittsburgh, May 15-20.

Multiple Presence through Auditory Bots in Virtual Environments

Martin Kaltenbrunner
FH Hagenberg
Hauptstrasse 117
A-4232 Hagenberg
Austria
modin@yuri.at

Avon Huxor (Corresponding author)
Centre for Electronic Arts
Middlesex University
Cat Hill, Barnet
United Kingdom
a.huxor@mdx.ac.uk

0. Abstract

This paper proposes that virtual environments that aim to support mutual presence for distributed work groups should allow for multiple partial presence. The introduction of teleworking, and the massive uptake of mobile phones have addressed a need for people to be virtually present (if not physically present) from many spatial locations. Conventional virtual environment systems, however, seem fixed to a notion of presence as being tied to 'being in' a specific place. It is this view of VR that we have modified with collaborative working in mind.

Specifically, we have extended a graphical Internet-based 3D world to allow for users to have multiple, 'proxy', avatars which provides a sense of partial presence in many locations in the virtual world, and/or in many different virtual worlds. Presence is now not tied to place, but to awareness of events. Users are connected to their proxies by audio cues, allowing for multiple locations to be attended to at once, creating a form of presence appropriate to the workplace.

1. Introduction

In previous work (Huxor, 1999), an application of shared virtual spaces to support distributed working was described. It is known that 'chance encounters', the unscheduled meetings between people that occur in such places as corridors, are crucial to the functioning of organisations (Backhouse & Drew, 1992). It is these encounters that any collaboration tool must support in addition to the more formal aspects of collaboration. It was suggested that such encounters can be better implemented in a shared 3D space, as opposed to than other collaborative web-based environments due to the importance of spatiality in managing interpersonal encounter. This idea has been explored through an example prototype, which employs the ActiveWorlds (http://www.activeworlds.com/) shared 3D-world technology, hyperlinked to work-related content held in BSCW (http://bscw.gmd.de/), a web-based document and collaboration management system. Certain physical spaces, known as 'locales' contain content and other people that are associated with a particular work-task: an association which manages encounters, and equally virtual 'locales' can facilitate online encounter between distributed work-groups.

One author has used this system for over three years to support such distributed working practices. It has indeed facilitated chance encounters not only for myself, but also for other colleagues. Although the system proved valuable, it was found that many potential encounters with other users were missed, as either:

- I had filled my monitor screen area with another application that required the space, making the virtual world not viewable, or
- My gaze was directed elsewhere: at a paper document, at a visitor to my physical office, out the window, etc.

Such concerns, however, also point to a larger issue. Unlike many virtual activities (such as gaming or training simulations) collaborative working requires that one has a variety of presences - of awarenesses. One can be both in a physical office and engaged in a telephone conversation: in this situation where is one's presence? It seems to be in both 'places' at once.

The problems of missed encounters were addressed by the addition of audio cues, so that activity in the virtual world generates various appropriate sounds that can be heard when the user is unable to view the 3D world. A user can set up avatar representations in a number of locations in the virtual world, each one specific to an ongoing work task of interest. While otherwise engaged, this user can now follow much of the 'coming and goings' and other activity that is taking place in all of these locations. The identity of which location the action occurs being indicated by variations in the sound cue.

One of the surprising results is that moving away from a purely visual presence to an auditory one creates new possibilities of presence. Specifically, we have extended the system to allow for multiple 'proxy' avatars in various spaces. This would be impracticable with a conventional visual interface - one window only on the virtual world takes up enough screen real estate, multiple windows would make other work impossible.

It seems strange that as technology in the workplace tries to liberate us from being fixed to a single location, through the use of mobile for, for example, that collaborative virtual environments should try to re-impose that constraint. The use of audio, and its ability to carry multiple steams, allows us to exploit the potential of VR technologies and support partial presence in many locations at the same time. With this in mind, we have extended the audio features of AlphaWorld. A user can now work in the physical office, and need not even have the 3D-world window on-screen. They can set a number of probes, or proxies, of their avatar in a variety of places. Each place in the environment is associated with particular content, in that the various object in the location are hyperlinked to online content, which is stored in BSCW. Most work consists of managing a number of tasks at the same time, dealing with a variety of projects. Just consider a typical day: writing a document, while reading and responding to new emails, phoning a colleague about an urgent problem, and so on. The AlphaWorld virtual office that we use follows this idea, creating task-related rooms and buildings in a larger social community.

Each proxy is a bot that generates suitable audio cues for important activities at each location, such as another user entering or leaving, another user text chatting, or a hyperlinked object being activated. This audio bot system is called Marvin. Figure 1 shows a screen shot from the AlphaWorld office, one of the proxy bots is in the space, represented by the avatar of a bird (this avatar was selected from the limited number available in AlphaWorld due to its connotation of being somewhat 'above it all', of partial presence).

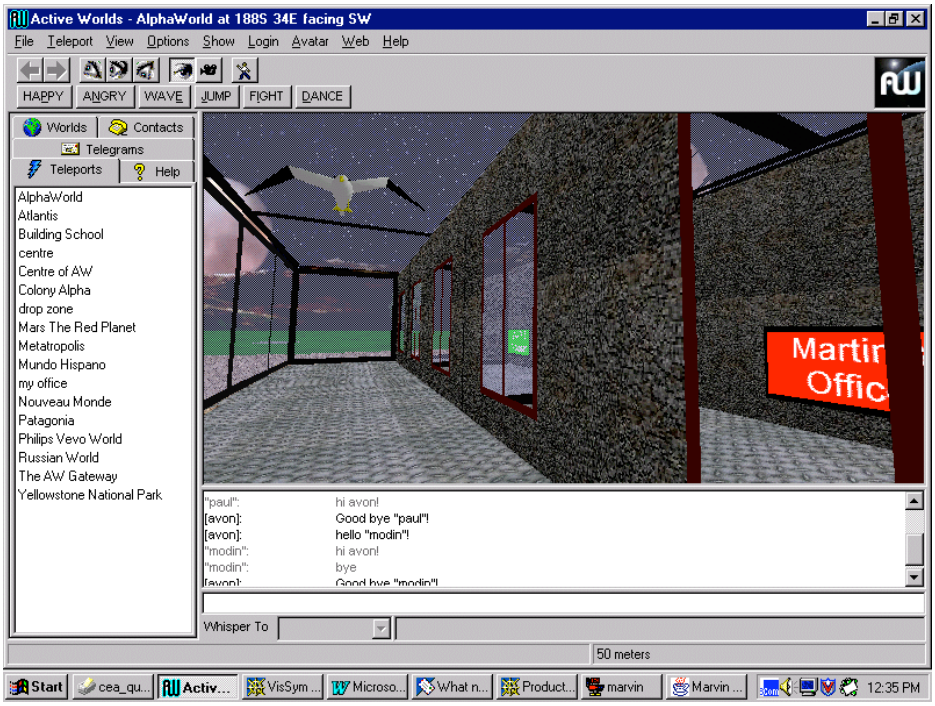


Figure 1. A Marvin proxy in the Virtual Office

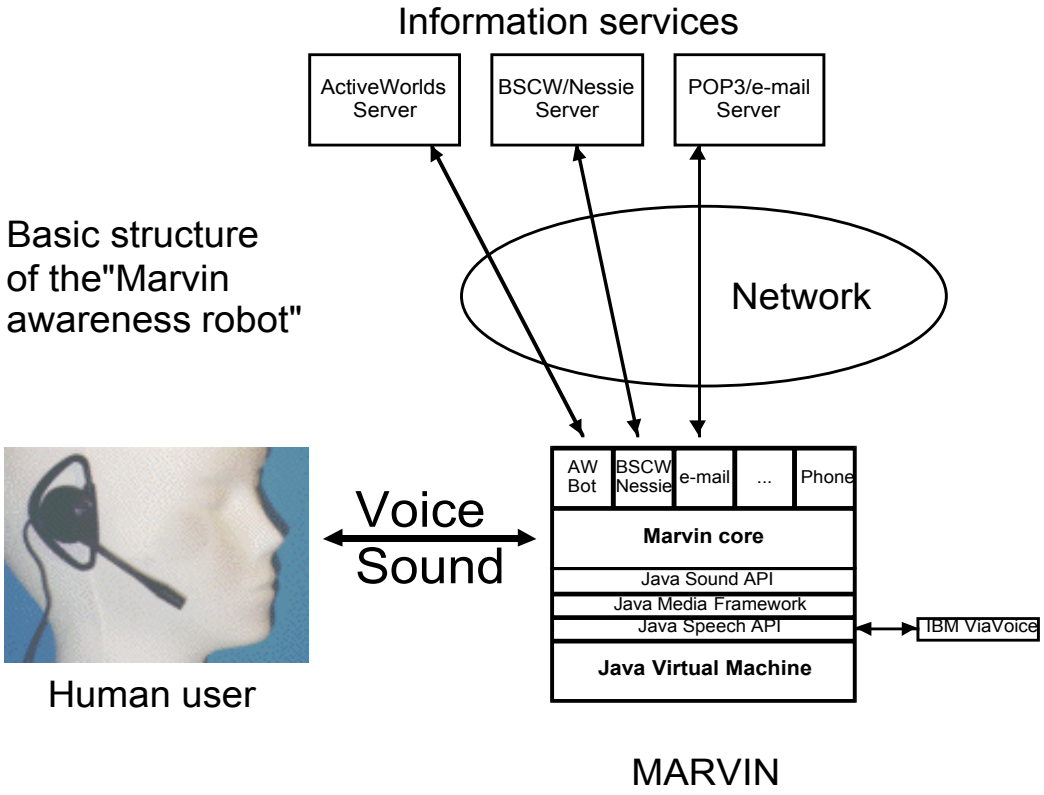
2. Marvin: an auditory bot

The Marvin bot is a simple programmable agent, who can enter multiple information servers (such as AlphaWorld) as a proxy and report noteworthy events via speech and audio to the user. Since with the help of the robot one is constantly aware what is going on, he can if desired then directly turn his attention to the appropriate application and enter the locale instead of the robot. The proxy itself isn't an independent intelligent agent, it is an extension to the user's senses for extended awareness of events in info-space.

Marvin is implemented in the Java programming language, version 1.1 or above. This decision was made mainly because of Java's platform independence and the availability of various multimedia features like Speech, Sound, and Media APIs. The application consists of two layers:

1. The Marvin core classes, which provide the basic features like speech recognition and synthesis, sound output, logging and so on. This core loads all present plug-ins after initialisation and starts them as independent threads. The plug-ins then can use Marvin's event processing methods. Depending on the event's priority, it will either only logged to a file or to the screen, or in case of the highest priority announced with speech and audio cues.

2. The plug-in interface, which allows the easy addition of any information service. At the moment we have only implemented an Active Worlds Robot. But due to this architecture and the fact that Marvin is released under the GPL (Gnu General Public License) it is easy for third party developers to add their own plug-ins for various other network information services or multi-user environments.



2.1 The Active Worlds Robot

The AWBot plug-in uses Thierry Nabeth's Java native interface for the Active Worlds Software Development Kit. Applications implementing this interface can place a remote-controlled avatar into the Active Worlds space, which can interact with other persons or events in this virtual environment. Once such a robot is placed in a certain predefined area of a world within the AW server, it will notice any event in its surroundings. These events are then caught be the robot application, analysed and then processed by the Marvin core according to their priority. Since, as mentioned above, a robot only can notice events in its nearest surroundings, multiple instances of the robot – called probes – were created and placed in various areas of the AW space. To ease the modification of the robots, all the variable parameters such as position, user names or sound files are stored in separate configuration files that are processed upon start-up.

3. Discussion

We have used readily available Internet 3D world technologies to explore the use of desktop VR for collaborative working applications for some years now. Throughout that time one issue constantly arises. The technology has been primarily developed for certain simulation tasks, for training and gaming. The notions of immersion and presence, which are appropriate for such applications, differ greatly from the needs of CSCW (computer-supported collaborative working) in which the virtual environment is part of ongoing work activities. Indeed, it may be better to consider such systems as a form of augmented reality, as users always require 'one foot in the physical world'.

Experience of the use of the audio bot system has confirmed many of our intuitions: Not only of the importance of multiple presence, but of the power of the audio channel. The response to sound is so very different to that of visual information. A virtual doorbell

sounds like a physical one – both has the same form of waves in air hitting the ear. A virtual visual door image, however, does not yet have that same equality of experience. In future work we hope to explore what the nature of this audio mediated presence. Two possible factors seem worth investigating. Firstly, the role of social presence (as discussed by Towell & Towell 1997), that is, to what extent is a sense of being there sustained by the possibility of interaction with other (interesting) users. Or secondly, could it be that the use of audio changes the nature of the space. In our own experience is seems as if the various locations which contain proxies are rooms next to my physical office. Just as I can overhear activity in physical rooms next to my own office, so it is with the virtual rooms. They often feel alike, as if both forms of space are in constellation around a primary physical location. It is hoped that extended use of the system will address which of these, or other factors, are crucial.

Acknowledgements

The authors would like to acknowledge the efforts of Thierry Nabeth of the Department of Technology Management at the Centre for Advanced Learning Technologies, INSEAD, France. His Java port of the AlphaWorld SDK made the Marvin system possible, and his speedy updates from our comments were invaluable.

Bibliography

Backhouse, A. & P. Drew (1992) The Design Implications of Social Interaction in a Workplace Setting. *Environment and Planning B: Planning and Design*, 19: 573-584.

Huxor, A (1999) The Role of 3D Shared Worlds in Support of Chance Encounters in CSCW. In: Vince, J. & Earnshaw, R. (eds.) *Digital Convergence: The Information Revolution.* Springer-Verlag

Towell, J. & Towell, E. (1997) Presence in Text-Based Networked Virtual Environments or "MUDS" *Presence* 6(5) 590-595.

Public Sound Objects

A shared musical space on the web

Álvaro Barbosa, Martin Kaltenbrunner
E-mail: {abarbosa; mkalten}@iua.upf.es
Music Technology Group - Pompeu Fabra University
Passeig de Circumval·lació 8 - 08003 Barcelona, España
http://www.iua.upf.es/mtg/

0. Abstract

In this paper we describe "The Public Sound Objects" project and its context. This project, which is currently under development, intends to approach the idea of collaborative musical performances over the Internet, going beyond most common paradigms where the network is mainly used as a channel to provide a connection between performative spaces. At its final stage this system will provide a public performance space within the network where people can be found participating in an ongoing collaborative sonic event. The users connected to this installation are able to control a server side synthesis engine through a web-based interface. The resulting "Sound Objects" form a sonic piece that is then streamed back to each user. The user takes the role of a performer and his contribution has a direct and unique influence on the overall resulting soundscape. This ongoing event is also played back at the installation site in the presence of a live audience, with added contextual elements such as sound spacialization and metaphorical visual representation of the current participants.

1. Introduction

Computer-Supported Cooperative Work (CSCW) is one of the major research fields in modern information society, and recent technological advances, specially in Internet computing, have allowed computer science researchers and developers to create different types of collaborative tools, such as white boards, shared editors, video conference systems or even e-mail based systems that are already part of our daily life.

On the other hand during the last decades we heave seen artists taking cutting edge technology and using it to maximize the aesthetics and conceptual value of their work, not only by enhancing the way they traditionally create, but also by using technology as a media itself to express meaningful artistic work.

The idea of using computer networks as an element in collective artistic creation and performance (or when both come together in improvisation) was no exception.

Collaboration paradigms have great relevance in music, since traditionally music performance is a result of joint synchronous events where musicians with their individual performance contribute in real time to a final piece.

Early experiments with musical computer networks at a local area scale date back to the late 1970's in California with performances by the League of Automatic Music Composers [1].

However with the massive worldwide growth of the Internet community, characterized by users strongly moved by music in many different ways, more appealing possibilities for music composers and performers came up in the 1990's.

1.1 Collaboration over the Internet

So far in music or sonic arts collaboration experiments over the Internet the biggest breakthrough has been the capability to provide remote communication between worldwide-displaced musicians and composers. This type of connectivity tremendously enhances the traditional collaboration paradigm for music production.

Early experimental systems based on this idea go back to the early 1990's with the Craig R. Latta's NetJam [2] from Berkley University. This system allowed a community of users to collaborate producing music in an asynchronous way by automatically exchanging MIDI files trough e-mail.

Other experimental systems focused more on the idea of having synchronous performances as close as possible to a real-time situation, like the 1998 *TransMIDI* [3] system, implemented using the *Transis* multicast group communication layer for CSCW applications [4], or Phil Burk's *TransJam* [5] that also allows this kind of interaction but going beyond the MIDI format and allowing low fidelity digital audio.

Recently commercial systems based on client-server architectures allow the collaboration on music pieces using MIDI and digital audio formats. Systems such as the *ResRocket Surfer* [6] or the *Tonos* system [7] have been highly successful receiving reasonable support from music industry manufacturers.

All these systems provide effective enhancements in the process of music production, however, they are mostly oriented towards a traditional studio production environment, leaving little space for more experimental forms of Sonic Arts, and constraining the potential of what the Internet can offer as a media for artistic expression.

Very few examples can be found where the Internet's possibilities, more than just allowing remote connections between two traditional events, were embraced by the artists as elements that actually contribute to their piece.

1.2 More than just Tele-Presence

Remote performance in a live public event incorporating low-cost public domain technology is one of the most appealing possibilities provided by the Internet.

Many public events with remote presence of musicians over an Internet connection have been performed in the last few years.

Different styles of music, instruments and technical setups have been tried like the *Telemusic* and the *Piano Master Classes* by John Young and Randall Packer [8] [9], the New York University's Cassandra Project [10] or Robert Rowe's demonstration of *Real-Time Internet Multi-Channel Audio* during the 107th AES Convention [11].

Situations of remote performance raise interesting questions about performance techniques and what should be the actual remote performer's visual and sonic representation on site.

In any case serious considerations should be made when integrating the inevitable network delay as an element into the resulting sonic soundscape, like we can find in the work of Chris Shafe and Greg Niemeyer in the *Ping* sonic installation project developed at Stanford University [12].

A more complex scenario than a unilateral remote performance is a performative collaboration between two or more simultaneous events. In October 2001, during the *Networkshop* festival in Dresden, Germany, several collaborative on-line concerts based on the FMOL Virtual Music Instrument [13] took place between there and Barcelona.

The concerts consisted of improvised duets, using a peer-to-peer version of the new FMOL system [14], which supported real-time network jamming. Attained delays were in the range of 100 ms using a conventional 56 kb modem connection, providing a very good feeling of playability. This condition of immunity to network delays in FMOL music is related to the nature of the resulting soundscape.

The sound sequencing technique used in this system, based on low frequency oscillators (LFO) excitation of sound generators, creates rhythmical and melodic progressions that to some extent support flexible reaction times and short lacks of synchronicity from the performing partners.

A different approach in collaborative Internet performances is to allow free access for the Internet community of *CyberNauts* as performers in public events for live audiences.

For a regular Internet user, having the possibility to perform over the Internet with one of Tod Manchover's *Hyper-Instruments* during a Brain Opera session in Vienna's *Haus der Musik* is a key factor for the highly successful results of these artistic proposals. Collaborating with others in Jorda's FMOL [15] over the Internet in a piece that could be selected for the music score of La Fura dels Baus's *Faust 3.0* opera premiered in Barcelona's Teatro del Liceu was equally appealing.

The concept of community-oriented music itself is also an interesting open research topic. It is yet to be clarified until which extent the average internet user is prepared to participate in a creative process, contributing meaningfully to an artistic event and what kind of constraints should be considered when designing such systems.

A very fruitful discussion related with this topic was held in December 2001 in the interactivity discussion panel during the MOSART Workshop in Barcelona [16].

1.3 Shared Musical Spaces

More recently other proposals of Sonic Art in the context of community-oriented music focused on collaboration in expressive Internet sound events exclusively for Internet audiences.

These proposals are shared musical spaces where people can be found performing in collective music pieces, given that everyone should be able to choose either to participate as a performer or simply as a member of the audience.

Pioneer Internet systems that convey the essence of a virtual community space are based on the original MUD ("Multiple-User Domain/Dungeon") software developed in the early 1990's by Pavel Curtis, Xerox Corporation [17]. In a MUD or in its successors like the MOO (MUD, Object-Oriented) or the IRC (Internet Relay Chat), the participants (usually called players) tend to develop a specific language for communication and collaboration amongst themselves, that evolves to some sort of virtual social behaviour, that only makes sense in these environments [18].

William Duckworth's 1997 Internet based Cathedral [19], is one of the first interactive music works created specifically for the World Wide Web. Other relevant examples of work developed in this context are the *WebDrum*, based on Phil Burke's *Javasyn* [20], or Atau Tanaka's *MP3Q* piece on the web [21].

2. The Public Sound Objects

The *Public Sound Object* project is being developed in the Music Technology Group of the Pompeu Fabra University. The project shares common ground with many of the previously mentioned proposals for Internet based Collaborative Virtual Environments focused on sonic arts and music creation. However, in our approach we explore the concept of a shared musical space in the sense of community-driven music creation, as in an art installation that brings together both a physical space and virtual presence in the Internet, allowing synchronous interaction amongst web users.

The overall system architecture is designed along the following key aspects:
It is a public event with characteristics that should be appealing both to a "real world" live audience and for a virtual audience of occasional *Cybernauts* visiting our server.
The tele-performers' contribution to the final musical piece should be adequately constrained in a way that the overall aesthetical coherence of the piece can be guaranteed.

The system should be scalable and modular enough to allow future extension and further experiments with different setups.
Besides the system implementation, in this paper we also discuss a proof of concept interface and on-site installation prototype that we are designing and implementing in parallel.

2.1. An ongoing public event

One of the questions that stand out when designing a system with these characteristics is whether it is reasonable to consider a music piece as an event limited in time?

In fact until now most of the artistic proposals for public events designed for community performance - even the most acknowledged art pieces like the *Vectorial Elevation* by Rarrael Lozano-Hemmer [22], awarded with the Prix Ars Electronica for interactive Art in 2000 - have been developed towards an event that takes place at a specific date during a certain period of time, when the presence of a physical and/or virtual audience in a theatre-like experience is guaranteed.

We argue that it is the Internet's essence to provide permanent connectivity, thus it makes sense that a public Internet event should go on permanently, and that the audience and performers are free to join and leave at any time they want.

Therefore this event is permanent and public, since it is continuously displayed to the public both via the Internet and on the installation site discussed later in this paper. It also provides the permanent possibility for the public to choose either the performer's or the spectator's role.

2.2 Sound Objects

In this project the raw material provided to the users for their contribution to the performance are Sound Objects. The definition of a Sound Object as a relevant element of the music creation process goes back to the early 1960's [23]. According to Schaeffer's theories, a Sound Object can be defined as "any sound phenomenon or event perceived as a coherent whole (…) regardless of its source or meaning" [24].

Although there are advantages in using logical formats like MIDI from the communications point of view in distributed sound systems, defining the universe of

sound events by subsets of Sound Objects is a very promising alternative for content-processing and transmission of audio [25]. In our system a user can choose from a set of digital sound samples, provided as a Sound Object when entering a session.

From a psychoacoustic and perceptual point of view, Schaefer's definition is extremely useful, since it provides a very powerful paradigm to sculpt the symbolic value conveyed in a sonic piece. The symbolic value of the Sound Object is a key element for the construction of sonic soundscapes.

Adding metaphorical value to a Sound Objects enables the user to identify it within the piece. On the other hand the symbolic value of the Sound Object might also change depending on the context where it is presented.

In many applications such as Auditory Users Interfaces, Sound Objects must be simple and straight forward, so that there is no ambiguous understanding of what they intend to represent. However in an artistic context the scope for the user's personal interpretation is wider, therefore such Sound Objects can have a much deeper symbolic value and represent more complex metaphors.

Often there is no symbolic value in a sound, but once there is a variation in one of its parameters it might then convey a symbolic value. A typical example is the use of white noise to synthesize wind sound. If we listen to continuous white noise it might not represent a very strong metaphor, although we could relate it with some meaning depending on its context. It can for instance be perceived as an offline transmission device.

However, if we apply a band pass filter to this sound, varying its central frequency, even without any special context we can perceive the result as the very familiar natural sound of wind blowing.

In our system a server-side real-time sound synthesis engine provides the interface to transform various parameters of a Sound Object, which enables the user to a certain extent to add symbolic meaning to his performance.

3. System architecture

As shown in the illustration below, the *Public Sound Object* system is based on classic client-server architecture. The server handles the actual sound synthesis computation and the interaction interface is implemented on the client side. One of the main characteristics of this implementation scheme is its modularity.

The main application at the server side is the synthesis engine, which is designed in a rather general way in order to allow its versatile use for different applications. This engine is configured and controlled by two interfaces: a configuration interface, which initialises the general sound installation set-up by reading from a configuration file and a real-time control interface which allows an external application to control the various parameters of the synthesis process during execution time.

The core technology of this synthesis engine is based on CLAM (C++ Library for Audio and Music), a set of audio synthesis C++ Classes, designed at the Music Technology Group in Barcelona. It allows flexible sound transformations, by providing a versatile interface for the modification of a large number of audio parameters. Conceptually the engine is a re-implementation of Jorda's FMOL synthesiser.

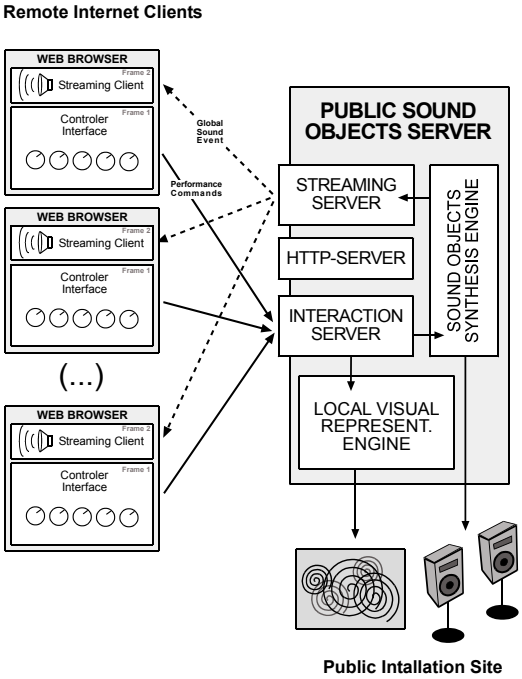


Fig. 1 -The PSO Architecture

The second server-side module is the interaction-server, which basically manages the sessions of the various connected users. It processes the received interface parameters and controls the according sound object synthesis processes within the synthesis engine. Additional components are a streaming audio server, which broadcasts the final audio stream back to the users and a standard HTTP server for delivering the web based interface.

Finally there is also a server side graphics component for the creation of the on-site visual representation of all current user sessions. The public sound server is part of a physical sound installation located in an appropriate museum space. There the resulting piece is played back by a local speaker system, and a visual representation of the installation is projected. The installation site provides as well some local client machines to allow local visitors the spontaneous participation in the piece.

On the user side the main application is a Java applet, embedded into the web interface. This applet upon loading connects to the interaction server, registers and initialises a user session. It provides the complete graphical user interface for the interactive control of the synthesis process. An additional component is the streaming audio client for the playback of the resulting musical piece.

3.1 The Synthesis engine

The synthesis engine incorporated into our system is a re-implementation of Sergi Jorda's original FMOL synthesizer. This project, currently carried out at the MTG in Barcelona, aims not only to re-implement but extend Jorda's synthesis concept using the CLAM C++ framework.

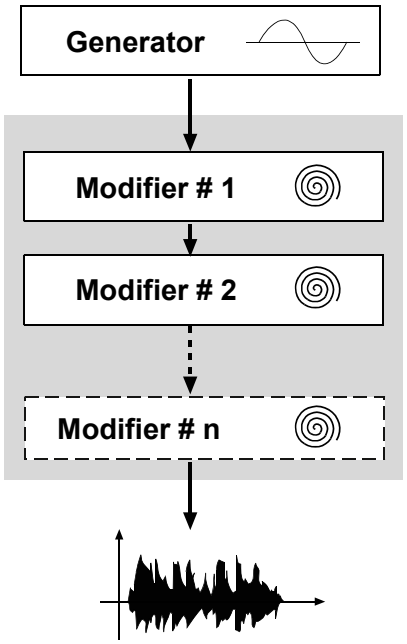


Fig. 2 -The Synthesis Engine

Basically this synthesizer provides a sound generator, which ranges from basic oscillators and modulations up to the simple use of digital sound samples. A chain of at least three modifiers then alters this generator's waveform. These modifiers implement a large toolbox of digital filters such as IIR or comb filters and effects such as pitch shifting etc.

These chains of modifiers provide a suitable amount of alterable parameters for our idea of mapping the interaction with an abstract visual model to the actual Sound Object. Finally the FMOL synthesizer can handle a large number of such tracks - those generator/modifier models that each actually represents an individual Sound Object. These independent sound tracks are then mixed together to the final musical piece.

3.2 The user interface

The user interface allows the interaction with the server-side synthesis engine and focuses on the manipulation of the actual sound synthesis parameters. Due to the modular nature of the system the component that is likely to vary in different setups is the graphical user interface (GUI).

In our system each GUI implementation, called Skin, should be developed along the following requirements:

- It should enable the user to contribute to the ongoing musical performance by transforming the characteristics of a visual Sound Object representation, sending normalized parameters to the synthesis engine over the network.
- The interface application should be able to manipulate the parameters for each of the modifiers in the synthesis engine according to the specific installation site setup.
- The GUI itself should be a behaviour–driven metaphorical interface, avoiding a flat mapping of parameters in a classical way, such as faders or knobs. The Sound Object representation has a default automatic periodical behaviour that can be conducted by the user.
- The auditory feedback conveys the performance of all currently connected users. Optionally there can be added some local auditory feedback, which is not part of the actual piece (the same way one would use a metronome).

As a proof of concept application we are currently developing a prototype based on a bouncing ball skin. This interface, shown below, is a metaphor for a ball that infinitely bounces on the walls of an empty room. When the ball hits one of the walls the corresponding Sound Object is triggered on the server.

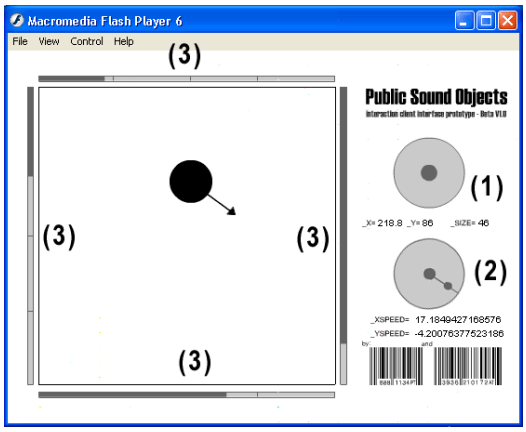


Fig. 3 -The Bouncing Ball Skin

The ball is moving continuously and the user can manipulate its size (1), its speed and direction (2) and each wall's acoustic texture (3).

The normalized values are then sent to the server where they are mapped to the following synthesis parameters:

Modifier #1: The **Wall's acoustic texture** is mapped to the **Sound Object's pitch**. Individual pitch values can be assigned to each wall, allowing the creation of melodic and rhythmic sound structures.

Modifier #2: The **Ball size** corresponds to the Sound Object's **reverberation**. The smaller the ball size, the higher reverberation, following a metaphor of an empty room with a bouncing ball. A bigger ball fills the room and therefore there is less reverberation.

Modifier #3: The **Ball speed** has an influence on the Sound Object's **amplitude**. The bigger the ball, the louder is the sound of the impact in each wall.

The first interface prototype was developed with Macromedia's Flash Action Script language; however the final version will be ported to Java2 mostly because of compatibility and network connectivity reasons.

3.3 The installation site

The installation site for the bouncing ball prototype will be suitable for a media art museum environment, where visitors can either watch the piece, or even participate using one of the provided local clients.

The scenario will be located in a dedicated room, which can hold several people. There will be a video projection showing a single local representation of the bouncing ball interface, visualizing the performance of all current participants.

Various loudspeakers positioned along the walls, create a spatial soundscape reproducing the sounds of the objects colliding with the walls.

4. Conclusions and future work

The Public Sound Objects project is still under development, however, the experiments realized so far with the bouncing ball prototype GUI, and the FMOL synthesis engine, are quite promising.

After this project overview we will finish the prototype implementation, and look for an adequate installation site to guarantee the success of this artistic proposal.

Once the system is operational we will have the opportunity to conduct research and evaluation about the user's behavior and the sonic event's results.

In future implementations we will experiment with the possibility of allowing the users to upload their own Sound Objects to the central server evaluating its musical results. We also intend to explore the possibilities of having different setups adapted to situations with large amounts of simultaneous users. For such scenarios *Micro-Sonic* music techniques [26], the use of banner clients or GRID computing could be interesting approaches.

5. References

[1] J. Bischoff, R. Gold and J. Horton, *Music for an Interactive Network of Microcomputers*. Computer Music Journal *2,* 24-29 (1978).

[2] C. Latta, *A New Musical Medium: NetJam*. Computer Music Journal *15,* (1991).

[3] D. Gang, V. Chockler, T. Anker, A. Kremer and T. Winkler. *TransMIDI: A System for MIDI Sessions Over the Network Using Transis*. 1997. Proceedings of the International Computer Music Conference (ICMC 1997).

[4] Y. Amir, D. Dolev, S. Kramer and D. Malki. *Transis: A Communication Sub-System for High Availability*. 1992. Proceedings of the 22nd Annual International Symposium on Fault-Tolerant Computing (FTCS).

[5] P. Burk. *Jammin' on the Web: A New Client/Server Architecture for Multi-User Musical Performance*. 2000. Proceedings of the International Computer Music Conference (ICMC 2000).

[6] M. Moller, W. Henshall, T. Bran and C. Becker. *The ResRocket Surfer - Rocket Network*. 1994. http://www.rocketnetwork.com/.

[7] Tonos - online musician's network. *TC8 - Music Collaboration Tool*. 2001. http://www.tonos.com.

[8] J. Young. *Using the Web for Live Interactive Music*. 2001. Proceedings of the International Computer Music Conference (ICMC 2001).

[9] J. Young and I. Fujinaga. *Piano master classes via de Intrenet*. 1999. Proceedings of the International Computer Music Conference (ICMC 1999).

[10] D. Ghezzo, J. Gilbert, A. Smith and S. Jacobson. *The Cassandra Project*. 1996. http://www.nyu.edu/pages/ngc/ipg/cassandra/.

[11] A. Xu, W. Woszczyk, Z. Settel, B. Pennycook, R. Rowe, P. Galenter, J. Bary, M. Geoff, J. Corey and J. Cooperstock. *Real-Time Streaming of Multichannel Audio Data over Internet*. 47[11]. 2000. Proceedings 108th Convention of the Audio Engineering Society.

[12] C. Chafe and G. Niemeyer. *Ping music installation, 2001, SFMOMA*. 2001. http://crossfade.walkerart.org/.

[13] S. Jordà and T. Aguilar. *FMOL: a graphical and net oriented approach to interactive sonic composition and real-time synthesis for low cost computer systems*. 1998. Proceedings of COST G6 Conference on Digital Audio Effects 1998.

[14] S. Jordà and A. Barbosa. *Computer Supported Cooperative Music: Overview of research work and projects at the Audiovisual Institute - UPF*. 2001. Proceedings of MOSART Workshop on Current Research Directions in Computer Music.

[15] S. Jordà, *Faust Music On Line (FMOL): An approach to Real-time Collective Composition on the Internet*. Leonardo Music Journal *9,* (1999).

[16] A. Barbosa. *Overview and conclusions of the Music Interfaces Panel Session at the MOSART Workshop (Barcelona, 2001)*. 2001. Report on the MOSART Workshop on Current Research Directions in Computer Music. http://www.abarbosa.org/docs/mosart-interactivity_pannel.pdf.

[17] P. Curtis. *Mudding: Social Phenomena in Text-Based Virtual Realities*. 1992. Proceedings of the 1992 Conference on the Directions and Implications of Advanced Computing.

[18] L. E. Marvin, *Spoof, Spam, Lurk and Lag: the Aesthetics of Text-based Virtual Realities*. Journal of Computer-Mediated Communication *1,* (1995).

[19] W. Duckworth. *Cathedral*. 1997. http://www.monroestreet.com/Cathedral/.

[20] P. Burk. *JSyn - A Real-time Synthesis API for Java*. 1998. Proceedings of the International Computer Music Conference (ICMC 1998).

[21] A. Tanaka. *MP3Q*. 2000. http://fals.ch/Dx/atau/mp3q/.

[22] R. Lozano. *Vectorial Elevation, Relational Architecture #4.* 2000. http://www.alzado.net/.

[23] P. Schaeffer, *Traité des Objets Musicaux.*, 1966.

[24] M. Chion, *Guide des Objets Sonores. Pierre Schaeffer et la Reserche Musicale.*, 1983.

[25] X. Amatriain and P. Herrera. *Transmitting Audio Content as Sound Objects*. 15-6-2002. Proceedings of the AES22 International Conference on Virtual, Synthetic and Entertainment Audio.

[26] C. Roads, *Microsounds*, MIT Press, 2001.

On the Use of FastMap for Audio Retrieval and Browsing

Pedro Cano, Martin Kaltenbrunner, Fabien Gouyon and Eloi Batlle
Music Technology Group
Universitat Pompeu Fabra
08002, Barcelona, Spain
+34 93 542 22 02
{pcano, mkalten, fgouyon, eloi} @iua.upf.es

0. Abstract

In this article, a heuristic version of Multidimensional Scaling (MDS) named *FastMap* is used for audio retrieval and browsing. *FastMap*, like MDS, maps objects into an Euclidean space, such that similarities are preserved. In addition of being more efficient than MDS it allows query-by-example type of query, which makes it suitable for a content-based retrieval purposes.

1. Introduction

The origin of this experiment is the research on a system for content-based audio identification. Details on the system are described in [1]. Basically the system decomposes songs into sequences of an alphabet of sounds, very much like speech can be decomposed into phonemes. Once having converted the audio into sequences of symbols, the identification problem results in finding subsequences in a superstring allowing errors, that is, approximate string matching. If we compare one sequence--corresponding to an original song in the database--to the whole database of sequences we retrieve a list of sequences sorted by similarity to the query. In the context of an identification system, this list reflects which songs the query - a distorted version of an original recording [1] - can be more easily confused with. Of course, studying this for each song is a tedious task and it is difficult to extract information on the matching results for the whole database against itself. Indeed, the resulting distances displayed in a matrix are not very informative at first sight. One possible way to explore these distances between songs by mere visual inspection is Multidimensional Scaling. MDS makes it possible to view a database of complex objects as points in an Euclidean space where the distances between points correspond approximately to the distances between objects. This plot helps to discover some structure in the data in order to study methods to accelerate the song-matching search. It can also be used as a test environment to compare different audio parameterization as well as their corresponding intrinsic distances independently of the metrics. Finally, FastMap's indexing capabilities also provide an interesting tool for content-based browsing and retrieval of songs.

2. Related Work

Research projects that offer visual interfaces for browsing are the *Sonic Browser [2]* and *Marsyas3D* [3]. The *Sonic Browser* uses sonic spatialization for navigating music or sound databases. In [2] melodies are represented as objects in a space. By adding direct sonification, the user can explore this space visually and aurally with a new kind of cursor function that creates an aura around the cursor. All melodies within the aura are played concurrently using spatialized sound. The authors present distances for melodic similarity but they acknowledge the difficulty to represent the melodic

distances in an Euclidean space. *Marsyas3D* is a prototype audio browser and editor for large audio collections. It shares some concepts with the *Sonic Browser* and integrates them in an extended audio editor. To solve the problem of reducing dimensionality and mapping objects into 2D or 3D spaces, Principal Component Analysis (PCA) is proposed. The drawback of this solution is that the object must be a vector of features and, consequently, it does not allow the use of e.g.: the edit distance, the inclusion of meta-data or other arbitrary distance metrics. In this article, the use of MDS, specifically *FastMap* is proposed to address this issue.

3. Mapping Complex Objects in Euclidean Space

3.1 Multidimensional Scaling

MDS [5] is used to discover the underlying (spatial) structure of a set of data from the similarity, or dissimilarity, information among them. It has been used for some years in e.g. social sciences, psychology, market research, and physics. Basically the algorithm projects each object to a point in a k-dimensional space trying to minimize the *stress* function:

where d_{ij} is the dissimilarity measure between the original objects and d_{ij} is the Euclidean distance between the projections. The *stress* function gives the relative error that the distances in k-dimensional space suffer from, on average. The algorithm starts assigning each item to a point in the space, randomly or using some heuristics. Then, it examines each point, computes the distances from the other points and moves the point to minimize the discrepancy between the actual dissimilarities and the estimated distances in the Euclidean space. As described in [4], the MDS suffers from two drawbacks:

- It requires $O(N^2)$ time, where N is the number of items. It is therefore impractical for large datasets.
- If used in a 'query by example' search, each query item has to be mapped to a point in the k-dimensional space. MDS is not well suited for this operation: Given that the MDS algorithm is $O(N^2)$, an incremental algorithm to search/add a new item in the database would be $O(N)$at best.

3.2 FastMap

To overcome these drawbacks, Faloutsos and Lin [4] propose an alternative implementation of the MDS: *FastMap*. *FastMap* considers the objects as points of some unknown k-dimensional space. The points are iteratively projected to the hyperplanes perpendicular to an orthogonal set of k-lines passing through the most dissimilar objects. The algorithm is faster than MDS (being linear, as opposed to quadratic, w.r.t. the database), while it additionally allows indexing. They pursue fast searching in multimedia databases: mapping objects into points in k-dimensional spaces, they subsequently use highly fine-tuned spatial access methods (SAMs) to answer several types of queries, including the 'Query by Example' type. They aim at two benefits: efficient retrieval, in conjunction with a SAM, as discussed above, visualization and data mining.

4. Results

To evaluate the performance of both least squares MDS and *FastMap*, we used a test bed consisting of 2 data collections. One collection consists in 1840 commercial songs

and the second collection in 250 isolated instrument sounds (from IRCAM's Studio OnLine). Several dissimilarity matrices were calculated with different distance metrics. The results of these experiments are shown in detail in *http://www.iua.upf.es/mtg/SongSurfer/*. In Figure 1 the representation of the song collection as points calculated with MDS and *FastMap* is shown. The MDS map takes considerably longer to calculate than the FastMap 's (894 vs. 18.4 seconds) although several runs of *FastMap a*re sometimes needed to achieve good visualizations. Although we did not objectively evaluate *FastMap* and MDS (objective evaluations of data representation techniques are discussed in [5]), MDS maps seem of higher quality. On the other hand, MDS presents a high computational cost and does not account for the indexing/retrieval capabilities of the *FastMap* approach.

5. Conclusions

We have presented the use of the existing *FastMap* method for improving a content-based audio identification system. The tool proves to be interesting, not only for audio fingerprinting research (visually exploring the representation space of audio data may reveal the possible weakness of a similarity measure), but also as a component of a search-enabled audio browser.

We first tested the tool with audio objects such as harmonic or percussive isolated sounds for which perceptually derived distances exist. In this case the results are excellent. But songs have a more complex nature, they account for many aspects of interest. Not only good similarity measures are hard to design but also to extract automatically from low-level audio features. Song repositories are usually described with heterogeneous mixes of attributes, descriptors range from physical feature vectors (e.g. MFCCs), up to subjective labels defined by experts (e.g. the "genre").

The advantage of MDS and *FastMap* lies in their generality: they can combine any type of data attributes, from low-level attributes to meta-data. We believe that this feature is relevant for improving browsing engines.

6. References

[1] P. Cano, E. Batlle, H. Mayer, and H. Neuschmied. "Robust Sound Modeling for Song Detection in Broadcast Audio." In: *Proceedings of the 112th Audio Engineering Society Convention*, Munich, 2002.

[2] D. Ó. Maidin and M. Fernström. "The Best Of Two Worlds: Retrieving and Browsing." In: *Proceedings of the Conference on Digital Audio Effects*, 2000

[3] G. Tzanetakis and P. Cook. "Marsyas3D: A Prototype Audio Browser-Editor using a Large Scale Immersive Visual and Audio Display." In: P*roceedings of the International Conference on Auditory Display*, 250-254. 2001.

[4] C. Faloutsos and K. Lin. "FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets." In: *Proceedings of the 1995 ACM SIGMOD*, 163-174. 1995.

[5] W. Basalaj. "*Proximity Visualization of Abstract Data.*" Technical Report 509, University of Cambridge Computer Laboratory, January 2001.

6. Bibliography

General:

[1] Arons B.: “A Review of the Cocktail Party Effect”. In: “Journal of American Voice I/O Society”, Vol. 12 p.35-50, July (1992).

[2] Gaver W.: *The Sonic Finder – An Interface that Uses Auditory Icons.* In: “The Use of Non-Speech Audio at the Interface” pp. 5.85-5.106, ACM Press, CHI ‘89 Austin, TX (1989).

[3] Dombois F.: „Using Audification in Planetary Seismology“, In: Proceedings of the 7th International Conference on Auditory Display, Espoo, Finland 2001

[4] Sturm, B.: “Surf Music: Sonification Of Ocean Buoy Spectral Data”, In: Proceedings of the 8th International Conference on Auditory Display, Kyoto, Japan 2002

[5] Hermann, Th. & Meinicke, P. & Bekel, H. & Ritter, H. & Müller, H. & Weiss, S.: “Sonifications For EEG Data Analysis”, In: Proceedings of the 8th International Conference on Auditory Display, Kyoto, Japan 2002

[6] Gaver W.: “Auditory Interfaces”. In Helander M.G, Landauer T.K., Prabhu P. (Ed.): Handbook of Human Computer Interaction. Elsevier Science, Amsterdam (1997).

[7] M. Barra, T. Cillo, A. De Santis, T. Matlock, U. F. Petrillo, A. Negro, V. Scarano, & P. P. Maglio: “Personal Webmelody: Customized Sonification of Web- Servers” “, In: Proceedings of the 7th International Conference on Auditory Display, Espoo, Finland 2001

[8] Brewster S.A.: “Using non-speech sounds to provide navigation cues.” In ACM Transactions on Computer-Human Interaction, 5(2), pp 224-259, 1998

[9] Raman, T.V.: “Emacspeak – A Speech Interface”, Proceedings of the Conference on Human Factors in Computing Systems. p66ff, ACM Press, 1996. URL, http://emacspeak.sourceforge.net/

[10] Gaver, W.: “Synthesising Auditory Icons”, Conference proceedings on Human Factors in Computing Systems, Amsterdam 1993

[11] Crease, M. & Brewster, S.: “Making Progress with Sounds. The Design Evaluation of an Audio Progress Bar”, Proceedings of the International Conference on Auditory Display, 1998

[12] Madhyastha T. M. & Reed D. A.: “A Framework for Sonification Design”, Kramer G. (Ed.) Auditory Display. Addison-Wesley, New York (1994).

From: "Marvin: Supporting Awareness through Auditor in Collaborative Virtual Environments"

Backhouse, A. & P. Drew (1992) "The design implications of social interaction in a workplace setting." *Environment and Planning B: Planning and Design*, 19: 573-584.

Ballas, J. A. (1994) "Delivery of Information through Sound." In: Gregory Kramer (ed.) *Auditory Display*, SFI Studies in the Sciences of Complexity, Proc. Vol. XVIII, Addison-Wesley, pp. 79-94.

Brewster, S.A. (1998). "Using non-speech sounds to provide navigation cues." *ACM Transactions on Computer-Human Interaction*, 5(2), pp 224-259.

Cohen, J. (1994) "Monitoring Background Activities." In: Gregory Kramer (ed.) *Auditory Display*, SFI Studies in the Sciences of Complexity, Proc. Vol. XVIII, Addison-Wesley, pp. 499-531.

Erickson, T. (1993) "From Interface to Interplace: The Spatial Environment as a Medium for Interaction." *Proc. of Conf. on Spatial Information Theory*. Heidelberg: Springer-Verlag.

Fitzpatrick, G., Mansfield, T. & S. M. Kaplan (1996) "Locales Framework: Exploring foundations for collaboration support." *IEEE Proc of the 6th Australian Conf on Computer-Human Interaction OZCHI'96*, Hamilton, NZ, pp. 34-41.

Gaver, W. W., (1989). "The SonicFinder, a prototype interface that uses auditory icons." *Human Computer Interaction* (4): 67 - 94.

Gaver, W. W. (1993) "What in the world do we hear? An ecological approach to auditory event perception." *Ecological Psychology*, 5(1): 1-29.

Hindus, D. et al. (1996) "Thunderwire: A Field Study of an Audio-Only Media Space." *Proc. ACM Conf. on Computer Supported Cooperative Work (CSCW'96)*, pp. 238-247.

Huxor, A (1999) "The Role of 3D Shared Worlds in Support of Chance Encounters in CSCW." In: Vince, J. & Earnshaw, R. (eds.) *Digital Convergence: The Information Revolution.* Springer-Verlag

James, F. (1996) "Presenting HTML Structure in Audio: User Satisfaction with Audio Hypertext." *ICAD '96 Proceedings*. Xerox PARC, 4-6 November 1996, pp. 97-10.

James, F. (1997) "AHA: Audio HTML Access." *Proceedings of The Sixth International World Wide Web Conference.* Santa Clara: CA, pp. 129-139.

Macaulay, C. and Crerar, A. (1998) Observing the Workplace Soundscape: Ethnography and Interface Design. *Proceedings of the International Conference on Auditory Display (ICAD '98)*, Glasgow, November 2-5.

Mynatt, E. D., Back, M. & R. Want (1998) “Design Audio Aura.” *Proceedings of the Computer-Human Interaction (CHI) Conference*, Los Angeles, April 1998, pp. 566-573.

Sawhney, N. & C. Schmandt. (1999) “Nomadic Radio: Scaleable and Contextual Notification for Wearable Audio Messaging.” *ACM SIGCHI Conference on Human Factors in Computing Systems*, Pittsburgh, May 15-20.

From: “Multiple Presence through Auditory Bots in Virtual Environments”

Backhouse, A. & P. Drew: The Design Implications of Social Interaction in a Workplace Setting. *Environment and Planning B: Planning and Design*, 19: 573-584, 1992

Huxor, A: The Role of 3D Shared Worlds in Support of Chance Encounters in CSCW. In: Vince, J. & Earnshaw, R. (eds.) *Digital Convergence: The Information Revolution.* Springer-Verlag, 1999

Towell, J. & Towell, E: Presence in Text-Based Networked Virtual Environments or "MUDS" *Presence* 6(5) 590-595, 1997

From: Public Sound Objects: A Shared Musical Space on the Web

J. Bischoff, R. Gold & J. Horton: *Music for an Interactive Network of Microcomputers*. Computer Music Journal *2,* 24-29 (1978).

C. Latta: *A New Musical Medium: NetJam*. Computer Music Journal *15,* (1991).

D. Gang, V. Chockler, T. Anker, A. Kremer & T. Winkler: *TransMIDI: A System for MIDI Sessions Over the Network Using Transis*. 1997. Proceedings of the International Computer Music Conference (ICMC 1997).

Y. Amir, D. Dolev, S. Kramer & D. Malki: *Transis: A Communication Sub-System for High Availability*. 1992. Proceedings of the 22nd Annual International Symposium on Fault-Tolerant Computing (FTCS).

P. Burk: *Jammin' on the Web: A New Client/Server Architecture for Multi-User Musical Performance*. 2000. Proceedings of the International Computer Music Conference (ICMC 2000).

M. Moller, W. Henshall, T. Bran & C. Becker: *The ResRocket Surfer - Rocket Network*. 1994. URL, http://www.rocketnetwork.com/.

J. Young: *Using the Web for Live Interactive Music*. 2001. Proceedings of the International Computer Music Conference (ICMC 2001).

J. Young & I. Fujinaga: *Piano master classes via de Intrenet*. 1999. Proceedings of the International Computer Music Conference (ICMC 1999).

A. Xu, W. Woszczyk, Z. Settel, B. Pennycook, R. Rowe, P. Galenter, J. Bary, M. Geoff, J. Corey & J. Cooperstock: *Real-Time Streaming of Multichannel Audio Data over Internet*. 47[11]. 2000. Proceedings 108th Convention of the Audio Engineering Society.

S. Jordà & T. Aguilar: *FMOL: a graphical and net oriented approach to interactive sonic composition and real-time synthesis for low cost computer systems*. 1998. Proceedings of COST G6 Conference on Digital Audio Effects 1998.

S. Jordà & A. Barbosa: *Computer Supported Cooperative Music: Overview of research work and projects at the Audiovisual Institute - UPF*. 2001. Proceedings of MOSART Workshop on Current Research Directions in Computer Music.

S. Jordà: *Faust Music On Line (FMOL): An approach to Real-time Collective Composition on the Internet*. Leonardo Music Journal *9,* (1999).

P. Curtis: *Mudding: Social Phenomena in Text-Based Virtual Realities*. 1992. Proceedings of the 1992 Conference on the Directions and Implications of Advanced Computing.

L. E. Marvin: *Spoof, Spam, Lurk and Lag: the Aesthetics of Text-based Virtual Realities*. Journal of Computer-Mediated Communication *1,* (1995).

P. Burk: *JSyn - A Real-time Synthesis API for Java*. 1998. Proceedings of the International Computer Music Conference (ICMC 1998).

P. Schaeffer: *Traité des Objets Musicaux.*, 1966.

M. Chion: *Guide des Objets Sonores. Pierre Schaeffer et la Reserche Musicale*, 1983.

X. Amatriain & P. Herrera: *Transmitting Audio Content as Sound Objects*. Proceedings of the AES22 International Conference on Virtual, Synthetic and Entertainment Audio. 15-6-2002

C. Roads: *Microsounds*, MIT Press, 2001.

From: On the Use of FastMap for Audio Retrieval and Browsing

P. Cano, E. Batlle, H. Mayer, and H. Neuschmied: "Robust Sound Modeling for Song Detection in Broadcast Audio." In: *Proceedings of the 112th Audio Engineering Society Convention*, Munich, 2002.

D. Ó. Maidin & M. Fernström: "The Best Of Two Worlds: Retrieving and Browsing." In: *Proceedings of the Conference on Digital Audio Effects*, 2000

G. Tzanetakis & P. Cook: "Marsyas3D: A Prototype Audio Browser-Editor using a Large Scale Immersive Visual and Audio Display." In: P*roceedings of the International Conference on Auditory Display*, 250-254. 2001.

C. Faloutsos & K. Lin: "FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets." In: *Proceedings of the 1995 ACM SIGMOD*, 163-174. 1995.

W. Basalaj: "*Proximity Visualization of Abstract Data.*" Technical Report 509, University of Cambridge Computer Laboratory, January 2001.