A Tool for Designing Sonifications of Background Monitored Information Sources

Sylvain Daudé
385 rue de la Bibliothèque
38041 Grenoble Cedex 9 - France
(+33) 4 76 51 44 40
sylvain.daude@imag.fr

Laurence Nigay
385 rue de la Bibliothèque
38041 Grenoble Cedex 9 - France
(+33) 4 76 51 44 40
laurence.nigay@imag.fr

ABSTRACT

In this article we present a software tool that allows neophyte sound designers to specify sounds that signal background monitored information sources. This tool is based on our decomposition of the sonification process (transformation of information into sounds) into steps, which are characterized by design parameters, modifiable by the user in our tool. Furthermore a plug-in mechanism permits the expert user to complement the existing design parameters by allowing the user to implement her/his own plug-ins or to download them from the Internet. From the specification of a sound source, an executable internal representation is automatically generated, which can then be played. Therefore our tool can either be used for designing or for running sound sources. We finally present an application specified and run using our tool, called the phearcons, which allows the user to place sound sources in space by means of physical objects representing the sound sources.

KEYWORDS

Design tool, sound signalling, information streams, sonification process, phearcon, physical object, space.

1. INTRODUCTION

Recently, the tremendous increase in computing power has come with the development of sound devices and with the apparition of new interaction paradigms in the Human-Computer Interaction field. However, in most existing commercial human-computer interfaces (HCI), the use of sound is generally restricted to associating binary events, such as alarms, to sound files, whereas more complex information is generally displayed using visual or vocal modalities. These HCI therefore neglect the enormous potential of information sound signalling (or sonification, i.e. transformation of information into sound) [1], as well as the ear's highly developed skills for information analysis. On the other hand, research applications involving information sound signalling [2, 3] suffer generally from rigidity, as the displayed information structure and the sonification parameters are generally fixed during the design process.

In this article we present a software tool that allows neophyte sound designers to specify sounds that signal background monitored information sources. This tool is based on our decomposition of the sonification process into steps, which allow the user to structure her/his design and to decompose the difficulties encountered into manageable sub-problems. Each step corresponds to an abstraction level and is described according to high level design parameters, understandable by neophyte sound designers and digital signal processing novices. Finally, a plug-in mechanism permits the expert user to complement the existing design parameters by allowing the user to implement her/his own plug-ins or to download them from the Internet.

This tool also integrates a mechanism that automatically generates an executable internal representation, or code, of the specified sound sources, which can then be played. The generated code is organised according to modules that correspond to the steps of the sonification process. Therefore our tool can be used for both designing and for running sound sources.

After presenting the sonification process and its implications in our design tool, we show that this process also permits sound applications developers to structure their sound sources internal representation. We finally present an application specified and run using our tool, called the phearcons, which allows the user to place sound sources in space by means of physical objects which represent the sound sources.

2. SONIFICATION PROCESS AND SOUND SOURCE SPECIFICATION

Defining a sound according to an information source requires knowing how to transform information into sound (the sonification process). In order to identify the sonification process steps and parameters, we have adapted the visualisation process as defined by Ed Chi [4] to sonification. Figure 1 shows the parallel between the two processes. The identified sonification process steps and parameters are common to every sound source[1] and explicit in our software tool.

The sonification of a data set can be broken down into five abstraction levels and four successive transformations, F1 to F4, between the levels.

F1 aims at extracting useful information from the original data. This step is common to both the visualization and the sonification processes and depends on both data semantics and users' tasks. For example, the AROMA system [5] distorts the sound signal coming from distant users talk (data semantics) while keeping enough information for speaker identification (a users' task). In our software tool, the initial data are referenced by their location –

[1] the sonification process was also used for classifying existing sound interfaces and for structuring the state of the art [10]

e.g. a socket, an URL or an audio recording device- and their type –e.g. audio or numerical-, whereas F1 is then represented by a serial sequence of filters.

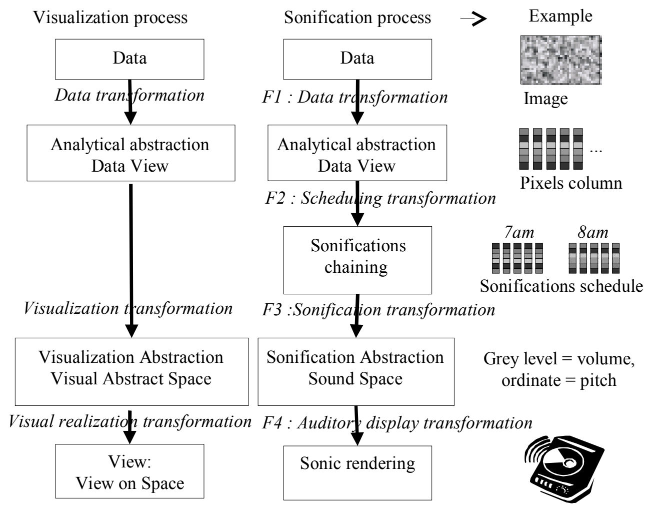

Visualization process
Sonification process
Example
Data
Data
Data transformation
F1 : Data transformation
Image
Analytical abstraction Data View
Analytical abstraction Data View
Pixels column
F2 : Scheduling transformation
Sonifications chaining
7am
8am
Sonifications schedule
Visualization transformation
F3 :Sonification transformation
Visualization Abstraction Visual Abstract Space
Sonification Abstraction Sound Space
Grey level = volume, ordinate = pitch
Visual realization transformation
F4 : Auditory display transformation
View: View on Space
Sonic rendering

Figure 1. Parallel between Chi's visualisation process and our sonification process

F2 defines a set of instants when the data provided by F1 will be, or have been, sonified. For instance it handles the repetition of sounds over time, the conditions for this repetition to stop and a history of the last played sonifications. In our software tool four parameters are kept: (i) the instant when the information will be displayed for the first time, which can be either relative to the time when the data are received or to time values included in the data, (ii) the frequency of the sounds repetition over time, (iii) the condition for the repetitions to stop, which can be either automatic (e.g. after 10 repetitions) or manual (i.e. when the user triggers a particular event) and (iv) the history duration, that is the duration during which the last played sonifications are memorised. Indeed, early experiments showed that users appreciated being able to replay the last sounds of a sound source [12].

F3 defines, from the abstract data set, an abstract sound localized in a Euclidean space. An abstract sound consists of a time interval and a set of acoustic cues (e.g. intensity, volume, timbre, frequency, position in space) that vary during this interval. In our tool, F3 is defined by two parameters: a synthesis algorithm that characterizes the transformation itself, such as a "wind" or "bubbles" algorithm, and an absolute sound volume which defines the gain in intensity to apply to the data generated by the synthesis algorithm. The absolute volume parameter aims at allowing the user to quickly equilibrate the different volumes of the monitored sound sources. The default set of synthesis algorithms proposed to the user was obtained by adapting an everyday sounds classification based on cognitive criteria [6]. Indeed, this adaptation guarantees to the user a set of familiar and easily distinguishable sounds. The algorithms used were mainly extracted from [7] and [8].

F3 involves two successive transformations: (1) the representation of each single element by an abstract sound, then (2) the coordination of the corresponding sounds in space and time. Our tool maps these two transformations with two kinds of synthesis parameters, namely (1) sound parameters, e.g. for a wind sound, the ratio between the wind strength and the data element value, and (2) coordination parameters, such as the silence duration or the spatial angle between two successive wind sounds. Though the coordination parameters may be common to several algorithms, both sound and coordination parameters generally vary from one algorithm to another.

Finally, the F4 transforms the abstract sound into a physical sound according to four steps: after having chosen a view point on the abstract sound space, device parameters are computed then transformed into a sound signal which is finally displayed on a physical device. In the tool, F4 uses two parameters: (i) the sound devices channels where the sound is sent; this is be chosen from all the available channels; this feature allows the user to listen to sources through different devices, for instance confidential sounds heard through headphones and music through loudspeakers; inversely, a four-channel sound can be created by selecting two stereo outputs on two different devices; (ii) the kind of physical device on which the sound is to be played, e.g. headphones or loudspeakers. The physical device can furthermore be defined more accurately; for example for headphones, the user can choose a HRTF filter from among a predefined set [9].

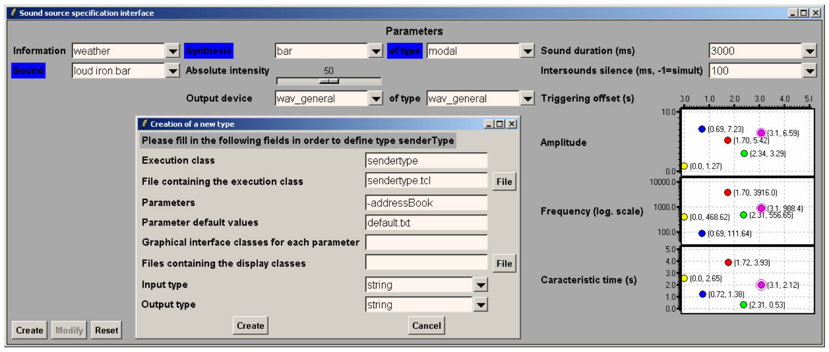
As mentioned in the introduction, our tool furthermore allows neophyte users to define complete sound sources. It also integrates three other usage levels for more expert users. The tool thus integrates four specification levels corresponding respectively to neophyte, casual, intermediate and expert users.

- The neophyte user can choose a complete sound source: the parameters for every sonification process step are then fixed. The user is thus provided with turnkey solutions for creating sound sources associated with predefined information sources. In the current implementation, the filters and sounds associated by default to the predefined information sources are chosen on a case-by-case basis; for instance an e-mail delivery is filtered to extract its size and sender category (colleague, family, friend); this category is then associated to a characteristic sound, a typing machine sound (of which the duration increases with the e-mails size) for a colleague, a recorded laugh for a family member and a bottle opening sound for a friend.

- At a more accurate level, the casual user can choose separately a data source and a sonification process from among predefined choices. The obtained sound source is thus relevant to the *user's* mental model and not to the designer's one. More generally, predefined choices provide values for groupings of parameters, such as the whole set of scheduling (F2) or synthesis (F3) parameters. It should be noticed that some choices could have been restricted relatively to the user's previous choices and to predefined heuristic rules, for example by proposing only sounds of which the semantics is related to the displayed information. This elegant solution was however excluded in our tool in order to enhance flexibility and extensibility.

- The intermediate user can modify the parameters for each predefined element, for example by setting the value of a threshold filter or the viscosity of a bubble sound. Figure 2a presents an example of such a setting.

- Finally, the expert user can add new features ("plug-ins"); for example, s/he can define a new bubbling sound, parameterized by liquid viscosity and bubbles frequency. In other words, the expert


Sound source specification interface

Parameters

Information: weather
Synthesis: bar
of type: modal
Sound duration (ms): 3000
Sound: loud iron bar
Absolute intensity: 50
Intersounds silence (ms, -1=simult): 100
Output device: wav_general
of type: wav_general
Triggering offset (s)

Amplitude: (0.69, 7.23) (3.1, 6.59) (1.70, 5.42) (2.34, 3.29) (0.0, 1.27)
Frequency (log. scale): (1.70, 3916.0) (3.1, 988.4) (0.0, 468.62) (2.31, 556.65) (0.69, 111.64)
Caracteristic time (s): (1.72, 3.93) (0.0, 2.65) (3.1, 2.12) (0.72, 1.38) (2.31, 0.53)

Creation of a new type

Please fill in the following fields in order to define type senderType

Execution class	sendertype
File containing the execution class	sendertype.tcl
Parameters	-addressBook
Parameter default values	default.txt
Graphical interface classes for each parameter	
Files containing the display classes	
Input type	string
Output type	string

Figure 2. Our specification tool while being used: (a) the background window shows the setting of sound synthesis algorithm parameters, increasingly accurate from left to right; (b) the foreground window shows the setting of a filter plug-in.

user can add specific parameters, such as the « viscosity » parameter, to the generic parameters of the sonification process. In concrete terms, importing a plug-in means specifying the algorithm method name, the list of parameters handled by the algorithm, the data types it uses and generates, and optionally the name of methods that display interfaces for specifying each parameter (figure 2b).

3. Sonification process and sound source execution

The specification of a sound source using our tool, as presented in the previous paragraph, produces a software object or generated code which represents the sound source and which can be played or executed. As shown in figure 3, the software object representing a sound source is made up of five *general* modules that model the sound source and of which the structure is independent of the sound source. The first module is dedicated to the reception of the initial data, whereas the four other modules respectively perform the four transformations of the sonification process. Indeed, this structure eases both sound source configuration and code modularity.

As shown in figure 3, the general modules are then connected to *specialised* modules which deal with some particular task and communicate through the general modules. The specialised modules are created according to the corresponding sound source configuration and can be classified into five categories: (i) the "acquisition" modules handle the original data acquisition; (ii) the "filter" modules filter the original data; (iii) the scheduling modules, when receiving the filtered data, order the following modules to transform the filtered data into a sound sample, then, when appropriate, to play the sample. Dissociating the sound sample preparation, which happens only once, from its playing, which can be repeated in time, allows a substantial saving in terms of computing time; (iv) the "synthesis" modules generate, from the filtered data, abstract sounds according to synthesis algorithms. These abstract sounds are independent from the sound device used. They generally consist of sound signals tagged with spatial data, but can also consist of abstract data, such as midi sequences; (v) finally, the "sound device" modules handle the loading of abstract sounds into samples, then their playing.

Each specialised module can be shared by several sound sources, allowing a coherent allocation of system resources. For example, a data receiver can feed several sound sources with data read on the network: this makes the network traffic lower than if each sound source had its own receiver.

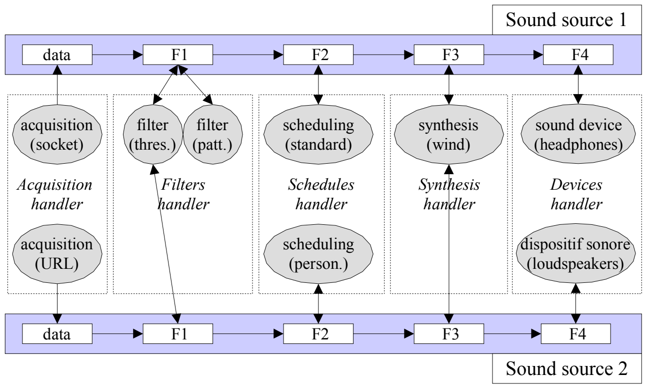


Figure 3. Internal representation of two sound sources and their associated modules. General modules are represented by white boxes, specialised modules by grey ellipses. Handlers are not represented.

Other digital objects, the *handlers*, provide the sound sources with the specialised modules they need. In particular, the handlers manage the sharing of specialised modules between sound sources. A particular handler deals with the user interactions; events managed by this handler include all the user actions that can affect the sound sources properties, such as the order to replay the last event of a sound source. Therefore, the specialised modules which have made the appropriate request can be informed of the user interactions and consequently modify their treatment; for example, the scheduling module can stop a sound repetition if the user has made a particular action. Furthermore, handlers ease the adding of plug-ins, which correspond to specialised modules. Indeed existing modules and handlers are not modified by adding a plug-in.

It should be noticed that our design and execution tool runs on several platforms (Windows, Mac and Linux) because it is exclusively based on cross-platform languages and APIs. In the following paragraph we present an example of an application developed and executed with our tool.

4. Phicon + Earcon = Phearcon

In order to completely define a sound interactive application, besides the definition of sound sources, we need to know how the user is going to interact with the sound sources. Indeed, it is quite obvious that the user should be able to switch off or lower the volume of each monitored sound source at any time, because the source becomes less important compared to the user's current task, or because the user does not want to disturb her/his neighbours. Furthermore, without an independent control of the different sound sources, the user may forget what information sources are sonified, and therefore have trouble differentiating the monitored sound sources.

Various kinds of interfaces have been proposed for manipulating several sound sources independently. For example, the physical mixing table allows a quick and multi-threaded access to the volume of several sound sources. However the number of controllable sound sources is limited by the number of tracks on the table; moreover this device does not allow placing the sound sources into space, which makes their discrimination harder. The graphical interfaces can solve these problems, however a study carried out by [11] shows that these interfaces are not adapted for sound sources monitoring, because they require the user's focus (a problem which does not occur in audio-only interfaces), and because they introduce a perceptual discontinuity between the sound sources visual and audio representations. This very study proposes an interaction based on physical objects placed on a bench equipped with sensors, the position of the objects on the bench determines the position of sound in space. However this system forces the objects to conform to a precise format and sound spatialisation is rendered through a relatively imprecise stereo loudspeaker system. As a matter of fact, the authors have observed that users were not using the spatialisation provided with this system for differentiating the monitored sound sources.

The interface proposed by our application also relies on a coupling between physical objects (phicons), disposed on a table and tracked by a computer vision system, and sonified information sources (earcons in its widest meaning), this coupling being materialised by the spatialisation of sounds according to the objects positions. We call the resulting combined object a "phearcon". As shown in figure 4 and unlike the previous system, this application allows the user to freely choose and place the physical objects on the table within the limits imposed by the recognition system.

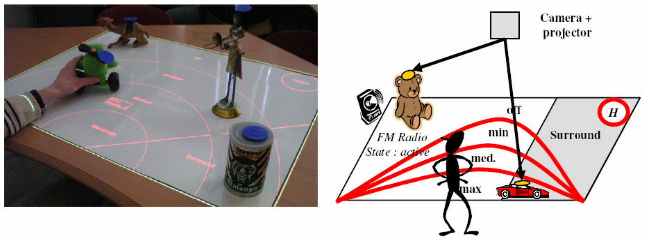


Figure 4. a. (left) The phearcons while being used.
b. (right) Sketch of the interaction with the phearcons.

The phearcons present several advantages. First they provide the user with control on the surrounding audio space. Indeed, by moving the physical objects, the user can quickly adjust the corresponding sound sources positions in space, and therefore make their understanding and discrimination easier. This may be of some help each time the monitored sources characteristics and/or the user's needs evolve; for instance if a monitored sound source becomes important for the user, by bringing the corresponding object into the foreground, the user can listen to the sound more carefully. Inversely, a noisy sound source can be "put aside" by moving the corresponding object. Finally, the objects provide a familiar visual support to sounds, making the memorisation of the sound sources locations and natures easier. The phearcons application is detailed in [12].

5. CONCLUSION

We have presented a software tool that enhances users of various skill levels to (i) design, specify sound sources that signal continuously monitored information sources, and (ii) run the specified sound sources. As regards the execution, we propose to structure the code generated by our tool into general and specialised modules that are directly related with the steps in our sonification process. After introducing the use of our tool, we have finally presented an application, called the phearcons, which allows the final user to place sound sources in space by means of physical objects that represent the sound sources.

Our ongoing work aims at evaluating the sonification design parameters presented previously, by measuring (i) the time for a user to specify a sound source, and (ii) the number of specification/execution loops between before the user gets a satisfying result in terms of usage.

6. REFERENCES

[1] S. Barrass and G. Kramer, "Using sonification", Multimedia Systems, Vol. 7, No 1, 1999.

[2] D. Malandrino, D. Mea, A. Negro, G. Palmieri and V. Scarano, "NeMoS: Network monitoring with sound", in Proc. ICAD'03, pp. 251-254, 2003.

[3] H. Ishii and B. Ullmer, "Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms," in Proc. CHI'97, pp. 234-241, 1997.

[4] E. Chi and J. Riedl, "An Operator Interaction Framework for Visualization Systems", in Proc. Info Vis '98, 1998.

[5] E. Pedersen and T. Sokoler, "AROMA: abstract representation of presence supporting mutual awareness", in Proc. CHI'97, pp. 51-58, 1997.

[6] W. W. Gaver, "What In The World Do We Hear ? An Ecological Approach to Auditory Event Perception", Journal of Ecological Psychology, Vol. 5, No 1, pp. 1–29. 1993.

[7] S. Conversy, "Conception d'Icônes Auditives Paramétrées pour les Interfaces Homme-Machine", thèse de Doctorat, Université de Paris-Sud, 2000.

[8] M. Russ, "Sound Synthesis and Sampling", Francis Rumsey (ed.), Focal Press, 1997.

[9] V. R. Algazi, R. O. Duda, D. M. Thompson and C. Avendano, "The CIPIC HRTF Database", in Proc. IEEE Workshop on Applications of Signal Processing to Audio and Electroacoustics, pp. 99-102, 2001.

[10] S. Daudé and L. Nigay, "Design Process for Auditory Interfaces", in Proc. ICAD'03, pp. 176-179, 2003.

[11] A. Singer, D. Hindus, L. Stifelman and S. White "Tangible Progress: Less Is More In Somewire Audio Spaces", in Proc. CHI'99, 1999.

[12] S. Daudé et L. Nigay, "Objets physiques pour manipuler des sources sonores - Phicon + Earcon = Phearcon", in Proc. Mobilité-Ubiquité, 2004.