The inaugural International Conference on Music Communication Science 5-7 December 2007, Sydney, Australia
http://marcs.uws.edu.au/links/ICoMusic

OVERCOMING SOFTWARE INERTIA IN DATA SONIFICATION RESEARCH USING THE SoniPy FRAMEWORK

David Worrall

Sonic Communications Research Group, University of Canberra

david.worrall@canberra.edu.au

ABSTRACT

It has been assumed that the much-needed development of data sonification software would occur from the adaptation of sound synthesis software, principally that developed for computer music. As the software demands of data sonification research grow, some limitations of this approach are becoming evident. This paper outlines an extendable software framework, called SoniPy, which attempts to redress some of those limitations.

1. INTRODUCTION

Whilst some data sonification software that has been engineered from first principles, for example for individual experiments entailing clearly defined tasks such as accurate monitoring [1] or graphic user-interaction [2], this type of software has become relatively uncommon. Cheaper data storage and faster computation has enabled the implementation of a certain level of abstraction away from low-level audio synthesis routines and the generality and reusability that this abstraction affords. The software tools being used for this purpose have principally been adopted from the field of computer music which, given the head-start created by the historical alliance between composers and engineers in its development, is perhaps not surprising.

Principal of these software tools was the Music series and its children, dominant of which today are Csound [3], Supercollider [4], Max/MSP [5] and its younger half-brother PD [6]. A defining feature of this family is the bifurcation of the music-making process: composing routines in which abstract musical ideas are expressed in (text and/or graphic) 'scores' on the one hand, and the use of those scores to control sound synthesis 'orchestras' that are optimized for the intensively repetitive tasks of generating digital soundwaves on the other.

Throughout this evolution, composers have used various computer languages for exploring musical ideas, much as they might use a piano in their 'dot-based' compositional practice. Software functions as a tool for thought; for generating and articulating abstract structural ideas, which are then expressed in sound. [7] Composer Barry Truax put it like this:

> *In order to work with computers, practically everyone had to become a programmer, or at least work with one, as there were no standard tools and few texts.*
>
> *... for me, the digital domain means something totally open-ended, limited only by my ability to program compositional ideas... In fact, I've never really used any one else's software for composition ... you'll probably never need to develop your own at all. Neither path seems preferable, but I wonder where the new ideas will come for you?* [8]

Today, irrespective of the field of endeavour, two computer-related characteristics can be recognised. Firstly, a manufacturer's global hardware or software system upgrade can render current or previous work unrealisable, especially when the work depends upon interfaces to low-level systems, such as those for audio processing. This seems to occur every five or so years and can lead to developers banding together to maintain recently outdated software, at least long enough for them to scan the horizon for other, hopefully more resilient tools. Once the new tools are adopted, the task of translating the essential parts from the older to the newer system is undertaken with varying degrees of enthusiasm.

Secondly, modern computational environments are now so complicated that it has become almost impossible for an individual to learn all that is needed, in the depth and diversity required, for them to undertake significant development work on their own.

2. COMPOSITION AND SONIFICATION COMPARED

Computer music composition and data sonification share the need for sophisticated real-time sound synthesis: the 'orchestras' referred to in the Introduction. However, whilst music's abstract structures can be expressed within a software environment used to generate 'scores' to be played by 'orchestras', in data sonification, concrete, sometimes voluminous, data has to be acquired, analysed and filtered in a timely manner before any such 'score' generation or direct application to resonant models can occur. Faced with the need to communicate with external musical data, the computer music community universally adopted the MIDI [9] and OSC [10] 'score' protocols. However no such protocols exist for non-musical data, nor, given it's diversity, are they ever likely to.

Furthermore, the sorts of software tools required for such numerical analysis and data storage and

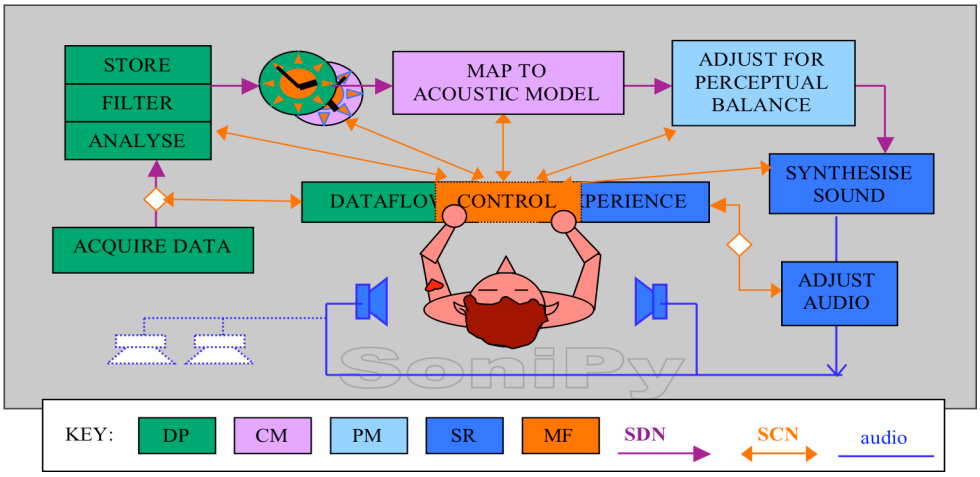

Figure 1: Flow Diagram of SoniPy's Five Module Sets and Two

retrieval are in the public domain and in use in other disciplines. So there is dilemma as to whether to adopt a set of tools appropriate for data handling and enhance them with sophisticated sound synthesis capabilities, or do the reverse. The latter is the most common approach, and that taken in *SonEnvir* [11], *OctaveSC* [12] and the *Interactive Sonification Toolkit* [13] for example.

Supporting such enhancements over the longer term is a major undertaking. When it comes to software maintenance, there is safety in numbers: from a third-party user's perspective, the fewer the number of people involved, the greater the risk of obsolescence, and the longer the lead time until someone fixes a bug or implements a new feature.

3. INTRODUCING SoniPy

In an attempt to break the impasse outlined, we are developing an open-source software framework called SoniPy. The SoniPy project evaluates and integrates various already existing independent components, such as those for data acquisition, storage and analysis, cognitive and perceptual mappings as well as sound synthesis and control, by encapsulating them, or control of them, as Python modules. The choice of Python was not arbitrary, as it is used in a wide variety of application areas from scientific computing to active website design. It possesses all the features of a modern modular object-oriented general-purpose programming language, which can also serve as a glue language for connecting together in a simple and flexible manner many separate software components, including subroutine libraries written in other languages [14][15]. The way these wrapping techniques work to integrate the different components of SoniPy is discussed in more detail elsewhere [16].

The power of this approach is that each module (such as one for multidimensional numerical processing, for example) can be integrated into a wide range of research fields whilst being independently maintained by a group of developers with a particular interest and expertise. Such development responsibility, when shared, makes it unnecessary for such expertise to be acquired by each research community that needs such a tool, nor responsible for its ongoing maintenance in the face of shifts in the underlying computing platforms whilst relying on other module developers to do the same.

Having access to an interpreter in order to construct a complete sonification by iteratively building on small tests is a powerful and user friendly aspect of the Python Framework because it allows the consequences of decisions at each stage to be tested incrementally, thus enabling a better understanding and control of emergent effects in an overall design. This is particularly important in sonification because of the non-linear nature of aural perception.

Following the sonification design process, as illustrated in Figure 1, the SoniPy design specifies five Module Sets communicating over two different networks The SoniPy design specifies five Module Sets that communicate over two different networks: the Sonipy Data Network (SDN) and the Sonipy Control Network (SCN). Modules are grouped according to their role in the data sonification process: Data Processing (DP), Conceptual Modeling (CM), Psychoacoustic Modeling (PM), Sound Rendering (SR) and Monitoring & Feedback (MF).

Depending on the dictates of a particular project, modules in a Set may be instantiated on different machines. A particular Module Set may be empty, i.e. contain no modules, or a particular module may belong to more than one Module Set.

3.1 The Data Module Set

By way of illustration, the Data Module Set serves the following purposes:

- Audification - writing data in formats acceptable as direct input to audio hardware,
- Simultaneous handling of multiple time-locked streams, such as from biomedical monitors,
- Deconstruction, analysis and filtering, including of complex meta-tag embedded multiplexed streams,

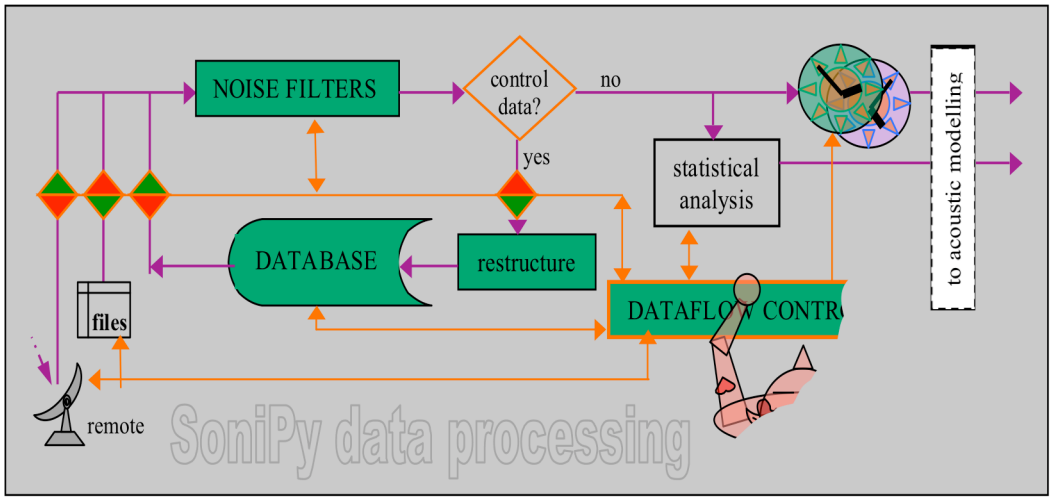

Figure 2: A representation of one configuration of SoniPy's Data Processing modules

such as a data feed from a stock-market trading engine,

- Model-based sonification involving user feedback, and
- Simulation of data feeds, including buffering with time compression and expansion.

Figure 2 illustrates a way the different components of the Data Module Set might be configured for a particular sonification.

3.2 SoniPy on Sourceforge.net

The SoniPy Project is established as a public-domain, community-based project so it can evolve as its components evolve. It uses only public-domain software and, although it is being initially developed on Macintosh OSX, the aim is to eventually release for all major hardware platforms.

The Copyright under which a particular software component is released into the public domain remains in tact; it is unaltered by being released by, or in association with, the SoniPy Project.

We recognise three types of public-domain software:

- That which is released as a complete downloadable click-and-install package,
- That which is released as downloadable source code that requires platform-specific compilation and/or installation, and,
- Software of our own design and/or software that has required significant modification such the conversion of source code to Python extension libraries, or the debugging, updating or compiling of code not supported by the original authors or maintainers.

Coordination of the Project occurs through Sourceforge.net, a collaborative revision control and software development management system that provides a range of software development lifecycle services to more than 150,000 registered projects. The SoniPy website [17] is the principal way through which users can access the latest module recommendations and related developments.

3.3 Limitations

The major limitation of the SoniPy approach is the intensity of the evaluation process needed to independently evaluate existing possible modules. Being public-domain software, its quality is variable, from the extremely well designed and over-performing to the badly planned and carelessly documented.

There is a diverse degree-of-difficulty in building and/or installing these third-party tools. This degree-of-difficulty exponentiates when a module has dependencies which themselves need to be built from source code.

We have developed a set of evaluation guidelines to assist future contributors and us in module evaluation. These guidelines need to extend to the evaluation of the possible negative interactions between different modules.

4. FUTURE DIRECTIONS

Currently, tools for undertaking Sonification research are either discipline-specific, with modifications to accommodate new interconnections, or ad-hoc collections of stand-alone software tools developed for a specific task.

Because SoniPy's open framework design can integrate modules conforming to widely accepted inter-process computation standards (wrappable libraries), it will be possible for it to grow in most directions its user-community needs it to. However, the SoniPy Project is a new venture and its success will depend on the extent to which the sonification community finds it useful and is prepared to contribute to its ongoing development.

There is currently, for example, a dearth of good public-domain software for experimental psychology

involving sound, and the integration of such a Module Set into SoniPy would be a welcome addition. By such inclusion it would be possible to design and conduct different types of computer-based empirical experiments, either in single-participant and/or Internet delivery mode, and analyse the results within a single framework.

A user-contributed library of experiments for evaluating a sonification design could assist in developing some standards for measuring the functionality, aesthetics, learnability, effectiveness, accuracy, expressiveness and other aspects of a design. These evaluations could result in sonification templates that can assist a wider community in choosing between different designs for a particular sonification task, much as templates are used for colour palettes or website designs.

By establishing SoniPy an open source project we hope to share our work with others who in return will contribute to a framework that is capable of great flexibility and general usefulness to the sonification community.

5. REFERENCES

1. Chafe, C. and Leistikow, R. “Levels of Temporal Resolution in Sonification of Network Performance,” in *Proceedings of the 2001 International Conference on Auditory Display,* Espoo, Finland, July 29-August 1, 2001

2. Walker, B. and Cothran, J.T. “Sonification Sandbox: A Graphical Toolkit for Auditory Graphs,” in *Proceedings of ICAD 2003, Boston,* 2003.

3. http://www.counds.com/

4. http://www.audiosynth.com/

5. http://www.cycling74.com/

6. http://crca.ucsd.edu/~msp/software.html/

7. Worrall, D. "Studies in metamusical methods for sound and image composition." *Organised Sound* Vol. 1 No. 3 Cambridge University Press (1996) pp 20-26.

8. Truax, B. "Music and Science Meet at the Micro Level: Time-Frequency Methods and Granular Synthesis," Presented at the Music Viva conference, Coimbra, Portugal, September 2003. Reproduced at http://www.sfu.ca/~truax/mviva.html

9. http://www.midi.org/

10. Wright, M., Freed, A. and Momeni, A. “Open sound control: State of the art 2003,” in *Proceedings of the 2003 Conference on New Interfaces for Musical Expression.* National University of Singapore , Singapore. 2003.

11. de Campo, A., Frauenberger, R. and Höldrich, R. “Designing a generalized sonification environment” in *Proceedings of ICAD 04-Tenth Meeting of the International Conference on Auditory Display,* Sydney, Australia, July 6-9, 2004.

12. Hermann, T. 2006. http://www.sonification.de/projects/sc3/index.shtml

13. Pauletto, S. and A. Hunt, A. “A toolkit for interactive sonification,” in *Proceedings of ICAD 04-Tenth Meeting of the International Conference on Auditory Display,* Sydney, Australia, July 6-9, 2004.

14. A. Watter, A., Van Rossen, G. and Ahlstrom, J. *Internet Programming with Python*. M&T Books, NY, NY. 1996.

15. http://www.python.org/

16. Worrall, D., Bylstra, M, Barrass, S. and Dean, R. "SoniPy: The design of an extendable software framework for sonification research and auditory display." *Proceedings of the 13th International Conference on Auditory Display, Montréal, Canada, June 26-29, 2007.*

17. http://sonipy.sourceforge.net/

(all URLs current as at 20070601)