THE SYSSON PLATFORM: A COMPUTER MUSIC PERSPECTIVE OF SONIFICATION

Hanns Holger Rutz *Katharina Vogt* *Robert Höldrich*

University of Music and Performing Arts Graz
Institute of Electronic Music and Acoustics (IEM)
Inffeldgasse 10, 8010 Graz, Austria
rutz | vogt | hoeldrich @iem.at

ABSTRACT

We introduce *SysSon*, a platform for the development and application of sonification. *SysSon* aims to be an integrative system that serves different types of users, from domain scientists to sonification researchers to composers and sound artists. It therefore has an open nature capable of addressing different usage scenarios. We have used *SysSon* both in workshops with climatologists and sonification researchers and as the engine to run a real-time sound installation based on climate data. The paper outlines the architecture and design decisions made, showing how a sonification system can be conceived as a collection of specialised abstractions that sit atop a general computer music environment. We report on our experience with *SysSon* so far and make suggestions about future improvements.

1. INTRODUCTION

The *SysSon* platform has been developed for the past two years as part of the eponymous research project [1] funded by the Austrian Science Fund (FWF), P 24159. It is an open source software[1] that provides both an application programming interface (API) and a standalone desktop application. Fig. 1 is a schematic view of its components and the types of users and activities it supports.

The current design is the result of an incremental refactoring that departed from the initial requirements of the research project, namely to be able to open and preprocess data files to become suitable for real-time sound synthesis based on *ScalaCollider*, a client for the *SuperCollider* server [2].

Another objective was to enable climatologists to integrate sonification with their typical workflows. Originally coupling to existing plotting software such as *Ncview* and building a simple domain specific language (DSL) for text-based interaction, the platform gradually acquired the shape of a full-fledged desktop application.

As the system became more complex, new requirements for higher-level resource management, caching, and a persistent description of sonification models emerged, and we repositioned the project on the foundation of *Sound Processes*, a computer music framework [3]. The platform was fully "infected" by the dataflow model of this framework, and the specific abstractions used for sonification were reimplemented within the object model of *Sound Processes*.

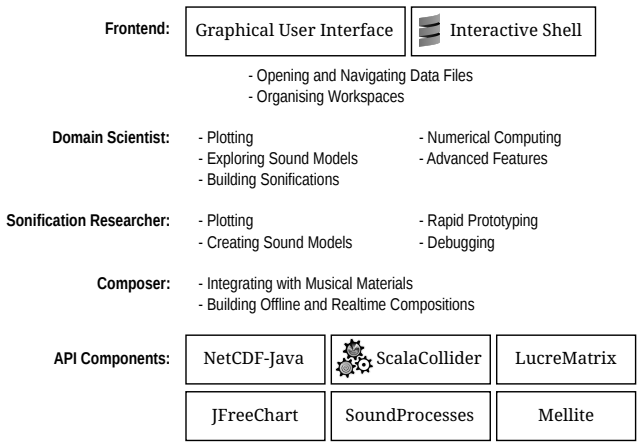

Figure 1: SysSon architecture

This shift was reinforced by the planning of a sound installation, where sonifications would be interactively composed. Consequently, as a last step, the GUI was integrated with the computer music front-end *Mellite*. What we arrived at is a perspective in which sonification becomes a particular methodology of computer music. What we hope for is that, as composers get acquainted with *Mellite*, they will discover and explore the possibility of sonified data as a new material element in their work. Of course, many composers and sound artists have utilised sonification processes (e.g. [4], [5], [6]), but there exists no general platform for the systematic experimentation with sonification that equally satisfies the needs of researchers and artists and lets them directly share their ideas.

In the following, we give an overview of our system. It is still a young project and not without obstacles, and we will conclude by summarising the experience with its usage so far.

2. HOST ENVIRONMENT

Sound Processes is a frameork that is the result of research into the observation of the compositional process. It provides data structures that trace the evolution of computer based composition over time, making these traces available for later inspection or for incorporation into the process itself. But it can also be transparently used as a foundation to build computer music systems.

[1] https://github.com/iem-projects/sysson

2.1. Common Objects

Sound Processes provides two core functionalities. First of all, it defines a protocol for reactive (data-flow like) objects, and it contains a number of useful objects, such as:

- atomic types: integer and floating point numbers, boolean expressions, strings, ...
- folders: they are containers for other objects and similar to a sub-patcher in *Max* or *Open Music*.
- artifacts and artifact locations: artifacts are logical references to external file resources. Artifacts are resolved through locations, and when a workspace is transported to a different computer, locations can be adjusted to resynchronise with external resources.
- audio-files: they are a specific type of artifact.
- proc (or sound process): a description of a sounding object. It encapsulates a DSP graph in the *ScalaCollider* language, which is a dialect of *SuperCollider*. It also contains ports (called scans) to connect to other processes.

The object protocol is extensible, and *SysSon* uses it to add its sonification abstractions.

The second functionality is sound synthesis. Model objects, such as an audio-file, a proc, an ensemble (group of objects) or a timeline may have corresponding *aural representations*. *Sound Processes* integrates a transactional memory model with the sound synthesis system and provides high level abstractions for dealing with graph topology and resources such as audio-buses and buffers.

2.2. Workspace

Mellite provides a graphical user interface for *Sound Processes* and defines the workspace as the basic organisational unit. A workspace in most cases is simply a root folder that can be thought of as the main "patcher". The creation and modification of objects inside this patcher are automatically synchronised with an underlying database. We thus ensure that the user can come back to the workspace at any later point and will find everything in its previous state, including for example the parametrisation of sonification models or plot objects. To preserve state, one can either duplicate objects or use an automatically versioned workspace that traces the evolution of all parameters over time.

The GUI uses metaphors of a standard desktop application such as point-and-click, drag-and-drop, and undo-redo. Fig. 2 shows an example workspace containing two locations, a folder with data-sources, a folder with sonification models, an audio-file and a plot. Except for locations and audio-file, these are objects introduced by the *SysSon* platform.

3. SYSSON ABSTRACTIONS

We now describe the main abstractions provided by *SysSon*.

3.1. Data-Sources

Data sets for sonification can become very large, and domain sciences have come up with file formats to store them. *SysSon* supports *NetCDF* (Network Common Data Form) [7], a format frequently used in atmospheric research. *NetCDF* files can easily grow to several hundreds of megabytes, and they are therefore not copied but merely linked to workspaces as external references through handles called data-sources.

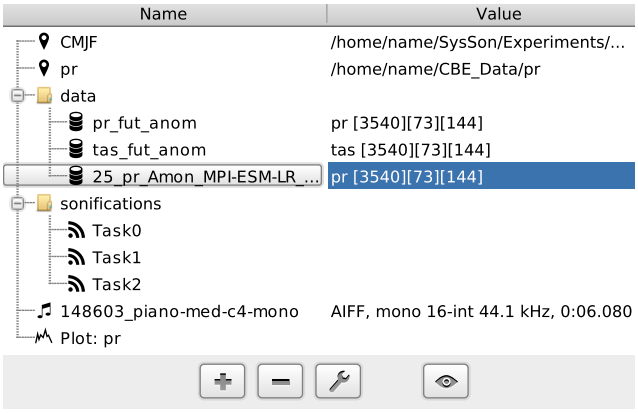

Figure 2: Workspace view

When a data-source object is created, its skeleton structure consisting of a number of variable descriptors (matrices) is stored with the workspace, allowing to operate even when the underlying *NetCDF* file is offline. A data-source is associated with an artifact which can be updated when a workspace is moved to a different computer.

Adding different types of data-sources in a future version should be simple. For example, at the moment a *CSV* file would have to be converted to a *NetCDF* or audio-file first, but there is no reason one could not add direct support.

3.2. Matrix Structure

A matrix is a regular one- or multi-dimensional structure of floating point cells. Dimensions are simply represented by other matrices. For example, a matrix of precipitation data may have dimensions *lon* (longitudes), *lat* (latitudes), *time* (time-series). Each of these dimensions then is another one-dimensional matrix (or vector) that stores the dimension's values, such as the series of latitudes with unit 'degrees-north'.

Matrices are composed and transformed through a data-flow graph. They usually originate from a data-source object. To be editable in the user interface, a variable placeholder is used that stores the current data-flow graph. Transformations then become new nodes in this graph. The most common transformation is a reduction of the matrix's size using a reduce object. The reduce object takes an input matrix, a dimension-selector and a reduction-operator. For example, to produce a time slice of the aforementioned precipitation matrix, the dimension-selector would indicate the time dimension and the reduction-operator is an index into the time dimension. The output matrix thus has a rank of one less than the input matrix. Each of the objects related to the reduction is again made editable through data-flow variables holding the dimension's name and the index integer position. This is illustrated in Fig. 3, where the resulting matrix expression at the very bottom could be a parameter of a sonification model.

Other operators take slices (ranges) of a dimension or perform sub-sampling by skipping samples using a stride parameter. Future versions shall include other commonly used operators such as dimensional reduction through scanning and sub-sampling using

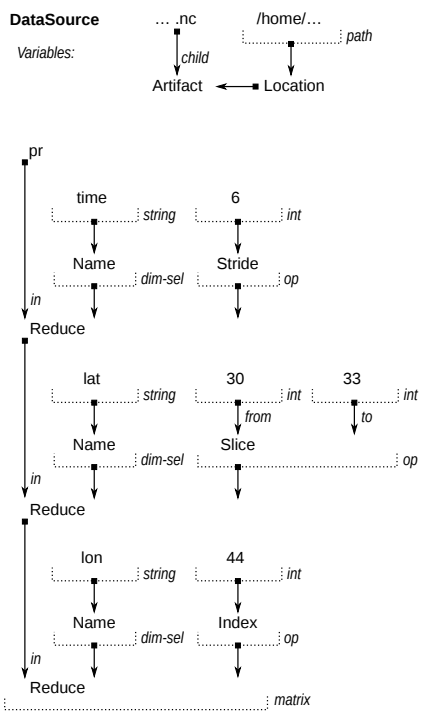

Figure 3: Composed matrix graph. Dotted trays represent variable (mutable) data-flow cells.

averaging or interpolation, as well as binary operations such as taking the element-wise differences between two matrices. Currently, those operations can only be carried out eagerly, producing new matrix files.

3.3. Plot Objects

Plot objects encompass a matrix, a mapping from dimensions to axes, and visual parameters such as colour palette and scaling. Fig. 4 shows an example plot of a time slice of precipitation data.

3.4. Sonification Objects

Sonification instances are encapsulated by a dedicated object type. This object is composed of

- a proc (sound process) object that describes the sound production in terms of a synthesis function.
- a dictionary of sources where a logical name in the sonification model is associated with a tuple of a matrix and a dimensional dictionary. The dimensional dictionary provides logical dimensions for the sound model that may want to use them for unrolling the matrix in time or to drive specific sound aspects such as timbre or spatialisation.
- a dictionary of controls which are user adjustable scalar parameters of the sound model. For example, a typical control would be the speed at which a sonification traverses a time series.

The user interface for a sonification object is shown in Fig. 5. The section labeled ‘Mapping’ shows that the model uses a single

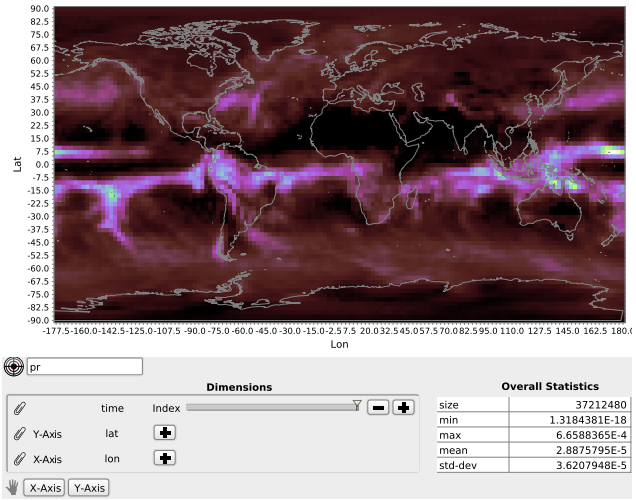

Figure 4: Plot view

source ‘data’ with which a matrix ‘pr‘ has been associated. This matrix has been reduced, the underlying graph corresponding with Fig. 3. The model also defines two logical dimensions ‘time‘ and ‘pan‘ which are associated with the matrix’s own ‘time‘ and ‘lat‘ dimensions. Using this dictionary-based decoupling, sonification models can be flexibly tested with different data inputs.

The sonification researcher or sound designer can open an integrated code editor within the workspace to develop the sound models. This is depicted in Fig. 6. Regular *ScalaCollider* expressions are augmented with user interface elements such as the ‘user-value’ object responsible for the ‘controls’ section of the sonification editor, and data specific elements such as matrix and dimension keys. Within the DSP graph, matrices and vectors may appear as scalar values or dynamic time-changing signals. When a sonification object is made audible, the system translates the matrix expressions into a cache of audio files which can then be streamed on the *SuperCollider* server. We exploit its multi-channel expansion feature and provide pseudo-UGens to easily align the matrix data with related data such as the axis dimensions.

4. EXPERIENCE

The platform has been used in multiple user tests, a workshop and for the creation of a sound installation.

4.1. As a Research Tool

Preliminary results show that both climatologists and computer music researchers had no problems navigating and configuring existing sonification models. During the workshop, it became clear that mastering the programming of sound models requires a longer learning process, as users need to gain an understanding of the nature of the data at hand, the architecture of *SysSon* and its model of data processing, as well as the sound synthesis language and the way it connects to the data inputs. The current focus on a graphical user interface without a direct equivalent in the interactive text console was seen as disadvantageous by some participants. Furthermore, experienced computer musicians felt the restriction of having one compound sound synthesis process limiting compared

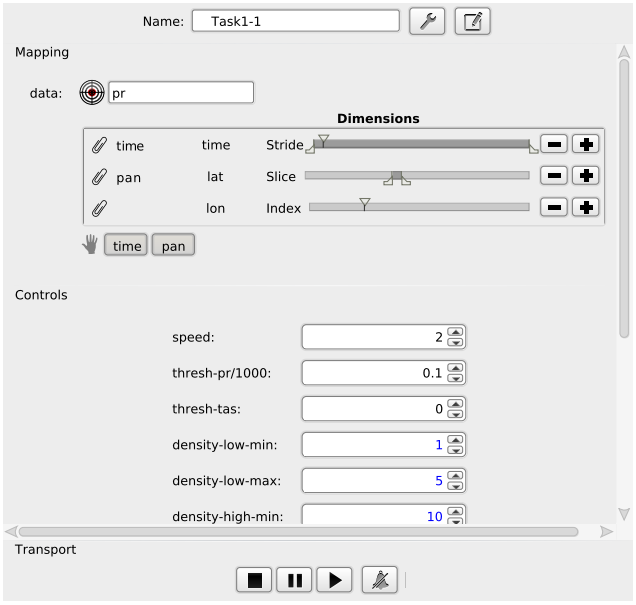

Figure 5: Sonification editor

to a more flexible client-side timing model provided for instance by *SuperCollider*'s pattern sequences.

4.2. Sound Installation

Using the platform in a real-time interactive sound installation was another stress test and rewarding experience. The piece *Turbulence. A climate sound portrait* was developed in collaboration with a group of visual artists for exhibition at the Forum Stadtpark, Graz in November 2014 [8]. The system is coupled to 42 speakers and 12 acceleration sensors that influence the selection of sound situations and the modulation of the exhibition space that is filled with a topography of threads and paper (see Fig. 7).

Embedding the sonification objects in a temporal form that changes according to compositional constraints and the signals of the sensors was an interesting challenge that especially led to the question of whether all structures should be composed inside *Mellite* or outside of it, using a conventional development environment. In favour of the former was the ability to directly evolve sound objects as the installation was running, in favour of the latter was the more mature editor and the ability to describe the interconnection of processes as text commands, something that is still cumbersome, as proc atoms must be instrumented with code fragments of actions that are triggered at points in time, revealing the limitations of the graphical user interface to represent relationships between the workspace objects.

In the end a hybrid approach was used to combine both advantages, where the layers of the composition were prototyped directly inside the environment but finally written back to regular class files. The seamless transition and interweaving in the sound composition from sonification based elements to purely musical macro form supported the perspective of sonification as one possible rendering among others in the repertoire of computer music.

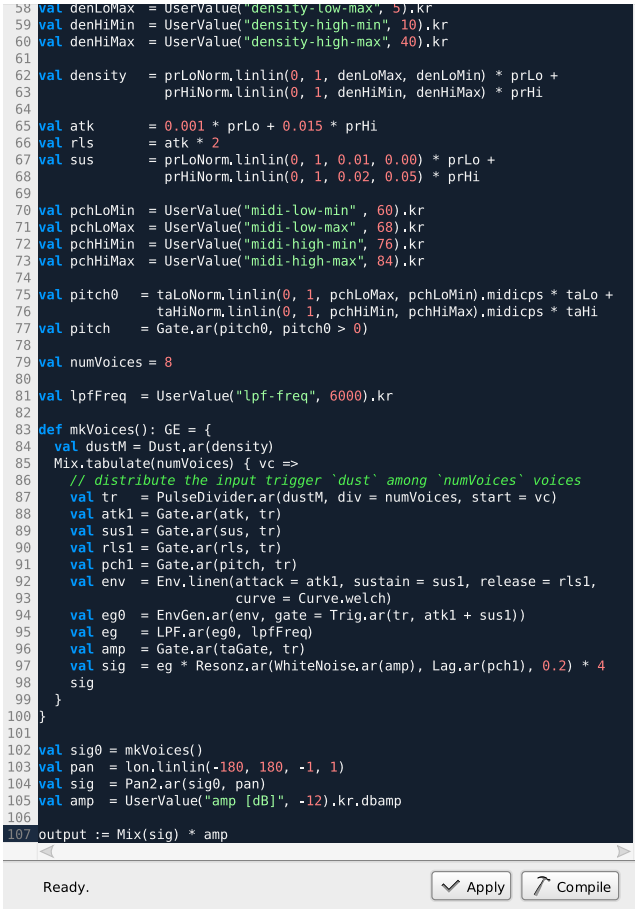

Figure 6: Synth Graph code editor

5. OUTLOOK

The future development of *Mellite* will have to focus on improving the possible oscillations between writing code fragments and connecting and arranging materials in the workspace, making this hybrid form more compelling to work with.

The *SysSon* platform will benefit from an expansion of the reactive matrix layer. Not only do we need more transformations—especially resampling—but ideally all matrix operations will be supported inside the definition of sound synthesis functions itself. For example, the sound designer should be capable of specifying a dimensional reduction of an input matrix or the calculation of anomalies, freeing the domain scientist from tedious preparations of the input data and yielding one unified API. Another iteration of the development could also focus on a more user-friendly DSL as a thin layer on top of the existing API to allow easier text-based access to the entire platform.

While *SysSon* has been validated within climate research, an interesting future task will be the application in other scientific areas. Here it will be seen if it can establish itself as a compelling alternative to other existing sonification solutions. From our perspective, its advantage over plain numerical computing environments such as *MATLAB* or *Octave* is the use of a well established sound synthesis system based on *SuperCollider* with a seamless transition to computer music paradigms and its potential for real-

Figure 7: View of the sound installation *Turbulence*

time control. More advanced numerical computing packages can be easily linked to the system, and the Scala language is establishing itself firmly in the "data science" community. The advantage over customised *Max/MSP* or *Pure Data* patches is the dynamic nature of the workspace, the sound models and the handling of data-sources (e.g. channel abstraction), something that is cumbersome if not impossible to achieve in static systems. Furthermore, we believe the dual perspective of visual front-end with desktop metaphors and the power of text-based API make our approach superior to patchers.

6. REFERENCES

[1] K. Vogt, V. Goudarzi and R. Höldrich, 'SysSon – a systematic procedure to develop sonifications', in *Proceedings of the 18th International Conference on Auditory Display*, Atlanta, GA, 2012, pp. 229–230.

[2] H. H. Rutz, 'Rethinking the SuperCollider client...', in *Proceedings of the SuperCollider Symposium*, Berlin, 2010.

[3] ——, 'A reactive, confluently persistent framework for the design of computer music systems', in *Proceedings of the 9th Sound and Music Computing Conference (SMC)*, Copenhagen, 2012, pp. 121–129.

[4] A. Polli, 'Atmospherics/Weather Works: a spatialized meteorological data sonification project', *Leonardo*, vol. 38, no. 1, pp. 31–36, 2005.

[5] F. Dombois and G. Eckel, 'Audification', in *The Sonification Handbook*, T. Hermann, A. D. Hunt and J. Neuhoff, Eds., Berlin: Logos Publishing House, 2011, pp. 301–324.

[6] N. Barrett and K. Mair, 'Sonification for geoscience: listening to faults from the inside', in *EGU General Assembly Conference Abstracts*, vol. 16, 2014, p. 4489.

[7] H. L. Jenter and R. P. Signell, 'NetCDF: a public-domain-software solution to data-access problems for numerical modelers', in *Proceedings of the 2nd International Conference on Estuarine and Coastal Modeling, American Society of Civil Engineers (ASCE)*, vol. 72, 1992, pp. 72–82.

[8] H. H. Rutz, 'Particularities and generalities. considerations on the sound composition', *Forecast:Turbulence. Workshop:Exhibition*, pp. 64–67, 2014.