# Soundium2: An Interactive Multimedia Playground

Simon Schubiger-Banz, Stefan Müller

DIUF, Université de Fribourg, CH-1700 Friburg
MultiMedia Lab, University of Zurich, CH-8057 Zürich
*email:* shoobee@corebounce.org

## Abstract

This paper gives an overview of Soundium2. Soundium2 is a unique combination of different areas of computer science ranging from real-time signal processing over programming languages to automatic software configuration. It not only serves as an experimental tool for exploring new ideas in computer science but also is frequently used in live multimedia performances in order to expose it to real-world conditions. Although similar systems exist, few were designed explicitly for live multimedia-performances. In this sense, Soundium2 is more like a multi-user, audio-visual instrument than a software system. Soundium2 is introduced and an overview of its architecture is given. The features of Soundium2 are outlined and compared with related work.

## 1 Introduction

This paper gives an overview of Soundium2, supported by the MultiMedia Lab of the University of Zurich, the Parallelism and Artificial Intelligence group of the University of Fribourg and the Corebounce Association.

Soundium2 brings together different areas of computer science in a unique multimedia system. The main motivation behind Soundium2 is the quest for a framework, which allows us to test new ideas in audio processing and visualization with minimal overhead. Nevertheless, the system should be rock solid and user friendly since we use it frequently during live performances. The current client server architecture in combination with data flow networks, signal processing, 3D graphics, versioned global state, and an advanced GUI manages to match these contradicting goals. The rest of this paper starts with background information on related work as well as the origins of Soundium2, then continues with an overview of the system and finally concludes with an outlook.

## 2 Background

Although several excellent multimedia systems exist today, none of them implement all features we consider as crucial for successful rapid development and live performances. This section lists a small selection of systems that perform well for certain aspects and are in turn very inspiring for our own work.

### 2.1 The Interactive Dance Club

The Interactive Dance Club (Ulyate and Bianciardi 2002) was an installation built for the 25th annual ACM SIGGRAPH Conference and includes many elements also found in Soundium2 (such as interactive, music synchronized live-visuals). On the one hand the interactive dance club is a mix of heterogeneous components, and thus lacks the homogeneity and consistency of Soundium2. On the other hand, it goes further than Soundium2 by strongly integrating the participants of an event through interactive zones. Although possible with Soundium2, we consider an integration of the audience more as an artist's choice than an integral part of a performance.

### 2.2 The Max Family and SuperCollider

The Max environment (Pukette 2002) has its origins in audio processing. Different extensions such as DIPS (Matsuda and Rai 2000) add multimedia capabilities. It shares with Soundium2 the split between a small, optimized processing core and a modular GUI. The main difference is that Max is a *visual programming environment*. Soundium2 has an integrated programming language too but it is mainly used for system manipulation (e.g. system state, or GUI) and it is completely independent of the engines, which implement the processing core. Max allows very fine-grained composition and control, resulting in a highly flexible system. Despite that, typically Max developers tend to include larger building blocks as "externs" instead of constructing them from basic operators. This is exactly the approach of Soundium2. Engines export only a set of "externs" that can be interconnected. Simple operators such as arithmetic are excluded.

Another example is SuperCollider (McCartney 2002) which is a highly evolved computer music language closely integrating processing and process description. The question of granularity, placement of processing objects, and process description is not easily answered; the deciding argument for Soundium2 is its main purpose: a live-performance system where complexity (at least for the user) has to be kept at a minimum

## 2.3 Parallel, Bomb, WinAmp plugins

All these tools have in common that they visualize aspects of an audio signal such as spectrum or level and process the audio signal with DSP (Digital Signal Processing) algorithms. User interaction is also possible to some degree but for both aspects they never go as far as Soundium2. Despite their rudimentary functionality compared to Soundium2 or Max, some of these tools produce very appealing output and are thus very inspiring from an artistic perspective. Especially the success of WinAmp plugins show that a large community is willing to develop visualization and audio processing tools and an even larger group is consuming their output.

## 2.4 Soundium2

In addition to the inspiring work in multimedia research in general, each of the project team members contributes its own expertise in a specific field. Several predecessors focusing on various aspects of sound processing and visualization were developed by individual members of the project. The experience gained from its ancestors laid the foundations for the project. Soundium2 was written entirely from scratch as an integrated multimedia framework. It retains the features of its predecessors considered crucial for an interactive multimedia tool (Juillerat 2001; Specht 2000; Parish and Müller 2001; Schubiger and Hirsbrunner 2003; Schubiger 2002):

– State of the art signal processing
– High-level information extraction e.g. beat-detection
– State of the art 2D and 3D visualization
– Strongly typed dataflow network
– RPC client/server architecture with clustering
– Automatic software configuration for network construction and media-format handling
– Graphical user interface optimized for interactive live-performances

# 3 A closer look at Soundium2

Figure 1 depicts the overall architecture of Soundium2. Basically, Soundium2 is based on a multi-tier client/server architecture with one client controlling a cluster of servers called "engines". The motivation for this separation is twofold: the first motivation is code complexity and code maintenance and the second is timing. The engines should handle all timing sensitive operations while keeping their code complexity small. All complex operations should run on the client, who may or may not control the engines in real-time. This allows us to rapidly develop and fine-tune individual engines for specific tasks such as 3D rendering or audio processing while performing more complex tasks like global-state and GUI management in the client.
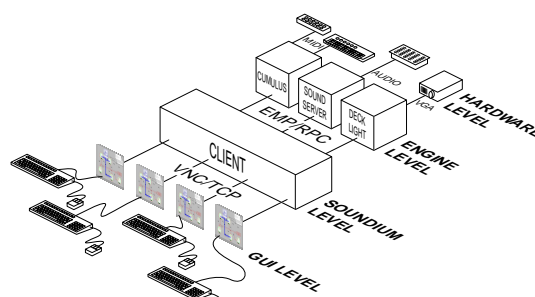


Figure 1 - Overview of the Soundium2 architecture

The lowest architectural level is represented by specialized *hardware* (hardware level) such as video projectors, DMX and MIDI equipment etc. All these devices are controlled by individual *engines* (engine level) which are optimized for a specific task. Each engine holds a part of the global dataflow network state which is manipulated through an *RPC* interface. This RPC interface is mainly used by the *client* (soundium level) to configure and parameterize the engines. Only the client holds the entire global system state composed of the per engine dataflow networks as well as the system wide state modification history. The client also renders a multi-user graphical user interface (GUI level) that can be accessed via the remote frame buffer (RFB) (Richardson and Wood 1998) protocol over the network for collaborative performances.

## Features

### Engine

The Soundium2 engine is a C++ framework based on a processing graph of building blocks (the nodes) that communicate which each other (along the graph edges). It has been designed with two main goals in mind: First, provide efficient real-time processing facilities to different media types. Second, provide a highly abstracted SPI (System Programming Interface) for the building blocks that allows non-experts to write signal and media processing code.

The first goal has been achieved by implementing a system that makes use of extensive multithreading. The building blocks are automatically assigned to schedulers, which handle processing characteristics of a specific media type. Dataflow analysis determines the execution order inside the different scheduling groups. The building blocks communicate trough connected ports allowing arbitrary data types. Inter-scheduler communication is automatically synchronized.

The application of rigorous OO methods in C++ almost completely hides scheduling and synchronization issues from programmers of building blocks. For instance, ports can be accessed like normal variables. To date, building blocks for sequencing, audio signal processing, MIDI I/O, parameterized OpenGL rendering graphs, and OpenGL Performer has been implemented.

**RPC interface**

The RPC interface is extremely simple, basically allowing the manipulation of the engine's dataflow network. The calls comprise creation and removal of nodes as well as edges and reading and writing of parameters. Every call takes a timestamp argument occurring in the future, which is used by the engine to properly schedule the execution of the call. Two auxiliary calls allow obtaining static engine information (supported node types and data formats) and querying the current engine time.

**SL1**

The global system state (the union of the per-engine dataflow networks) is versioned and stored as a revision tree. Every state manipulation is saved in the revision tree and it is thus possible to rollback to any previous state and starting a new branch from there. Even forward propagation of state changes is supported if the changes are non-conflicting with later revisions of that state. The integration of versioning in a multimedia system ensures that no artwork is lost and that new artwork can be easily evolved from existing work. The storage format is called Soundium Language 1 (SL1), a full-fledged imperative programming language. Besides encoding the system state, SL1 is also used for client side scripting, GUI customization, and performance preparation.

**Resources**

The management of resources such as 3D models, textures, and audio clips is entirely HTTP (Hyper Text Transfer Protocol) based. In addition to flat file resources, a proprietary extended resource format (XRS) allows association of SL1 code with media files, thereby transforming passive resources into active objects that can react to events like resource loading.

**Software Configuration**

Resource classification[1] is the foundation for the automatic software configuration capability used in Soundium2. A modular scheme allows the addition of resource transformations (such as format conversion), which are automatically aggregated based on their semantics in order to construct high-level transformations. Software configuration relies on an ontology, which is planned to include meta information beside the current media related information such as music style, mood, etc.


Figure 2 - The Soundium2 graphical user interface

**Graphical User Interface**

The graphical user interface (GUI) allows interactive inspection and manipulation of the *system state* as well as editing SL1 code. Figure 2 shows a screenshot of the Soundium2 GUI with a selection of user interface elements. The system state represents the union of all dataflow networks on each engine as a graph. *SL1 scripting* is used for modifications of SL1 code. The *revision tree* holds the trace of all state changes in the system and allows arbitrary jumps between system states[2]. Node *parameters* can either be modified through dedicated GUI elements or associated with hardware devices such as *MIDI controllers*.

## Live Performance with Soundium2

Using Soundium2 for live performances mainly consists of preparing a set of global states for the event in question. The preparation takes into account information such as expected music-style, social and cultural aspect of the audience, spatial and technical constraints of the location (projection surfaces and on-site equipment). Hours before the event, the engines are hooked up to the input[3] and output equipment[4] and clustered with the client in an IP network. During the event, the artists use the graphical user interface to change parameters, reconnect inputs, or add new nodes in order to adapt the multimedia experience. The modifications are the result of the artist's perception of the event and constitute his live contribution to the performance. Usually each artist focuses on an aspect like

---

[1] By classification we understand the process of determining resource properties such as content type as well as extracting features like beat positions.
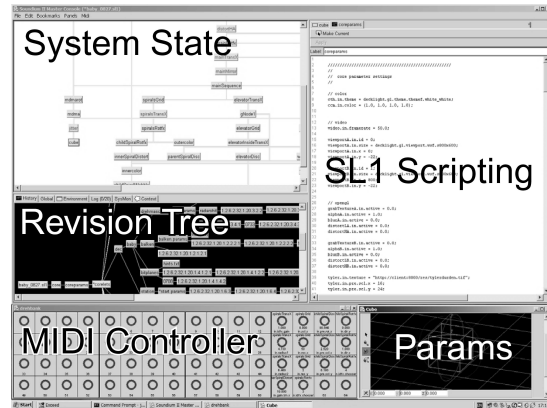
[2] The revision tree can also be seen as a multi-level redo/undo tree.

[3] Commonly used input equipment includes audio, video, MIDI, EEG (Electro-Encephalogram), environmental sensors, etc.

[4] Output equipment are e.g. video projectors, audio-system, laser projectors.

visualization or sound[5] and the final multimedia experience is the result of a strong interaction between the artists as well as the audience.

# 4    Conclusion and Outlook

Soundium2 combines input from various domains such as signal processing, client/server architecture, software configuration, programming languages and user interface design. Besides being used for live performances (see figure 3) which is a very satisfactory part of the work, it also serves as an excellent playground for exploring new ideas. This is mainly thanks to its modular design and its strong visualization and interactive manipulation capabilities. For example, Soundium2 is also often used just as a framework for rapid prototyping like the phoneme-recognition / facial-animation demonstration given at Imagina'03 (Kalberer, Müller and van Gool 2002). It is also used as a MMS (Multi Media Messaging) demonstration system at Swisscom.

Currently the system is expanding in many directions but the basic architecture proved to be stable for a long time. On the graphics processing part, extensions are under the way to take into account recent enhancements of OpenGL. To fully immerse into the on-site equipment, DMX 512 support is currently being implemented. On the audio part, we work on higher-level music analysis as well as improving the interactive mixing and virtual DJ (Disk Jockey) capabilities. A long-term endeavor is certainly the addition of media meta information such as style, audience perception, cultural aspects, or mood to the ontology.



Figure 3 -Live visuals produces by Soundium2

# References

N. Juillerat. 2001. SoundEngine "Un serveur d'effets sonores en temps réel." *Master's thesis*, Université de Fribourg, 1700 Fribourg, Switzerland

G. A. Kalberer, P. Müller, and L. van Gool. 2002. "Speech Animation Using Viseme Space." *Vision Modeling and Visualization 2002*, pp 463–470, Erlange, Germany

S. Matsuda and T. Rai. 2000. "DIPS: the real-time digital image processing objects for Max environment." *Proceedings of the International Computer Music Conference 2000*. International Computer Music Confernece

J. McCartney. 2002. "Rethinking the Computer Language: SuperCollider." *Computer Music Journal*, 26(4):61–68

Y. I. H. Parish and P. Müller. 2001. "Procedural Modeling of Cities". *Proceedings of SIGGRAPH'01*, pp 301–308, Los Angeles, CA, USA

M. Pukette. 2002. "MAX at Seventeen". *Computer Music Journal*, 26(4):31–43

T. Richardson and K. R. Wood. 1998. "The RFB Protocol." *ORL*, Cambridge, UK

S. Schubiger. 2002. "Automatic Software Configuration – A Model for Service Provisioning in a Dynamic and Heterogenous Environment." *PhD thesis*, University of Fribourg, No. 1393

S. Schubiger and B. Hirsbrunner. 2003. "JRPC - An Efficient RPC Implementation for Java." *Java Developer's Journal*, submitted

M. Specht. 2000. "Paralleler Software-Synthesizer – Cumulus." *Master's thesis*, ETHZ, Eidgenössische Technische Hochschule, 8092 Zürich, Switzerland

R. Ulyate and D. Bianciardi. 2002. "The Interactive Dance Club: Avoiding Chaos in a Multi-Participant Environment." *Computer Music Journal*, 26:3:40–49

---

[5] Not all artists have to use Soundium2 to participate – thanks to the wide range of supported input equipment and signal processing inside Soundium2, Soundium2 can be linked to almost any data source.