MEIG

A CONTROL SYSTEM FOR INTERACTIVE MULTIMEDIA PERFORMANCE

Carlo BARBAGALLO[1,2,3] and **Francesco BIANCHI**[1,2,3]

[1]**Tempo Reale**, Florence, Italy
[2]**Conservatorio Giuseppe Verdi**, Turin, Italy
[3]**COMET**, Turin, Italy

ABSTRACT

Artistic multimedia performances (live, networked, or virtualized) involve a great amount of data management. An efficient, adaptive, and immediate control system is needed to design, rehearse and perform the dynamic interactive behaviors of different kinds of synchronized devices parameters inside a logical and time-related structure.

MEIG (*multimedia Event interactive Generator*) is a multi-layered event management platform, inspired by a collaborative approach, based on editable and triggerable cues. The system allows to program dynamically the parameters, set the interactions among different devices and use them on a cue basis.

OSC (*Open Sound Control*) as a communication protocol guarantees a plain management of various media devices in a networked environment; the client/server architecture, based on web technologies (NODE.JS with a backend integrated SQL database and a browser-based navigable frontend), provides a collaborative system management, a wide operative system compatibility, and the integration into multimedia programming environments as CYCLING '74 MAX.

The project arises in the context of TEMPO REALE research scholarship in "Relational systems for the management of live events" as a development of the MEEG (*Max Electronic Events Generator*).

1. INTRODUCTION

Composing for multimedia before the advent of personal computers was predominantly based on human live performance: an operator took care of managing one or more electronic devices by acting directly on their hardware controls (Figura 1) The indications relating to the performance were placed on a score (or inspired by improvisation) and the operator used to play exactly like an instrumentalist.

The possibility to record the *electronic performance* on storage mediums and then recursively manipulate it allowed electronic media artists to produce purely electronic fixed works, on one hand to be performed on playback, on the other to integrate playback and live electronic performance features into more complex interactive practices. More and more, merging deferred and real-time approaches to process and control electronic devices has become crucial in the fields of live electro-acoustic mixed music, multimedia theatre, interactive installation, and in between these disciplines.

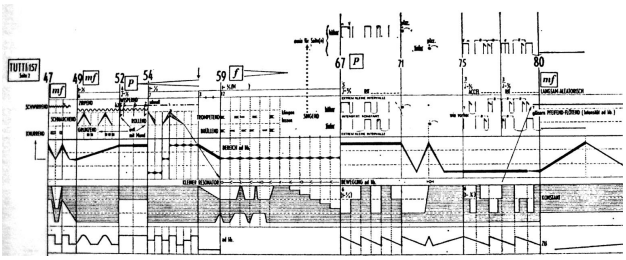

Figure 1. Karlheinz Stockhausen, *Mikrophonie 1* for Tam-Tam, 2 Microphones, 2 Filters and Potentiometers (6 players) (Universal, 1964)

With the mass advent of personal computer, device control can be delegated to predisposed algorithms that can automate the physical gestures. The complexity of an electronic score is no longer proportional to the number of operators and devices but it is determined by the computational capacity. Managing multi-layered actions over time through algorithms permits to dynamically configure the controls so that they move according to profiles established.

Electronic control notation becomes virtual: the media artist designs the dynamic behaviors of algorithms. The great amount of data control management can be very complex to handle and a flexible control system to efficiently compose, rehearse and perform multimediality is crucial. Media artists, especially during the composing process, as well as in the rehearsals, need to focus on specific problems and test significant changes to control the events immediately. In an *event-oriented* approach to performance, the need is to define and trigger the points of attack (*cues*) of pre-structured groups of control configurations. Furthermore, it is essential to define the level of interaction of the performer with the control system.

2. FROM MEEG TO MEIG

The MEEG application (*Max Electronic Event Generator*) [1] developed by Tempo Reale was born to face these problems, in particular in the field of *live electronics* for contemporary mixed music performances.

MEEG aims to integrate the CYCLING '74 MAX software with a relational database implemented using SQL (*Structured Query Language*) in order to aggregate a series of device configurations into cue events to trigger according to a score [2]. Optimized by music without rigid timing in which the electronic musician can follow the performance, the system is a generic platform for the automatic construction of specific messages for the Max software based on the device global settings that are managed by Max.

In the context of TEMPO REALE research scholarship in "Relational systems for the management of live events" the intent is to update and develop the MEEG system. To date, the result of the development is the *work in progress* project MEIG (*multimedia Event interactive Generator*) which aims to expand MEEG in the following directions:

- abstract the system from the Max integration, in order to be able to interact with any software equipped with OSC protocol;
- provide a notational system that allows the media composer to think a score in terms of structure, with the possibility of arbitrarily stratify and handle the logical subdivisions of it;
- define abstract control entities that can directly talk to any hardware or software device (audio, video, lights, prototyping boards, or whatever) to use the system for a wide range of multimedia art projects;
- integrate the system with autonomous playback features besides the export of the entire score to file: manual activation of cues, playback of automatic sequences of cues, and interactive external timecode playback management.

3. WORKFLOW

MEIG workflow proceeds both in a top-down and in a bottom-up direction. Top-down, from the macro-structural *Score Level* to ever greater detail of formal segmentation (*Sections*) till the *Event Level* that host the dynamic behaviors of all controls; bottom-up, starting from the definition of the devices (*Device Level*) and their control parameters, up to the writing of Events to be placed in the structure (Figure 2).

3.1 Top-down flow

Following the top-down flow, the first step is to define a multi-layered global structure (*Score*) formed by a certain number of logical, configurable, and recursive subdivisions (*Sections*).

Logical as they satisfy the need to structure the score, configurable because they allow configuring parameters

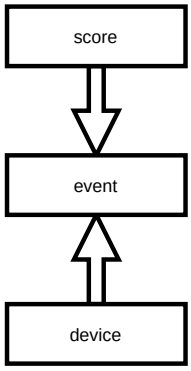

Figure 2. mEiG workflow.

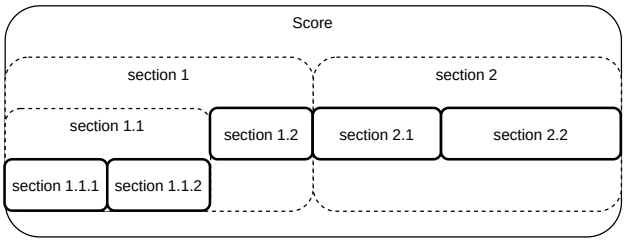

Figure 3. Recursive stratifications of *Section* layer.

and interconnections between devices at a section level, recursive as they can be divided into other sections. The low-level sections constitute new partitions of the top-level parent section and inherit the static configurations from them unless new default configurations are stored in the lowest layer (Figure 3).

Further on this structure, there is the *Event Level*(Figure 4). An Event is an entity that allows the definition of time-based automation of groups of parameters and dynamic interactions between devices. One or more events refer to a parent *Section* temporally via an attack offset and inherit their parameters configurations (if not changed by the event itself). The *Event Level* is also multi-layered: events can overlap without limits, thus allowing a *polyphonic* dynamic handling of a group of parameters. Inside a single event it's possible to write automation of parameters belonging to different devices, thus allowing to focus on a multi-device approach (rather than *by device*) to design multimediality.

Sections are unrelated to time but they have to be configured to host Events that are related to. It's required a global configuration of the Sections to a timecode format.

3.2 Bottom-up flow

On the bottom-up direction, the starting point is the *Device Level* (Figure 5). Here the user creates abstract labels that refer to the actual hardware or software that he intends to control. The abstract device initialization is completed by the definition of a group of parameters (*Params*), their default values (that will be stored on the Main Layer of the Score Level), and their interconnection points (I/O). When a typical OSC message is created by the system, the first part of the address consists of the device label followed by

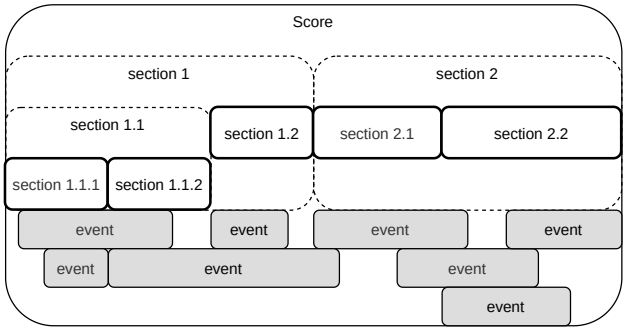

Figure 4. *Event layer*. Every event refers to a section just above.

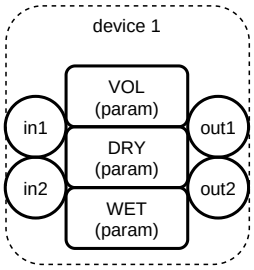

Figure 5. *Devices*. Parameters and interconnection points.

parameter and value.

There are also *Super-Parameters* (Figure 6), i.e. aggregations of device parameters: it is possible to link parameters to manage them from a single control. As described before, static configuration of parameters and super-parameters values can be done at a Score and Section Level, while the time-related dynamic behavior of them is managed by the *Event Level*.

Once all the controls are created into the MEIG system, the user can populate the *Events* (Figure 7). Values can be formatted in various ways according to the needs: the user can insert a single alphanumeric value or a list or edit a time-related automation curve.

4. CUE LIST

Once defined, Sections and Events can be inserted into *Cue Lists*. Cue i.e. the markers that contain the metadata that specify what must be performed. A cue is nothing more than a reference to one or more entities of the score, which can be recalled through a triggering action.

There are many ways in which triggers are defined. A cue can be triggered manually or automatically at the end of another.

MEIG provides a flexible cue activation system, which allows the user to define arbitrary levels of detail on activation, from the single event to the automated triggering of the entire score, even in a non-linear way.

5. OSC PROTOCOL

The MEIG communicates with the outer world in real-time via an OSC-compliant messaging system. The parser generates messages with an Address Space obtained

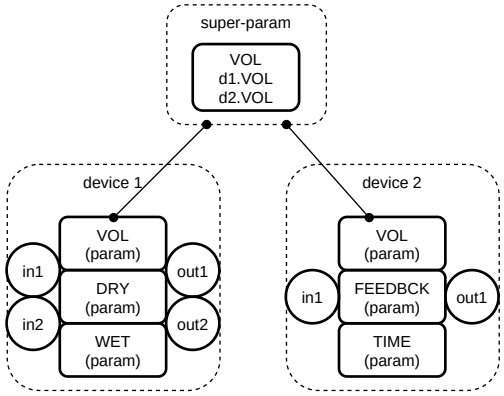

Figure 6. *Super-Parameters*. Here you can aggregate parameters of different devices.

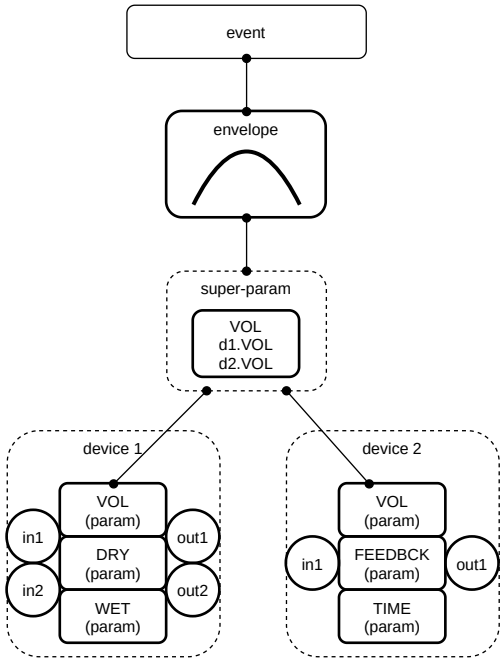

Figure 7. A Super-parameter, in conjunction with a control shape, can populate an event.

from the labels attributed to the devices, and where the parameters constitute the OSC Methods. The timing of the message output is reconstructed by the system starting from the offsets defined within the events.

MEIG allows also to generate output files of the entire score, which contains the list of OSC messages and the timing indications on the multimedia score. It is also possible to define some output data formats (yaml, xml, json, coll for max, and others).

6. A COLLABORATIVE SYSTEM

MEIG is a collaborative system, both from an editing and playback point of view. The application defines a network (local or on the internet) of nodes in which nominal roles are established. Conceptually, in editing mode, all the nodes work on a copy of the project which is updated when changes occur. In this context, all the participants

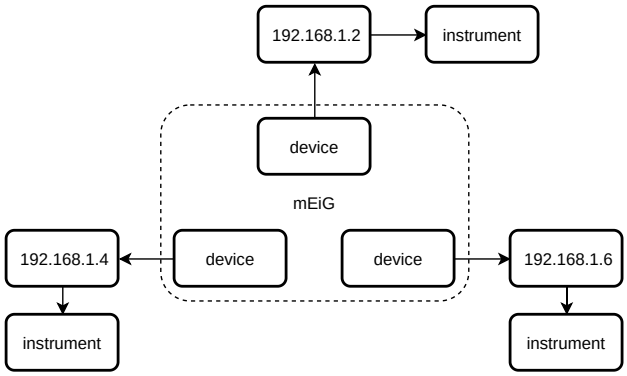

Figure 8. mEiG devices can be routed to differents IP addresses.

could play the same role of *creators*, without particular hierarchical constraints.

On playback mode, on the other hand, a node assumes the role of master and directs the activation of the cues. In a simple scenario, the master is the only node in the network and can be approached by another that acts as a backup node. Always synchronized, the backup node replaces the master when some malfunction occurs. In a more complex situation, the nodes can be connected to hardware or software real devices: on playback, the OSC messages generated by the system are automatically routed to the right destination (Figure 8).

7. TECHNOLOGIES

MEIG, open-source and multi-platform, will be based on a client-server architecture (Figure 9), following the model of a web app and built on the following technologies plus support frameworks:

- NODE.JS (with express.js framework) for the backend (`https://nodejs.org`);
- SQLITE as the database engine (`https://www.sqlite.org`);
- GRAPHQL as the query language (`https://graphql.org/`);
- VUE.JS for the frontend (`https://vuejs.org/`).

The choice of sqlite was motivated by the need for flexibility and portability. Moreover, the relational model seemed to be the most suitable to represent the structure of the Score and all the entities deriving from it. The exposed APIs are built on a GraphQL server, which allows the targeted and efficient querying of the data needed from a single endpoint. Unlike the Representational State Transfer (REST) model, which operates by identifying each resource with a specific endpoint, GraphQL acts according to a scheme that defines the resources on which it operates. At the client level, it is therefore possible to request exactly the resources you need, defining real queries with the fields of interest.

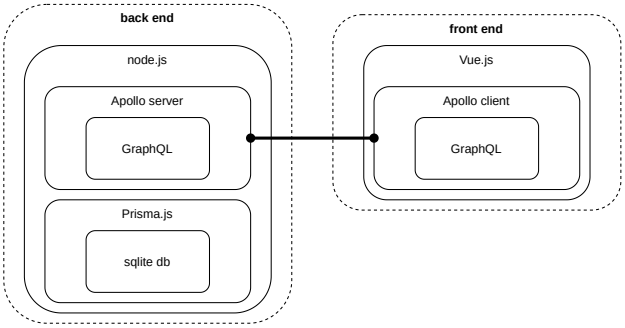

Figure 9. Diagram of software stack.

The user interface, which can be navigated from a common browser, is implemented in Vue.js, incorporating a GraphQL client for resource requests.

8. CONCLUSIONS

MEIG is currently a *work in progress* draft project born from the collaboration between the writers and *Tempo Reale* (Damiano Meacci e Francesco Canavese), discussing the guidelines and critical issues of the development.

The subsequent in-depth work, to be carried out in parallel with the implementation, concerns equipping the system with external interaction with other protocols than OSC, parsing different types of data structures (ex. MIDI) directly into the system and at its output. This implies the possibility of bypassing brokerage systems and communicating directly with any kind of device involved in the performance.

Considering the age-old problem of technological obsolescence in media arts [3], a system as imagined and described here could work as a remedy for the risk of losing the heritage of the multimedia composition and performance practices.

9. REFERENCES

[1] F. Canavese, F. Giomi, D. Meacci, and K. Schwoon, "An sql-based control system for live electronics," in *Proceedings of the 2005 International Computer Music Conference*, Barcelona, September 2005.

[2] D. M. F. Giomi and K. Schwoon, "Live electronics in luciano berio's music," *Computer Music Journal*, vol. 27, no. 2, pp. 30 – 46, 2003.

[3] N. Bernardini and A. Vidolin, "Sustainable live electro-acoustic music," in *Proceedings of the International Sound and Music Computing Conference*, Salerno, November 2005.