VIRAGE : Designing an interactive intermedia sequencer from users requirements and the background

Antoine Allombert, Raphaël Marczak, Myriam Desainte-Catherine, Pascal Baltazar, Laurent Garnier

VIRAGE : DESIGNING AN INTERACTIVE INTERMEDIA SEQUENCER FROM USERS REQUIREMENTS AND THEORETICAL BACKGROUND

Antoine Allombert,Raphël Marczak, Myriam Desainte-Catherine
LaBRI

Pascal Baltazar
GMEA

Laurent Garnier
Blue Yeti

ABSTRACT

We present the unrolling of the *Virage* project and its main achievement : a sequencer for writing and performing interactive intermedia scenarii. This two years long project addressed the question of writing and controlling interactive scenarii involving several heterogeneous digital media, in the context of the performing arts. We show here how we adapted a theoretical background, thought for the interpretation of musical scores, to specific requirements in collaboration with field agents. We also describe how these exchanges about the features of the sequencer leaded the software development.

1. INTRODUCTION

The *Virage* project attempted to address the questions of writing time and interaction in the context of stage management for living arts, and also for museography tools. For several years, stage managers and creators have used more and more digital contents and tools. For this purpose the field workers use software and tools designed for each type of contents, which often come from other fields (computer music as an example). However, more and more complex cases challenge the possibilities of writing the time organization of heterogeneous contents, as some ways to interact with them.

The *Virage* project proposed some answers to these questions, by designing an interactive sequencer for writing time and interaction with heterogeneous digital contents. The results of the study and the participation of field workers fueled the the discussion of the theoretical model. In the same manner, each step of the implementation was confirmed by every members of the project.

We first present some elements of the study. Then we focus on the theoretical background of the project, and how the field requirements influenced the design of the model and its implementation. At last, we propose some applications and perspectives.

2. CONTEXT AND CHALLENGES

This project can be considered as a practical reflection on time-based media scripting in digital artistic creation, particularly in relation to live performance. Previously to the project, a study has been carried out and published [4] in order to analyze the limits of existing software systems for interactive performing arts situations. This preliminary study was then completed by a field survey[1] conducted at the very beginning of the project, that leaded to write initial specifications that will be updated at the very term of the project (February 2010). The final objective of the project is not to produce effective software, but these specifications, in order to allow future development. Though, the key point of the research statement has been to make generate these specifications through the development of usable prototypes, and feedback from their confrontation to real-world cases

2.1. Interoperability, modularity

The insularity between the existing multi/intermedia software environments, has been stated through the preliminary studies as a major bottleneck for creative work flow in the context of the performing arts.

The Virage sequencer[2] has then been proposed as an experimental way to overcome this insularity by acting as a hub between these environments, dedicated to the design of temporal parameter management. This has been achieved by combining the Virage sequencer with the existing software and hardware environments that are actually used in production by the field agents, and that correspond to the habits. In order to do that, a plugin interface has been developed, that will be further explained in 4 and allows the coexistence and simultaneous usage of several protocols, including : Open Sound Control, Minuit [3], CopperLan [4], potentially any network or device protocol can be added to the list by developing third-party dedicated plugins.

In respect to this concern of interoperability, the Virage sequencer focuses on the scripting of behaviors of abstract parameters, and not of the media themselves (as in all existing sequencers, except Iannix [7], which has been reviewed and didn't show as relevant to the studied field, namely the performing arts) This design choice allows a certain independence from the medias' intrinsic durations, similarly to

[1] `http://www.plateforme-virage.org/wp-content/uploads/2009/11/Virage_Analyse_des_Usages.pdf`
[2] freely downloadable after registration at : `http://www.plateforme-virage.org/?p=1297`
[3] specifications available at : `http://www.plateforme-virage.org/?p=1444`
[4] `http://www.copperlan.org/`

what was experienced with MIDI sequencers before the introduction of DSP.

2.2. Flexible time

Existing creative multi/intermedia software environments generally address time-scripting by one of the next two strategies : a fixed time line (DAWs and Video editing softwares), that allows a very precise scripting of media behaviors or cue lists (Theater Cue Managers such as Qlab[5] but also in Ableton Live[6] which is very frequently used in performing arts projects)

Time is then managed as a monolithic fixed flow (as in the time line model), or as a set of unrelated discrete events (as in the cue list model). It is to note that Ableton Live proposes both of these models in the software, but that they are completely unrelated temporally.

The field agents expressed a need for a third strategy that might be a mix of these ones. As soon as the comedians or the technical staff modify their behavior during a given performance, the fixed time line representation appears to be too strict. On the other side, the cue representation is lacking of possibilities of designing complex time structures.

Thus we developped a system in which one can express complex and precise time organization : time relations between events of the scenario, due to artistic choices or real-life constraints, and also some indicative values for the time intervals between the events. On the other side, it might allow following the behavior of the agents, by changing some characteristics of the temporal organization during a performance. For this purpose, the system should adapt the scenario by respecting the time relations and accepting real-time modifications. We use the term “flexible time” to describe this type of behavior.

In such a system, a scenario is no more the representation of a single performance, but it represents a set of possible performances that share temporal proprieties. Then, our system can be compared to other works as *Harp* [5] or *DoubleTalk* [9], which considered a musical score as the representation of several possible executions with shared proprieties.

3. THEORETICAL MODEL

The sequencer of the *Virage* project is based on an hybrid temporal paradigm that has been designed for musical composition

For a specific musical approach and a complete presentation, one can refer to [1].

We assume that an interactive scenario is the temporal organization of a set of *events*, in which the temporal characteristics are partly specified to allow some modifications during the performance. During the writing, the user (the director and/or the stage manager) can specify some temporal proprieties that he wants to be respected during each performance : temporal orders between some events and ranges of value for the time intervals between the events. He also

[5] http://figure53.com/qlab/
[6] http://www.ableton.com/

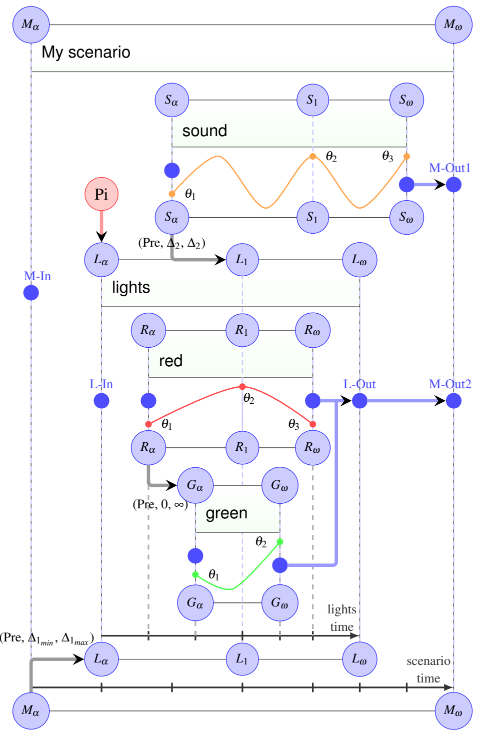

Figure 1. An interactive scenario

specifies what can be changed during a performance. This way of writing the temporal organization is close to “constraints programming”, since a performance constitutes an instance among possible scenarii that respect some given rules.

3.1. The writing model

We present here the graphical language used for writing the interactive scenarii. An example of such a scenario is given on the figure 1.

Since the system is though as a hub that can send abstract data to other applications, the writing formalism is based on *processes* able to compute and produce abstract values. The user can define how the processes will run during the performance. For this purpose, he defines the temporal organization of his scenario by using *temporal objects*.

The *temporal objects* are the basic element of the temporal model which is inspired from the hierarchical models proposed by Mira Balaban [3]. A *temporal object* with no children, called a *texture*, represents the execution in time of a given *process*. A *temporal object* with children, a *structure*, represents the temporal organization of its children. Each *structure* has its own time line with scaled with its own time unit. A scenario is represented by a root *structure*. Each object is associated with a set of *control points* which represent particluar moments of its execution.

On the figure 1, the *objects* “sound”, “red” and “green” are *textures* that represent the execution of processes, which read tables in order to send values to specialized applications ; “lights” and “My scenario” are structures.

3.1.1. Temporal relations

The user can define the temporal proprieties of his scenario through temporal relations between the *control points* of the *temporal objects*. Thus these relations are taken from points algebra. There are 2 qualitative relations : precedence (*Pre*) and posteriority (*Post*). In addition, the user can specify quantitative constraints, by giving a range of possible values for the time interval between two points bound by a relation.

The system will maintain the proprieties imposed by the temporal relations, during the writing process as well as during the performance. In the first case they can help the editing operations, in the second case, they define the limits for the modifications introduced by the performer through the *interaction points*.

3.1.2. Interaction Points

During the writing process, the user can define some *control points* to be dynamically triggered by the performer during the execution. These *control points* are said *dynamic* (as the beginning of the “lights” *structure* on the figure 1), while the other *control points*, the *static* ones, will be triggered by the system. The written date of a dynamic *control point* is indicative.The possibilities are limited by the temporal relations. Then, on the figure 1, the system will refuse the dynamic triggering of the point L_α before the duration $\Delta 1_{min}$ has elapsed, in order to respect the relation between M_α and L_α. In like manner, the system will automatically trigger L_α if the performer did not before the duration Δ_{max} has elapsed.

3.2. The execution model

For the execution side of the system, we propose a generic abstract machine, called the *ECO* Machine, for running the scenarios written with the formalism.

Inside this machine, we represent the temporal organization of the scenario with Petri nets structures (precisely hierchical time stream Petri nets [8]), which are runable place/transition representations of partialy ordered set of events ; they have been used several times in computer music applications [6]. Other representation could be explorered such as concurrent constraints [2].

4. IMPLEMENTATION

In this section we present the *IScore* library, an implementation of the formalism described in section 3. Thus, the library is the union of 2 parts which corresond to the sides of the model : an editor engine to write the temporal constraints based scenarii, an execution engine to compile and to run the scenarii. This library, developed in C++ and currently working on *Mac OSX* and *Linux*, is open source.

4.1. Architecture

The architecture of the implementation correpsonds to the theorical model.

4.1.1. Authoring

For the authoring part, we use a constraints solver solver library (*GeCode*[7]. The dates of the *control points* and the temporal relations are turned into a constraints problem. Each time a date is changed, the solver is called to find a new values for the other dates and tht scenario is updated.

4.1.2. Execution

For this side, we directly implemented the ECO machine as it is theorically described. Thus, we implemented a Petri nets runner. During the execution, the Petri nets is run according to its temporal caracteristics. Whenn triggering an event, the runner sends a message to a part of the software which lanches and stops the processes associated to the *temporal objects*.

4.1.3. Controller library

The Controller library is implemented to manage network communications. Indeed the *Virage* software has to control many different technologies by sending messages. This library hides the protocols used to communicate with aother applications. When authoring a scenario, user only see some abstract devices that represent external environements. He can write the time evolution of the parameters of these environments without knowing how the values will travel trought the network during the execution.

4.2. Users involvement

Users have been implied in the research process all along the project, and at all its stages : from the initial definition of features through directed experimentations to the conception stage by giving constant feedback and defining priorities of the development tasks. Then, once the first stable prototype had been produced, all successive steps of the development were the opportunity to confront the prototypes to a diversity of real-world situations generated by experimental workshops leaded by the 6 artistic partners of the research platform, ranging from the performing arts to digital interactive arts. These experimentations of the prototypes have been done within the existing creative environments of the field agents and developed around actual artistic questions. These experiments have been done in actual production situations, but out of the production time, in order to avoid stressing the research process with production deadlines and constraints.

4.3. Graphical interface

The main space of the graphical interface is dedicated to the edition of the temporal scenario, using the forthcoming elements : *Temporal objects* are represented by boxes on the time line, that can be connected at each end by *Intervals*, may they be fixed (bold lines), flexible (dashed lines) or semi-flexible with bounds (combination of bound and dashed lines representing the respective fixed and flexible durations). *Trigger Points* can be created and attached to

[7] `http://www.gecode.org/`

the temporal objects, allowing the user to interactively trigger them. They are currently placed on an upper rail for more straightforward ordering and triggering by the operator, when executing the scenario. Though, it has been noticed that this representation was in some way restrictive to the creative temporal possibilities of the model. This placement of Trigger Points will be alternatively moved against the objects or on the time line rail depending on the editing mode (creation or execution) in a future version.

Apart from this main space, the user has access to a Namespace browser for exploring the remote environments.

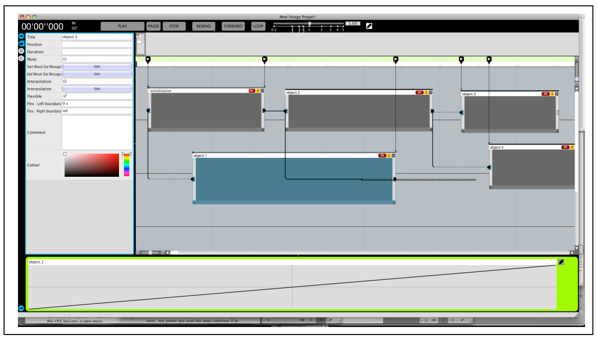

Figure 2. A screenshot of the graphical interface.

5. PERSPECTIVES

Consolidation of the prototype as a usable cross-platform open source software by a further structuration of the project's consortium and associated partners as a cooperative in order to keep the collaborative spirit developed throughout the research project

This consolidation phase will mostly focus on robustness (including porting the graphical interface to the Qt framework), ergonomy and usability

A few features that were not developed in the project will be added, including routing and conditioning of data flows or events from and to remote addresses, in order to allow to script continuous interaction and edition of hierarchical scenarios (this feature is actually implemented in the engine)

Further research has been submitted to focus on the integration of conditional objects, non-linearity, multi-user scenarii.

6. CONCLUSION

We presented a research project that consisted in gathering researchers and field agents to propose solutions to real problems. Both groups came with a background, theoritical models and results on one side, needs and experience on the other side. The all along communication allowed us to fuel the research process with new questions that leaded to adapt the theoretical background, but also to improve it for new developments. The scientific approach helped the field agents to clarify their point of view on their own practices, which evolve rapidly. It is clear that this parternship was successful because both parts were enough mature, existing models for writing interaction on the one hand, real cases and newly established practices on the other hand.

A least, the software prototype gave an experimental tool and initiated the development of a really efficient software.

7. ACKNOWLEDGMENTS

This research project has been funded by French National Agency, in the "Audiovisual and Multimedia program. The authors would also like to thank all of the partners of the Virage platform[8] that have been contributing by feedback, advising and workforce in order to achieve this project.

8. REFERENCES

[1] A. Allombert, "Aspects temporels d'un système de partitions musicales interactives pour la composition et l'interprétation," Ph.D. dissertation, Université de Bordeaux, 2009.

[2] A. Allombert, G. Assayag, M. Dessainte-Catherine, and C. Rueda, "Concurrent constraints models for interactive scores," in *Pr. of the 3rd Sound and Music Computing Conference (2006), GMEM, Marseille, France*, May 2006.

[3] M. Balaban and N. Murray, "Interleaving time and structure," *Computers and Artificial Intelligence*, vol. 17, no. 1, pp. 1–34, 1998.

[4] P. Baltazar and G. Gagneré, "Outils et pratique du sonore dans le spectacle vivant," in *Actes des Journées d'Informatiques Musicales (JIM07), Lyon, France*, Avril 2007, pp. 153–162.

[5] A. Camurri, A. Catorcini, C. Innocenti, and A. Massari, "Music and multimedia knowledge and reasoning: the harp system," *Computer Music Journal*, vol. 19, no. 2, pp. 34–58, 1995.

[6] A. Camurri, G. Haus, and R. Zaccaria, "Describing and performing musical processes by means of petri nets," *Interface*, vol. 15, pp. 1–23, 1986.

[7] T. Coduys and G. Ferry, "Iannix - aesthetical/symbolic visualisations for hypermedia composition," in *Proc. of the* 1*st* *Sound and Music Computing Conference (SMC04), Paris, France*, 2004.

[8] R. W. P. Sénac, P. De Saqui Sannes, "Hierarchical time stream petri nets : A model for hypermedia systems," in *Proc. of the 16th International Conference on Applications and Theory on Petri Nets, Turino, Italy*, G. D. Michaelis and M. Diaz, Eds., 1995, pp. 451–470.

[9] S. Pope, "Music notations and the representation of musical structure and knowledge," *Perspectives of New Music*, vol. 24, no. 2, 1986.

[8] `http://www.plateforme-virage.org/?page_id=14`