ABSTRACT
We present the research process of the Virage project and its main achievement: a sequencer for authoring and performing interactive intermedia scenarios. This two year long project addressed the question of authoring and controlling interactive scenarios dealing with several heterogeneous digital media, in the context of the performing arts. We show here how we adapted a theoretical background, meant for the interpretation of musical scores, to specific requirements in collaboration with practitioners in the field. We also describe how these exchanges about the features of the sequencer guided the software development.

1. INTRODUCTION
The Virage project attempted to address the questions of authoring time and interaction in the context of media management (sound, lighting, image, machinery, etc.) for the performing and interactive arts. For several years, media managers and artists have used more and more digital contents and tools. For this purpose the practitioners use software and tools designed for each type of content, or adapt software coming from other fields (e.g. computer music). However, more and more complex cases challenge the possibilities of scripting the time organization of heterogeneous contents, as well as ways to interact with them.

The Virage project proposed some answers to these questions, by designing an interactive sequencer for scripting time and interaction with heterogeneous digital contents. The results of the study and the participation of practitioners fueled the discussion of the theoretical model. In the same manner, each step of the implementation was approved 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. Finally, 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. Prior to the project, a study was carried out and published 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 conducted at the very beginning of the project, that lead to write initial specifications that were updated at the term of the project (February 2010). The final objective of the project was indeed not to produce concrete software, but rather to write these specifications — in order to allow future development — 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, was stated through the preliminary studies as a major bottleneck for creative workflow in the context of the performing arts.

The Virage sequencer was then 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 was achieved by combining the Virage sequencer with the existing software and hardware environments that are currently used in production by the practitioners in the field. In order to do that, a plugin interface was developed, that will be further explained and allows the coexistence and simultaneous usage of several protocols, including: Open Sound Control, Minuit, CopperLan, potentially any network or device protocol can be added to the list by developing third-party dedicated plugins.

With respect to this concern of interoperability, the Virage sequencer focuses on the scripting of behavior of abstract parameters, and not of the media themselves (as in all existing sequencers, except Iannix, which was reviewed and was not considered as relevant to the studied field, namely the performing arts) This design choice allows a certain independence from the intrinsic duration of
the media, similarly to 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 timeline (DAWs and Video editing software), that allows a very precise scripting of media behavior or cue lists (Theater Cue Managers such as Qlat\(^5\) or SFX 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 timeline 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 temporally completely unrelated.

The practitioners in the field expressed a need for a third strategy that might be a mix of these ones because, on the one hand, the fixed timeline representation appeared to be too strict to adapt to the live character of performing arts. On the other hand, the cue representation is lacking possibilities of designing complex time structures.

Thus we developed a system in which one can express complex and precise time organization: temporal relations can be drawn between events of the scenario, according to artistic choices or technical constraints, with optional duration for the intervals between the events. Additionally, the behavior of the live action can be followed by changing some characteristics of the temporal organization during a performance, including the execution speed of the scenario. 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 properties. 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 properties.

3. THEORETICAL MODEL

The Virage sequencer is based on an hybrid temporal paradigm that has been initially designed for musical composition.

For a specific musical approach and a complete presentation, 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 authoring phase, the user (the media manager or artist) can specify some temporal properties that he wants to be respected during each performance: temporal orders between some events and ranges of values for the time intervals between the events. He also specifies what can be changed during a performance. This way of scripting the temporal organization is close to “constraints programming”, since a performance constitutes an instance among possible scenarios that respect some given rules.

3.1. The authoring model

Here, we present the graphical language used for authoring the interactive scenarios. An example of such a scenario is given in figure 1.

Since the system is thought of as a hub that can send abstract data to other applications, the authoring 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 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 timeline, 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 particular moments of its execution.

In 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;

\[^{5}\text{http://www.figure53.com/qlab/}\]
\[^{6}\text{http://www.ableton.com/}\]
“lights” and “My scenario” are structures.

3.1.1. Temporal relations

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

The system will maintain the properties imposed by the temporal relations, during the authoring 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 authoring 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 in 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, in the same example, the system will refuse the dynamic triggering of the point \( L_a \) until the duration \( \Delta_{\text{min}} \) has elapsed, in order to respect the relation between \( M_a \) and \( L_a \). Similarly, the system will automatically trigger \( L_a \) if the performer did not do it until the duration \( \Delta_{\text{max}} \) has elapsed.

3.2. The execution model

On the execution side of the system, we propose a generic abstract machine, the ECO Machine, that runs the scenarios written with the formalism.

Inside this machine, we represent the temporal organization of the scenario with Petri net structures (or more precisely hierarchical time stream Petri nets [8]), which are runnable place/transition representations of partially ordered set of events; they have been used several times in computer music applications [6]. Other representation could be explored 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 corresponding to both sides of the model: an editor engine to write the temporal constraints based scenarios and an execution engine to compile and run the scenarios. This library, developed in C++ and currently working on Mac OS X and Linux, is open source[7].

4.1. Architecture

The architecture of the implementation corresponds to the theoretical model.

4.1.1. Authoring

For the authoring part, we use a constraint solver library (GeCode[8]). 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 new values for the other dates and the scenario is updated.

4.1.2. Execution

On this side, we directly implemented the ECO machine as is theoretically described. Thus, we implemented a Petri net runner. During the execution, the Petri net is run according to its temporal characteristics. When triggering an event, the runner sends a message to a part of the software which launches and stops the processes associated with the temporal objects.

4.1.3. DeviceManager library

The DeviceManager library was implemented for the purpose of managing network communications. Indeed the Vírage software has to control many different types of technology by sending messages. This library hides the protocols used to communicate with other applications. Communication protocols quickly evolve so we had to think about an extendable solution based on a plugin system conceived in the Tulip project[9] and develop dedicated plugins for each of the supported protocols.

4.2. Users involvement

Users have been implicated 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 with a diversity of real-world situations generated by experimental workshops lead by the 6 artistic partners of the research platform, ranging from the performing arts to digital interactive arts. These experiments of the prototypes were done within the existing creative environments of the practitioners in the field, and developed around actual artistic questions, in actual production situations, but out of 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, and that can be connected at each end by Relations, may they be fixed (bold lines), flexible (dashed lines) or semi-flexible with bounds (combination of bold and dashed lines representing the respective fixed and flexible durations). Trigger Points can be created and attached to 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 towards the objects or on the time line rail depending on the editing mode (authoring 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](image_url) A screenshot of the graphical interface.

5. PERSPECTIVES

The consolidation of the prototype as a usable cross-platform open source software is currently contemplated by a further consolidation 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), ergonomics 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. Also, the edition of hierarchical scenarios will be added to the features of the graphical interface.

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

6. CONCLUSION

We presented a research project that consisted of gathering researchers and practitioners in the field, to propose solutions to real problems in the field of interactive media arts. The constant communication along the way allowed us to fuel the research process with new questions that lead to adapt the theoretical background, but also to improve it for new developments. The research situation helped the field agents to clarify their point of view on their own constantly-evolving practices, by stepping back from their production-oriented habits to a more experimental situation questioning the very nature of the temporal aspects of their creation.

This partnership proved successful because both sides were mature enough and brought along existing models for scripting interaction on the one hand, real cases and newly established practices on the other hand. Finally, the software prototype represents an experimental tool and initiated the development of really efficient software.

7. ACKNOWLEDGMENTS

This research project has been funded by French National Agency, in the “Audiovisual and Multimedia” 2007 program. The authors would also like to thank all of the partners of the Virage platform[10] that have been contributing by feedback, advising and workforce in order to achieve this project.

8. REFERENCES
