INScore Time Model

D. Fober Y. Orlarey S. Letz
Grame
Centre nationale de création musicale
Lyon - France
{fober, orlarey, letz}@grame.fr

ABSTRACT

INScore is an environment for augmented interactive music score design, oriented towards unconventional uses of music notation, without excluding conventional approaches. In this environment, although all the objects of a score have a temporal dimension, the time remains fixed *i.e., the date (or duration) of an object does not change, except when a message is received (sent from an external application or resulting from events handling). Thus, INScore does not include a time manager in the classic sense of the term. This choice was based on the fact that the system was originally designed to be used with sound production software (e.g., Max/MSP, Pure Data), that have more strict real-time constraints than INScore's graphical environment. However, the need to introduce dynamic time has gradually emerged, leading to an original model, both* continuous *and* event based. *The paper presents this model and its properties in the frame on INScore.*

1. INTRODUCTION

There are numerous non-computer based scores that allow the performers to move around the piece in a non-linear time fashion (e.g. John Cage - Variations series [1], Boucourechliev - Archipel series [2], Stockhausen - Refrain [3],...). This approach to time and musical form had its golden age from the end of the 1950s. To some extent, it can be seen as the premises of contemporary interactive music. It is emblematic of a musical thought that breaks with the traditional approaches of time and needs to be taken into account in computer environments for musical composition. Time manipulations in these environments can be expressed in very different ways.

In OpenMusic, there are three main types of time expression: "musical time (i.e., time of symbolic notation, relative to a tempo), proportional time (expressed in milliseconds), and continuous time (defined by mathematical functions)" [4].

In other environments such as iScore [5], time is also *event based* and can be described in terms of Allen relations [6], which allows to introduce reactive aspects in the musical pieces. More recently, these reactive aspects have also been introduced in OpenMusic [7], testifying to their importance in contemporary compositional approaches.

With Antescofo [8], time management approach is a sophisticated combination of events, tempo, continuous functions, and allows processes synchronisation as well.

INScore differs from the above environments in that the system is only dedicated to music score design. Time in INScore [9] is *static*: all objects have a date and a duration expressed in musical time that are fixed by the score description. The common temporal dimension of all objects of a score allows to represent their temporal relations in the graphical space [10]. The *animation* of time (e.g., the modification of the temporal attributes of an object) occurs when OSC messages are received. These messages are necessarily transmitted from an external application.

INScore was originally designed to work with sound production software (e.g. Max/MSP, Pure Data), that have strict real-time constraints and offer effective and accurate time management mechanisms. Nevertheless, the reactive aspects were quickly taken into account with the introduction of *events*, which allow to interact with the musical score [11]. Initially, the typology of these events includes classical user interactions events (e.g., mouse events), and introduces events in the temporal domain (notably time intervals monitoring). This mechanism of *events* paved the way for an original approach to music score programming by allowing to place messages - and thus interactions - in the temporal space. Nevertheless, as time was always *static*, the use of an external application remained necessary to actuate it.

The need for a *dynamic* representation of time has thus emerged. It has been solved in a simple way, with the introduction of an additional attribute to the objects of a score: *tempo*. This simple addition, however, has an important impact on the design of time in INScore and on the dynamic aspects of the score. We will first present the musical time model, which is controlled by the tempo. We will then see how this time combines with event-based time, and allows to program *autonomous* dynamic scores. We will finally show some use cases before concluding and giving some perspectives for the extension of the temporal dimensions of the objects of a score.

2. THE MUSICAL TIME MODEL

From the user point of view and for a given object, time is active when its tempo attribute is non-zero: a non-zero tempo value moves the object in musical time, according to its tempo and the flow of absolute time.

We will talk about *date* to refer to the musical time of an object, and about *time* to refer to a point in the flow of absolute time.

Let t_0 be the time of activation of an object time and v the value of its tempo, the date d_t of the object at a time t is given by a time function f such that:

$$f(t) \rightarrow d_t = d_{t0} + (t - t_0) \times v \times k, \quad t \geqq t_0 \qquad (1)$$

where d_i is the date of the object at time t_i and k a constant to convert absolute time to musical time. In practice, the absolute time is expressed in milliseconds and the unit of musical time is the whole note. Therefore, the value of the constant k is $1/1000 \times 60 \times 4$.

Each object of a score has an independent tempo. The tempo value is signed, meaning that an object can move in any direction of time (forward or backward). There are few environments allowing to express this reversal of time in a simple and natural way, we can quote IanniX [12], a graphical sequencer inspired by the work of Iannis Xenakis and that can be viewed as a *time writing system*.

2.1 Implementation

In INScore, the time granularity is that of the display: the system runs asynchronously and a periodic time task processes the incoming messages and computes the graphical rendering. In practice, since the effects of time are visible only in the graphical space, we consider that it isn't necessary to adopt a finer time resolution.

On the other hand, the frequency of the time task can be variable:

- by default, it is performed every 10 ms but the user can freely change this value by sending the `rate` message to the application, followed by a value in milliseconds.
- the time task is not re-entrant and the complexity of the display can potentially slow it down.

Thus, and at each time task, the system measures the actual elapsed time. Then, objects whose tempo is non-zero use this value to compute an offset expressed in musical time as indicated in (1), then self-send a `ddate` message (moving the date in relative mode) with the corresponding value.

```
/ITL/scene/obj tempo 60

-> /ITL/scene/obj ddate f(r_i)
-> /ITL/scene/obj ddate f(r_{i+1})
-> /ITL/scene/obj ddate f(r_{i+2})
-> ...
```

Figure 1. A sequence of messages activating the time of an object `obj`. Messages prefixed by `->` are generated by the object itself. r_i is the time elapsed between task i and $i-1$.

To be consistent with the overall system, this implementation is entirely message-based. Thus it is compatible with all INScore mechanisms, such as the message *forwarding* system.

3. THE EVENTS-BASED TIME.

The *events-based* approach of time in INScore preceded the musical time model and was first presented in [11]. As a reminder, the process of event interaction relies on the association of messages and events of the system. These messages are sent when the events they are associated with occur. The general message format for creating such associations is described in figure 2.

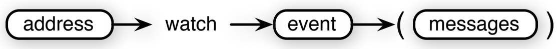

Figure 2. Format of an interaction message : the `watch` method installs a list of messages associated to the event `event`.

With the first version presented, the event's typology was limited to classical UI events (such as mouse clicks), extended in the time domain (see table 1).

Graphic domain	Time domain
mouseDown	timeEnter
mouseUp	timeLeave
mouseEnter	durEnter
mouseLeave	durLeave
mouseMove	

Table 1. Main events of the system in its initial version.

This typology has been significantly extended to include:

- any attribute of an object: the modification of an attribute value may generate the corresponding event, that carries the name of the attribute (e.g., `x`, `y` `date`, etc. .).
- the intrinsic data of an object, i.e., those defined by a `set` message. That's the `newData` event, introduced for the purpose of symbolic scores composition [13].
- arbitrary events defined by the user.

Any event can be triggered by sending an `event` message followed by its parameters. Conceptually, this message is equivalent to calling a parametrized function that generates OSC messages as output. This logic is particularly consistent for user events, which can take a variable number of parameters, that are then available to the associated messages under the form of variables named $1...$$n$. Figure 3 shows a 2-arguments user event example.

```
/ITL/scene/obj watch MYEVENT (
  /ITL/scene/t1 set txt $1,
  /ITL/scene/t2 set txt $2
);
/ITL/scene/obj event MYEVENT
            "This text is for t1"
            "This one is for t2";
```

Figure 3. Definition of a user event named `MYEVENT` expecting 2 parameters referenced by $1 and $2. This event is then triggered with 2 strings as arguments.

The temporal dimension of this *event-based* logic allows to place *functions* in the time space, under the form of events activating messages, which can modify the state of the system and/or be addressed to external applications using INScore extended OSC addressing scheme. Figure 4 gives an example.

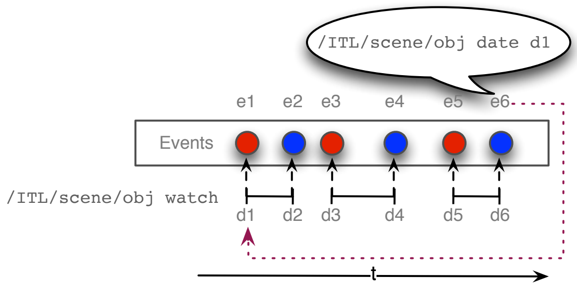

Figure 4. Example of events placed in the time space for an object `obj`. These events are associated with time intervals (`timeEnter` and `timeLeave`) and are triggered when entering (red) or leaving (blue) these intervals. The last event (`e6`), sends a `date` message that creates a loop by positioning the object at the beginning of the first interval.

3.1 Implementation

Each object of a score handles relationships between a set of events $\mathbb{E}$ and a set of messages $\mathbb{M}$. The set of events $\mathbb{E}_o$ of an object o is a polymorphic set defined as:

$$\mathbb{E}_o = \mathbb{A}_o \cup \mathbb{U}_o \cup \mathbb{T}_o \cup \mathbb{T}'_o \cup \mathbb{K}$$

where:

- $\mathbb{A}_o$ is the set of the object attributes,
- $\mathbb{U}_o$ is the set of the object user defined events,
- $\mathbb{T}_o$ and $\mathbb{T}'_o$ are respectively the set of time intervals monitored at input and at output,
- $\mathbb{K}$ is the set of UI events (`mousexxx, touchxxx`).

An event e will be called a trigger when:

$$\exists(e, m) \in \mathbb{E}_o \times \mathbb{M}_o$$

The messages from m are then sent.
Computation of an event e depends on the event type:

- for a message x addressed to an object o, an event will be triggered if $x \in \mathbb{A}_o$,
- for a message `event` x addressed to an object o, an event will be triggered if $x \in \mathbb{U}_o \cup \mathbb{A}_o \cup \mathbb{K}$,
- for a message `date` x addressed to an object o that has the date d, an event will be triggered if:
  $\exists i \in \mathbb{T}_o \mid (x \in i \wedge d \notin i)$
  or $\exists i \in \mathbb{T}'_o \mid (x \notin i \wedge d \in i)$,
- events from $\mathbb{K}$ are computed by the host system.

4. DYNAMIC MUSICAL SCORES

The combination of musical and event time allows to design autonomous dynamic scores. In the following example, 3 objects `o1, o2` and `o3` activate each other (Figure 5) according to the time diagram described in Figure 10. It makes use of the 2 *user* events `START` and `STOP`, that are described in Figure 6.

```
/ITL/scene/o1 watch timeLeave 0 $t2
  ( /ITL/scene/o2 event START );
/ITL/scene/o1 watch timeLeave 0 $d1
  ( /ITL/scene/o1 event STOP );
/ITL/scene/o2 watch timeLeave 0 $d2
  ( /ITL/scene/o2 event STOP,
    /ITL/scene/o3 event START );
/ITL/scene/o3 watch timeLeave 0 $d3
  ( /ITL/scene/o3 event STOP,
    /ITL/scene/o1 event START );
```

Figure 5. Mutual activation of 3 objects `o1, o2, o3` according to a predefined time scheme.

The variables `$t1, $d1, $d2` and `$d3` are instantiated with values corresponding to Figure 10. Note that the object `o3` loops the system by triggering the `START` event of the object `o1`.

```
/ITL/scene/o* watch STOP (
  /ITL/scene/$self tempo 0,
  /ITL/scene/$self alpha $alpha
);

/ITL/scene/o* watch START (
  /ITL/scene/$self tempo 120,
  /ITL/scene/$self alpha 255,
  /ITL/scene/$self date 0
);
```

Figure 6. Definition of the `START` and `STOP` user events.

The `START` and `STOP` events respectively activate and deactivate the receiver object time by changing the value of its tempo and modifying its graphical appearance (alpha channel) to account for the activation of time. The `START` event also sets the date of the object to 0, which allows looping the system by `o3`.

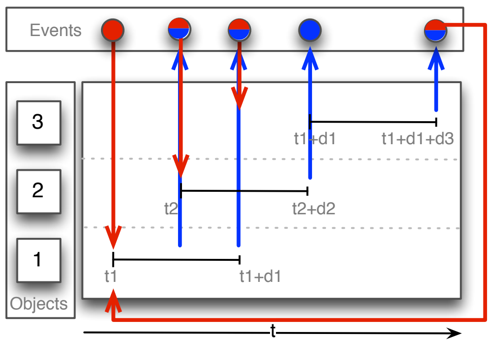

Figure 7. Representation of the time activation for 3 objects of the score. The red color is used for `START` events, blue for `STOP`

4.1 Jeu est un autre

Jeu est un autre from Vincent Carinola is a pedagogical music piece for computer and a variable number of performers, where Max/MSP is used to compute sound processes and to activate a dynamic musical score intended to performers.

```
/ITL/scene watch P1 (
  /ITL/scene/g1 alpha 255,
  /ITL/scene event STATION $tempo '$1'
);

/ITL/scene watch STATION (
  /ITL/scene/cursor show 0,
  /ITL/scene/cursor watch timeLeave
               0 '$2' $stop_s,
  /ITL/scene/cursor date 0,
  /ITL/scene/cursor tempo '$1'
);

stop_s = (
  # set the tempo to 0
  /ITL/scene/cursor tempo 0,
  # remove any watched event
  /ITL/scene/cursor watch,
  # and reset the alpha channel
  /ITL/scene/g* alpha $alpha
);
```

Figure 8. Definition of the event `P1` activating the *station* represented by the object `g1`. Note: the use of the variable `stop_s`, which is evaluated by the parser and allows to share the stop behavior between several events. The `tempo` and `alpha` variables are defined globally.

```
/ITL/scene watch P5 (
  /ITL/scene/a3 duration '$1',
  /ITL/scene event PATH a3 $tempo '$1'
);

/ITL/scene watch PATH (
  /ITL/scene/cursor show 1,
  /ITL/scene/sync cursor '$1' syncFrame,
  /ITL/scene/cursor watch timeLeave
               0 '$3' $stop_c,
  /ITL/scene/cursor date 0,
  /ITL/scene/cursor tempo '$2'
);

stop_c = (
  /ITL/scene/cursor show 0,
  /ITL/scene/cursor tempo 0,
  /ITL/scene/cursor watch,
  # remove the synchronization
  /ITL/scene/sync cursor
);
```

Figure 9. Definition of the event `P5` activating the *path* represented by the object `a3`.

The music score (Figure 10) consists of a set of drawings with labels (sculpture, zebra, pace, texture, turn) connected by *paths*. Each drawing (that we will call *stations*) is associated to an instrumental play mode and to specific sound processes, based on materials prepared by the students. For example, the path *whisper* indicates that *the passage from one drawing to another is done by crossing a path inhabited by whispers. You should find here, in the instrumental and vocal play, something like a story that would be told very softly, whispering.* [1]

To some extents, the computer plays the role of the conductor: it computes sound sequences based on pre-recorded materials and activates *paths* or *stations* for variable durations, dynamically computed. The *paths* and possible *stations* are indicated in the Figure 10 by labels P1 to P13.

The dynamic aspects of the score are defined with user events that carry the name of the corresponding labels. These events take a duration as parameter, so that the following message is enough to activate a *path* or a *station*:

```
      /ITL/scene Px d
```

where `Px` is the user event name (`P1...P13`)
and `d` the corresponding duration.

Thus, it's a high-level approach allowing to *conduct* the score in real time from Max/MSP.

Visually, the score accounts for these events in the graphical space in two ways:

- for *stations* (events OSC P1 ... P4 and OSC P13): using the alpha channel of the corresponding drawing to indicate its activation.
- for *paths* (events OSC P5 ... P12): by moving a cursor along the path, in the corresponding direction.

Schematically, the implementation of these events relies on an object named `cursor` that moves in time at a given tempo, depending on a time unit fixed by convention.

Activation of a *station* consists in changing the alpha channel of the corresponding drawing, activating the cursor time and instantiating an event for the end of the duration computed by Max/MSP (see Figure 8).

For *paths* activation, the duration of the corresponding arc is set to the duration of the event, the cursor is synchronized to that arc in `syncframe` mode (which consists of synchronizing an object on the border of another), its time is activated and an event is instantiated for the end of the arc duration (see Figure 9).

5. CONCLUSIONS

The temporal approach proposed by INScore provides an original support to dynamic musical score design. It allows to express arbitrary behaviors in both musical and event times. However, it remains limited: unlike environments such as Antescofo for example, events do not have a temporal dimension, preventing them from being composed with logical operations. It would also be interesting to extend the temporal dimensions of an object to all its graphic attributes, so that its displacements in time could be translated into the graphical space by the variation of its attributes.

[1] Excerpt from the instructions given with the score.

6. REFERENCES

[1] J. Cage, "Variation I-VI," Henmar Press, New York, 1960-1966.

[2] A. Boucourechliev, *Archipel 1*. Universal, 1967.

[3] K. Stockhausen, "REFRAIN for 3 players," Universal Edition, Vienna, 1959.

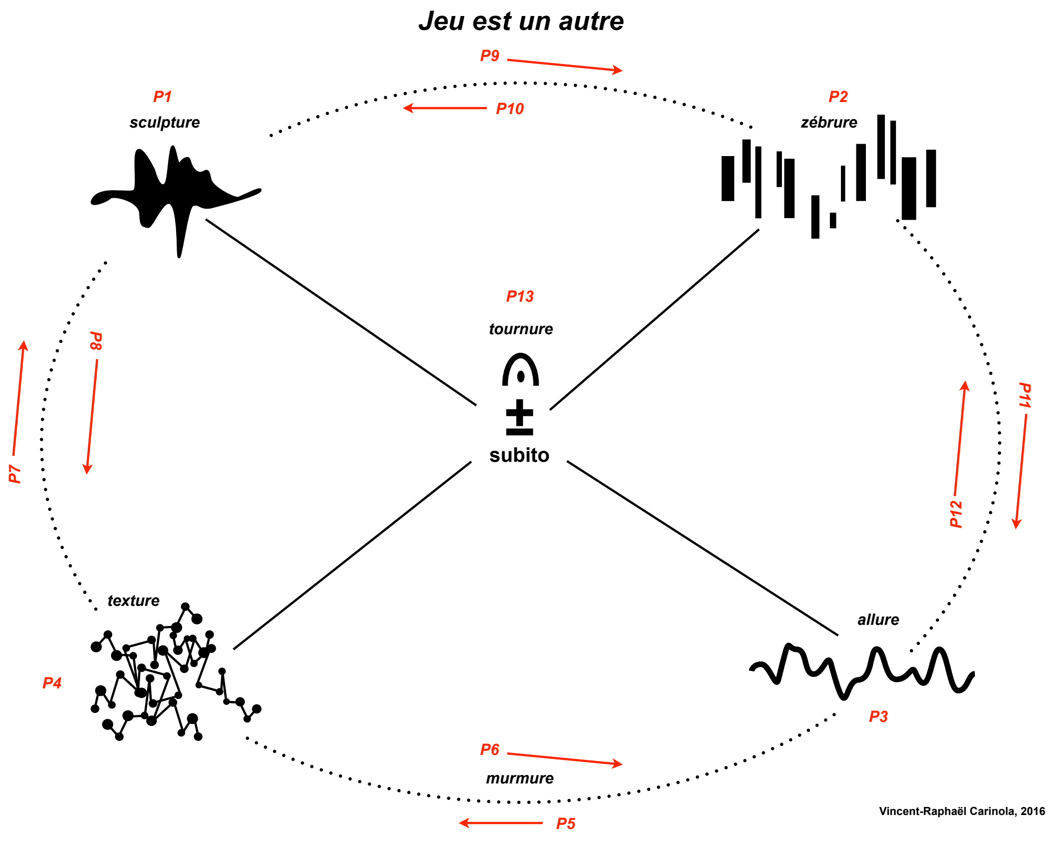

Figure 10. The music score of *"Jeu est un autre"* annotated (in red) with the possible *ways* and *stations*. The annotations are implemented as INScore's user events, they do not appear on the performer score.

[4] J. Bresson and C. Agon, "Scores, Programs, and Time Representation: The Sheet Object in OpenMusic," *Computer Music Journal*, vol. 32, no. 4, pp. 31–47, 2008.

[5] A. Allombert, M. Desainte-Catherine, and G. Assayag, "Iscore: a system for writing interaction," in *Proceedings of the Third International Conference on Digital Interactive Media in Entertainment and Arts, DIMEA 2008, 10-12 September 2008, Athens, Greece*, ser. ACM International Conference Proceeding Series, S. Tsekeridou, A. D. Cheok, K. Giannakis, and J. Karigiannis, Eds., vol. 349. ACM, 2008, pp. 360–367.

[6] J. F. Allen, "Maintaining knowledge about temporal intervals," *Commun. ACM*, vol. 26, pp. 832–843, November 1983. [Online]. Available: http://doi.acm.org/10.1145/182.358434

[7] J. Bresson and J.-L. Giavitto, "A Reactive Extension of the OpenMusic Visual Programming Language," *Journal of Visual Languages and Computing*, vol. 25, no. 4, pp. 363–375, 2014. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00965747

[8] A. Cont, "ANTESCOFO: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music." in *Proceedings of International Computer Music Conference*, ICMA, Ed., 2008.

[9] D. Fober, Y. Orlarey, and S. Letz, "INScore – An Environment for the Design of Live Music Scores," in *Proceedings of the Linux Audio Conference – LAC 2012*, 2012, pp. 47–54.

[10] D. Fober, C. Daudin, S. Letz, and Y. Orlarey, "Time Synchronization in Graphic Domain - A new paradigm for Augmented Music Scores," in *Proceedings of the International Computer Music Conference*, ICMA, Ed., 2010, pp. 458–461.

[11] D. Fober, S. Letz, Y. Orlarey, and F. Bevilacqua, "Programming Interactive Music Scores with INScore," in *Proceedings of the Sound and Music Computing conference – SMC'13*, 2013, pp. 185–190. [Online]. Available: fober-smc2013-final.pdf

[12] T. Coduys and G. Ferry, "IanniX Aesthetical/Symbolic Visualisations For Hypermedia Composition," in *Proceedings of the first Sound and Music Computing Conference SMC'04 - Paris/France*. IRCAM, 2004, pp. p. 105–110.

[13] G. Lepetit-Aimon, D. Fober, Y. Orlarey, and S. Letz, "INScore expressions to compose symbolic scores," in *Proceedings of the International Conference on Technologies for Music Notation and Representation - TENOR2016*, R. Hoadley, C. Nash, and D. Fober, Eds. Cambridge, UK: Anglia Ruskin University, 2016, pp. 137–143.