TIME SYNCHRONIZATION IN GRAPHIC DOMAIN - A NEW PARADIGM FOR AUGMENTED MUSIC SCORES

D. Fober, C. Daudin, Y. Orlarey, S. Letz

Grame

Centre national de création musicale, Lyon, France

{fober,daudin,orlarey,letz}@grame.fr

ABSTRACT

We propose a simple method for synchronization of arbitrary graphic objects, based on their time relations. This method relies on *segmentation* and *mappings* that are *relations* between segmentations. The paper gives a formal description of *segmentations* and *mappings* and presents "Interlude", a framework that implements the proposed method under the form of an *augmented music score viewer*, opening a new space to music notation.

1. INTRODUCTION

Music notation has a long history and evolved through ages. From the ancient neumes to the contemporary music notation, western culture is rich of the many ways explored to represent the music. From symbolic or prescriptive notations to pure graphic representation [6], the music score has always been in constant interaction with the creative and artistic process. However, although the music representations have exploded with the advent of computer music [7, 16, 11], the music score, intended to the performer, didn't evolved in proportion to the new music forms. In particular, there is a significant gap between interactive music and the static way it is generally notated.

Music notation by computer has always been and remains a complex task [2]. Outside the closed commercial world, the most advanced systems, i.e. those trying to go beyond classical western music notation, propose engraving-based solutions [10, 14]. Although it doesn't aim at producing music scores, the ENP approach [13] is the most opened to score extensions due notably to the Lisp language support. But none of the current approaches are suited to dynamic music notation.

Considering that time is a constant and common property of all musical objects, we propose to give a status of music score to any graphic object having required time properties and we name *augmented music score*, the graphic space that provides representation, composition and manipulation of heterogeneous music objects, both in the graphic and time domains. The definition of these time and graphic properties defines a whole class of music scores.

Actually, we aim to consider arbitrary graphic objects (music scores, images, text, signal representation...) as possible scores candidates, and to give them a time position and dimension, to transform them into music score elements, which also implies to make the time relations graphically visible. This concern - to organize the graphic space consistently to the time space - which is at the basis of the classic western music notation, has been stirred through the last century of music notation history [4], but also with composing languages due to the very different nature of algorithmic composition. However, this concern is back in recent research [5], and notably to synchronize various media [1].

Synchronization of heterogeneous medias in the graphic domain raises issues related to non-linearity, non-continuity, non-bijectivity. We propose to approach the problem using *segmentation* and the description of relations between *segments* - we will next use the term *mappings* to refer to these relations.

The paper presents the segmentation and mapping formalism. Next, it introduces "Interlude", a framework that implements *augmented music scores*, making use of the segmentation and mapping formalism. Some examples of applications are finally given.

2. SEGMENTATION AND MAPPINGS

2.1. Definitions

We will first introduce the notions of graphic and time segments. Next we will generalize these concrete definitions in an abstract and generic segment definition.

2.1.1. Time Segment

A *time segment* is an interval $i = [t_0, t_1[$ such as $t_0 \leqslant t_1$. An interval $i = [t_0, t_1[$ is said to be empty when $t_0 = t_1$. We will use the notation $\oslash$ for the empty time segment.

Intersection between time segments is the largest interval defined as:

$$\forall i,\ \forall j,\ i \cap j := i' \mid \forall t \in i'\ t \in i \ \wedge\ t \in j$$

2.1.2. *Graphic Segment*

A *graphic segment* is defined as a set of two intervals $\{g_x, g_y\}$ where g_x is an interval on the x-axis and g_y, on the y-axis.

A graphic segment $g = \{g_x, g_y\}$ is said to be empty when $g_x = \emptyset$ or $g_y = \emptyset$

Intersection $\cap$ between graphic segments is defined as:

$$\forall g = \{g_x, g_y\},\ \forall h = \{h_x, h_y\},\ g \cap h = \{g_x \cap h_x,\ g_y \cap h_y\}$$

2.1.3. *Segment Generalization*

We extend the definitions above to a general definition of a *n*-dimensional segment. A *n*-dimensional segment is a set of *n* intervals $s^n = \{i_1, ..., i_n\}$ where i_j is an interval on the dimension *j*.

A *n*-dimensional segment s^n is said to be empty when $\exists i \in s^n \mid i = \emptyset$

Intersection between segments is defined as the set of their intervals intersection:

$$s_1^n \cap s_2^n := \{i^j\} \mid i^j = i_1^j \cap i_2^j \ \wedge\ 0 \leqslant j < n \tag{1}$$

2.2. Resource Segmentation

A *segment-able* resource *R* is a *n*-dimensional resource defined by a *n*-dimensional segment S^n. The segmentation of a resource *R* is the set of segments $Seg(R) = \{s_1^n, ...s_i^n\}$ such as:

$\forall i, j \in Seg(R)$	$i \cap j = \emptyset$	the segments are disjoints
$\forall i \in Seg(R)$	$i \cap S^n = i$	segments are included in R

2.3. Mapping

A *mapping* is a relation between *segmentations*.

For a mapping $M \subseteq Seg(R_1) \times Seg(R_2)$ we define two functions:

$$M^+(i) = \{i' \in Seg(R_2) \mid (i, i') \in M\} \tag{2}$$

that gives the set of segments from R_2 associated to the segment *i* from R_1.

and the reverse function:

$$M^-(i') = \{i \in Seg(R_1) \mid (i, i') \in M\} \tag{3}$$

that gives the set of segments from R_1 associated to the segment i' from R_2.

2.4. Mappings composition

Mappings composition is quite straightforward.
For a mapping $M_1 \subseteq Seg(R_1) \times Seg(R_2)$
and a mapping $M_2 \subseteq Seg(R_2) \times Seg(R_3)$, then :

$$M_1 \circ M_2 \subseteq Seg(R_1) \times Seg(R_3)$$

3. AUGMENTED MUSIC SCORE

Augmented music scores have been developed in the framework of the Interlude project[1], which objective is to investigate gestural interaction with music content, both in the audio and symbolic domains.

An augmented score is a graphic space that provides representation, composition and manipulation of heterogeneous music objects, both in the graphic and time domains. It supports arbitrary resources like music scores, images, text, vectorial graphics and signals representations (not detailed in this paper). It focuses on the graphic synchronization of its components, according to their time relations.

3.1. The Interlude Library

Augmented score capabilities are provided under the form of a C++ platform independent library. It supports the Guido [12] and the MusicXML formats [9] as music notation input. It makes use to the Guido Engine [8, 15] for the score layout and relies on the Qt framework [3] for the rendering of the other resource types and to ensure platform independence. All of the library features are available as C++ API as well using OSC [17] messages.

3.2. Augmented Score Segmentations and Mappings

All the resources that are part of an augmented score have a graphic dimension and a temporal dimension in common. Thus, they are *segment-able* in the graphic and time spaces. Unless specified otherwise, time is referring to music time (i.e. metronomic time).

In addition, each resource type is *segment-able* in its specific space: audio frames linear space for an audio signal, two dimensional space organized in lines/columns for text, etc.

type	segmentations and mappings required
text	*graphic* ↔ **text** ↔ **time**
score	*graphic* ↔ *wrapped time* ↔ *time*
image	*graphic* ↔ **pixel** ↔ **time**
vectorial graphic	*graphic* ↔ **vectorial** ↔ **time**
signal	*graphic* ↔ **frame** ↔ **time**

Table 1. Segmentations and mappings for each component type

Table 1 lists the segmentations and mappings used by the different component types. Mappings are indicated by arrows (↔). Note that the arrows link segments of different types (the *segment* qualifier is omitted). Segmentations and mappings in *italic* are automatically computed by the system, those in **bold** must be given externally. This typology

[1]Interlude - ANR-08-CORD-010

could be extended to any kind of resource, provided that a mapping exists to go from the graphic space to the time space.

Note that for music scores, an intermediate time segmentation, the *wrapped time*, is necessary to catch repeated sections and jumps (to sign, to coda, etc.).

Composing these mappings is at the basis of the mechanisms to address and synchronize the components both in the graphic and time spaces.

3.3. Time to graphic conversion and synchronization

Let's consider two score components A and B with their corresponding graphic and time segmentations:
$Seg(A_g)$, $Seg(A_t)$, $Seg(B_g)$, $Seg(B_t)$.
In addition, B has an intermediate segmentation $Seg(B_l)$ expressed in the resource local space units (e.g. frames for an audio signal). The mappings
$M_A \subseteq Seg(A_g) \times Seg(A_t)$ and $M_B \subseteq Seg(B_t) \times Seg(B_l)$
give the correspondence between graphic and time space for A and between time and local space for B.

When synchronizing objects and to decide on what position should be used as base position, the Interlude library introduces a master/slave relation between components: a slave is constrained to its master graphic space.

3.3.1. Graphic alignment of time positions

Let's consider that B is A slave and that we want to graphically align B to A at a time t. Let $s = [s_0, s_1[$ be the A segment containing the time t. Then the corresponding graphic segment is:

$$M_A^-(s) = \{g \in Seg(A_g) \mid (g,s) \in M_A\}$$

For each $M_A^-(s)$ segment, B graphic position can be computed by simple linear interpolation i.e.:

$$(x_B, y_B) = (g_{x0} + (g_{x1} - g_{x0}).\delta,\ g_{y0})$$

where g_{x0} and g_{x1} are the graphic segment first and last x coordinates and $\delta = (t - s_0)/(s_1 - s_0)$.

y_B is arbitrary fixed to g_{y0} but it is actually controlled by a synchronization mode (over, above, below).

3.3.2. Segments graphic alignment

The basic principle of segments alignment consists for each of the master graphic segment, to retrieve the corresponding slave segment expressed in the slave local coordinates and to render this slave segment in the space of the master graphic segment. Provided that $Seg(A_t) = Seg(B_t)$, the operation may be viewed as the mapping composition

$$M_A \circ M_B \subseteq Seg(A_g) \times Seg(B_l)$$

Figure 1 gives an example of a bitmap aligned to different score locations.

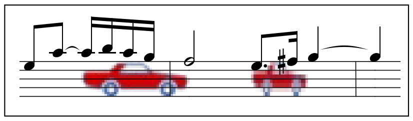

Figure 1. The same car bitmap synchronized to different time positions. The car has a quarter note duration, it is stretched to the corresponding score graphic segments.

3.4. The Interlude viewer

The Interlude viewer is build on top of the Interlude library. It has no user interface since it has been primarily designed to be controlled via OSC messages i.e. using external applications like Max/MSP or Pure Data.

3.4.1. Messages general format

An Interlude OSC message is made of an OSC address, followed by a message string, followed by 0 to n parameters. The message string could be viewed as the method name of the object identified by the OSC address. The OSC address is a string or a regular expression matching several objects. The OSC address space includes predefined static nodes:
`/ITL` corresponds to the Interlude viewer application
`/ITL/scene` corresponds to the rendering scene, actually the augmented score address.

The next section presents an example of OSC messages setting up a score including synchronized components. Note that the messages list corresponds strictly to the file format of a score. Note also that this example is static while realtime interaction is always possible, for example to move objects in time by sending `clock` or `date` messages.

3.4.2. Nested synchronization example

This example uses 3 components; the first one is master of the second, which is master of the third one. Lines beginning with a '#' are comments interleaved with the messages.

```
# creates a score using an image
/ITL/scene/score set img "score.jpg"
# sets the score graphic to time mapping
/ITL/scene/score mapf "score.map"

# creates a text using a text file
/ITL/scene/text set txtf "comment.txt"
# changes the text scale
/ITL/scene/text scale 3.0
# and the text color
/ITL/scene/text color 0 0 240 255
# put the text in front
/ITL/scene/text z 0.5
# and sets the text to time mapping
/ITL/scene/text mapf "comment.map"

# creates a ball as vectorial graphic
```

```
/ITL/scene/ball set ellipse 0.2 0.2
# puts it in front
/ITL/scene/ball z 0.4
# changes the ball color
/ITL/scene/ball color 250 50 0 255

# sets all the objects date
/ITL/scene/* date 4 1
# sets the text slave of the score
/ITL/scene/sync text score
# sets the ball slave of the text
/ITL/scene/sync ball text
```

Note the use of a wildcard in the OSC address to set all the objects date with a single message. The corresponding result is given by figure 2.

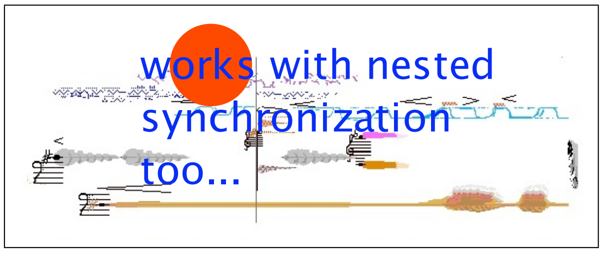

Figure 2. A score with nested synchronization.

4. CONCLUSION

The proposed method for synchronizing arbitrary objects in the graphic space according to their time relations combines the advantages of simplicity and flexibility: a great variety of behaviors may be obtained depending on the defined segmentations and mappings. There are many potential application domains, including pedagogic applications, games, but also new music forms like interactive music.

Acknowledgments
The Interlude project is funded by the Agence Nationale pour la Recherche [ANR-08-CORD-010].

5. REFERENCES

[1] D. Baggi and G. Haus, "Ieee 1599: Music encoding and interaction," *COMPUTER*, vol. 42, no. 3, pp. 84–87, March 2009.

[2] D. A. Bird, "Music notation by computer," Ph.D. dissertation, Indiana University, 1984.

[3] J. Blanchette and M. Summerfield, *C++ GUI Programming with Qt 4 (2nd Edition)*. Prentice Hall, 2008.

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

[5] 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.

[6] E. Brown, *December 1952*. AMP/G.Schirmer, 1952.

[7] R. B. Dannenberg, "Music representation issues, techniques and systems," *Computer Music Journal*, vol. 17, no. 3, pp. 20–30, 1993.

[8] D. Fober, S.Letz, and Y.Orlarey, "Open source tools for music representation and notation," in *Proceedings of the first Sound and Music Computing conference - SMC'04*. IRCAM, 2004, pp. 91–95.

[9] M. Good, "MusicXML for Notation and Analysis." in *The Virtual Score*, W. B. Hewlett and E. Selfridge-Field, Eds. MIT Press, 2001, pp. 113–124.

[10] K. Hamel, "Noteability, a comprehensive music notation system." in *Proceedings of the International Computer Music Conference.*, 1998, pp. 506–509.

[11] W. B. Hewlett and E. Selfridge-Field, Eds., *The Virtual Score; representation, retrieval and restoration*, ser. Computing in Musicology. MIT Press, 2001.

[12] H. Hoos, K. Hamel, K. Renz, and J. Kilian, "The GUIDO Music Notation Format - a Novel Approach for Adequately Representing Score-level Music." in *Proceedings of the International Computer Music Conference*. ICMA, 1998, pp. 451–454.

[13] M. Kuuskankare and M. Laurson, "Expressive notation package," *Computer Music Journal*, vol. 30, no. 4, pp. 67–79, 2006.

[14] H.-W. Nienhuys and J. Nieuwenhuizen, "Lilypond, a system for automated music engraving," in *Proceedings of the XIV Colloquium on Musical Informatics*, 2003.

[15] K. Renz, "Algorithms and data structures for a music notation system based on guido music notation," Ph.D. dissertation, Technischen Universität Darmstadt, 2002.

[16] E. Selfridge-Field, Ed., *Beyond MIDI: the handbook of musical codes*. MIT Press, 1997.

[17] M. Wright, *Open Sound Control 1.0 Specification*, 2002. [Online]. Available: http://opensoundcontrol.org/spec-1_0