Partitions musicales augmentées

Dominique Fober, Christophe Daudin, Stéphane Letz, Yann Orlarey

PARTITIONS MUSICALES AUGMENTÉES

D. Fober, C. Daudin, S. Letz, Y. Orlarey
{fober,daudin,letz,orlarey}@grame.fr
Grame - Centre national de création musicale

RÉSUMÉ

Une *partition musicale augmentée* est une partition mettant en relation un objet musical symbolique avec différentes représentations de son interprétation. La partition musicale est à considérer au sens large, comme un objet graphique permettant de représenter un objet temporel. L'interprétation représente une instance sonore ou gestuelle particulière de la partition. Nous présenterons les fondements théoriques qui sous-tendent la *partition augmentée*, ainsi qu'une application sous forme d'*afficheur* mettant en oeuvre les solutions proposées.

1. INTRODUCTION

La notation musicale s'inscrit dans une longue histoire et a beaucoup évolué à travers les âges. Des neumes aux notations contemporaines, notre culture est riche de toutes les voies explorées pour représenter la musique. Des formes symboliques ou prescriptives de la notation aux représentations purement graphiques [5], la partition musicale reste en constante interaction avec le processus de création artistique.

Cependant, bien que l'on ait assisté à une explosion des représentations avec l'avènement de l'informatique musicale [7, 15, 11], la partition, destinée à l'interprète, n'a pas évolué en proportion des nouvelles formes musicales. En particulier, il y a un fossé important entre les musiques interactives et la manière statique de les représenter. L'interprête ne dispose souvent que d'une partition traditionnelle à laquelle s'ajoute une représentation électronique minimale de l'état du système, sous forme rudimentaire de compteur ou de lettres.

La notation musicale par ordinateur a toujours été et demeure une tâche complexe [2]. En dehors du monde commercial de l'édition de partition, les systèmes les plus aboutis, i.e. ceux qui vont au delà de la notation traditionnelle, proposent des approches de type gravure musicale [10, 14]. Bien que n'ayant pas l'ambition de produire des partitions mises en page, l'approche proposée par ENP [12] est certainement la plus ouverte à l'extension de la notation, notamment grâce au langage Lisp sur lequel s'appuie ENP. Toutefois aucune de ces approches ne permettent la prise en compte d'aspects dynamiques dans la partition.

A partir du constat que le temps est une propriété commune à tous les objets musicaux, nous proposons d'étendre la partition musicale à tout objet graphique possédant des propriétés temporelles. Nous appellerons *partition musicale augmentée*, l'espace permettant de représenter, composer et manipuler des objets musicaux hétérogènes, aussi bien dans le domaine graphique que temporel. La définition des propriétés graphiques et temporelles requises définit dans le même temps toute une classe de partitions musicales.

Nous allons considérer des objets arbitraires (partitions musicales, images, texte, représentation du signal, graphiques vectoriels) comme candidats possibles d'une partition augmentée en leur donnant une position et une dimension temporelles nécessaires à leur statut de partition musicale, ce qui implique également de rendre les relations temporelles visibles.

L'organisation consistante de l'espace graphique par rapport à l'espace temporel est à la base de la notation musicale traditionnelle. Elle a toutefois été bousculée dans les cinquante dernières années [3] et particulièrement avec les languages informatiques pour la composition musicale, en raison notamment de la nature particulière de la composition algorithmique. Cependant, la question de cette consistance reste ouverte, y compris dans les recherches récentes [4] aussi bien que pour satisfaire les besoins de synchronisation de différents médias [1, 13].

La synchronisation dans le domaine graphique de médias hétérogènes soulève des questions de non-linéarité, discontinuité, non-bijectivité. Nous avons abordé ces questions par le biais de la *segmentation* et de la description de relations entre *segments*.

Enfin, la représentation de l'interprétation musicale, soit gestuelle ou sonore, est abordée avec un renversement de perspective : la représentation graphique d'un signal y est vue comme un *signal graphique*, c'est à dire comme un signal composite comportant toute l'information nécessaire à son rendu graphique. Cette approche, qui abstrait le calcul de la représentation, constitue un système ouvert et dynamiquement extensible.

Nous nous attacherons tout d'abord à décrire les principes de base des *mappings* ainsi que le contexte de leur utilisation dans le cadre de la partition augmentée. Nous expliquerons ensuite comment le système traite les signaux de l'interprétation pour en faire des *signaux graphiques*. Enfin nous présenterons l'*afficheur* de partitions augmentées en nous attachant plus particulièrement à son API de contrôle qui est une API de messages OSC[16].

2. RELATIONS TEMPORELLES ET GRAPHIQUES

Nous parlerons de *synchronisation temporelle dans le domaine graphique* pour faire référence à la représentation graphique des relations temporelles entre composants d'une partition. Notre expérience dans ce domaine [6, 9] nous a conduit à aborder la question de ces relations par le biais de la *segmentation* et de la description de relations entre *segments*. Nous utiliserons par la suite le terme *mappings* pour faire référence à ces relations.

Le rôle d'un *mapping* est d'établir des correspondances entre les *segments* de ressources différentes. Un *segment* est une zone contiguë d'une ressource. Ainsi que mentionné en introduction, ces ressources peuvent être de nature arbitraire (images, texte, notation musicale, signaux...). Ces mappings permettent typiquement de lier des positions graphiques, du temps musical et des positions dans des ressources audio.

Un mapping entre un enregistrement audio et une partition musicale permet par exemple d'exprimer une correspondance entre une position audio (exprimée en frames) et un temps musical (exprimé en subdivision de la noire). Un mapping entre une partition musicale et sa représentation graphique permet d'exprimer une correspondance entre une position exprimée en temps musical en une position graphique. La combinaison de ces mappings suffit alors à exprimer des relations entre toutes les ressources ayant une base temporelle.

Les sections qui suivent définissent un cadre général pour les notions de *segment, segmentation* et *mapping*. Ces définitions sont indépendantes de toute implémentation et de toute information spécifique aux différentes ressources. Elles sont suivies de cas concrets d'utilisation, implémentés dans le cadre de l'afficheur de partition augmentée.

2.1. Definitions

Nous allons tout d'abord introduire les notions de segments graphiques et temporels. Puis nous généraliserons ces définitions concrètes en une version abstraite et générique.

2.1.1. Segment temporel

Un *segment temporel* est défini comme un intervalle $i = [t_0, t_1[$ tel que $t_0 \leqslant t_1$.

Un intervalle $i = [t_0, t_1[$ est dit vide quand $t_0 = t_1$. Il sera noté $\oslash$.

L'intersection de segments temporels est telle que communément définie : c'est le plus grand intervalle tel que

$$\forall i_m,\ \forall i_n,\ i_m \cap i_n := \{j \mid j \in i_m\ \wedge\ j \in i_n\}$$

2.1.2. Segment graphique

Un *segment graphique* g est défini comme un rectangle donné par deux intervalles $g = (i_x, i_y)$ où x est un intervalle sur l'axe des abscisses et y, sur l'axe des ordonnées.

Un segment graphique $g = (i_x, i_y)$ est dit vide quand $i_x = \oslash$ ou $i_y = \oslash$

L'opération d'intersection $\cap$ entre segments graphiques est définie telle que :

$$\forall g = (i_x, i_y),\ \forall g' = (i'_x, i'_y),\ g \cap g' = (i_x \cap i'_x, i_y \cap i'_y)$$

2.2. Generalisation de la notion de segment

Nous allons étendre les notions de segment temporel et graphique à une définition plus générale de segment à n dimensions. Un segment de dimension n, noté s^n, est défini comme une liste de n intervalles $s^n = (i_1, ..., i_n)$ où i_j est un intervalle sur la dimension j.

Un segment s^n de dimension n est dit vide quand $\exists i \in s^n \mid i = \oslash$

L'intersection de segments de dimension n est définie comme la liste des intersections de leurs intervalles :

$$s^n_1 \cap s^n_2 = (i_1 \cap j_1, ..., i_n \cap j_n) \tag{1}$$

où $s^n_1 = (i_1, ..., i_n)$ et $s^n_2 = (j_1, ..., j_n)$

2.3. Segmentation d'une ressource

Une ressource R de dimension n est *segmentable* quand elle peut être vue comme un segment S^n de dimension n. La segmentation d'une ressource R est l'ensemble des segments $Seg(R) = \{s^n_1, ...s^n_i\}$ tels que :

$\forall i, j \in Seg(R)$	$i \cap j = \oslash$	les segments sont disjoints
$\forall i \in Seg(R)$	$i \cap S^n = i$	tous les segments sont inclus dans la ressource

2.4. Mapping

Un *mapping* est une relation entre *segmentations*.

Pour un mapping $M \subseteq Seg(R_1) \times Seg(R_2)$ nous définissons deux fonctions :

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

qui donne l'ensemble des segments de R_2 associés au segment i de R_1.

et la fonction inverse :

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

qui donne l'ensemble des segments de R_1 associés au segment i' de R_2.

Ces fonctions sont définies sur un ensemble de segments comme l'union des mappings de chaque segment :

$$M^+(\{i_1, ...i_n\}) = \{M^+(i_1) \cup M^+(i_2)... \cup M^+(i_n)\} \tag{4}$$

et

$$M^-(\{i_1, ...i_n\}) = \{M^-(i_1) \cup M^-(i_2)... \cup M^-(i_n)\} \tag{5}$$

2.5. Composition de mappings

La composition de mappings se fait de manière naturelle. Pour les mappings $M_1 \subseteq Seg(R_1) \times Seg(R_2)$ et $M_2 \subseteq Seg(R_2) \times Seg(R_3)$:

$$(M_1 \circ M_2)^+(i) = M_2^+(M_1^+(i)) \quad (6)$$

De manière similaire :

$$(M_1 \circ M_2)^-(i') = M_1^-(M_2^-(i')) \quad (7)$$

2.6. Segmentations et mappings d'une partition augmentée

Toutes les ressources qui composent la partition augmentée ont en commun une dimension graphique et une dimension temporelle exprimée en temps musical. Elles sont donc toutes *segmentables* dans l'espace graphique et dans l'espace temps musical. Par ailleurs, chaque type de ressource est *segmentable* dans un espace qui lui est propre : espace linéaire des frames audio pour un signal audio, espace à 2 dimensions organisé en lignes/colonnes pour du texte, etc.

La table 2.6 décrit les segmentations et mappings utilisés par les différents types de composants. Les mappings sont indiqués par des flèches ($\leftrightarrow$). Les segmentations et mappings en *italique* sont automatiquement calculés par le système, ceux en **gras** sont fournis de manière externe.

C'est la composition de ces mappings qui permet d'adresser et de synchroniser les composants d'une partition augmentée aussi bien dans l'espace graphique que temporel. La segmentation de l'espace temporel constitue la colonne vertébrale du système : l'espace temporel est commun à tous les composants et permet, par le biais de la composition, de mettre en relation toutes les autres segmentations.

type	segmentations et mappings requis
texte	*graphic* $\leftrightarrow$ **text** $\leftrightarrow$ **temps**
partition	*graphic* $\leftrightarrow$ *temps enroulé* $\leftrightarrow$ *temps*
image	*graphic* $\leftrightarrow$ **pixel** $\leftrightarrow$ **temps**
graph. vectoriel	**graphic** $\leftrightarrow$ **temps**
signal	*graphic* $\leftrightarrow$ **échantillon** $\leftrightarrow$ **temps**

Table 1. Segmentations et mappings pour chaque type de composant.

Pour un élément de type *texte* par exemple, cela revient à établir la composition suivante :

(graphic $\leftrightarrow$ text) $\circ$ (text $\leftrightarrow$ temps)

Le temps fait référence au temps musical métronomique (i.e. relatif à un tempo). La relation "*temps enroulé* $\leftrightarrow$ *temps*" est nécessaire pour traiter les sections d'une partition avec reprises ou avec des sauts (à la coda, au signe...).

2.7. Exemples de synchronisation

Soit deux composants A et B d'une partition, ainsi que leurs segmentations graphiques et temporelles :

$Seg(A_g)$, $Seg(A_t)$, $Seg(B_g)$, $Seg(B_t)$.

De plus, B possède une segmentation intermédiaire $Seg(B_l)$ exprimée dans les coordonnées de son espace local (par exemple : frames pour un signal audio). Les mappings

$M_A \subseteq Seg(A_g) \times Seg(A_t)$

et $M_B \subseteq Seg(B_t) \times Seg(B_l)$

donnent la correspondance entre l'espace graphique et temporel pour A et entre l'espace temporel et local pour B.

Lors de la synchronisation et afin de spécifier quelle position graphique doit être utilisée comme base, nous avons introduit une relation maître/esclave entre composants : un esclave est contraint à la position de son maître.

2.7.1. Alignement graphique de positions temporelles

Considérons que B est l'esclave de A et que nous voulons aligner graphiquement B et A à un temps t. Soit s, le segment de A qui contient la date t, alors le segment graphique correspondant est :

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

Quand $M_A^-(s)$ est constitué d'un seul segment, la position de B peut être calculée par simple interpoation linéaire i.e. :

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

où g_{x0} et g_{x1} sont les coordonnées x de début et de fin du segment graphique et $\delta = (t - s_0)/(s_1 - s_0)$.

y_B est fixé de manière arbitraire à g_{y0}. Dans la pratique, il est contrôlé par le mode de synchronisation (*over, above, below*).

Quand $M_A^-(s)$ contient plusieurs segments, l'opération peut-être répétée pour chacun de ces segments.

2.7.2. Alignement de segments graphiques

Le principe de base de l'alignement par segment consiste, pour chaque segment du composant maître, à prendre le segment correspondant de l'esclave, exprimé dans son espace local, et à faire le rendu de ce segment dans l'espace graphique correspondant du maître.

Si $Seg(A_t) = Seg(B_t)$, l'opération peut être vue comme une composition de mappings :

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

La figure 1 donne l'exemple d'une même image alignée sur une partition à des dates différentes.

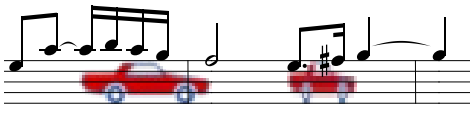

Figure 1. La même image d'une voiture synchronisée à des dates différentes. L'image a la durée d'une noire, elle est étirée sur la longueur du segment graphique de la partition qui lui correspond.

3. REPRÉSENTATION DE L'INTERPRÉTATION

Le travail sur la représentation de l'interprétation s'appuie sur des expériences de visualisation du jeu instrumental réalisées dans un contexte pédagogique [8]. Dans ce cadre, nous avons développé un moteur de rendu graphique prenant des signaux ainsi qu'un type de représentation en entrée pour calculer l'image correspondant au type désiré et aux valeurs des signaux. L'inscription statique des types de représentation supportés dans le moteur de rendu constitue une des limitations importante de cette approche : elle implique une modification du moteur de rendu pour chaque nouveau type de représentation.

Dans le cadre du travail sur la partition augmentée, nous avons souhaité lever cette limitation et définir un système extensible dynamiquement. A cet effet, la représentation graphique d'un signal est vue comme un *signal graphique*, c'est à dire comme un signal composite comportant toute l'information nécessaire à son rendu graphique.

3.1. Des signaux graphiques

Nous définissons un signal graphique comme un signal composite constitué :

- d'un signal y : les coordonnées en y du graphique
- d'un signal h : qui décrit l'épaisseur du graphique à la position y
- d'un signal c : qui décrit la couleur du graphique à la position y

Pour simplifier, nous supposons que l'espace des couleurs décrit par c n'a qu'une dimension. La figure 2 donne un exemple de ces paramètres dans l'espace graphique.

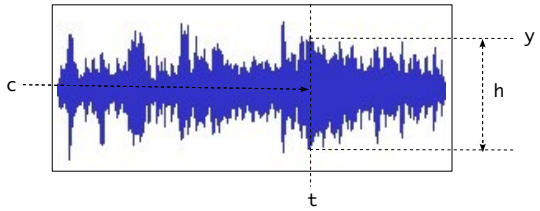

Figure 2. Paramètres graphiques d'un signal.

Considérons maintenant que nous disposons d'un signal S défini comme une fonction du temps :

$$f(t) : \mathbb{R} \rightarrow \mathbb{R}^3 = (y, h, c) \mid y, h, c \in \mathbb{R}$$

alors ce signal contient toute l'information pour être dessiné directement i.e. sans calcul supplémentaire.

Une autre manière de voir revient à considérer le système comme un *oscilloscope* qui prendrait les composantes d'un signal graphique en entrée.

3.2. Composition de signaux

Afin de construire des signaux composites pouvant servir de signaux graphiques, nous avons introduit une opération de parallélisation de signaux.

Soit $\mathbb{S}$, l'ensemble des signaux $s : \mathbb{N} \rightarrow \mathbb{R}$.
Nous définissons l'opération *parallèle* '$/$' comme :

$$s_1/s_2/.../s_n : \mathbb{S} \rightarrow \mathbb{S}^n \mid s_i \in \mathbb{S} \tag{8}$$

La fonction du temps d'un signal parallèle $s^n \in \mathbb{S}^n$ est la mise en parallèle de la fonction du temps de chaque signal :

$$f(t) = (f_0(t), f_1(t), ...f_n(t)) \mid f_i(t) : \mathbb{N} \rightarrow \mathbb{R} \tag{9}$$

3.3. Types de signaux parallèles

Pour les besoins de l'afficheur de partition augmentée, nous avons défini plusieurs types de signaux parallèles :

- le type *signal de couleur* qui utilise le modèle HSBA [hue, saturation, brigthness, transparency] de représentation des couleurs :

$$c ::= \overrightarrow{(h, s, b, a)} \mid h, s, b, a \in \mathbb{R}$$

- le type *signal graphique* qui comporte un signal y, un signal d'épaisseur th suivi des 4 composantes du signal de couleur :

$$g ::= \overrightarrow{(y, th, h, s, b, a)} \mid y, th, h, s, b, a \in \mathbb{R}$$

- en enfin le type *signaux graphiques parallèles* qui permet de composer plusieurs *signaux graphiques* en parallèle :

$$g^n ::= \overrightarrow{g} \mid g \in \mathbb{R}^6$$

3.4. Exemples de signaux graphiques

A des fins de validation du modèle, nous allons décrire plusieurs types de représentations qui étaient implémentées de manière statique dans notre approche précédente.

3.4.1. Représentation de la hauteur de note

Consiste à représenter les hauteurs de note à partir de leur fréquence fondamentale, sur l'axe des y (figure 3).

Figure 3. Graphique des hauteurs de notes.

Le signal graphique correspondant s'exprime de la manière suivante :

$$g = S_{f0} \ / \ k_t \ / \ k_c$$

où S_{f0} : fréquence fondamentale exprimée en 1/2 tons
k_t : signal d'épaisseur constant
k_c : signal de couleur constant

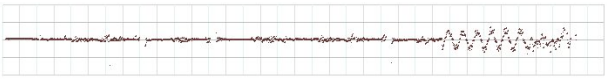

Figure 4. Graphique de justesse.

3.4.2. Représentation de la justesse

Consiste à représenter la différence entre une fréquence fondamentale et une fréquence de référence (figure 4).

Le signal graphique correspondant s'exprime de la manière suivante :

$$g = S_{f0} - S_{fr} \ / \ k_t \ / \ k_c$$

où S_{f0} : fréquence fondamentale exprimée en 1/2 tons
S_{fr} : fréquence de référence exprimée en 1/2 tons
k_t : signal d'épaisseur constant
k_c : signal de couleur constant

3.4.3. Représentation sous forme d'enveloppe

Consiste à utiliser les valeurs RMS d'un signal pour contrôler l'épaisseur du graphique. (figure 5).

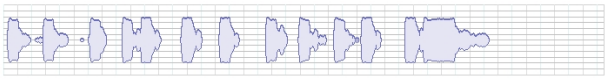

Figure 5. Graphique d'articulations.

Le signal graphique correspondant s'exprime de la manière suivante :

$$g = k_y \ / \ S_{rms} \ / \ k_c$$

où k_y : signal y constant
S_{rms} : signal RMS
k_c : signal de couleur constant

3.4.4. Combinaison d'enveloppe et de hauteur

Consiste à utiliser la fréquence fondamentale et les valeurs RMS d'un signal dessiner un signal d'articulation modulé en fonction de la hauteur de note. (figure 6).

Figure 6. Combinaison de hauteur et d'enveloppe.

Le signal graphique correspondant s'exprime de la manière suivante :

$$g = S_{f0} \ / \ S_{rms} \ / \ k_c$$

où S_{f0} : fréquence fondamentale exprimée en 1/2 tons
S_{rms} : signal RMS
k_c : signal de couleur constant

3.4.5. Combinaison d'harmoniques et de hauteur

Consiste à combiner la fréquence fondamentale et l'intensité des premiers harmoniques. (figure 7). Chaque harmonique est représenté avec une couleur différente.

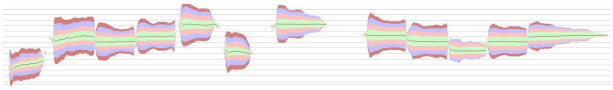

Figure 7. Combinaison de la hauteur et de l'enveloppe des harmoniques.

Le signal graphique correspondant s'exprime en plusieurs étapes. On construit d'abord le graphique de la fréquence fondamentale comme précédemment (voir section 3.4.4) :

$$g0 = S_{f0} \ / \ S_{rms0} \ / \ k_c0$$

où S_{f0} : fréquence fondamentale exprimée en 1/2 tons
S_{rms0} : valeurs RMS du signal f0
k_c0 : signal de couleur constant

Puis on construit le graphique de l'harmonique 1 :

$$g1 = S_{f0} \ / \ S_{rms1} + S_{rms0} \ / \ k_c1$$

S_{rms1} : valeurs RMS de l'harmonique 1
k_c1 : signal de couleur constant

Puis de l'harmonique 2 :

$$g2 = S_{f0} / \ S_{rms2} + S_{rms1} + S_{rms0} \ / \ k_c2$$

S_{rms2} : valeurs RMS de l'harmonique 2
k_c2 : signal de couleur constant

etc.

Pour finalement les combiner en un signal graphique parallèle :

$$g = g2 \ / \ g1 \ / \ g0$$

4. L'AFFICHEUR DE PARTITION AUGMENTÉE

Les travaux sur la partition augmentée ont été réalisés dans le cadre du projet Interlude [1] . Ils ont fait l'objet d'une implémentation sous forme de librairie C++ - la librairie Interlude - ainsi que sous forme d'un afficheur de partitions, construit au dessus de cette librairie. Cet afficheur n'a pas d'interface utilisateur car il a été conçu pour être contrôlé par des messages OSC i.e. par des applications externes (typiquement Max/MSP ou Pure Data).

4.1. Format général des messages

Un message OSC/Interlude est constitué d'une adresse OSC, suivie d'un nom de message, suivi de 0 à n paramètres. Le nom du message peut être vu comme le nom d'une méthode de l'objet identifié par l'adresse OSC. Cette adresse peut comporter des expressions régulières désignant un ensemble de destinataires.

1 . ANR-08-CORD-010

L'espace d'adressage OSC inclut des nodes statiques prédéfinies :

- `/ITL` correspond à l'afficheur Interlude.
- `/ITL/scene` correspond à la scène de rendu graphique, dans les faits, l'adresse de la partition augmentée.

La section qui suit présente un exemple de messages OSC décrivant une partition comportant des éléments synchronisés. La liste des messages correspond strictement au format de fichier d'une partition augmentée. Notez que l'exemple ci-dessous est statique mais qu'une interaction dynamique avec la partition est toujours possible, par exemple en déplaçant des objets dans le temps en leur envoyant des messages `date` ou `clock` (d'une sémantique similaire aux message `clock` MIDI).

4.2. Exemple de synchronisation imbriquée

Cet exemple met 3 composants en oeuvre : le premier est maître du second, qui est maître du troisième. Les lignes qui commencent par '#' sont des commentaires imbriqués avec les messages.

```
# crée une partition basée sur une image
/ITL/scene/score set img "score.jpg"
# décrit son mapping entre les espaces
# temps et graphique
/ITL/scene/score mapf "score.map"

# crée un texte à partir d'un fichier
/ITL/scene/text set txtf "comment.txt"
# change l'échelle du texte
/ITL/scene/text scale 3.0
# ainsi que sa couleur
/ITL/scene/text color 0 0 240 255
# met le texte en 'avant' (z order)
/ITL/scene/text z 0.5
# et décrit le mapping du texte au temps
/ITL/scene/text mapf "comment.map"

# crée un cercle (graphique vectoriel)
/ITL/scene/ball set ellipse 0.2 0.2
# le mets en 'avant'
/ITL/scene/ball z 0.4
# change la couleur du cercle
/ITL/scene/ball color 250 50 0 255

# change la date de tous les objets
# la date est exprimée sous forme
# de rationnel (1 valant une ronde)
/ITL/scene/* date 4 1

# rend le texte esclave de la partition
/ITL/scene/sync text score
# rend le cercle esclave du texte
/ITL/scene/sync ball text
```

Le résultat correspondant est donnée par la figure 8.

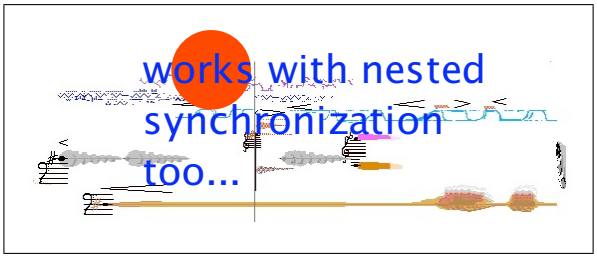

Figure 8. Une partition avec synchronisation imbriquée.

5. CONCLUSION

La méthode proposée pour la synchronisation graphique d'objets arbitraires en fonction de leurs relations temporelles combine les avantages de la simplicité et de la flexibilité : une grande diversité de comportements peut être décrite par la simple définition de segmentations et de mappings. Cette méthode est décrite indépendamment de toute implémentation.

La simplicité et la flexibilité caractérisent également l'approche mise en oeuvre pour intégrer la représentation de l'interprétation au sein de la partition. Elle consiste à abstraire le calcul de la représentation du moteur de rendu, ce qui en fait un système ouvert et dynamiquement extensible.

Il y a de nombreux domaines d'application potentiels au concept de partition augmentée - domaine pédagogique, jeux,... - mais nous souhaitons également que ce travail puisse servir les besoins de la création contemporaine et notamment de ses nouvelles formes telles que les musiques interactives.

Remerciements
Cette recherche a été menée dans le cadre du projet Interlude qui est soutenu par l'Agence Nationale pour la Recherche [ANR-08-CORD-010].

6. 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] A. Boucourechliev, *Archipel 1*. Universal, 1967.

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

[6] Y. Chapuis, D. Fober, S. Letz, Y. Orlarey, and C. Daudin, "Annotation de partitions musicales dynamiques," in *Actes des Journées d'Informatique Musicale JIM'07 - Lyon*, Grame, Ed., 2007, pp. 18–27.

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

[8] C. Daudin, D. Fober, S. Letz, Y. Orlarey, and Y. Chapuis, "Visualisation du jeu instrumental," in *Actes des Journées d'Informatique Musicale JIM'07 - Lyon*, Grame, Ed., 2007, pp. 64–72.

[9] D. Fober, S. Letz, and Y. Orlarey, "Vemus - une école de musique européenne virtuelle," in *Actes des Journées d'Informatique Musicale JIM'07 - Lyon*, Grame, Ed., 2007, pp. 57–63.

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

[13] L. A. Ludovico, "An XML multi-layer framework for music information description," Ph.D. dissertation, Università degli Study de Milano, 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] E. Selfridge-Field, Ed., *Beyond MIDI : the handbook of musical codes*. MIT Press, 1997.

[16] M. Wright, *Open Sound Control 1.0 Specification*, 2002.