LE MODÈLE DU TEMPS DANS INSCORE.

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

RÉSUMÉ

INScore est un environnement pour la conception de partition interactives augmentées, tourné vers des usages non conventionnels de la notation musicale, sans exclure pour autant les approches classiques. Dans cet environnement, bien que tous les objets de la partition aient une dimension temporelle, le temps reste *fixe* i.e. que la date (ou la durée) d'un objet ne change pas, sauf à réception d'un message qui ne peut être produit que de manière externe ou événementielle. INScore n'inclut donc pas de gestionnaire du temps au sens classique du terme. A l'origine, ce choix a été dicté par le fait que le système fut conçu pour des usages couplés avec des logiciels de production sonore (e.g. Max/MSP, Pure Data), qui ont des contraintes de temps-réel plus strictes que l'environnement graphique d'INScore. Toutefois, la nécessité d'introduire un temps dynamique a progressivement émergé, conduisant à un modèle original, à la fois *événementiel* et continu. C'est ce modèle qui est présenté et ses propriétés dans l'environnement d'INScore.

1. INTRODUCTION

Les manipulations du temps dans les environnements pour la composition musicale peuvent s'exprimer de manière très différentes.

Dans OpenMusic, il y a 3 principaux types d'expression du temps : "le temps musical (i.e. le temps de la notation symbolique, relatif à un tempo), le temps proportionnel (exprimé en millisecondes) et le temps continu (défini par des fonctions mathématiques)" [3].

Dans d'autres environnements comme Antescofo [5] ou iScore [2], le temps est également *événementiel* et peut notamment se décrire en terme de relations de Allen [1], ce qui permet d'introduire des aspects réactifs dans les oeuvres. Plus récement, ces aspects réactifs ont été également introduits dans OpenMusic [4], témoignant de leur importance dans les approches compositionnelles contemporaines.

Le temps dans INScore [8] est *fixe* : tous les objets possèdent une date et une durée exprimée en temps musical qui sont fixés dans la description de la partition. La dimension temporelle commune des objets d'une partition permet de représenter leurs relations temporelles dans l'espace graphique [6]. L'*animation* du temps (i.e. la modification des attributs temporel d'un objet) se fait à la réception de messages OSC, nécessairement émis depuis une application externe.

Ce choix initial d'un temps *fixe* a été dicté par une conception orientée vers des usages couplés avec des logiciels de production sonore (e.g. Max/MSP, Pure Data), qui ont des contraintes de temps-réel strictes et de ce fait, offrent des mécanismes de gestion du temps efficaces et précis. Néanmoins, les aspects réactifs ont été rapidement pris en compte avec l'introduction d'*événements* qui permettent d'interagir avec la partition [7]. Initialement, la typologie de ces événements inclut des interactions classiques de type "interface utilisateur" (e.g. interactions avec la souris) et introduit des événements dans le domaine temporel avec notamment la *surveillance* d'intervalles temporels. Ce mécanisme d'*événements* a ouvert la voie à une approche originale de la programmation de partitions en offrant à l'utilisateur, la possibilité de placer des messages - et donc des interactions - dans l'espace temporel. Il n'en demeurait pas moins que le temps étant toujours *fixe*, l'usage d'une application externe restait nécessaire pour l'actionner.

La nécessité d'un temps *dynamique* a de ce fait émergé. Elle a été résolue de manière simple, avec l'introduction d'un attribut supplémentaire aux objets d'une partition : le *tempo*. Ce simple ajout a néanmoins un impact important sur la conception du temps dans INScore et sur les aspects dynamiques des partitions. Nous présenterons tout d'abord le modèle du temps musical, qui est contrôlé par le tempo. Nous verrons ensuite comment ce temps se combine avec le temps événementiel et permet de programmer des partitions dynamiques *autonomes*. Nous montrerons enfin quelques cas d'utilisation avant de conclure et de donner quelques perspectives pour l'extension des dimensions temporelles des objets de la partition.

2. LE MODÈLE DU TEMPS MUSICAL

Du point de vue utilisateur et pour un objet donné, le temps est actif quand son attribut tempo est non nul : une valeur non nulle du tempo a pour effet de déplacer l'objet dans le temps musical, en fonction de son tempo et de l'écoulement du temps absolu.
Nous parlerons de *date* pour faire référence au temps musical d'un objet et de *moment* pour faire référence à un

point dans l'écoulement du temps absolu.

Soit t_0, le moment d'activation du temps d'un objet et v la valeur de son tempo, la date d_t de l'objet à un instant t est donnée par une fonction du temps f telle que :

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

où d_{ti} est la date de l'objet au moment t_i et k une constante permettant de convertir le temps absolu en temps musical. Dans la pratique, le temps absolu est exprimé en millisecondes et l'unité de temps musical est la ronde. De ce fait, la valeur de la constante k est $1/1000 \times 60 \times 4$.

Chaque objet de la partition a un tempo indépendant. La valeur du tempo est signée, ce qui signifie qu'un objet peut se déplacer dans tous les sens du temps (vers le futur ou vers le passé). Il y a peu d'environnements permettent d'exprimer ce renversement du temps de manière simple et naturelle, on peut citer IanniX [9], un séquenceur graphique inspiré des travaux de Iannis Xenakis et que l'on peut voir comme un système d'écriture du temps.

2.1. Implémentation

Dans INScore, la granularité du temps est celle de l'affichage : le système fonctionne de manière asynchrone et une tâche du temps périodique traite les messages entrants et calcule le rendu graphique. Dans la pratique, comme les effets du temps ne sont visibles que dans l'espace graphique, nous considérons qu'il n'est pas nécessaire d'adopter une granularité plus fine.

Par ailleurs, la fréquence de la tâche du temps peut être variable :

- par défaut, elle est effectuée toutes les 10 ms mais l'utilisateur peut changer librement cette valeur en adressant le message `rate` à l'application, suivi d'une valeur en millisecondes.
- cette tâche n'est pas réentrante et la complexité de l'affichage peut potentiellement la ralentir.

De ce fait et à chaque tâche du temps, le système mesure le temps réellement écoulé. Les objets dont le tempo n'est pas nul utilisent alors cette valeur pour calculer un offset en temps musical comme indiqué en (1), puis s'auto-envoient un message `ddate` (déplacement de la date en mode relatif) avec la valeur correspondante.

```
/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. Une séquence de messages qui activent le temps d'un objet `obj`. Les messages préfixés par `->` sont générés par l'objet lui même. r_i représente la valeur du temps écoulé entre la tâche i et $i - 1$.

Pour être consistante avec l'ensemble du système, cette implémentation est entièrement basée sur des messages. De ce fait, elle est compatible avec tous les mécanismes d'INScore, tels que le système de *forwarding* des messages.

3. LE TEMPS ÉVÉNEMENTIEL

L'approche *événementielle* du temps dans INScore a précédé le modèle du temps musical et a été présentée en [7]. Pour rappel, le processus d'interaction événementielle repose sur l'association de messages à des événements du système. Ces messages sont émis lorsque l'événement auquel ils sont associés se produit. Le format général des messages pour créer de telles associations est décrit en figure 2.

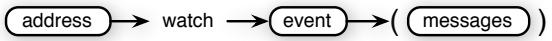

Figure 2. Format général d'un message d'interaction : le message `watch` installe une liste de messages associés à l'événement `event`.

Dans la première version présentée, la typologie des événements était limitée à des événements classiques d'interface utilisateur (tels que clics de souris), étendus dans le domaine temporel (voir table 1).

Domaine graphique	Domaine temporel
mouseDown	timeEnter
mouseUp	timeLeave
mouseEnter	durEnter
mouseLeave	durLeave
mouseMove	

Table 1. Principaux événements du système dans sa version intiale.

Cette typologie a été largement étendue pour inclure :

- n'importe quel attribut d'un objet : la modification de la valeur d'un attribut est potentiellement génératrice de l'événement correspondant, qui porte le nom de l'attribut (e.g. `x, y date`, etc.).
- les données intrinsèques d'un objet i.e. fixées par un message `set`. Il s'agit de l'événement `newData`, introduit pour les besoins de la composition de partitions symboliques [10].
- des événements arbitraires définis par l'utilisateur.

Tout événement peut être arbitrairement déclenché avec l'émission d'un message `event` suivi de ses paramètres. Conceptuellement, ce message `event` est équivalent à l'appel d'une fonction paramétrée qui génère des messages OSC en sortie. Cette logique est particulièrement consistante pour les événements utilisateurs, qui peuvent prendre un nombre variable de paramètres, ceux-ci étant

ensuite accessibles dans les messages associés sous la forme de variables nommées $1...$$n$. La figure 3 présente un exemple d'événement utilisateur à 2 arguments.

```
/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. Définition d'un événement utilisateur nommé `MYEVENT` qui attend 2 paramètres référencés par $1 et $2. Cet événement est ensuite déclenché avec 2 chaines de caractères comme arguments.

La dimension temporelle de cette logique *événementielle* permet de placer des *fonctions* dans l'espace temporel, sous forme d'événements qui activent des listes de messages pouvant modifier l'état du système et/ou s'adresser à des applications externes via l'adressage OSC étendu. La figure 4 en donne un exemple.

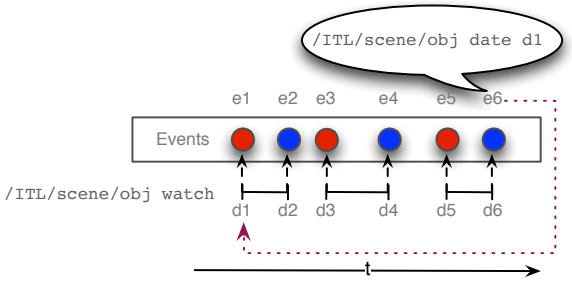

Figure 4. Exemple d'événements placés dans l'espace temporel pour un objet `obj`. Ces événements sont associés à des intervalles de temps (`timeEnter` et `timeLeave`) et sont déclenchés à l'entrée (en rouge) ou à la sortie (en bleu) de ces intervalles. Le dernier événement (`e6`), émet un message `date` qui crée une boucle en renvoyant l'objet au début du premier intervalle.

3.1. Implémentation

Chaque objet de la partition gère des relations entre un ensemble d'événements $\mathbb{E}$ et un ensemble de listes de messages $\mathbb{M}$. L'ensemble des événements $\mathbb{E}_o$ d'un objet o est un ensemble polymorphe défini comme :

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

où :

- $\mathbb{A}_o$ est l'ensemble des attributs de l'objet o,
- $\mathbb{U}_o$ est l'ensemble des événements utilisateur de l'objet o,
- $\mathbb{T}_o$ et $\mathbb{T'}_o$ sont respectivement l'ensemble des intervalles temporels surveillés en entrée et en sortie,
- $\mathbb{K}$ est l'ensemble des événements associés à l'interface utilisateur (`mousexxx, touchxxx`).

L'ensemble $\mathbb{K}$ est indépendant de l'objet o alors que l'ensemble $\mathbb{A}$ dépend de ses attributs et que les ensembles $\mathbb{U}$, $\mathbb{T}$ et $\mathbb{T}'$ sont dynamiques (vides par défaut).

Un événement e sera dit déclencheur quand :

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

Les messages de m sont alors envoyés.

Le calcul d'un événement e est dépendant du type de l'événement :

- pour un message x adressé à un objet o, un événement sera émis si $x \in \mathbb{A}_o$,
- pour un message `event` x adressé à un objet o, un événement sera émis si $x \in \mathbb{U}_o \cup \mathbb{A}_o \cup \mathbb{K}$,
- pour un message `date` x adressé à un objet o qui porte la date d, un événement sera émis si :
  $\exists i \in \mathbb{T}_o \mid (x \in i \ \wedge \ d \notin i)$
  ou $\exists i \in \mathbb{T'}_o \mid (x \notin i \ \wedge \ d \in i)$,
- les événements de $\mathbb{K}$ sont calculés par le système hôte.

4. PARTITIONS DYNAMIQUES

La combinaison du temps musical et événementiel permet de concevoir des partitions dynamiques autonomes. Dans l'exemple suivant 3 objets `o1, o2` et `o3` s'activent mutuellement (figure 5) selon le schéma temporel décrit par la figure 10. Il fait usage des 2 événements *utilisateur* `START` et `STOP`. Ceux-ci sont décrits en 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. Activation mutuelle de 3 objets `o1, o2, o3` selon un schéma temporel prédéfini.

Les variables `$t1, $d1, $d2` et `$d3` sont instanciées avec des valeurs correspondant à la figure 10. On notera que l'objet `o3` boucle l'ensemble du système en activant l'événement `START` de l'objet `o1`.

Les événements `START` et `STOP` activent et désactivent respectivement le temps de l'objet récepteur en changeant la valeur de son tempo et en modifiant son aspect graphique (canal alpha) pour rendre compte de l'activation du temps. L'événement `START` positionne par ailleurs la date de l'objet à 0, ce qui permet le bouclage du système par `o3`.

```
/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. Définition des événements utilisateur `START` et `STOP`.

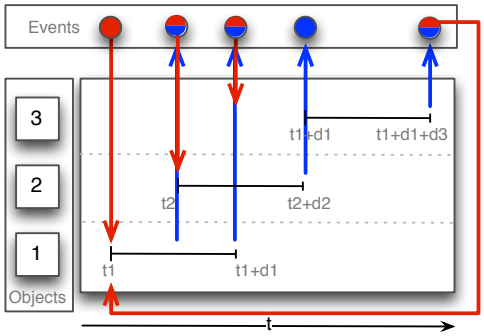

Figure 7. Représentation de l'activation du temps pour 3 objets de la partition. La couleur rouge est utilisée pour les événements `START`, le bleu pour les événements `STOP`.

4.1. Jeu est un autre

Jeu est un autre est une pièce pédagogique pour ordinateur et effectif variable de Vincent Carinola, où Max/MSP est mis en oeuvre pour calculer des processus sonores et activer une partition dynamique à l'intention des instrumentistes.

La partition (figure 10) comprend un ensemble de figures munies de labels (sculpture, zébrure, allure, texture, tournure), reliées par des chemins. A chaque figure (que nous appelerons *stations*) est associée un mode de jeu instrumental ainsi des processus sonores spécifiques, basés sur des matériaux préparés par les élèves. Par exemple, le chemin *murmure* indique que *le passage d'une figure à une autre se fait en traversant un chemin habité par des murmures. Il faut ici trouver, dans le jeu instrumental et vocal, ce qui s'apparenterait le plus à une histoire que l'on raconterait très doucement, en chuchotant.* [1] .

Dans cette pièce, l'ordinateur joue en quelque sorte le rôle de chef d'orchestre : il calcule des séquences sonores basées sur les matériaux pré-enregistrés et active des chemins ou des stations pour des durées variables, calculées dynamiquement. Les chemins et stations possibles sont indiqués sur la figure 10 par les labels P1 à P13.

Les aspects dynamiques de la partition sont définis avec des événements utilisateurs qui portent le nom des labels correspondants. Ces événements prennent une durée en paramètre, de sorte que pour activer un chemin ou une station, il suffit d'émettre le message :

`/ITL/scene Px d`

où `Px` est le nom de l'événement (`P1...P13`)
et `d` la durée correspondante.
C'est donc une approche de haut niveau qui permet de *conduire* la partition en temps réel depuis Max/MSP.

Visuellement, la partition rend compte de ces événements dans l'espace graphique de deux manières :

- pour les stations (événements `P1...P4` et `P13`) : en utilisant le canal alpha de la figure correspondante pour indiquer son activation.
- pour les chemins (événements `P5...P12`) : en déplaçant un curseur le long du chemin, dans le sens correspondant.

L'implémentation des événements utilisateur repose sur un objet nommé `cursor` qui se déplace dans le temps à un tempo fixé de manière conventionnelle.

Pour les stations, elle consiste à changer le canal alpha de la figure correspondant, à activer le temps du curseur et à instancier un événement pour la fin de la durée calculée par Max/MSP (figure 8).

```
/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. Définition de l'événement `P1` qui active la station représentée par l'objet `g1`. A noter : l'utilisation de la variable `stop_s`, qui est évaluée par le parser et qui permet de partager le comportement d'arrêt entre plusieurs événements. Les variables `tempo` et `alpha` sont définies de manière globale.

Pour les chemins, la durée de l'arc correspondant est fixée à la durée de l'événement, le curseur est synchronisé sur cet arc en mode `syncframe` (qui consiste à synchroniser un objet sur la bordure d'un autre), son temps est activé et un événement est instancié pour la fin de la durée de l'arc (figure 9).

1 . extrait des instructions données avec la partition.

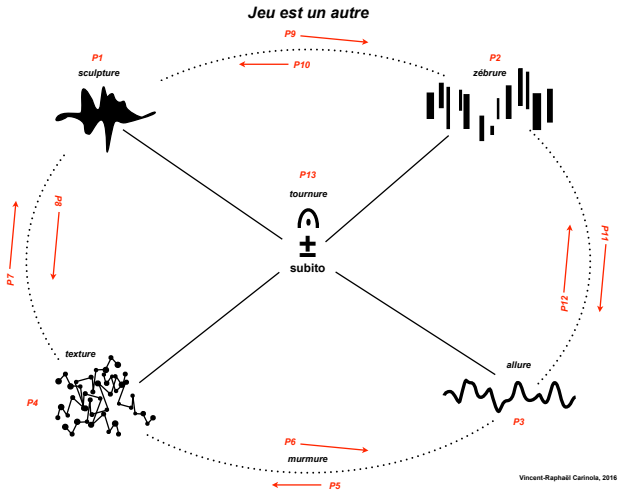

Figure 10. Partition de *"Jeu est un autre"* annotée (en rouge) avec les chemins et les stations possibles. Ces annotations sont implémentées comme des événements utilisateur d'INScore, elles ne figurent pas sur la partition de l'instrumentiste.

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

/ITL/scene watch CHEMIN (
  /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. Définition de l'événement `P5` qui active le chemin représenté par l'objet `a3`.

5. CONCLUSIONS

L'approche temporelle proposée par INScore fournit un support original à la conception de partition dynamiques. Elle permet d'exprimer des comportements arbitraires à la fois dans le temps musical et événementiel. Elle reste toutefois limitée : à la différence d'environnements comme Antescofo par exemple, les événements n'ont pas de dimension temporelle ce qui interdit de les composer avec des opérations d'ordre logique. Il serait également intéressant d'étendre les dimensions temporelles d'un objet à tous ses attributs graphiques, de sorte que ses déplacements dans le temps puissent se traduire dans l'espace graphique par la variation de ses attributs.

6. REFERENCES

[1] James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26 :832–843, November 1983.

[2] Antoine Allombert, Myriam Desainte-Catherine, and Gérard Assayag. Iscore : a system for writing interaction. In Sofia Tsekeridou, Adrian David Cheok, Konstantinos Giannakis, and John Karigiannis, editors, *Proceedings of the Third International Conference on Digital Interactive Media in Entertainment and Arts, DIMEA 2008, 10-12 September 2008, Athens, Greece*, volume 349 of *ACM International Conference Proceeding Series*, pages 360–367. ACM, 2008.

[3] Jean Bresson and Carlos Agon. Scores, programs, and time representation : The sheet object in openmusic. *Computer Music Journal*, 32(4) :31–47, 2008.

[4] Jean Bresson and Jean-Louis Giavitto. A Reactive Extension of the OpenMusic Visual Programming Language. *Journal of Visual Languages and Computing*, 25(4) :363–375, 2014.

[5] Arshia Cont. Antescofo : Anticipatory synchronization and control of interactive parameters in computer music. In ICMA, editor, *Proceedings of International Computer Music Conference*, 2008.

[6] D. Fober, C. Daudin, S. Letz, and Y. Orlarey. Time synchronization in graphic domain - a new paradigm for augmented music scores. In ICMA, editor, *Proceedings of the International Computer Music Conference*, pages 458–461, 2010.

[7] Dominique Fober, Stéphane Letz, and Yann Orlarey. Programmation événementielle de partitions musicales interactives. In *Actes des Journées d'Informatique Musicale JIM2013, Paris*, 2013.

[8] Dominique Fober, Yann Orlarey, and Stephane Letz. Inscore – an environment for the design of live music scores. In *Proceedings of the Linux Audio Conference – LAC 2012*, pages 47–54, 2012.

[9] Guillaume Jacquemin and Thierry Coduys. Partitions rétroactives avec iannix. In *Actes des Journées d'Informatique Musicale JIM2014, Bourges*, pages 35–41, 2014.

[10] Gabriel Lepetit-Aimon, Dominique Fober, Yann Orlarey, and Stéphane Letz. Composition de partitions symboliques dans inscore. In *Actes des Journées d'Informatique Musicale JIM'16 - Albi*, pages 54–60, 2016.