Numéros / n° 5 - Informatique et musique : Recherche et Création 1

« Figures spatiales composées avec séquences finies »

Mario Lorenzo

Résumé

À travers quelques exemples composés avec *SuperCollider*, nous exposerons dans ce texte une approche personnelle quant aux avancées théoriques sur la décorrélation microtemporelle faites par Horacio Vaggione. Affranchie du cadre de la modélisation acoustique, la définition des intervalles de temps, ainsi que la composition des originaux et des répliques avec patterns, en tant que séquences d'événements finies, nous permet de réfléchir à une méthode de travail avec laquelle nous mettons en relation des figures à échelles multiples en suivant une direction musicale.

Introduction

Ce texte est consacré à l'exposé de quelques exemples de composition écrites dans l'environnement *SuperCollider* [(1)] (SC). Ces derniers nous permettront d'illustrer la manière dont nous faisons notre une partie de la théorie exposée par Horacio Vaggione dans l'article « Décorrélation microtemporelle, morphologies et figurations spatiales [(2)] », en particulier, la façon dont le couple « original-réplique » se dissout, peu à peu, pour suivre une direction musicale à travers la composition d'une multiplicité de figures de tailles diverses [(3)].

Dans une perspective musicale (plutôt que de programmation [(4)]), l'objectif ici est de participer à une réflexion menée au sein du CICM cherchant à donner une orientation musicale aux techniques de décorrélation microtemporelle [(5)]. La composition des figures spatiales en tant que séquences finies stockées comme variables d'environnement se dégage, pour nous, comme une piste complémentaire à approfondir, car elle permet de penser avec précision (à travers un langage alphanumérique) la création d'un réseau sonore à plusieurs échelles spatio-temporelles.

Décorrélation microtemporelle « arbitraire »

Dans le domaine de la composition musicale, la décorrélation micro-temporelle (DMT) constitue un moyen de définir les attributs sonores d'ordre spatial. Elle s'affranchit des systèmes de spatialisation globale (qu'elle prend comme cadre de référence), dans le but de « cre?er une directionnalite? souhaite?e pour des raisons musicales » [(6)]. Elle est constituée d'un ensemble de techniques dans lesquelles on travaille les pistes séparées par des intervalles micro-temporels différents. Dans un contexte polyphonique, ces intervalles ne sont pas basés sur des données d'analyse spectrale concernant les valeurs de phase en vue d'une « reproduction "re?aliste" du contenu spatial de sons pre?existants », mais distribués arbitrairement [(7)].

Patterns

Nous nous servons abondamment de la classe *Pattern* en tant que séquence finie, associée à une variable d'environnement. Il n'est pas inutile d'en dire quelques mots. Au sens large, un *pattern* [8] permet de générer, d'après un modèle, une séquence d'objets, généralement des valeurs [9]. On fait appel au pattern pour contrôler les variables d'un ou plusieurs synthétiseurs préalablement définis et envoyés dans le serveur. *Pbind* est le pattern de base qui permet de lier plusieurs séquences de valeurs en une seule séquence d'événements (*Event Patterns*). Il est constitué d'une série de paires « clé, valeur ». Dans l'illustration 1, la variable p se réfère à un *Pbind* qui inclut trois autres patterns : *Pseq*, qui envoie une séquence fixe de valeurs à la clé dur [10] (ligne 4) ; *Prand*, envoie les valeurs de la liste à la fréquence, avec une fonction aléatoire (ligne 5) ; de même que *Pwhite* dont les valeurs envoyées à l'amplitude se situent entre 0.01 et 0.6. (ligne 6).

Illustration . Un *Pbind* qui se réfère à l'instrument « un synth » et à ses paramètres avec trois autres patterns générant une séquence d'événements

```
1 (
2 p = Pbind (
3     \instrument, \un_synth,
4     \dur, Pseq ([0.06, 0.17, 0.36, 0.31,0.7], inf),
5     \freq, Prand ([497, 502, 505, 499], inf),
6     \amp, Pwhite (0.01,0.6, inf)
7 )
8 )
9 p.play;
```

Séquence finie

À la ligne 9 du code présenté ci-dessus, on associe la variable (p) à la méthode *play*. Quand on l'exécute, on obtient une séquence infinie (*infinite streams*) d'événements, car nulle part nous n'avons déterminé un arrêt. En effet, tous les patterns de l'illustration 1 prennent comme deuxième argument la valeur *inf* (infini). Pour l'arrêter, on doit utiliser la méthode *stop*. (p.stop). Si on veut écrire une séquence finie, il faut le préciser. Pour cela, la plupart des patterns ont en commun une variable de répétitions (*repeats variable*). Dans l'illustration 2 nous avons une séquence qui s'arrête au bout de 6 événements (3 valeurs * 2), car le septième message *next* (ligne 2) retourne la valeur *nil*.

Illustration . Une séquence finie dans laquelle on répète deux fois la liste

```
1 z = Pseq ([1, 2, 3], 2).asStream;
2 z.nextN (7);//-> [ 1, 2, 3, 1, 2, 3, nil ]
```

Lorsqu'il y a plusieurs séquences dans un *Pbind*, si l'une est finie, les autres s'arrêtent avec elle, même si celles-là sont définies comme infinies. Dans l'illustration suivante *Pseq* détermine le nombre d'événements de toutes les autres séquences, car, une fois les cinq valeurs passées, le pattern s'arrête. La dernière valeur de la séquence liée à la fréquence est 497 (ligne 5).

Illustration . Séquence d'événements finie

```
1 (
2 p = Pbind (
3     \instrument, \un_synth,
4     \dur, Pseq ([0.06, 0.17, 0.36, 0.31,0.7], 1),
5     \freq, Prand ([497, 502, 505, 499], inf),
6     \amp, Pwhite (0.01,0.6, inf)
7 ).play;
8 )
```

Si nous soulevons cette particularité des patterns c'est parce que, comme nous le verrons avec les exemples, la détermination d'un arrêt est, selon nous, fondamentale C'est avec des séquences finies qu'on peut composer une figure et avec elle, une multiplicité spatiale et temporelle (11). En termes informatiques, nous clôturons un objet, permettant sa circulation dans un réseau (12). Notons d'ailleurs que l'ensemble du code est entre deux parenthèses, ce qui permet son évaluation, indépendamment d'autres parties, facilitant ainsi le travail contextuel et le cycle « écriture-réécriture » propre au processus de composition (13).

Variable d'environnement

Nous associons le pattern fini à un nom. Plus précisément, nous le définissons comme une « variable d'environnement » (*Environment variable*) (14), car cette dernière permet d'ajouter un contexte particulier à l'intérieur de l'environnement en cours (*current Environment*). Il s'agit d'une variable dans la mesure où elle peut être invoquée dans n'importe quel autre contexte, à condition que l'environnement global ne soit pas redéfini (15). Dans l'illustration 4 il y a deux variables d'environnement, dont la deuxième inclut la première.

Illustration . Deux séquences déclarées comme variables d'environnement

```
~mon_env_01 = Pbind (\freq, Pwhite (30, 35), \dur, Pxrand ([0.1, 0.7, 0.2, Rest (1)], 4));
~mon_env_02 = Pn (~mon_env_01, 5);
```

En l'associant à un pattern de référence (*Pdef*), nous allons nous en servir abondamment pour composer les figures et les faire évoluer à l'intérieur d'autres figures, créant ainsi un réseau à échelles spatio-temporelles multiples.

Composition de figures spatiales : l'« original »

Revenons maintenant à la DMT. Nous avons vu que la première condition de la décorrélation microtemporelle est de jouer entre les originaux et leurs répliques. Pour cela, il faut donc pouvoir déjà disposer d'un « original ». Voici les trois parties principales du code qui constituent notre original (le *Buffer*, le *SythDef* et le *Pattern*) :

1 - Nous écrivons deux variables globales « b » et « y » correspondant, respectivement, à un fichier son (16) et à une enveloppe de type percussion, que nous plaçons dans le *buffer*.

Illustration . Définition des deux variables (b, y) avec la classe Buffer

```
(
b = Buffer.read(s, Platform.resourceDir +/+ "Sounds" +/+ "68-v05.wav");
y = Buffer.sendCollection(s, Env.perc.discretize, 1);
)
```

2 - Ensuite nous définissons un synthétiseur (illustration 8). Il s'agit d'un moteur granulaire qui nous permettra de traiter le fichier son stocké dans la mémoire. Il a deux variables « src » (source) et « window » (fenêtrage). La source est composée principalement de l'Ugen *PlayBuf* et la génération d'une fenêtre avec *BufRd* et *EnvGen* (17). Il prend huit arguments (out, pos, buf, windowbuf, graindur, rate, pan, amp) et un nom (« bufgrain_01 ») qui recevront les messages des patterns.

Illustration . L'instrument « grainbuf_01 » dans le serveur auquel nous enverrons les valeurs à travers les patterns

```
(
SynthDef(\grainbuf_01, {

    arg out = 0, pos = 0, buf = 0, windowbuf = 1, graindur = 0.2,
    rate = 1, loop = 1, pan = 0, amp = 1;
    var window, src;

    src = PlayBuf.ar(1, buf, BufRateScale.kr(buf) * rate,1, round(pos *
BufFrames.kr(buf)), loop, 2);

    window = BufRd.ar(1, windowbuf,EnvGen.ar(Env([0, BufFrames.kr(windowbuf)],
[graindur]),doneAction: 2), loop, 4);

    OffsetOut.ar(out, Pan2.ar(src, pan, amp) * window);
}).add;
)
```

Il est important de remarquer que la sortie du signal se fait à travers l'objet *OffsetOut* [(18)], qui associé à la méthode *ar* prend comme deuxième argument une liste de canaux. Dans notre cas, nous utilisons *Pan2* qui permet d'envoyer le signal sur deux canaux [(19)]. Comme nous le verrons à travers les exemples, en suivant les conditions de la DMT, nous allons travailler sur une grande quantité de paires stéréophoniques, ce qui nous éloigne de la simple technique de *panning*.

3 - Et finalement, le pattern de l'original proprement dit.

Illustration . Définition de l'environnement « original » avec les valeurs et les patterns

```
(
~original = Pdef (\orig,
    Pbind (
        \instrument, \grainbuf_01,
        \graindur, 0.05,
        \dur, 0.01,
        \windowbuf, y,
        \buf, b,
        \amp, 0.9,
        \pos, Pseq(Pseq ([0.0, 0.8, 0.7], inf),Pseq ([0.15, 5,0.4],1), \lin),
        \rate, Pwhite (3.98, 4.02, inf)
    )
).play;
)
```

Il s'agit d'une variable d'environnement (~original) que nous utiliserons pour répliquer [(20)]. Elle est associée à un pattern de référence (*Pdef*) qui permet réévaluer la séquence, indépendamment, dans d'autres contextes. Il prend comme argument un nom et un pattern, généralement *Pbind* pour pouvoir envoyer la séquence d'événements au serveur.

Il est important de noter que l'objet « original » est déjà, à part entière, une figure, dans ce sens qu'il est une séquence finie, portant une forme composée, et avec elle, une direction musicale. L'élément le plus caractéristique de la composition de cet objet, relativement simple, est le parcours sur la position de lecture du fichier son (ligne 10) avec le pattern *Pseg*, qui permet, justement, de passer d'une valeur à une autre, fixée dans une première séquence, dans un temps défini avec un deuxième pattern (le deuxième *Pseq*). C'est d'ailleurs à travers celui-ci, en tant que pattern fini, que nous déterminons la durée globale de l'original. La durée totale est de 5,55 secondes. La valeur 1 détermine que la liste des durées va être envoyée au serveur une seule fois. Cela arrête la machine et nous permet de réfléchir à la suite.

Original + répliques

Nous composons après une première figure spatiale (~figure_01) avec deux répliques séparées par un intervalle de 15 millisecondes.

Illustration . Première figure spatiale

```
1 (
2 ~figure_01 = Pdef (\fig_01,
3     Ptpar ([
4         0.0, ~replique_01 = Pdef (\rep_01, Pbindf (~original, \pan, 1)),
5         0.015, ~replique_02 = Pdef (\rep_02, Pbindf (~replique_01, \pan, -1))
6     ], 1)
7 ).play;
8 )
```

Nous avons utilisé la sous-classe *Ptpar* (ligne 3), car elle remplit la condition principale de la DMT, à savoir définir un intervalle microtemporel entre l'original et sa réplique. En effet, il prend comme premier argument une liste de temps et de patterns en parallèle (time, pattern, time, pattern, etc.), permettant ainsi le décalage. Il est important de remarquer qu'il s'agit d'un décalage global, c'est-à-dire, qui ne change pas pendant la lecture. Si on travaille figure par figure, l'*offset* étant toujours différent dans chacune d'entre elles, nous pouvons contrôler la variété temporelle (*time-varying*), en évitant ainsi un effet global venant affaiblir la morphologie de l'original.

Comme nous pouvons le voir, toujours dans l'illustration 8, chacune des répliques est composée avec la classe *Pbidnf* qui nous permet d'appeler l'original en changeant seulement pour celui-ci, dans cet exemple, la valeur de l'argument pan, définie dans le *SynthDef* (0), par 1 et -1 respectivement (ligne 4 et 5). Notons que la réplique 02 est déjà une copie de la réplique 01 et non de l'original. Dorénavant, toutes les répliques vont être une copie d'autres répliques. On peut noter, par ailleurs, que nous avons également défini deux nouvelles variables d'environnement (~réplique_01, et ~réplique_02). Cela va dans le sens de notre approche compositionnelle, car bien que les répliques soient interdépendantes vis-à-vis de l'attribut spatial du son, rien ne nous empêche d'y faire appel plus tard individuellement, en tant que porteuses d'un trait singulier, à l'intérieur d'autres figures, d'autres contextes. C'est toute l'utilité de nos variables, car elles nous permettent de rappeler l'ensemble des valeurs, modifiées ou pas, seulement avec son nom au moment où on le veut.

La figure 02 de l'illustration suivante est composée comme la figure 01, avec un changement de canal et d'intervalle du temps.

Illustration . Figure spatiale 02

```
1 (
2 ~figure_02 = Pdef (\fig_02,
3     Ptpar ([
4         0.0, ~replique_02,
5         0.012, ~replique_01
6     ], 1)
7 ).play;
8 )
```

Dans la figure suivante (illustration 10), la valeur du début de chaque réplique reçoit la méthode *rand*. À chaque fois que cet environnement sera utilisé, l'intervalle de temps entre les répliques sera différent, à l'intérieur d'une échelle de 40 millisecondes. (voir figure 05, illustration 12).

Illustration . Méthode « rand » envoyée aux valeurs de l'offset

```
1 (
2 ~figure_03 = Pdef (\fig_03,
3     Ptpar ([
4         {0.04.rand}, ~replique_01,
5         {0.04.rand}, ~replique_02
6     ], 1)
7 ).play;
8 )
```

Par la suite, nous composons une nouvelle figure avec les figures précédentes, en faisant appel à une variante de *Ptpar*, *Ppar*, qui permet de faire jouer des patterns en parallèle sans la spécification du temps, ce qui n'est pas nécessaire, car chaque figure en a déjà une.

Illustration . Première figure des figures avec Ppar

```
(
~figure_04 = Pdef (\fig_04,
    Ppar ([~figure_01,~figure_02,~figure_03
    ], 1)
).play;
)
```

Jusqu'ici, nous avons travaillé « manuellement », réplique par réplique. Une autre possibilité est d'utiliser une fonction pour reproduire automatiquement le nombre de répliques que l'on veut. Pour cela, nous faisons appel au pattern *Pspawn*. Dans l'exemple suivant, la figure 03 est à l'intérieur d'un pattern fonction (*Pfunc*), qui permet la « reproduction ». Dans la clé delta (ligne 12), on détermine le temps d'attente et le nombre de chaque réplique. Puisque nous avons déjà défini le décalage temporel de la figure 03 (avec une composante aléatoire), nous spécifions, avec le pattern *Pn*, seulement le nombre de copies (8).

Illustration . Multiplication automatique des répliques avec Pspawn

```
(
~figure_05 = Pdef (\fig_05,
    Pspawn (
        Pbind(
            \method,\par,
            \pattern, Pfunc {
                Pbindf (~figure_03,
                    \amp, 0.15
                )

            },
            \delta, Pn (0, 8)),
)
).play;
)
```

Pspawn fait partie, sans doute, des abstractions très puissantes dans SC, mais son utilisation, comme tous les automatismes, demande un certain dosage. Les « séquences enfants » (*child events*) engendrées ne sont pas indépendantes. Ce sont des copies de la « séquence mère » (*parent events*) qu'on peut reproduire autant qu'on veut et placer à des distances voulues, mais on ne peut pas, comme nous le faisons dans les autres exemples, les modifier individuellement et les utiliser dans d'autres contextes.

Variations d'autres attributs

Étant donné que nous ne sommes pas, depuis le début, dans le cadre de la modélisation acoustique, mais dans celui d'un processus de composition, nous pouvons, toujours « arbitrairement », non seulement ajouter des répliques à des intervalles temporels voulus, mais aussi les faire varier autant que l'on veut. Nous continuons donc la direction musicale prise déjà en composant l'original. L'illustration 13 montre la figure 06, composée avec la figure 01 et deux nouvelles répliques, la figure 03, dans laquelle il y a un changement dynamique (linaire) de vitesse de lecture (ligne 6), et la réplique 04 qui en est la copie, mais avec une décorrélation de 15 millisecondes et placée dans l'autre canal.

Illustration . Nouvelles figures avec deux nouvelles répliques (03 et 04)

```
(
~figure_06 = Pdef (\fig_06,
    Ptpar ([
        0.0, ~figure_01,
        0.035,~replique_03 = Pdef (\rep_03, Pbindf (~replique_02,
            \rate, ~a = Pseg (Pseq ([4.02, 3.1], inf), Pwhite (4.5,5.5), \lin))),
        0.05, ~replique_04 = Pdef (\rep_04, Pbindf (~replique_03,
            \pan, 1))
    ], 1)
).play;
)
```

Dans la figure 07, j'introduis des patterns aléatoires (*Pexprand* et *Pwrand*) sur les arguments de la durée entre les grains, de la durée de chaque grain et de l'amplitude.

Illustration . Premières variations avec des patterns aléatoires

```
(
~figure_07 = Pdef (\fig_07,
    Ptpar ([
        {0.06.rand}, ~replique_05 = Pdef (\rep_05,
            Pbindf (~replique_01,
                \graindur, Pwrand ([0.011,0.08], [12, 1].normalizeSum, inf),
                \dur, Pexprand (0.01,0.015),
                \amp, Pwrand ([0.46,0.8], [9, 1].normalizeSum, inf)*1.5)),
        {0.06.rand}, ~replique_06 = Pdef (\rep_06,
            Pbindf (~replique_05,
                \pan, -1))
    ], 1)
).play;
)
```

La figure 08 est une copie de la figure 07 avec le rate de la figure 06.

Illustration . Mixage de valeurs des figures 06 et 07

```
(
~figure_08 = Pdef (\fig_08,
    Ppar ([Pbindf (~figure_07, \rate, ~a)
    ], 1)
).play;
)
```

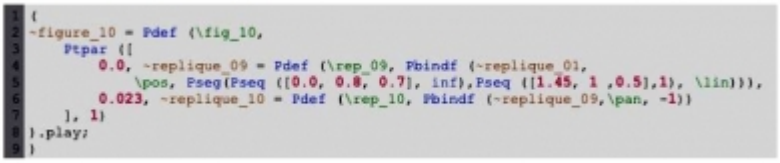
Dans l'exemple suivant, nous ajoutons deux autres répliques, qui sont une variante de la figure 03, avec un changement dans la durée de pos. Il s'agit des « allers-retours » sur le fichier son entre les positions 0.2 et 0.8, à une vitesse aléatoire variant entre 0.034 et 0.042 (ligne 6). Compte tenu de la différence d'amplitude entre les deux positions, cela produit une modulation d'amplitude qui varie légèrement entre un canal et l'autre. Dans cette figure, la durée globale diminue avec le nombre de répétitions de la liste.

Illustration . Deux nouvelles répliques

```
(
~figure_09 = Pdef (\fig_09,
    Ptpar ([
        {0.018.rand}, ~replique_07 = Pdef (\rep_07,
            Pbindf (~replique_02,
                \pos,Pseg(Pseq ([0.2, 0.8], 70), Prand ([0.034,0.042], inf), \lin))),
        {0.018.rand}, ~replique_08  = Pdef (\rep_08,
            Pbindf (~replique_07, \pan, 1)
        )
    ], 1)
).play;
)
```

Dans la figure suivante (illustration 17), nous changeons encore les valeurs de la durée du parcours dans la position (ligne 5). La durée globale de la figure diminue davantage. Toujours en suivant une direction musicale, nous nous éloignons de plus en plus de l'original.

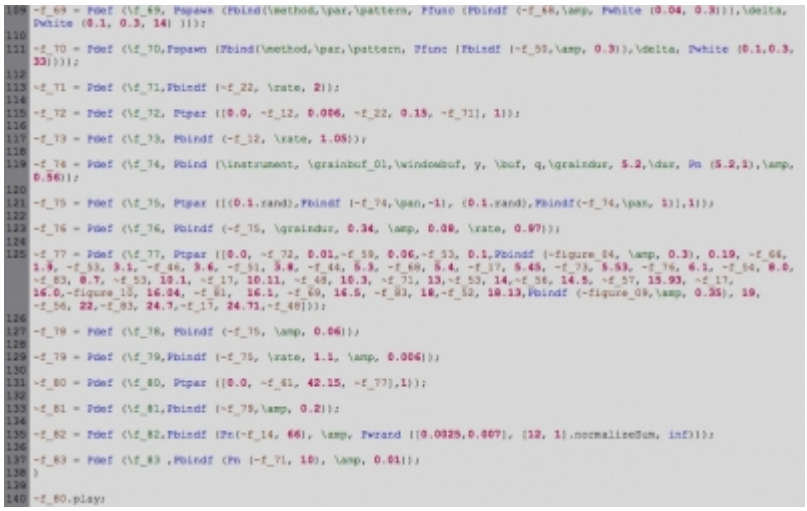
Illustration . Modification de la durée globale



Une polyphonie de figures spatiales

Dans ce qui suit, on verra un petit extrait d'une composition, il s'agit d'une séquence d'une durée de 1'10" environ. Il s'agit d'une figure (~f_80), composée en deux parties (~f_61 et ~f_77), elles-mêmes à leur tour composées d'autres figures et ainsi de suite. Au total, il y a quatre-vingt-trois figures de différentes tailles, chacune ayant une ou plusieurs paires stéréophoniques. Plus précisément, il s'agit de plusieurs originaux avec leurs répliques respectives, lesquelles sont de?corre?le?es avec des valeurs d'offsets et des directionnalite?s variées. Un véritable réseau de figures (ou de « contextes locaux »), indépendantes et interdépendantes. Nous avons donc composé une polyphonie spatialisée, dernière condition de la DMT.

Illustration . Extrait d'une polyphonie de figures spatiales, dernière condition de la DMT. C'est la figure 80 qui reçoit le message *play*



Conclusion

La composition musicale à multiples échelles spatio-temporelles dans laquelle on tisse un grand nombre de relations demande, à nos yeux, une attention particulière quant à la méthode qu'il faut adopter pour y faire face. Dans ce texte nous avons abordé un aspect, celui de la composition des attributs d'espace suivant une partie des recherches d'Horacio Vaggione sur la décorrélation microtemporelle. En prenant comme point de départ le concept de figure en tant que séquence d'événements finis stockée comme variable d'environnement, nous avons cherché à clarifier le rapport entre les originaux et leurs répliques. Cela étant dit, il est important de souligner que le couple « original-réplique » n'a de sens que dans le cadre théorique. Du moment où nous déclarons les attributs sonores d'ordre spatial en même temps que le reste de la figure, selon la direction musicale que nous prenons, la distinction perd sa pertinence pour laisser la place à la composition d'une multiplicité de figures que nous avons appelées systématiquement figure (~f_n). On pourrait décrire notre approche comme la « décorrélation des figures » plutôt que de signaux. Une figure spatiale est déjà une composition, un « contrepoint » entre deux ou plusieurs figures. Bien entendu, cette relation entre figures ne se limite pas au niveau microtemporel. Le « décalage macrotemporel » entre un contexte et un autre est étroitement lié. C'est cette imbrication entre espace et

temps à multiples niveaux suivant une « directionnalite? souhaitée pour des raisons musicales » que nous avons voulu montrer avec nos exemples.

Bien sûr, savoir ce qu'est une qu'une raison musicale, et comment nous y accédons sont deux questions majeures tout aussi importantes pour notre pratique que les techniques que nous avons utilisées.

---

1. http://supercollider.github.io/. Bien que nous n'utilisions pas exclusivement *SuperCollider* dans notre travail de composition, et que notre approche ne soit pas celle du codage à la volée *(live coding)* ni celle de l'algorithmique (dans son sens plus traditionnel), sa flexibilité et sa syntaxe apparaissent à nos yeux particulièrement bien adaptées à notre démarche de clarification conceptuelle autour de déterminations dans le processus de la composition. Voir M. Lorenzo, *Choix et composition musicale : dans l'espace des raisons*, Université Paris 8, Saint Denis, 2014.

2. H. Vaggione, « Décorrélation microtemporelle, morphologies et figurations spatiales », *in Actes des Journées d'Informatique Musicale (JIM), 2002*, Marseille, 2002.

3. Nous nous concentrerons surtout au travail de « décalage manuel » sous forme de code, en laissant de côté les techniques de filtrage qui sont mentionnées dans l'article.

4. Comme le notent Blackwell et Collins, avec les techniques du code en temps réel (*live coding*) le codage musical devient indépendant de l'ingénierie logicielle. Voir A. Blackwell et N. Collins, « The Programming Language as a Musical Instrument », Proceedings of PPIG05 (Psychology of Programming Interest Group) 2005. Cela dit, il est important de préciser que le sens de « live » se réfère au fait de changer le code « sur la marche » (ou « à la volée »), indépendamment du fait que cela ait lieu dans le studio ou dans une performance (*laptop performance*). Nous nous servons de l'aspect dynamique du langage pour favoriser le cycle « écriture-réécriture » propre au processus de composition. Autrement dit, nous « arrêtons la machine » après chaque évaluation. Voir plus bas « Séquence finie ».

5. Outre le travail de Vaggione, voir J. Colafrancesco, « L'ambisonie d'ordre supérieur et son appropriation par les musiciens, présentation de la bibliothèque Hoa.lib », *in Actes des Journées d'Informatique Musicale (JIM), 2012*, Mons, Belgique, ainsi que celui d'A. Sedes, « Approche musicale de la décorrélation microtemporelle dans la bibliothèque HOA », *in Actes des Journées d'Informatique Musicale (JIM) 2015*, Montréal, 2015.

6. H. Vaggione, « Décorrélation microtemporelle, morphologies et figurations spatiales », *op. cit.*

7. Il faut noter que Vaggione ne rejette pas les procédures par analyse/synthèse à partir des données spectrales concernant les valeurs de phase, mais il qualifie le travail d'arbitraire, et le juge « plus intéressant » lorsque le but n'est pas la modélisation.

8. La classe *Pattern* est associée a? un vaste ensemble de sous-classes. Sans compter les extensions, SC compte plus de 120 sous-classes de patterns. Elles ont la particularité de commencer par la lettre P. (*Pbind*, *Pseq*, *Prand*, etc.)

9. Plus précisément, le pattern est une représentation de haut-niveau d'une fonction (routine), pouvant retourner séquentiellement (avec les méthodes asStream puis next) les valeurs d'une série d'expressions depuis un modèle. Voir « Understanding Streams, Patterns and Events » dans http://doc.sccode.org/Browse.html#Tutorials%3EStreams-Patterns-Events.

10. La clé dur fait partie des clés par défaut de tous les *Pbind*.

11. M. Lorenzo, « Composition des figures en tant que séquences d'événements finies déclarées comme variable d'environnement » *in Actes des Journées d'Informatique Musicale (JIM) 2016*, Albi, 2016.

12. Voir « Abstraction, héritage » *in* H. VAGGIONE, « Son, temps, objet, syntaxe. Vers une approche multi-échelle dans la composition assistée par ordinateur », *in* A. Soulez (dir.), Musique, rationalité, langage?: l'harmonie, du monde au matériau, Paris, L'Harmattan, coll. « Cahiers de philosophie du langage », no 3, 1998.

13. Avoir une réponse immédiate est sans doute important dans le processus de composition. Cela dit, on n'est pas toujours dépendant des sons physiques. En effet, avec de l'entraînement, on développe une certaine écoute interne nous permettant de nous en passer.

14. Une variable d'environnement (commençant toujours par ~) est une « collection » ou un « dictionnaire », c'est-à-dire une liste permettant d'associer une clé ou plusieurs clés à une valeur ou une séquence de valeurs.

15. C'est pourquoi on la définit aussi comme « pseudo variable ».
http://doc.sccode.org/Classes/Environment.html

16. Afin de simplifier la présentation, nous travaillons avec un seul son enregistré. Il s'agit d'un son de guitare classique, plus précisément, la note Sol# jouée sur la première corde, *mf*, et d'une durée de 1.56 secondes. Nous avons volontairement choisi un son très simple par rapport à sa morphologie, pour mettre en valeur le travail avec les patterns. Il va sans dire que composer avec des sons plus complexes enrichit davantage notre palette.

17. Héritée des programmes Music n de Max Mathews, l'expression *Unit generator*, désignant une unité? de génération, de lecture, d'enregistrement ou de traitement de signal, est reprise dans plusieurs environnements d'informatique musicale, dont SC avec la classe *UGen*.

18. Plus précis que *Out* par rapport à la réactivité temporelle.

19. Des situations multiphoniques sont a? concevoir comme des situations dans lesquelles plusieurs paires stéréophoniques sont assignées a? des plans spatiaux spécifiques.

20. Le tempo des valeurs dans tous les exemples est celui par défaut : 1 = 1sec.

Pour citer ce document:
Mario Lorenzo, « Figures spatiales composées avec séquences finies », *RFIM* [En ligne], Numéros, n° 5 - Informatique et musique : Recherche et Création 1, Mis à jour le 24/07/2017
URL: http://revues.mshparisnord.org/rfim/index.php?id=394