Entre temps réel et temps différé - Pratiques, techniques et enjeux de l'informatique dans la musique contemporaine

Karim Barkati

THÈSE DE DOCTORAT DE L'UNIVERSITÉ PARIS 8

École Doctorale Esthétique, Sciences et Technologies des Arts

Discipline : Musique

Auteur :
Karim Barkati

Entre temps réel et temps différé

— Pratiques, techniques et enjeux de l'informatique dans la musique contemporaine —

Directeur de thèse :
M. Horacio Vaggione

Soutenue le 27 mars 2009 devant le jury composé de :

M. Horacio Vaggione
M. Martin Laliberté
M. Emmanuel Saint-James

Version du 2 septembre 2009

Résumés

Entre temps réel et temps différé

Cette thèse examine les notions de temps réel et de temps différé telles qu'on les rencontre en informatique musicale et dans la musique contemporaine.

La première partie commence par une approche historique, montrant la concomitance de leur apparition avec l'essor de l'informatique au début des années 60. Puis elle développe une critique du cloisonnement du temps réel et du temps différé en tant que catégories dichotomiques, à partir de définitions du seuil de latence qui sont incompatibles entre elles. Elle formalise enfin une représentation alternative : un axe « latentiel », qui intègre temps réel et temps différé. Cette représentation est illustrée par trois analyses axiales, utilisant un logiciel d'aide au classement relatif et subjectif.

À la suite de cette réflexion théorique, la seconde partie présente cinq collaborations musicales dans lesquelles j'ai réalisé la partie informatique avec Max/Msp :

- **CBox** avec le compositeur Mario Lorenzo, sur un projet d'exploration des possibilités de la circularité ;
- **Mimi** et **Rose amère** avec l'altiste Stéphanie Réthoré, sur un projet d'alto augmenté pour l'improvisation ;
- **FeedItBack** et **Iviv** avec le compositeur Santiago Quintáns, principalement sur la pièce *In Vivo / In Vitro*, pour caisse claire augmentée ;
- **a2m**, **Loterie**, **Emzed** et **Nappe** avec le compositeur Mauricio Meza, sur la pièce *Woes-war-sollichwer-den*, pour clarinette, public, et ordinateur ;
- **Plugiscope**, **Ifso**, et **Ultraviolette** avec le clarinettiste et compositeur Iván Solano, respectivement sur un premier projet pour voix, un deuxième pour clarinette basse, et un troisième pour percussion.

La thèse s'achève avec une brève réflexion sur le réalisateur en informatique musicale.

* * *

Mots clés : Temps réel, Temps différé, Informatique musicale, Max/Msp, Musique contemporaine, Réalisateur en informatique musicale.

Between real time and differed time

This thesis analyses the notions of real time and differed time such as they are to be found in contemporary music and in computer music.

The first part analyses the historical background of this two notions and how related they were with the raising of computer science in the early 60's. Then it develops a critical analysis of the split of real and differed times into two dichotomic categories, based on the coexistence of incompatible threshold definitions. It then formalises an alternate representation, building up a "latential" axis integrating real and differed times. This representation is finally illustrated by means of three axial analysis, using a program dedicated to relative and subjective sorting.

Beside this theorical development, the second part presents five musical collaborations in which I have programed the computer part with Max/Msp. Those five collaborations gave birth to several programs:

- ***CBox**, on a project about exploration of circularity possibilities, with composer Mario Lorenzo;*
- ***Mimi** and **Rose amère**, on a project for an augmented viola towards improvisation, with violist Stéphanie Réthoré;*
- ***FeedItBack** and **Iviv**, mainly on his musical piece* In Vivo / In Vitro, *for augmented snare drum, with composer Santiago Quintáns;*
- ***a2m**, **Loterie**, **Emzed** and **Nappe**, on his musical piece* Woes-war-sollichwer-den, *for clarinet, audience and computer, with composer Mauricio Meza;*
- ***Plugiscope**, **Ifso**, and **Ultraviolette**, respectively on a first project for voice, a second one for bass clarinet, and a third one for percussion, with clarinetist and composer Iván Solano.*

At last, the thesis briefly ponders over the role of the computer music designer.

* * *

Keywords: *Real time, Differed time, Computer music, Max/Msp, Contemporary music, Computer music designer.*

CICM / Centre de recherche Informatique et Création Musicale
Université Paris 8 / Vincennes – Saint-Denis
2, rue de la Liberté 93526 - Saint-Denis cedex 02

Remerciements

En tout premier lieu, je tiens à remercier les musiciens qui ont accepté de collaborer avec moi durant ce doctorat, et sans qui cette thèse n'aurait tout simplement pas existé : Mario Lorenzo, le premier à m'avoir fait confiance, Stéphanie Réthoré, aujourd'hui mon épouse, Santiago Quintáns, Mauricio Meza, et Iván Solano, qui se sont investis très généreusement, mais aussi Clarissa Borba et Pedro Castillo-Lara pour leur implication.

Ensuite, je veux remercier ceux qui m'ont transmis le goût de l'informatique musicale : Horacio Vaggione, Martin Laliberté, Anne Sedes, José Manuel López-López, Makis Solomos, et Jean-Claude Risset ; le goût de la programmation : Harald Wertz, Emmanuel Saint-James, Vincent Lesbros, Pierre Audibert et Daniel Goossens ; et le goût de la musique : Nicolas Baldeyrou, Cécile Gilly, Jean-Claude Bouveresse, Patricia Boulay, Rémy Salaün, Lisa Erbès et Thomas Julou.

Enfin, je remercie chaleureusement tous ceux qui m'ont soutenu et accompagné dans cette épreuve de longue haleine que reste une thèse : à nouveau Stéphanie Réthoré, pour son soutien indéfectible, ma mère Sylvie Delsart, bien présente ces derniers mois, mon père Arezki Barkati, qui me manque mais dont les valeurs continuent de me guider, les petits coups de pouce bien utiles de Julien, Fabienne et Stéphane Réthoré, l'accueil bienveillant de Martine et Serge Villard, et Caroline Chardonnet, Iván Solano, et Sophie Dumas, courageux relecteurs.

Avant-propos

Depuis le point de vue du programmeur

Cette thèse de musique adopte le point de vue de la programmation pour aborder la problématique du couple temps réel / temps différé et de leur relation dans la création musicale contemporaine. Or on peut s'interroger sur la légitimité d'un tel point de vue pour une thèse de musique, car après tout, au regard de sa fonction utilitariste, la technologie peut nous paraître aux antipodes de l'art...

Cependant, d'une part, on peut arguer du fait que la musique contemporaine fait un usage spécialement intensif des nouvelles technologies, et en particulier de l'informatique. À ce titre, les questions de programmation de logiciels musicaux trouvent une place de fait, et à plus forte raison lorsqu'ils s'intègrent dans des projets de création contemporaine comme c'est le cas ici.

D'autre part, la musicologie elle-même se nourrit de disciplines autres, comme Makis Solomos le souligne dans son entretien intitulé *Une musicologie et son temps* :

> S'il existe bien aujourd'hui une discipline que l'on appelle « musicologie » et si les musicologues savent se reconnaître entre eux, cependant, ils ont tous, en cachette ou publiquement, des lectures qui les obligent sans cesse à faire éclater et recomposer leur discipline. Si l'on s'accorde sur ce constat, on devra donc dire qu'il n'y a pas de musicologie « traditionnelle », mais une pléthore de discours sur la musique. [Solomos, 2004]

Cette thèse déploie en effet un « discours sur la musique » particulier, formulé depuis le point de vue d'un programmeur – moi-même –, et emprunte ainsi un angle d'approche souvent technologique, distinct de celui du musicologue, du compositeur, de l'interprète, de l'historien, ou du philosophe. Parfois, ce discours singulier peut s'apparenter à de la musicologie ou plus exactement à de la « proto-musicologie », en tant qu'il tente d'éclaircir des relations profondes entre la musique et la technologie. Néanmoins, dans sa plus grande part, ce mémoire traite simplement de la musique telle qu'elle s'est faite tout au long de mon doctorat.

Ce que cette thèse n'est pas

Le titre de cette thèse – *Entre temps réel et temps différé* – sonne plein de promesses tant la question du temps habite et agite celle de la musique. À vrai dire, c'en est vertigineux ! Comment parler sérieusement du temps sans connaître la conception des penseurs fondamentaux sur cette question que sont Aristote, Saint Augustin, Bachelard, Bergson, Canguilhem, Deleuze, Descartes, Einstein, Fraisse, Hegel, Husserl, Jacob, Janet, Kant, Lavelle, Levinas, Merleau-Ponty, Newton, Nietzsche, Pascal, Platon, Plotin, Prigogine et Stengers, Ricœur, Sartre, Sénèque, ou Spinoza ?

En réalité, la longueur de cette liste[1] non exhaustive répond à cette question, car chacun de ces auteurs semble incontournable, avec au moins un ouvrage majeur portant directement sur la question du temps. Par ailleurs, la plupart de ces ouvrages contiennent une pensée difficile à appréhender, ce qui multiplie d'autant le temps nécessaire à une compréhension même rudimentaire... Par suite, on mesure aisément que la « simple » tâche de se familiariser avec ces différentes pensées du temps sort déjà du cadre d'étude d'une thèse de musique.

En outre, il semblerait que la musique puisse se passer en partie de la connaissance extensive ou érudite de ces conceptions du temps, ainsi que le signale Pascal Dusapin :

> Le temps du philosophe, celui du physicien, ne sont pas ceux du musicien. Aucune des démonstrations spatio-temporelles de la philosophie ou des sciences ne convient à la musique. Les musiciens préjugent que le temps est leur matière principale mais peuvent en avoir une connaissance presque rudimentaire. [...] on n'écrit pas de la musique avec du temps mais avec des durées. [Dusapin, 2007]

Par conséquent, cette thèse évitera purement et simplement la référence à ces penseurs pourtant majeurs de la question du temps, à regret cependant[2], afin de se concentrer sur les notions de *temps réel* et de *temps différé*, en particulier telles qu'elles existent et opèrent dans le domaine de la création musicale contemporaine et de l'informatique musicale.

Deux jambes

Ce travail n'a pu progresser qu'à travers une marche « bipède » : tantôt pratique, tantôt théorique. Ainsi, bien que l'organisation de ce mémoire soit clairement scindée en une première partie théorique et une seconde partie pratique pour des raisons de lisibilité, l'avancement de la thèse s'est déroulé au long d'un processus, quant à lui, inextricable.

De fait, les cinq collaborations – avec Mario Lorenzo, Stéphanie Réthoré, Santiago Quintáns, Mauricio Meza, et Iván Solano – et le corpus des douze logiciels qui en a résulté, ont véritablement permis la réflexion théorique, tout autant que la réflexion théorique a nourri ces collaborations et ces logiciels, permettant en retour aux logiciels de prendre leur forme actuelle.

Cette intrication nous a paru indispensable à expérimenter et à exprimer, même si, au niveau du mémoire, elle entraîne une disproportion entre les deux parties et une certaine

1. Partiellement inspirée du compendium d'Alban Gonod intitulé *Le temps* [Gonord, 2001].

2. Mais je ne désespère pas de pouvoir approfondir ma connaissance de ces auteurs dans les années à venir, pour de futures recherches ?

artificialité dans la séparation de ces deux parties. En effet, d'une part, la partie pratique reste plus volumineuse que la partie théorique ; ceci se justifie entre autres par la quantité des collaborations entreprises, et par les nécessaires copies d'écran qui illustrent les programmes. D'autre part, parfois, des aspects pratiques (p. ex. les logiciels) apparaissent dans la première partie, et des aspects théoriques (p. ex. les notions de *performabilité* et de *composabilité*) sont développés dans la seconde.

Table des matières

Introduction

La musique et le temps partagent ceci de singulier qu'ils résistent tous deux à la définition : nous savons de quoi il s'agit sans toutefois être capables de le dire. Le XX^{e} siècle qui vient de s'écouler, fort de son cortège d'avancées prodigieuses dans le domaine de la connaissance, n'a d'ailleurs paradoxalement pas facilité la définition ni du temps – les sciences physiques nous ont même appris qu'il n'y a pas de temps absolu – ni de la musique – on ne s'étonne plus aujourd'hui qu'il existe des « musiques » qui peuvent se passer de mélodie, d'harmonie et de rythme mesuré, mais aussi de partition et d'interprète... Bien sûr, ces incertitudes patentes n'empêchent ni la science de continuer d'avancer, ni la musique de continuer de se faire, ni non plus les progrès technologiques de continuer de bouleverser les arts et les sciences : de fait, la « révolution informatique » a profondément modifié les pratiques scientifiques, par exemple avec l'avènement de la simulation, tout comme les pratiques artistiques, de la création à la monstration.

Dans ce contexte – celui de l'essor exponentiel de l'informatique et de son intégration généralisée (quelle discipline se prive encore de ses apports ?) –, la notion de « temps réel » a d'abord émergé dans les domaines de la finance et de l'industrie, dès le début des années soixante, avant de se diffuser progressivement dans bien d'autres champs. Parallèlement, la notion de « temps différé » s'est constituée en contrepoint de cette toute nouvelle notion de « temps réel », sans doute sous une mue opportuniste de la notion plus ancienne du « différé » telle qu'elle existait dans le domaine des télécommunications. Dont acte d'un nouveau couple de notions, profondément liées à l'informatique : le temps réel et le temps différé.

L'importance flagrante de ce couple en informatique musicale – et tout particulièrement en musique contemporaine –, ajoutée à des débats esthétiques houleux et à des contradictions récurrentes, a motivé pour une large part ce travail de recherche. Ainsi, ce mémoire de thèse entame un examen de ces deux notions dans sa première partie, théorique, à l'aune d'une activité de programmation musicale soutenue dont rend compte sa seconde partie, pratique.

Une première partie théorique

Le premier chapitre de cette thèse étudie ce couple temps réel / temps différé sous différents aspects : d'abord la question de la nature de ces notions, puis la question de l'histoire de ces notions, jusqu'à leur migration dans le champ de l'informatique musicale, et enfin la question du sens actuel « musical » de ces notions.

Le deuxième chapitre développe une critique épistémologique du cloisonnement du temps réel et du temps différé lorsqu'ils sont considérés en tant que catégories dichotomiques – ce qui est pratiquement toujours le cas. Ce chapitre discute d'abord du seuil sensé séparer ces deux catégories, puis propose de démêler la polysémie des expressions « temps réel » et « temps différé » en distinguant trois champs sémantiques : le champ technique, le champ pratique et le champ musical. Il rappelle ensuite la multiplicité des temporalités technologiques, puis se penche sur quelques cas de contradiction entre temps réel et temps différé afin d'éprouver la polysémie proposée.

Le troisième chapitre propose une représentation alternative à la représentation dichotomique critiquée au chapitre précédent : un axe « latentiel » qui intègre les deux aspects du temps réel et du temps différé. Pour opérer ce changement de représentation, ce chapitre montre d'abord en quoi le temps réel et le temps différé peuvent être considérés comme deux pôles idéaux, puis propose de munir cet axe d'une relation d'ordre, terme à terme et subjective. Un outil logiciel d'aide au classement relatif est ensuite présenté, puis appliqué à trois analyses axiales qui permettent de discuter de la validité de cette nouvelle représentation.

Ces trois chapitres forment ainsi la première partie de cette thèse, partie théorique dirigée vers la formalisation de cet axe de localisation latentielle des logiciels musicaux.

Une seconde partie pratique

La réflexion théorique développée dans la première partie s'appuie sur une pratique importante de l'informatique musicale au cours du doctorat, à travers la réalisation d'une douzaine de programmes en collaboration, avec principalement cinq musiciens (compositeurs et/ou interprètes). Afin d'éviter les risques de monotonie au cours de la lecture de cette longue partie pratique organisée en sept chapitres, les cinq chapitres centraux consacrés aux logiciels développés en collaboration (de 5 à 9) sont présentés de manière différente.

Le quatrième chapitre, introductif, traite de nos collaborations musicales pour ce qu'elles partagent en commun : notamment le choix de l'interaction, la prémisse acoustique, l'autonomie de l'interprète avec l'ordinateur, et le dispositif « podophonique » utilisé par plusieurs de ces projets.

Le cinquième chapitre révèle des points de passage entre la première partie théorique et la seconde partie pratique de la thèse, en rapportant les réflexions issues de la collaboration avec le compositeur Mario Lorenzo au cours du développement de la **CBox**. En particulier, ce chapitre dégage les concepts de *performativité* et de *composabilité* à partir d'un examen de l'idée d'interactivités plurielles.

Le sixième chapitre rapporte une collaboration orientée vers l'improvisation autonome à l'alto augmenté, avec l'altiste Stéphanie Réthoré. Après avoir identifié trois contraintes majeures pour l'improvisation – l'équilibre sonore, la variété musicale et la promptitude réactionnelle –, ce chapitre présente **Mimi**, un premier prototype d'auto-échantillonneur, puis délivre un mode d'emploi de **Rose amère**, un logiciel interactif plus abouti et qui a déjà pu faire ses preuves lors de plusieurs concerts.

Le septième chapitre entre davantage dans les arcanes de la programmation sur Max/Msp, à travers les deux logiciels développés en collaboration avec le compositeur Santiago Quintáns : **FeedItBack**, un logiciel imitant une pédale de délai, et **Iviv**, un auto-échantillonneur interactif. Ces notes de programmation sont toutefois précédées par une brève analyse musicale de la pièce *In Vivo / In Vitro*, pour caisse claire augmentée, créée par Clarissa Borba.

Le huitième chapitre présente le projet ambitieux et inachevé entrepris avec le compositeur Mauricio Meza : *Woes-war-sollichwer-den*, une pièce pour clarinette, public et ordinateur. Il montre d'abord en quoi ce projet peut être considéré comme complexe, puis présente les quatre logiciels développés pour ce projet : **a2m**, un outil précompositionnel de génération de matériau, **Loterie**, un petit utilitaire de tirage au sort, **Emzed**, un auto-échantillonneur interactif, et **Nappe**, un générateur de texture interactif et programmable.

Le neuvième chapitre dresse un panorama succinct des logiciels développés en collaboration avec le compositeur et clarinettiste Iván Solano : **Plugiscope**, une interface généraliste pour les plugiciels Vst, **Ifso**, un lecteur polyphonique interactif, et **Ultraviolette**, un générateur de flux rythmique interactif[3]. Par ailleurs, un entretien conséquent avec Iván a été retranscrit en annexe B page 253, portant entre autres sur les projets avec Ifso et Plugiscope, sur les questions du temps réel et du temps différé, et sur des questions liées à la collaboration.

Le dixième chapitre, pour finir, opère un « pas de côté » par rapport à cette partie pratique, en faisant part de quelques réflexions personnelles sur la collaboration en tandem, et en particulier sur mon travail de réalisation de la partie informatique au sein ces projets musicaux. Il clôt ainsi la seconde partie de cette thèse, avant le bilan et les perspectives de la conclusion générale.

3. Une présentation générale des principaux logiciels développés au cours de ce doctorat est donnée à la section 3.4.1 page 78.

Première partie

Un axe de localisation latentielle des logiciels musicaux

Chapitre 1

La dichotomie du temps réel et du temps différé

Résumé du chapitre : Temps réel et temps différé constituent dans le langage commun de notre discipline deux catégories antagonistes de classement des logiciels. Cependant, la frontière entre ces deux catégories historiquement pertinentes tend à se déliter à mesure des évolutions de l'informatique en général et avec celles de l'informatique musicale en particulier. Nous nous proposons dans ce chapitre d'étudier les conséquences de ces évolutions sur la façon d'envisager ces deux notions individuellement et en rapport.

1.1 Remarques préalables

1.1.1 Une ontologie informatique

Tout d'abord, nous voudrions montrer que les expressions « temps réel » et « temps différé » n'ont de sens en informatique musicale que relativement à la machine, et plus précisément à l'ordinateur.

Nous pensons en effet qu'en musique, l'expression « temps réel » n'a pas de raison d'exister en dehors des pratiques liées à l'informatique, pas plus que le « temps différé », pour la raison que toute musique sans ordinateur peut être classée de façon triviale dans un camp ou dans l'autre. Par exemple, dire que l'exécution, l'interprétation ou l'improvisation ont lieu en temps réel, ou bien dire que la composition, l'écriture ou la copie ont lieu en temps différé relève simplement de la tautologie, selon les définitions que nous retiendrons. C'est donc bien l'usage d'un même outil, l'ordinateur, et à distinguer parmi ses différentes utilisations, qui fonde et justifie ces deux expressions, d'où l'idée d'une « ontologie informatique » (ou la *nature* informatique si l'on préfère) du temps réel et du temps différé.

Précisons qu'il ne s'agit pas ici de l'ontologie informatique « technique » telle que l'a définie Thomas Gruber en 1993 dans son article *Towards Principles for the Design of Ontologies Used for Knowledge Sharing*, comme la spécification d'une conceptualisation d'un domaine de connaissance[1], mais bien d'une ontologie au sens philosophique, en posant l'informatique comme condition préalable à l'existence des notions de « temps réel » et de « temps différé ».

1.1.2 L'ancrage analogique du couple « direct » / « différé »

Les expressions de « temps réel » et de « temps différé » existaient dans d'autres domaines avant d'apparaître en musique, en particulier ceux de la finance et de l'industrie dès le début des années 1960. Cependant, l'idée de deux temporalités en opposition – entre du « direct » et du « différé » – s'était déjà banalisée plusieurs décennies auparavant à travers les émissions de radio dans les années 20 puis de télévision dans les années 50[2]. Mais plus tôt encore le téléphone avait imposé l'habitude de la quasi-instantanéité communicationnelle, dès les débuts de l'ère de l'électricité, à la fin du XIX^{e} siècle... On constate alors dans cet ancrage des habitudes communes la puissance du lien entre machine et instantanéité[3].

Pour appréhender la nature et la spécificité des vocables « temps réel » et « temps différé », nous pouvons interroger un couple de notions proches : quelles différences peut-il y

1. Gruber propose de formaliser des connaissances à partir de cinq critères : clarté, cohérence, extensibilité, déformation d'encodage minimale, et engagement ontologique minimal. La description formelle ainsi obtenue doit permettre de représenter un corpus de connaissances sous une forme utilisable par une machine.

2. La RTF – Radiodiffusion-Télévision Française – remplace la RDF – Radiodiffusion Française – en 1949.

3. Sauf exception, nous utiliserons ici le terme « instantanéité » au sens d'instantanéité *perceptive* et non pas absolue, puisqu'elle dépend dans ce contexte technologique de la vitesse du courant électrique dans les circuits, ou, au mieux, de la vitesse de la lumière.

avoir entre l'idée familière de l'opposition « direct » / « différé » et l'opposition « temps réel » / « temps différé » ? Sans doute plusieurs différences, mais l'ontologie informatique nous paraît suffisamment englobante pour définir à elle seule une ligne de partage entre ces deux oppositions. Autant le couple « direct » / « différé », issu des technologies analogiques, est susceptible de qualifier des réalisations informatiques au-delà de son domaine analogique d'origine, comme la radio sur internet, autant le couple « temps réel » / « temps différé » appartient plus strictement, tout en le caractérisant, au champ de l'informatique, dans le vocabulaire courant comme dans le vocabulaire musical. Jean Cristofol[4] précise la notion de « direct » :

> Ce qui nous est donné à voir, dans une « émission » en direct, c'est ce qui se passe, en ce moment même, ailleurs. Le direct renvoie à l'immédiateté d'une transmission qui ne transite plus par l'intermédiaire d'un enregistrement, d'une fixation de l'image ou de son sur un support qui les pérennise et les arrache à leur actualité. [Cristofol, 2003]

Le direct est donc associé à la *transmission d'une émission*, telle que la technologie analogique (essentiellement hertzienne) l'a permise, avec comme première caractéristique l'absence d'enregistrement sur un support de mémoire dans le processus de diffusion, et comme deuxième caractéristique le corollaire de « plus-de-vérité » ou de « plus-de-transparence » sous-entendu par l'absence des étapes de montage (ce qui n'exclut pas la possibilité d'autres formes de manipulations médiatiques). Voici un court commentaire assez représentatif (extrait d'un journal hebdomadaire français), qui témoigne de l'attache sémantique profonde du terme « direct » à la notion « d'émission » médiatique dans le vocabulaire courant. En effet, dans cet extrait, le terme « direct » se prolonge dans le champ informatique comme s'il s'agissait d'émissions analogiques :

> Selon Médiamétrie, le nombre d'internautes écoutant la radio sur le Net a augmenté de 22 % en un an. Alors que les vidéos sont plutôt consultées en différé (32,5 % contre 9 % en direct), la radio est, elle, encore principalement consultée en direct (32,1 % contre 8.7 % en différé). [L'Espresso, 2007, c'est nous qui soulignons]

Cependant, si le terme « direct » pénètre parfois le champ informatique, en particulier pour les activités médiatiques qui héritent de la tradition analogique et ne font que s'y prolonger, l'expression « temps réel », elle, reste foncièrement liée à ce champ informatique et désigne bien, finalement, tout autre chose que le direct. Intuitivement, le temps réel désigne effectivement d'autres types d'activités : un premier type informationnel, des cours de la bourse au GPS[5] en passant par les flux RSS[6], et un deuxième type plus calculatoire,

4. Philosophe, épistémologue, enseignant à l'Esa Aix-en-Provence, Jean Cristofol propose une réflexion sur la notion même de « temps réel », et le cadre théorique et pratique dans lequel elle prend sens, celui qu'engage les dispositifs technologiques interactifs. Il est l'auteur de plusieurs textes, au sein d'ouvrages souvent collaboratifs, dont *Inscriptions et singularité*, Art / Cognition (Cyprès – École d'Art, Aix en Provence, 1993), *Prédiction, Variation, Imprévu*, Laboratoire Plot (Agglo – programme de recherche, 2003–2006), et *Écritures, dispositifs, expériences*, publié dans *Nouveaux média, nouveaux langages, nouvelles écritures* (l'Entretemps, 2005).

5. *« Global Positioning System »*, système de positionnement par satellite, récemment porté au grand public pour la trajectographie en *temps réel* dans les automobiles.

6. Un flux RSS (*« RSS feed »* en anglais) est un format de syndication de contenu Web, acronyme de *Really Simple Syndication*. Ce système permet de diffuser en *temps réel* les nouvelles des sites Web, ce

de la téléchirurgie[7] aux performances artistiques avec ordinateur (souvent dites *« live »*) en passant par les MMORPG[8]. Par ailleurs, il est à remarquer la différence entre « le différé » et « le temps différé »... En effet, le même principe semble valoir qu'entre « le direct » et « le temps réel », à savoir la référence aux activités de l'époque analogique pour le premier, et la référence aux activités de l'époque informatique (dite aussi époque *numérique*) pour le second. Le « différé » relève aussi de la transmission des émissions en général, alors que « le temps différé » qualifie d'abord des activités informatiques.

1.1.3 Des difficultés sémantiques liées à l'informatique

La rapidité de l'évolution de l'informatique met souvent en difficulté le vocabulaire qui s'y rapporte, et les expressions de « temps réel » et de « temps différé », dont on vient de voir qu'elles relèvent essentiellement de l'informatique, n'échappent pas à ces difficultés.

Un domaine aux mutations particulièrement fulgurantes

Andrew Tanenbaum rend compte dans son livre *Architecture de l'ordinateur* d'une spécificité de l'informatique concernant la rapidité de son évolution, en s'appuyant sur les constats de Gordon Moore et de Richard Hamming :

> L'industrie informatique se développe comme nulle autre. Les fabricants de circuits sont principalement motivés par la possibilité d'intégrer chaque année toujours plus de transistors sur une même puce. Plus il y a de transistors, c'est-à-dire de minuscules commutateurs électroniques, plus les mémoires sont importantes et les processeurs puissants. Comme a dit en plaisantant Gordon Moore, le cofondateur et ancien président d'Intel : « Si la technologie aéronautique avait progressé aussi rapidement que celle des ordinateurs, les avions coûteraient aujourd'hui 500 dollars et feraient le tour de la Terre en 20 minutes avec 19 litres de carburant... mais auraient la taille d'une boîte à chaussures. » [Tanenbaum, 2005, p. 27]
>
> L'ancien chercheur de Bell Labs, Richard Hamming, a un jour constaté qu'une multiplication par dix en termes de quantité provoquait un saut qualitatif. Une voiture de course pouvant atteindre 1000 km/h dans le désert du Nevada est fondamentalement différente d'un véhicule classique conçu pour rouler à 100 km/h sur

qui permet de consulter rapidement ces dernières sans visiter le site. La spécification RSS 2.0 de Dave Winer est disponible sur le site d'Harvard [Winer, 2003], sous licence *Creative Commons*.

7. La première intervention chirurgicale tansatlantique chez l'homme, baptisée « Opération Lindbergh », a eu lieu le 7 septembre 2001, réalisée par les professeurs Marescaux, Leroy et Gagner entre New York et Strasbourg (environ 7 000 km). La notion de temps réel est ici vitale : *« Whether up-to-date, telecommunication technologies have allowed the sharing of information, voice, and images, here we show for the first time that complex gestures can be performed with high precision and in real time over long distances. »* [...] *« Despite a round-trip distance of more than 14,000 km, the mean time lag for transmission during the procedure was 155 ms. »* [Marescaux *et al.*, 2002]

8. *« Massively Multiplayer Online Role-Playing Game »*, ou jeu de rôle en ligne massivement multijoueur, désigne les jeux vidéo liant le jeu de rôle avec le jeu en ligne. Les MMORPG se distinguent des jeux de rôle classiques par le très grand nombre de joueurs impliqués, mais aussi par le monde virtuel dit *persistant* dans lequel ils évoluent. Depuis 2005, *World of Warcraft* est le plus populaire des MMORPG, avec plusieurs millions de joueurs actifs. Signalons aussi la simulation *Second Life*, qui utilise une monnaie virtuelle qui peut être échangée contre de la monnaie *réelle*...

> l'autoroute. De la même façon, un gratte-ciel de cent étages n'est pas un simple immeuble de dix étages dessiné à une plus grande échelle. Or, en matière d'ordinateur, il s'agit, non pas d'un facteur de dix, mais d'un facteur de 1 million sur une période de trente ans. [Tanenbaum, 2005, p. 29]

Un vocabulaire en défaut

Il faut tirer toutes les conséquences de la connaissance des spécificités d'une telle évolution. En particulier, nous pensons qu'il faut anticiper des difficultés de vocabulaire et de sens. Le premier écueil consisterait à ignorer que ce domaine connaît des lacunes lexicales, entre autre à cause de la rapidité de son innovation qui réduit d'autant le temps de création, de maturation et d'ajustement du vocabulaire pour des technologies et des principes versatiles et proliférants.

Le second écueil consisterait à ignorer que ce domaine connaît des glissements sémantiques fréquents, parce que les réalités désignées changent. Par exemple, ce que nous nommons aujourd'hui « internet » ne désigne plus ce que désignait ce mot ne serait-ce qu'il y a cinq ans, sans le « haut débit » (ADSL et câble[9]) ni l'explosion du « réseautage social numérique[10] », bien que ce réseau utilise toujours la même norme TCP/IP[11]. Intrinsèquement, il y a une véritable difficulté à désigner des réalités mouvantes, mobiles, ce qui est le cas en informatique.

Dans ce domaine où le temps semble plus court qu'ailleurs, plus contracté, on constate à la fois des déplacements sémantiques flous pour les « anciens » mots, un retard des néologismes, des « faux-amis » lors des transpositions directes du vocabulaire anglo-saxon (qui domine l'informatique actuelle) dans une autre langue, et finalement un retard à penser induit par des références dépassées ou impropres. Bien sûr, il ne s'agit pas de diaboliser la mobilité du vocabulaire, puisque c'est le propre de toute langue vivante, mais il s'agit de pointer la rapidité exceptionnelle qui caractérise l'évolution de l'informatique, rapidité tout à fait critique et inédite depuis l'avènement du réseau généralisé, puisque tout changement local peut se répercuter sur le réseau entier.

Enfin, il faut noter d'une part que le Journal Officiel donne une définition du temps réel qui reste très sommaire (cf. page 14), et d'autre part qu'il ne propose pas du tout de définition pour le temps différé[12]...

Ainsi, la prise en compte de l'ensemble de ces difficultés spécifiques nous amènera à

9. En France, l'internet à haut débit comptait 9 465 600 abonnés en 2005 contre 197 911 en 2000, selon l'observatoire de l'ARCEP (l'Autorité de Régulation des Communications Électroniques et des Postes, http://www.arcep.fr), soit près de 47 fois plus d'abonnés (une croissance de 4 683 % en cinq ans) [ARCEP, 2006].

10. Le « réseautage social » ou *« social networking »* désigne les activités humaines massivement collaboratives reposant sur les technologies du « Web 2.0 », dont les blogs, Wikipedia, l'*open-source*, et le *peer-to-peer* constituent quelques représentants (cf. l'article *We are the web* [Kelly, 2005]).

11. *« Transmission Control Protocol / Internet Protocol »* : norme qui permet à des ordinateurs différents de communiquer aisément entre eux, à partir des protocoles IP et TCP inventés en 1974 pour le réseau de télécommunication ARPANET du ministère de la défense américaine.

12. Le site officiel des termes publiés au Journal Officiel, FranceTerme, répond « Aucun résultat ne correspond à cette recherche (temps différé). ».

reconsidérer la signification et la façon de donner du sens au vocabulaire actuel de « temps réel » et de « temps différé », d'essence informatique, au long de ce chapitre.

* * *

Les notions « temps réel » et « temps différé » relèvent d'une ontologie informatique, à la différence des notions de « direct » et de « différé ». Elles subissent ainsi les difficultés sémantiques inhérentes à l'informatique. Les sections suivantes étudieront donc la question de leur sens, d'abord aux niveaux historique et technologique, puis plus particulièrement dans le champ musical.

1.2 Une dichotomie historique technologique

S'il est impossible de déterminer précisément la naissance des notions de « temps réel » et de « temps différé », nous pouvons tout de même avancer des repères historiques généraux, puis des éléments plus spécifiques à leur diffusion en art et en musique.

Ainsi, après une recontextualisation sur la lenteur caractéristique des débuts de l'informatique, cette section présentera d'abord une brève histoire du temps réel en informatique, puis la notion générale du temps réel en art. Ensuite, revenant sur la musique, elle montrera l'inévitabilité du souhait du temps réel en musique, puis sa conquête musicale dans les années 70, jusqu'à la formation de la dichotomie de l'informatique musicale telle que nous la connaissons aujourd'hui, partagée par le critère de la latence. Enfin, cette section rappellera les problématiques liées à ce seuil de latence, relatif à la perception.

1.2.1 De la lenteur des débuts

Aujourd'hui, tout le monde ne se souvient pas combien les débuts de l'informatique furent intensément chronophages. Avec l'extrait suivant, certes un peu long – mais justement ! –, Andrew Tanenbaum transmet bien cette sensation de lourdeur et d'inertie dans la tâche des premiers programmeurs [Tanenbaum, 2005, p. 9] :

> Au début de l'ère informatique, la plupart des ordinateurs étaient en libre-service. En regard de chaque ordinateur se trouvait une fiche de présence. La personne souhaitant exécuter un programme réservait une plage horaire, par exemple mercredi entre 3 et 5 heures du matin (la plupart des programmeurs préféraient profiter du silence de la nuit). À l'heure dite, elle se présentait dans la salle informatique avec un paquet de cartes perforées de 80 colonnes (l'un des premiers supports de données) et un simple crayon à papier, expulsait gentiment le précédent programmeur et prenait sa place au pupitre de l'ordinateur.
>
> Pour exécuter un programme FORTRAN, on procédait comme suit :
>
> 1. On se procurait le gros paquet de cartes vertes intitulé compilateur FORTRAN dans l'armoire où était conservée la bibliothèque des programmes, on le plaçait dans le lecteur de cartes et on appuyait sur le bouton START (démarrer).
>
> 2. On plaçait le programme FORTRAN dans le lecteur de cartes et on appuyait sur CONTINUE (poursuivre). Le programme était lu.

> 3. À l'arrêt de l'ordinateur, on relançait une seconde fois le programme FORTRAN. Certains compilateurs n'avaient besoin que d'un seul passage des instructions, mais la plupart en nécessitaient deux ou même plus. À chaque passage, un épais paquet de cartes devait être lu.
>
> 4. Plus la traduction approchait de la fin, plus la tension augmentait car si le compilateur identifiait une erreur dans le programme, la procédure devait être recommencé depuis le début. Si le programme ne contenait pas d'erreur, le compilateur perforait des cartes pour inscrire le programme traduit en langage machine.
>
> 5. On plaçait alors le programme en langage machine dans le lecteur de cartes ainsi que le paquet de cartes de sous-programmes. Les deux étaient lus.
>
> 6. Enfin, le programme démarrait. Le plus souvent, il ne fonctionnait pas et s'interrompait en plein milieu. Après examen des voyants lumineux et bataille avec les commandes du pupitre de l'ordinateur, on avait éventuellement la chance de trouver la solution du problème. On corrigeait alors l'erreur, on retournait prendre le gros paquet vert du compilateur FORTRAN et on recommençait l'ensemble du processus. Si la solution restait un mystère, on lançait l'impression du contenu de la mémoire, appelée vidage mémoire, et on rentrait chez soi pour étudier tout cela.
>
> À quelques détails près, cette façon de travailler était habituelle. C'est ainsi qu'ont fonctionné de nombreux centres de calcul pendant des années. Les programmeurs étaient obligés d'apprendre à exploiter les machines et devaient savoir quoi faire lorsqu'elles tombaient en panne, ce qui était fréquent. De plus, les ordinateurs étaient souvent inutilisés, le temps que les personnes aillent et viennent pour transporter les paquets de cartes perforées ou réfléchissent à la solution d'un problème ayant interrompu leur programme.

Même s'il arrive toujours à nos ordinateurs de tomber en panne, cette description assez fidèle de l'aspect laborieux de la programmation à ces débuts nous permet de mesurer la distance qui sépare notre informatique actuelle de la « proto-informatique » ; pour se représenter cette activité aujourd'hui, on pourrait invoquer un temps « ultra-différé[13] »... L'invention du système d'exploitation, vers 1960, permit de gagner du temps sur les premières étapes. Puis, pendant que le matériel vit sa vitesse de calcul augmenter constamment et de manière impressionnante[14], l'architecture logicielle se développait en couches successives, vers des langages toujours de plus haut niveau afin de faciliter et donc d'accélérer le travail de l'humain avec l'ordinateur.

13. Aujourd'hui, cette lenteur est quasiment devenue inimaginable. Dans son essai *La société immédiate*, Pascal Josèphe avance un *paradigme de l'immédiateté* pour décrire notre temps postmoderne, précisément à partir des progrès de l'informatique et de la télécommunication : « La révolution numérique engendre de nouveaux systèmes communicationnels, qui soumettent les évolutions sociales à leur propre rythme et à un nouveau langage. Il est inéluctable que cela ait en retour un impact sur le rapport de chacun à son environnement le plus immédiat, sur son rapport au monde, et donc sur la vie collective. L'activité médiatique s'exerce désormais dans un contexte caractérisé par deux tendances de fond. D'une part, une tendance à l'accélération et à la compression du temps, qui fait de l'immédiateté le nouveau paradigme des techniques et des pratiques sociales. Entre l'énonciation d'un projet et son accomplissement, l'attente est vécue comme une frustration, un retard, contraires à l'idéal de fluidité généralisée. On le constate dans la vie publique, dans la consommation, dans l'activité créatrice, dans la relation amoureuse et sexuelle, etc. Notre époque n'aime pas que l'on "perde son temps". [...] » [Josèphe, 2008, p. 71]

14. La loi de Moore, ou plus exactement l'observation empirique érigée en prédiction de Gordon Moore, indique un doublement de la puissance de calcul et de la capacité mémoire tous les 18 mois, ce qui s'est vérifié durant plus de trente ans, dès avant 1970 jusqu'à récemment. Le ralentissement actuel de la « loi » de Moore provient des problèmes de gestion de l'échauffement des microprocesseurs dûs à leur miniaturisation.

1.2.2 Une brève histoire du temps réel en informatique

En dehors du domaine musical, nous pouvons dater approximativement l'apparition du concept de « temps réel » avec la création de l'*Association for Real Time Systems* en 1963, l'organisation du *Real-Time Meeting* par l'*American Management Association* en 1964, et la publication de plusieurs ouvrages sur le sujet à cette même époque :

- *Real-time Business Systems*, de Robert V. Head, en 1964 ;
- *Real Time Data Processing Systems*, de William Herbert Desmonde, en mars 1965 ;
- *Programming Real Time Computer Systems*, de James Martin, en décembre 1965.

Ce concept a donc émergé au sein des domaines informatiques industriels et financiers dans les années 60, en tant que sujet de recherche informatique, mais aussi comme objectif technologique à conquérir pour des applications cruciales... Aujourd'hui, le temps réel touche effectivement des domaines très variés tels que le pilotage ou la navigation (bateau, avion, train, automobile), le contrôle de processus industriels, la robotique, mais également le domaine des transactions bancaires (où les dates de valeurs doivent être prises en compte) et bien sûr, celui des télécommunications. Mais de quoi parle-t-on exactement ? Peut-on définir de façon univoque ce temps réel aux domaines d'application multiples ?

Le Journal Officiel donne une définition plutôt laconique du temps réel[15], centrée sur l'*immédiateté* :

> Mode de traitement qui permet l'admission des données à un instant quelconque et l'obtention immédiate des résultats.

En revanche, dans leur ouvrage *Introduction aux systèmes temps réel* [Bonnet et Demeure, 1999, p. 18–19], les auteurs rassemblent pas moins de six définitions de référence, énoncées dans un intervalle de trente ans, de 1965 à 1995[16] :

> Définition 1. [Stankovic 88] : En informatique temps réel, le comportement correct d'un système dépend, non seulement des résultats logiques des traitements, mais aussi du temps auquel les résultats sont produits.
>
> Définition 2. [Glass 80] : Un logiciel temps réel est un logiciel qui pilote un ordinateur qui interagit avec des dispositifs ou objets externes en fonctionnement. Il est dit « temps réel » parce que ses actions logicielles dirigent les activités d'un processus en cours d'exécution. Les concepts de temps réel et d'embarqué sont relativement interchangeables si ce n'est pour le fait qu'un système embarqué est inclus dans le système qu'il contrôle.
>
> Définition 3. [Mellishamp 83] : L'informatique temps réel impliquera l'utilisation d'un ordinateur en relation avec un ou plusieurs processus externes. Le but de cette mise en relation sera d'obtenir des informations sur le ou les processus (suivre leur fonctionnement au travers de relevés des variables importantes) et peut-être d'intervenir sur son fonctionnement (en modifiant son comportement en fonction des informations collectées).
>
> Définition 4. [Saller 75] : Le traitement temps réel est le traitement de données qui met en jeu la collecte de données produites à l'extérieur du système et qui ne peuvent pas être reproduites au gré des besoins en respectant des contraintes temps

15. Journal officiel du 22/09/2000 ; source : arrêté du 22 décembre 1981.

16. Ces définitions sont reproduites ici dans l'ordre choisi par les auteurs, différent de l'ordre chronologique.

> réel, ou le traitement de données qui peut répondre en exerçant un contrôle effectif qui produit un résultat physique.
>
> Définition 5. [Martin 65] : Un système informatique temps réel peut être défini comme un système qui contrôle un environnement en recevant des données, en les traitant, et en produisant une action ou des résultats de façon suffisamment rapide pour intervenir dans le fonctionnement du système à ce moment-là.
>
> Définition 6. [Gillies 95] : Un système temps réel est un système dans lequel l'exactitude des applications ne dépend pas seulement de l'exactitude du résultat mais aussi du temps auquel ce résultat est produit. Si les contraintes temporelles de l'application ne sont pas respectées, on parle de défaillance du système. Il est donc essentiel de pouvoir garantir le respect des contraintes temporelles du système. Ceci nécessite que le système permette un taux d'utilisation élevé, tout en respectant les contraintes temporelles identifiées.

Résumons les points saillants de chacune de ces définitions :

1. (Stankovic) spécificité de la *contrainte du temps* de production des résultats ;
2. (Glass) *extériorité* du système en fonctionnement à contrôler ;
3. (Mellishamp) *obtention d'informations* sur des processus externes ;
4. (Saller) *séparation* entre le traitement de données extérieures collectées et le contrôle effectif physique ;
5. (Martin) contrôle d'un environnement grâce à une *rapidité suffisante* ;
6. (Gillies) taux d'utilisation élevé et *respect garanti* des contraintes temporelles.

Ces définitions diffèrent les unes des autres soit en se voulant plus restrictives ou plus larges par rapport aux autres, soit en mettant l'accent sur un aspect en particulier. Bonnet et Demeure, les auteurs du livre, choisissent de retenir la définition de Gillies[17], et ajoutent une remarque importante :

> Contrairement à une idée fausse assez généralement répandue, la vitesse de traitement n'est pas un critère essentiel des systèmes temps réel. Par exemple, sera considéré comme temps réel un système susceptible de fournir des prévisions météo sous 24 heures, si cette limite est respectée quelles que soient ses données initiales (dans un domaine fini).

C'est donc bien la *garantie du respect d'une contrainte temporelle préalablement établie* qui permet essentiellement de définir le temps réel dans le domaine de l'informatique industrielle et des télécommunications, et non, comme le suggère le sens commun, l'immédiateté.

1.2.3 La notion générale de temps réel en art

Qu'en est-il dans le domaine de l'art ? Peut-on adopter la même définition ? Examinons et comparons la définition généraliste du « temps réel » dans le domaine des arts proposée par le *Dictionnaire des arts médiatiques*[18] :

17. « La définition que nous retiendrons (et qui est la plus généralement acceptée) est la dernière. » [Bonnet et Demeure, 1999, p. 19]

TEMPS RÉEL
multimédia, musique, vidéo - (n.m.)

1. Modalité temporelle des systèmes de traitement de l'information dans lesquels il n'y a pas de délai entre la sortie d'informations et l'entrée de données, ou, si l'on veut, dans lesquels l'*output* suit immédiatement l'*input*. Le temps réel est caractéristique du mode interactif.
2. Instantanéité apparente d'une tâche effectuée par un appareil électronique. C'est ce qui se produit, par exemple, avec la contre-réaction, principe bien connu qui consiste à prélever un signal à la sortie d'un circuit électrique et à le renvoyer à l'entrée. Cette opération se passe trop rapidement pour que l'on perçoive l'infime délai qui se produit réellement. Certains traitements ou synthèses sonores informatiques exigent trop de calculs pour être effectués en temps réel. On obtient donc le résultat avec un délai qui peut parfois être très long : c'est le « temps différé ». Mais la puissance actuelle des ordinateurs est telle que de plus en plus de systèmes de synthèse réagissent en « temps réel », ce qui permet au compositeur d'avoir un contrôle perceptif immédiat et continu de son travail.
3. Durée qui est effective au moment de la radiodiffusion ou de la télédiffusion d'une émission.
4. Technique vidéo permettant la correction de l'écart entre le code horaire et le temps réel d'enregistrement (NTSC).

Première observation : les parties 3 et 4 de cette définition du Gram échappent à la définition systémique précédente (Gillies, n° 6) par leur attachement restrictif à un domaine médiatique (radiodiffusion, télédiffusion, ou correction vidéo horaire). Par ailleurs, concernant la vidéo, la partie 3 se rapproche de la notion de *direct*, telle que nous l'avons resituée depuis son ancrage analogique dans les remarques préalables (section 1.1.2 page 8).

Deuxième observation : ces définitions spécifiques aux arts médiatiques diffèrent sensiblement des définitions précédentes, qui concernaient le point de vue de l'ingénieur ou de l'informaticien. Certes, en tant que *« modalité temporelle des systèmes de traitement de l'information »*, la partie 1 de la définition du Gram se rapproche de la définition informatique retenue précédemment (celle de Gillies), mais elle s'en éloigne ensuite avec deux distinctions. D'une part, le caractère d'immédiateté du temps réel (cf. partie 1 *« il n'y a pas de délai entre la sortie d'informations et l'entrée de données »*, et partie 2 *« Instantanéité apparente d'une tâche effectuée par un appareil électronique »*) s'oppose ici avec la remarque sur le caractère secondaire de la vitesse de traitement (*« la vitesse de traitement n'est pas un critère essentiel des systèmes temps réel »*). D'autre part, on peut raisonnablement penser que la mention interactive de la partie 1 (*« le temps réel est caractéristique du mode interactif »*) fait référence à l'interaction homme-machine, alors qu'une seule de nos six définitions informatiques mentionne l'interaction (celle de Glass), mais d'un point de vue non-humain (*« logiciel qui pilote un ordinateur qui interagit avec des dispositifs ou objets externes en fonctionnement »*).

Troisième observation : la partie 2, la plus développée, aborde plusieurs notions connexes essentielles, notamment dans le domaine de la musique. Elle définit le temps différé par opposition au temps réel, du point de vue du traitement et de la synthèse sonores : *« Certains*

18. Élaboré par le Gram – Groupe de recherche en arts médiatiques –, de l'université du Québec à Montréal, en 1996.

traitements ou synthèses sonores informatiques exigent trop de calculs pour être effectués en temps réel. On obtient donc le résultat avec un délai qui peut parfois être très long : c'est le "temps différé". » Nous développerons cette idée, qui s'appuie implicitement sur la perception et le jugement humains, dans les sections suivantes. Cette partie présente une deuxième idée sur laquelle nous reviendrons, celle du contrôle perceptif immédiat et continu : *« la puissance actuelle des ordinateurs est telle que de plus en plus de systèmes de synthèse réagissent en "temps réel", ce qui permet au compositeur d'avoir un contrôle perceptif immédiat et continu de son travail. »* Il s'agit là de l'identification d'un besoin précis qui, une fois atteint, modifie le travail compositionnel en améliorant la vitesse de l'interaction homme-machine visée, ici de la synthèse sonore. Enfin, cette partie introduit l'idée d'une directionnalité dans l'évolution de l'informatique en art : une expansion du temps réel (*i. e.* des processus informatiques calculables sans délai perceptible) grâce à l'augmentation de la puissance des ordinateurs.

Pour résumer, nous pouvons considérer que *l'instantanéité perceptive* et *l'interaction homme-machine* constituent à la fois des différences avec la définition générique systémique de Gillies et des traits caractéristiques essentiels du temps réel tel qu'il est appréhendé dans le domaine des arts.

1.2.4 De l'inévitable souhait du temps réel...

Les souhaits d'immédiateté de l'audition dans la relation avec l'ordinateur, puis de modification en temps réel des résultats de l'ordinateur, se justifiaient parfaitement lorsque naquit l'informatique (donc à partir de la fin de la seconde guerre mondiale), par la présence et l'utilisation déjà très répandues des autres machines – les machines mécaniques, électriques et électroniques – qui, elles, ne fonctionnaient qu'en « temps réel » (si bien que le terme n'existait pas encore !), sans compter les instruments de musique traditionnels. En effet, en 1945, nombre de machines sonores mais analogiques ont déjà vu le jour, beaucoup sont d'ailleurs déjà obsolètes, d'autres occupent le devant de la scène musicale, et toutes fonctionnent de façon interactive, immédiate, souvent tactile (voire sensuelle, comme le theremin[19]), « en temps réel » pourrait-on dire, avant la lettre[20].

19. Cf. l'article de Marc Battier, *L'approche gestuelle dans l'histoire de la lutherie électronique – Etude de cas : le theremin* [Battier, 1999a].

20. Le phonographe date de 1877 (inventé parallèlement par Thomas Edison aux Etats-Unis et Charles Cros en France) ; le premier Telharmonium électrique de l'américain Taddeus Cahill, inventé en 1886, est fabriqué en 1900 ; l'américain Lee de Forest développe son Audio Piano, le premier instrument de musique à tube électronique, en 1915 (après avoir inventé en 1906 l'Audion, la première triode, initiant ainsi l'ère de l'électronique) ; le russe Lev Serguéiévitch Termène invente le Theremin (aussi écrit *Thérémine*, mais d'abord nommé *Etherophone* par son inventeur) en 1919, un synthétiseur qui module la hauteur et l'amplitude de son oscillateur en fonction de la position des mains par rapport à deux antennes (cf. Fig. 1.1 page suivante), utilisé entre autres par Edgar Varèse (*Ecuatorial* pour deux thérémines, voix de basse, vents et percussions, composé en 1934) et Bohuslav Martinů (Fantaisie pour thérémine, hautbois, quatuor à cordes et piano, composée en 1945) ; en France, les Ondes Martenot inventées en 1928 par Maurice Martenot remplacent rapidement le Theremin grâce à leur clavier, et sont utilisées par de nombreux compositeurs dont Messiaen (avec son quatuor d'Ondes Martenot *Oraison*, en 1937, première œuvre d'importance pour ensemble entièrement électronique), Jolivet, Honegger, Varèse, Koechlin, Martinů, Bloch, Boulez, Levinas, Risset, Murail, et bien d'autres (Thomas Bloch, instrumentiste spécialisé, rapporte que plus de 1200 œuvres ont été composées pour les Ondes Martenot). En 1935 déjà, tandis que Colon Nancarrow écrit une des premières œuvres *mixtes*, sa toccata pour violon et piano mécanique à papier

FIG. 1.1 – *Léon Theremin jouant de son instrument* [21]

Il faut encore attendre les années 1956 avec Lejaren Hiller et 1957 avec Max Mathews pour voir, ou plutôt pour entendre, les tout premiers pas de l'informatique musicale, l'un du côté de l'organisation des sons, la composition musicale assistée par ordinateur (CMAO), et l'autre du côté de la construction des sons [22], la synthèse sonore, sur les énormes dinosaures de l'informatique de cette époque ; l'Illiac I (cf. figure 1.2 page ci-contre) utilisé par Hiller mesurait 3 m de hauteur, 60 cm de profondeur et 2,60 m de largeur, pesait 5 tonnes, mais comportait moins de composants que la plupart des montres d'aujourd'hui... Autant dire que le temps réel restait loin de pouvoir figurer à l'ordre du jour à cette époque !

perforé, Laurens Hammond commercialise un synthétiseur qu'il compte vendre à un large public : l'orgue Hammond, qui a reçu le succès qu'on lui connaît. Dès lors, l'invention, la fabrication et la démocratisation des machines acoustiques électroniques ne cessent de progresser et de se diversifier – avec les synthétiseurs, microphones, enregistreurs, haut-parleurs, boîtes à effets, tables de mixages, sans compter les autres appareils de télécommunication.

21. D'après http://www.emfinstitute.emf.org/exhibits/theremin.html ; *« In the photo at the left, Leon Theremin performs in New York in 1927. Courtesy Robert Moog. »*

22. Cette catégorisation binaire reste encore d'actualité, comme l'indique Miller Puckette dans sa préface de *The OM Composer's Book* en 2006 : *« The field of computer music can be thought of as having two fundamental branches, one concerned with the manipulation of musical sounds, and the other concerned with symbolic representations of music. The two are iconized by Max Mathews's MUSIC program and Lejaren Hiller's ILLIAC Suite, both of 1957, although both have important antecedents. The two branches might provisionally be given the names "Computer Generated Music" (Denis Baggi's term for it) and "Computer Aided Composition" – or CGM and CAC for short. »* [Puckette, 2006].

23. D'après http://ems.music.uiuc.edu/history/illiac.html, un article de James Bohn présentant l'histoire de cette machine à l'université de l'Illinois.

FIG. 1.2 – *ILLIAC I* [23]

Ce nonobstant, plusieurs objets technologiques non-informatiques de grande consommation possèdent déjà à la fin des années 50, intrinsèquement, une relation interactive au sonore qui tranche sévèrement avec la lenteur de l'informatique naissante. Le cas musical des synthétiseurs évoqué précédemment s'inscrit dans une évolution technologique plus globale, touchant y compris le grand public et à laquelle participent vivement plusieurs objets/technologies emblématiques ayant un rapport au sonore, dont le téléphone, le disque, et la radio, que nous allons brièvement resituer.

De fait, lorsque Graham Bell dépose le premier brevet de téléphone en 1876 aux Etats-Unis, son appareil connaît immédiatement un destin commercial : 100 000 abonnés dans le monde 10 ans plus tard, 12 millions en 1912 (année du commutateur automatique qui permet de se passer des opératrices), et vers 1956 apparaissent déjà les premiers téléphones mobiles, dont un radiotéléphone pour automobile. À l'exclusion du temps de mise en communication, une fois cette dernière établie, la pratique téléphonique commune relève de l'instantanéité perceptive, au sens où elle permet le dialogue et la compréhension réciproque.

Le principe du disque a été adopté en 1887 par Emile Berliner, dont les premiers Gramophones sont commercialisés dès 1889. Berliner fonde en Allemagne la *Deutsche Grammophon Gesellschaft*, qui vend les premiers disques 78 tours dupliqués par pressage. On découvre en 1901 une méthode économique pour copier en grande série des enregistrements sur cylindres à destination des classes moyennes, mais ce n'est qu'après 1910 que la gravure sur disque a pris le pas sur la gravure sur cylindre. Ébranlée par l'apparition de la radiodiffusion au début des années 20, l'industrie du tourne-disque acoustique sombre complètement au début de la Grande Crise de 1929, avant de réapparaître après la Seconde Guerre mondiale, grâce à une nouvelle ère de prospérité. En 1956, Elvis Presley sort son premier album et Gene Vincent enregistre *Be-Bop-A-Lula*... c'est l'envol du rock'n'roll.

La naissance de la radio date quant à elle de 1895, inventée par Guglielmo Marconi. Elle sert d'abord aux liaisons maritimes, puis aux télécommunications (télégraphes radio), et enfin les années 20 voient naître les premières stations de radio sur tous les continents, avec les premières émissions civiles en 1922 pour la BBC (*British Broadcasting Company*) à Londres, et 1923 pour Radiola et Paris PTT (Postes, Télégraphes et Téléphones) en France, qui fonde d'ailleurs l'Orchestre National de radiodiffusion en 1934. La radio entre rapidement dans tous les foyers, au point de devenir cinq ans plus tard le média privilégié des propagandes d'une Seconde Guerre mondiale surnommée « la Guerre des Ondes ». En 1944, Pierre Schaeffer crée un studio consacré à l'expérimentation radiophonique à Paris – le Studio d'Essai – qui achète son premier magnétophone destiné à la création musicale, en 1950.

Ce rapide panorama du téléphone, du disque et de la radio vise à montrer comment l'usage généralisé des technologies liées à l'audio place le rapport de l'homme à la machine sonore sous le signe de *l'instantanéité*, de l'évidence de l'immédiateté temporelle, depuis la fin du XIX^{e} siècle jusqu'à l'arrivée de l'informatique au milieu du XX^{e} siècle. En effet, le téléphone permet de converser à distance normalement, c'est-à-dire sans délai sensible ou pénalisant ; la manipulation du poste de radio procède par ajustement de la position des boutons (souvent rotatifs) en fonction de ce qu'on entend, pour régler le volume sonore et choisir la fréquence, dans une boucle action/perception typique ; il en va de même pour l'utilisation du tourne-disque, où le son est audible aussitôt que la pointe de la tête de lecture entre en contact avec le disque en rotation. Bref, avec ces appareils, l'instantanéité va de soi.

À la fin des années 50, l'invention de ces appareils remonte à plus d'un demi-siècle, et les pratiques associées relèvent d'une véritable culture de masse depuis plusieurs décennies. Aussi, cette instantanéité propre aux pratiques électriques et électroniques jure avec la lenteur des pratiques de l'informatique naissante : les temps d'attente de l'informatique à ses débuts introduisent une rupture importante dans l'usage de la technologie, où l'instantanéité cède la place à un délai informatique important, systématique et inexorable. Les programmeurs doivent alors faire preuve d'une patience exceptionnelle entre l'introduction des données dans l'ordinateur et l'obtention des résultats (cf. section 1.2.1 page 12), tandis que les studios analogiques s'épanouissent... Cette rupture dans le rapport d'immédiateté entre les machines analogiques et les machines numériques appelle naturellement un souhait proportionnel à la frustration vis-à-vis de l'utilisation de ces dernières : le « temps réel » ; car, comme le confirme Edmond Couchot : « l'importance capitale du temps réel en informatique vient de ce qu'il est la condition nécessaire de toutes les techniques dites interactives ou conversationnelles. » [Couchot, 1985]

1.2.5 . . . vers la conquête du temps réel en musique. . .

Dans le domaine de la musique, l'usage de l'informatique accusait des débuts difficiles, comme toutes les activités de programmation, comme Curtis Roads le rappelle dans son article *L'art de l'articulation : la musique électroacoustique d'Horacio Vaggione* :

> Rares étaient dans les années 1960 les ordinateurs capables de générer du son. Il fallait une persévérance peu ordinaire pour acquérir les compétences nécessaires en matière de programmation ainsi que pour avoir accès aux installations. [Roads, 2007, p. 69]

Avec les progrès informatiques, l'expansion du temps réel en art s'est aussi appliquée en musique : on imagine aisément l'intérêt de pouvoir ouïr sans délai le résultat d'une programmation ou d'un paramétrage, de différentes modifications plus ou moins intempestives selon sa fantaisie, mais, surtout, *selon ce qu'on entend*[24] dans cette boucle *perception/action*[25] caractéristique du temps réel. Apparaît alors un couplage homme-machine plus fort, une nouvelle dimension : la rétroaction (ou *feedback*), dans un sens cybernétique étendu[26], où l'humain fait partie intégrante du système, en tant qu'il modifie le système en fonction de ce qu'il perçoit, dans une échelle temporelle proche de celle de sa propre perception pour assurer une sensation d'immédiateté dans la rétroaction.

Les systèmes informatiques (combinaisons matériel / logiciel) capables d'offrir cette rétroaction relativement rapide ouvrent la voie d'une part à l'expérimentation interactive – l'exploration vive, l'ajustement multiple – du côté de la composition, et d'autre part au jeu sonore et musical avec l'ordinateur réactif pendant une performance, du côté tant de l'interprétation que de l'improvisation, ainsi que des installations sonores interactives. Il s'agit donc bien d'une révolution des pratiques musicales lorsqu'elles utilisent l'informatique, et l'enjeu de celle-ci n'est rien de moins que de donner à la technologie les moyens de ses promesses : en plaçant la « perception rétroactive » – du compositeur ou du musicien – au cœur du processus de production sonore[27]. Concernant cette conséquence décisive de la possibilité technologique d'ouïr dans l'action, Horacio Vaggione précise sa place dans l'acte compositionnel[28] :

---

24. Pierre Schaeffer distingue « l'entendre » de « l'ouïr » dans *Les quatre écoutes* (chapitre VI du *Traité des objets musicaux* [Schaeffer, 1966, p. 116]) : il rapproche « l'ouïr », subjectif et concret, de la *réception* du son, et « l'entendre », subjectif et abstrait, de la *sélection* de certains aspects particuliers du son.

25. Francisco Varela parle aussi, peut-être plus justement, d'une boucle *interprétation*/action : « Ce n'est que dans les plus récents travaux de certains penseurs continentaux (plus particulièrement M. Heidegger, M. Merleau-Ponty et M. Foucault) que la critique explicite de la représentation a commencé. Ces penseurs se préoccupent du phénomène de l'*interprétation* tout entier, dans son sens circulaire de lien entre action et savoir, entre celui qui sait et ce qui est su. Nous nous référons à cette circularité totale de l'action/interprétation par le terme de faire-émerger. [...] il convient d'appeler cette nouvelle approche des STC [Sciences et Technologies de la Cognition] *l'enaction*. » [Varela, 1989, p. 92]

26. La notion cybernétique de *feedback* apparaît au sortir de la Seconde Guerre Mondiale, dès 1948, quand Norbert Wiener publie la première édition de son livre *Cybernetics, or Control and Communication in the Animal and the Machine*.

27. On peut comparer l'impact de cette révolution technologique en musique avec celui des magnétophones, qui, en leur temps, avaient permis à Pierre Schaeffer de remarquer l'avènement de la perception du sonore dans le travail compositionnel.

28. [Barbanti *et al.*, 2004, p. 332-346], traduction française de : *Some Ontological Remarks about Music Composition Processes*, Computer Music Journal, vol. 25 n°1, 2001, p. 54-61.

> Le compositeur en tant qu'auditeur constitue le corrélat du compositeur en tant que producteur : afin de produire de la musique, un acte d'audition est nécessaire, qu'il s'agisse de l'audition « intérieure » (la situation d'écriture silencieuse) de la composition instrumentale pure ou de l'audition « concrète » de la composition musicale électroacoustique. Ces situations impliquent des variants (il en existe plusieurs autres) d'une boucle de *feedback* action/perception, qui peut être définie comme une instance de validation propre aux processus musicaux.

Combien de temps va mettre l'informatique musicale pour accéder au temps réel ? En musique, par rapport à d'autres domaines, l'ontologie informatique du couple temps réel / temps différé se ressent d'autant plus vivement que le temps réel s'y caractérise par une immédiateté « serrée », notamment à cause du taux d'échantillonnage élevé, ce qui complique sa réalisation. Curtis Roads expose de façon détaillée, dans son ouvrage de référence *L'Audionumérique*, à quoi correspond la notion de temps réel en informatique musicale, puis celle de temps différé, en partant de la synthèse par ordinateur [Roads, 1998, p. 69-70] :

> Chaque étape d'un algorithme de synthèse demande un certain temps d'exécution. Pour un algorithme de synthèse compliqué, l'ordinateur ne peut pas toujours achever les calculs nécessaires à un échantillon pendant l'intervalle d'une période d'échantillonnage.
>
> Pour rendre ce point plus concret, examinez ci-dessous les étapes nécessaires au calcul d'un échantillon de son par la méthode de lecture de table.
>
> 1. Ajouter un incrément à la position en cours de lecture de table d'onde pour obtenir une nouvelle position.
> 2. Si la nouvelle position dépasse la fin de la table d'onde, retirer la longueur de table d'onde. En d'autres termes, effectuer une opération de modulo.
> 3. Stocker la nouvelle position pour l'utilisation du calcul du prochain échantillon. Voir étape 1.
> 4. Lire la valeur dans la table d'onde à la nouvelle position.
> 5. Multiplier cette valeur par l'entrée d'amplitude.
> 6. Envoyer le produit à la sortie.
>
> Le point important ici est que chaque étape prend une certaine quantité de temps pour être exécutée. Par exemple, un ordinateur peut prendre une microseconde pour effectuer les calculs ci-dessus. Mais si nous utilisons un d'échantillonnage de 50 000 échantillons par seconde, le temps disponible pour chaque échantillon n'est que de 1 / 50 000 de seconde, soit 20 microsecondes (20 000 nanosecondes). Ceci signifie qu'il est difficile à un ordinateur d'achever les calculs nécessaires pour plus de quelques oscillateurs simples en *temps réel*. Qu'entendons-nous par temps réel ? Dans ce contexte, le temps réel signifie que nous pouvons achever les calculs pour un échantillon en un temps inférieur à une période d'échantillonnage.
>
> Certaines techniques de synthèse et de traitement du signal sont gourmandes en calcul et sont ainsi difficiles à réaliser en temps réel. Ceci signifie qu'il existe un retard d'au moins quelques secondes entre le moment où nous commençons à calculer un son et le moment où nous pouvons l'écouter. Un système avec un tel retard est appelé sytème *différé*.
>
> La synthèse différée était la seule option possible au début de la musique informatique. Par exemple, une portion de deux minutes de l'œuvre de J.K. Randall *« Lyric Variations for Violin and Computer »*, réalisée entre 1965 et 1968 à l'université de Princeton (Cardinal Records VCS 10057), demanda neuf heures de calcul.

> Bien sûr, à la moindre erreur, le processus entier devait être repris à zéro. Bien que cela ait été une technique extrêmement laborieuse, une poignée de compositeurs passionnés qui avait accès au matériel adéquat fut capable de créer des œuvres de musique d'une certaine longueur, entièrement synthétisées par ordinateur.

Les premières avancées en informatique musicale de Hiller et de Mathews en 1956-1957, différentes[29] mais toutes deux qualifiables de « tout différé » à cause du délai informatique considérable à l'époque, sont à comparer avec les productions et les pratiques issues des studios analogiques qualifiables quant à elles de « tout réel », grâce à l'immédiateté de la réponse des appareils de ces studios, alors déjà opérationnels depuis plusieurs années. En effet, les musiques électroacoustiques prennent un tournant décisif au début des années 50 avec l'efflorescence de grands studios : Pierre Schaeffer crée le Club d'essai dès 1944, rejoint par Pierre Henry en 1949, Club qui deviendra successivement le Groupe de Recherche de Musique Concrète en 1951, puis le Groupe de Recherches Musicales en 1958 ; le studio de Cologne est fondé officiellement en 1951 par Werner Meyer-Eppler[30], Robert Beyer et Herbert Eimert, avec la participation de Karlheinz Stockhausen ; en 1955, Luciano Berio et Bruno Maderna fondent le *Studio di fonologia musicale* à la RAI (*Radio Audizioni Italiane*) de Milan. Les appareils électroniques et électroacoustiques de ces studios – microphones, amplificateurs, haut-parleurs, tables de mixage, synthétiseurs, magnétophones, chambres de réverbération et d'écho, et autres dispositifs d'effets – fonctionnent tous dans une relation immédiate, instantanée (même si certains appareils à lampes nécessitent un temps de préchauffage avant de pouvoir être utilisés).

En extrapolant, on pourrait dire que dans les studios analogiques « tout est temps réel », si la notion de temps réel pouvait s'appliquer aux technologies analogiques. *A contrario*, on pourrait dire que dans les laboratoires informatiques de cette époque : « au départ, le temps réel n'existait pas ». Cette différence fondamentale de l'expérience du studio selon sa nature analogique ou numérique a pu générer une tension suffisante pour éveiller des aspirations fortes et orienter les recherches vers ce qui devenait l'objectif du temps réel, comme à l'Ircam :

> Le rêve du musicien « informatique », dans les années 70, c'était une machine capable de générer en même temps des centaines de fréquences sinusoïdales. [...] Aussi, en 1976, le tout nouvel Institut de recherche et coordination acoustique/musique (Ircam) dévoile un prototype : le synthétiseur numérique en temps réel 4A, inventé par Giuseppe Di Giugno, licencié de physique de l'université de Naples. [Roads, 1992, p. 112]

Historiquement, le temps réel a ainsi souvent porté le statut d'accession privilégiée, de victoire technologique ; c'est-à-dire que l'utilisation de l'ordinateur sur scène, au moment même de la performance, constituait à l'époque un véritable défi technologique, comme en témoignent les efforts considérables d'instituts tels que l'Ircam avec la 4A dès 1976 ou du Grm avec le Syter (« *SY*stème *TE*mps *R*éel ») à partir de 1977, développé par Jean-François Allouis :

29. Anne Veitl rappelle la profondeur de cette divergence : « Avec l'utilisation des ordinateurs, l'affrontement entre deux idées opposées de la musique et de la création se radicalisa encore : formaliser toujours plus l'écriture d'une partition ou refonder la musique sur le total sonore ? » [Veitl, 2004]

30. Meyer-Eppler, chercheur au département de phonétique de l'université de Bonn, est l'inventeur de l'expression *« Elektronische Musik »*, en 1949.

> L'originalité [de Syter] résidait dans le fait d'avoir mis en œuvre des procédés de traitement issus des concepts du travail en studio, utilisés depuis les débuts de la musique concrète au GRM, et de les rendre facilement disponibles, sans nécessités d'apprentissage de langages de programmation ou de présence permanente d'un assistant. En d'autres termes l'originalité résidait dans les algorithmes et les interfaces. [...] Ainsi, des interfaces graphiques, aujourd'hui à la base de tout traitement informatique, mais alors méconnues, permettaient de visualiser le son et les paramètres de contrôle ; il y avait même un écran d'interpolation qui permettait d'explorer les territoires intermédiaires entre deux états du traitement. [Teruggi, 2005]

À partir du milieu des années 70, le nom de « temps réel » s'est alors imposé en musique pour désigner ce nouveau domaine de recherche, et de passions aussi parfois, partisanes ou détractrices.

1.2.6 . . . jusqu'à la dichotomie de l'informatique musicale.

Du même coup, la nécessité de baptême de ce nouveau champ attractif en « temps réel » définissait le « temps différé », par opposition logique et fonctionnelle. Ainsi, ces adjectifs « réel » et « différé » ne se suffisent donc pas à eux-même indépendamment l'un de l'autre ; ils se définissent mutuellement par opposition (comme dans la 2e définition du Gram, page 16). Il s'agit alors d'envisager un couple inséparable au sein de l'informatique, représenté sur la figure 1.3.

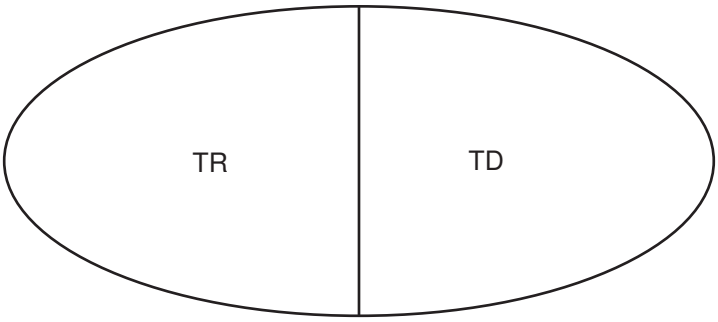


Fig. 1.3 – *Deux catégories informatiques dichotomiques*

Le champ de l'informatique se divise ainsi en deux, selon un critère d'instantanéité perceptive, ou de son envers, le *temps de latence*, défini comme le « délai entre la sortie d'information et l'entrée de données » (cf. définition du Gram, section 1.2.3 page 15) ; cette définition cybernétique de la latence dérive sans doute de l'acception psychologique de la latence : « Intervalle qui sépare le stimulus et la réponse au stimulus[31] ». Afin d'envisager une perspective nouvelle des notions de temps réel et de temps différé dans l'informatique musicale, la théorie naïve des ensembles devrait permettre de présenter assez intuitivement la situation actuelle et ses évolutions.

31. Définition extraite du dictionnaire *Le Trésor de la Langue Française* du Centre National de Ressources Textuelles et Lexicales, créé par le Cnrs.

Définition (Temps réel et temps différé technologiques)**.** Soit $\mathbb{L}$ l'ensemble des logiciels d'informatique musicale, TR et TD deux sous-ensembles de $\mathbb{L}$, t la fonction qui associe à un logiciel x son temps de latence caractéristique $t(x)$, et s la constante du seuil de latence maximum admis, en-dessous duquel un logiciel peut être considéré comme appartenant à l'ensemble TR des logiciels temps réel.

$$\boxed{\text{Soit } t : \left| \begin{array}{l} \mathbb{L} \longrightarrow \mathbb{R}^+ \\ x \longmapsto t(x) \end{array} \right. \text{, alors } \left\{ \begin{array}{l} TR = \{x \in \mathbb{L} \mid t(x) \leqslant s\} \\ TD = \{y \in \mathbb{L} \mid t(y) > s\} \end{array} \right.} \tag{1.1}$$

A partir de cette définition technologique minimale du temps réel et du temps différé, il apparaît d'une part que c'est le seuil de latence s qui permet de déterminer les ensembles TR et TD, et il apparaît d'autre part que ces ensembles sont complémentaires dans $\mathbb{L}$.

Propriété (Complémentarité)**.** Rappelons que deux ensembles sont dits *complémentaires* si et seulement si ils sont disjoints entre eux et que leur réunion vaut exactement l'ensemble $\mathbb{L}$. Or, on peut déduire des équations 1.1 que TR et TD sont disjoints et que leur réunion vaut exactement $\mathbb{L}$.

$$\boxed{\begin{array}{l} \text{Soit les trois ensembles } \mathbb{L}, TR \text{ et } TD \text{ tels que } TR \subset \mathbb{L}, TD \subset \mathbb{L}. \\ \\ \left\{ \begin{array}{l} TR \cap TD = \emptyset \\ TR \cup TD = \mathbb{L} \end{array} \right. \Rightarrow \left\{ \begin{array}{l} TR = \complement_{\mathbb{L}} TD \\ TD = \complement_{\mathbb{L}} TR \end{array} \right. \end{array}} \tag{1.2}$$

Dans ce modèle ensembliste, la latence est à la fois ce qui sépare et ce qui fonde ces deux catégories complémentaires, mais il reste à définir quantitativement le seuil de latence s, car il n'y a pas encore véritablement de consensus sur sa valeur...

1.2.7 Un seuil relatif à la perception

La difficulté majeure pour s'accorder sur une valeur précise du seuil de latence tient au fait que ce seuil est relatif à la perception. Or, lorsqu'il s'agit de perception, les choses apparemment simples ou banales se révèlent souvent étonnamment compliquées lorsqu'on tente de les étudier. D'une part, nos connaissances sur le fonctionnement du cerveau restent encore largement lacunaires, et d'autre part, les résultats qui concernent la perception varient fréquemment d'une recherche à l'autre selon la population étudiée, le protocole expérimental, les définitions du sujet de recherche et d'autres contingences qui échappent à la mesure. Ces difficultés d'étude et de consensus dans le champ de la perception se vérifient à propos du seuil de latence, qui, à notre connaissance, n'a d'ailleurs pas été directement étudié en tant que « critère de séparation quantitatif entre le temps réel et le temps différé ».

Cependant, la latence a fait directement ou indirectement l'objet de nombreuses études de psychologie expérimentale, de psychoacoustique, d'acoustique musicale, de conception d'interfaces homme-machine ou encore de neurosciences, avec des résultats souvent différents et contestés par articles interposés. En effet, ce domaine de recherche est actif depuis fort longtemps – Henri Piéron a publié par exemple un article intitulé *Recherches sur les*

lois de variation des temps de latence sensorielle en fonction des intensités excitatrices dans *La revue psychologique* en 1923 – mais les sociétés téléphoniques s'intéressaient déjà à ce sujet depuis la fin du XIX^e^ siècle ; par ailleurs, les progrès des appareils de mesure et des sciences expérimentales ont régulièrement renouvelé l'intérêt des études sur la perception du temps jusqu'à aujourd'hui.

Quelques résultats actuels

D'une manière générale, les chiffres avancés dans les différentes études sont compris entre 1 ms et 100 ms (parfois jusqu'à 250 ms), ce qui donne une première idée des échelles de temps concernées. Mais plus précisément, dans le domaine psychoacoustique, Stephen McAdams évoque un intervalle de 30 ms à 50 ms pour le seuil de fusion et de ségrégation perceptive de deux sources acoustiques :

> Les sons provenant de sources indépendantes commencent et s'arrêtent rarement en même temps. Une fois dépassé un seuil limite de décalage d'attaque (à peu près 30 à 50 ms), l'oreille détecte la présence de deux sources. Cette régularité correspond au principe du destin commun de la théorie de la *Gestalt*. [...] Les composantes qui commencent avec un décalage de moins de 30 ms ont tendance à fusionner perceptivement en un seul événement auditif (Rasch, 1978). Lorque ce décalage dépasse 50-60 ms, il en résulte souvent une perception de deux sources distinctes. Même dans un son complexe harmonique continu, si un harmonique s'arrête et recommence, il est perçu séparément comme un son pur superposé sur un son complexe. [McAdams, 1994, p. 314–315]

En fait, le seuil de latence au-delà duquel une activité interactive ne peut plus avoir lieu dépend largement des modalités de cette activité, de sa nature et de la quantité des sens impliqués dans l'interaction. Dans leur article *The Quest for Low Latency*, Lago et Kon pointent avec ironie l'obsession de la latence la plus faible possible, pour tenter de revenir à une définition qui prend en compte les véritables besoins par rapport à une utilisation particulière d'un système :

> *Low latency processing is usually a goal in real-time audio applications ; however, it is not clear how little latency is to be considered low enough. [...] So, in order to assess the quality of an interactive system regarding its latency and jitter*[32] *characteristics, we need to understand their effects on the user's perception so that we can define maximum acceptable values for these parameters on such system. The acceptable limits for latency and jitter on an interactive system may vary a lot.* [Lago et Kon, 2004]

Ils rappellent quelques valeurs pour indiquer les ordres de grandeurs en jeu dans les applications *musicales*, à partir de plusieurs résultats de recherche en psychoacoustique :

– la perception stéréophonique peut descendre jusqu'à 20 μs dans les déviations temporelles interaurales pour analyser le positionnement spatial (Pierce[33]) ;

32. Le grand dictionnaire terminologique de l'OQLF donne pour *jitter* les traductions suivantes : « scintillement », « fluctuation » ou « instabilité » (d'un signal). Il s'agirait ici plutôt de l'instabilité.

33. Pierce, J. (1999). Hearing in time and space. In P. Cook (Ed.), *Music, Cognition, and Computerized sound : an Introduction to Psychoacoustics*, pp. 89–103. Cambridge : MIT Press.

- la variation dans nos battements d'une pulsation régulière avec le bout des doigts peut descendre jusqu'à 4 ms (Rubine et McAvinney[34]) ;
- la détection consciente de variations temporelles dans une séquence de pulsations entendues a lieu vers 6 ms (Friberg et Sundberg[35]) ;
- la perception du délai entre une action (manuelle) et sa réaction (sonore) peut descendre jusqu'à 20 ms, par une compensation cependant non-consciente (Wing[36]) ;
- des asynchronies de 50 ms sont tout à fait courantes pour des notes supposées simultanées dans un contexte musical, même en musique de chambre (Rasch[37]) ;
- le simple fait de placer les haut-parleurs à 3 ou 4 mètres ajoute une latence de 10 ms (étant donnée la célérité du son), et l'éloignement relatif de 10 mètres entre les sections d'un orchestre (par exemple entre les cors et les violons) induit une latence de 30 ms pour le public ;
- au piano, le temps qui s'écoule entre l'enfoncement d'une touche et l'attaque effective varie notablement : de 30 ms pour une note jouée *staccato* et *forte*, jusqu'à 100 ms pour une note jouée *piano* (Askenfelt et Jansson[38]).

Finalement, les auteurs proposent de s'en tenir à une latence minimum comprise entre 20 et 30 ms pour les applications musicales :

> *We hope we have been able to argue convincingly that somewhat large latencies, maybe up to 20–30ms, are pretty much acceptable for most multimedia and music applications.* [Lago et Kon, 2004]

Il faut préciser plusieurs points. D'une part, les possibilités de synchronisation sont variables selon le *type de son*, en particulier la morphologie de l'attaque (les seuils diffèrent sensiblement entre un tuba et un glockenspiel...) ; à ce titre, l'expérience la plus exigeante reste sans doute celle du marteau[39]). D'autre part, les résultats perceptifs sont toujours très variables *selon les individus*[40].

34. Rubine, D. and P. McAvinney (1990). Programmable finger-tracking instrument controllers. *Computer Music Journal* 14(1), 26–40.

35. Friberg, A. and J. Sundberg (1995). Time discrimination in a monotonic, isochronous sequence. *Journal of the Acoustical Society of America* 98(5), 2524–2531.

36. Wing, A. M. (1977). Perturbations of auditory feedback delay and the timing of movement. *Journal of Experimental Psychology : Human Perc. and Performance* 3(2), 175–186.

37. Rasch, R. A. (1979). Synchronization in performed ensemble music. *Acustica* 43, 121–131.

38. Askenfelt, A. and E. V. Jansson (1990). From touch to string vibrations. I : Timing in the grand piano action. *Journal of the Acoustical Society of America* 88(1), 52–63.

39. Lire par exemple l'article *Sensitivity to haptic-audio asynchrony* [Adelstein *et al.*, 2003] : *« This paper describes a psychophysical study of detectable time delay between a voluntary hammer tap and its auditory consequence (a percussive sound of either 1, 50, or 200 ms duration). The results show Just Noticeable Differences (JNDs) for temporal asynchrony of 24 ms with insignificant response bias. »*

40. Dans la même étude avec le marteau, les auteurs rapportent par exemple des différences importantes entre les sujets. Ils suggèrent d'ailleurs de ne prendre en considération que les minima (de 5 à 8 ms) dans le développement des interfaces « haptiques » : *« While the average JNDs measured are relatively small, one participant in particular had 5-8 ms JNDs with 75% thresholds of only 8-10 ms. As a practical consideration for the design of multimodal haptic-auditory displays and auditory enhancements to haptic interfaces, the data for this individual observer suggest synchronization requirements that may undercut those for unimodal tactile temporal separation (10-30 ms) and begin to approach those for auditory fusion (1-2 ms) (Gescheider, 1967). »* [Adelstein *et al.*, 2003].

Ponctualité ou épaisseur ?

Fixer de façon autoritaire le seuil de latence s à une valeur unique permet d'utiliser le modèle ensembliste le plus simplement possible, par exemple pour étudier l'histoire du processus de la migration logicielle vers le temps réel. Néanmoins, le fait que la plupart des valeurs numériques concernant la latence sont indiquées sous la forme d'un intervalle semble nous enjoindre de considérer un intervalle de type $]s_{min}; s_{max}]$ au lieu d'une valeur unique. Il existe d'ailleurs un parallèle topologique entre le problème de la ponctualité ou de l'épaisseur du seuil de latence et celui de la ponctualité ou de l'épaisseur de la notion du présent [41]. Ainsi, l'idée d'une frontière floue, qui entame du point de vue de la définition dichotomique traditionnelle le cloisonnement entre temps réel et temps différé, pourrait mieux correspondre à la réalité qu'un seuil fixé arbitrairement. Or, prendre en compte un intervalle pour appréhender la latence fait apparaître une nouvelle catégorie-limite ou catégorie-frontière à interroger.

Si nous appelons TI cette troisième catégorie (pour *Temps Intermédiaire*), que nous conservons le formalisme ensembliste précédent, et que nous reprenons les conclusions de Lago et Kon sur ce que peut être une latence raisonnable pour les applications musicales, alors cette nouvelle catégorie s'écrit de la façon suivante : $TI = \{x \in \mathbb{L} \mid 20\,ms < t(x) \leqslant 30\,ms\}$. Les valeurs des deux bornes peuvent évidemment être remplacées par d'autres en fonction des besoins, car il s'agit seulement d'un principe de substitution du seuil s à valeur unique par un intervalle de la forme de TI ($]s_{min}; s_{max}]$).

Ainsi, l'épaisseur du seuil de latence permet sans doute de désigner à travers la catégorie TI soit des logiciels à vocation temps réel mais faillibles – c'est-à-dire dont le temps de réaction ou de calcul pourrait parfois être perçu comme gênant lors de leur utilisation –, soit des logiciels à vocation temps différé avec un potentiel interactif important, éventuellement en passe de devenir des logiciels temps réel. On pressent que cette catégorie intermédiaire ne peut caractériser durablement que relativement peu de logiciels, et constituerait davantage une étape provisoire dans l'accession de certains logiciels au temps réel. Autrement dit, cette nouvelle catégorie ne devrait pas remettre en cause la dichotomie des deux catégories principales (le deuxième chapitre approfondira cette question de la séparation entre temps réel et temps différé).

En tout état de cause, c'est l'usage du système qui peut valider ou invalider une valeur précise (ou un intervalle précis), en tenant compte de l'ensemble du contexte d'utilisation (types de sons, périodicité, acoustique de la salle, individus, etc.) ; les « usagers » du système restent les meilleurs juges de l'acceptabilité concernant le temps de latence global.

* * *

De cette étude sur les définitions du temps réel et du temps différé en informatique en général, puis plus particulièrement en art et en musique, se dégagent plusieurs points :

41. « Etant essentiellement continu, le temps peut être divisé – comme Aristote l'avait déjà noté – entre l'avant et l'après, mais l'"instant" qui les sépare ne peut davantage être un morceau de temps qu'un point ne peut être confondu avec un segment de droite. [...] Topologiquement, les instruments d'une partition comme les instants et les points ne peuvent être de même nature que ce qu'ils divisent. En bref, si le présent est pour nous un morceau mobile de temps, il ne peut être qu'une espèce de présent "épais" » [Lestienne, 2007, p. 233]

- l'apparition de la notion de temps réel au milieu des années 60 dans les domaines industriels et financiers ;
- l'importance de la garantie du respect d'une contrainte temporelle préalablement établie, pour la définition générale du temps réel en informatique ;
- l'importance de l'instantanéité perceptive et de l'interaction homme-machine, pour la définition du temps réel en art ;
- la rupture de l'immédiateté provoquée par les ordinateurs aux débuts de l'informatique par rapport aux machines analogiques dans années 50 (dont le téléphone, le disque ou la radio) ;
- la tension entre l'expérience du studio analogique et celle de la lenteur du studio numérique, jusqu'à l'apparition des premiers systèmes musicaux temps réel au milieu des années 70 ;
- la dichotomie résultante du temps réel et temps différé, qui pose le problème de la définition quantitative du seuil de latence ;
- les difficultés intrinsèques à la définition du seuil de latence lorsqu'il est relatif à la perception.

1.3 Glissement sémantique vers la fonction musicale

Philippe Manoury rappelait dans un article récent – *Considérations (toujours actuelles) sur l'état de la musique en temps réel* [Manoury, 2007, p. 6] – que « ces notions de temps réel et de temps différé ne sont pas une chasse gardée de la technologie informatique, mais appartiennent aussi à la pratique musicale traditionnelle. »

En effet, le couple temps réel / temps différé implique les activités de toute une population de gens concernés par la musique et par la composition dans leur rapport avec l'ordinateur, dont essentiellement des musiciens, des compositeurs, des chercheurs, et des informaticiens. Ici, le progrès technologique participe directement de l'humain, et, pire – si l'on peut dire –, de l'esthétique. Incidemment, Jean-Claude Risset explicite la façon dont l'art et la science peuvent se nourrir mutuellement, en particulier en musique [Risset, 1998].

1.3.1 Deux univers méthodologiques

Les activités humaines concernées par l'informatique musicale ont donc rapidement connu un clivage à partir de ce couple catégoriel, selon les centres d'intérêts de chacun. Certains ont pris des positions radicales pour ou contre le temps réel ou le temps différé : et c'est à ce moment-là que s'est opéré le glissement sémantique vers la fonction musicale, à partir des sens technologiques vus précédemment. Il semblerait que pour toute une période (et peut-être toujours aujourd'hui), le « temps réel » et le « temps différé » ne se sont pas adressés aux mêmes personnes, mais pas non plus aux mêmes pensées de la musique, comme nous le verrons dans les sections suivantes.

Chacune de ces deux catégories acquît une certaine autonomie vis-à-vis de l'autre au contact des diverses pratiques musicales et compositionnelles, impliquant davantage deux univers tout à la fois technologiques et esthétiques plutôt que seulement technologiques.

C'est-à-dire qu'une fois que les progrès technologiques ont permis à l'informatique musicale de développer des systèmes temps réel, l'usage des expressions *temps réel* et *temps différé* a subit un glissement métonymique où la caractéristique de l'outil remplace l'outil lui-même (le système informatique) et la pratique de cet outil (typiquement la performance musicale ou bien la composition, cf. chapitre 5 page 103).

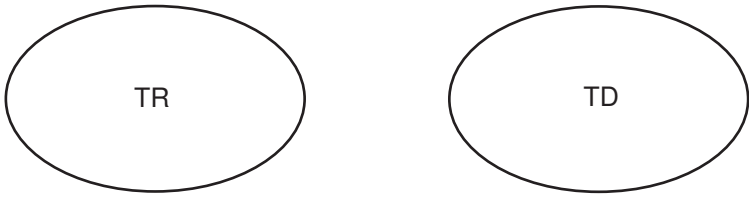


Fig. 1.4 – *Deux mondes autonomes et distincts*

On peut supposer que ce glissement s'est opéré à partir de la différence de méthodologie impliquée par la pratique d'une technologie plutôt que de l'autre, car ces deux méthodologies répondaient à des schémas traditionnellement séparés, à savoir le modèle de *l'instrumentiste* contre celui du *compositeur* :

> Traditionnellement, la composition est une activité en temps « différé ». C'est l'interprète qui inscrira dans le temps la partition que le compositeur écrit « hors-temps », suivant l'expression d'Iannis Xenakis. La partition, code d'actions et représentation « idéale » ou « virtuelle », permet au compositeur de préparer à loisir un voyage dans le temps sans être assujetti au temps de la musique. Le « temps réel » est celui de l'interprétation – ou de l'improvisation : mais cette dernière activité repose sur une longue accumulation mémorielle de figures musicales associée à des réflexes gestuels. [Risset, 1999b, p. 142]

De ce point de vue, les catégories du temps réel et du temps différé acquièrent une certaine autonomie l'une vis-à-vis de l'autre, en tant qu'adhésion à un mode opératoire plutôt qu'à l'autre, selon les oppositions classiques résumées dans le tableau 1.1.

Temps réel	*Temps différé*
modèle de l'instrumentiste	modèle du compositeur
interprétation ou improvisation	partition
éphémère	pérennité
signal	code

Tab. 1.1 – *Oppositions classiques entre temps réel et temps différé en musique*

1.3.2 Le temps réel comme passerelle vers l'instrumental

Andrew Gerzso, dans le livret qui accompagne le CD « Boulez : *Répons*[42] » avance certaines vertues du temps réel, telles qu'elles ont pu intéresser Pierre Boulez :

42. Pierre Boulez : *Répons* (1981-1984), *Dialogue de l'ombre double* (1985) ; Alain Damiens, Ensemble Intercontemporain, Pierre Boulez, enregistré en 1996 à Paris. Deutsche Grammophon 1998. Réalisation électro-acoustique : Andrew Gerzso.

> Dès les années 50, Boulez avait expérimenté dans le domaine électro-acoustique, mais l'aspect figé d'une musique « préfabriquée », qui rendait l'interprète prisonnier de la technique, ne l'avait jamais vraiment satisfait. Avec l'IRCAM, où tout ou presque semblait désormais réalisable, le compositeur put s'aventurer au-delà des sonorités instrumentales traditionnelles. Grâce à l'électronique en temps réel, qui permet d'associer en toute liberté sons synthétiques et sons instrumentaux, il put créer de nouveaux timbres et distribuer les sons dans l'espace à son gré. Boulez imagine ainsi une « musique spatiale », donnant naissance à des dimensions et des couleurs inédites, et c'est ce qu'illustrent merveilleusement *Répons* et *Dialogue de l'ombre double*. Quiconque associe à l'électronique froideur et dénaturation changera de point de vue en écoutant ces œuvres : on ne peut qu'être fasciné par leur fantaisie, leur spontanéité et leur richesse sonore.
>
> [...] Si Boulez a utilisé des nouvelles technologies dans *Répons*, c'est qu'il souhaitait conserver une cohérence entre le monde instrumental et l'électronique : il avait été fréquemment gêné, durant les années 70, par les premières tentatives de mélanger les sons des instruments avec ceux produits par l'ordinateur dans la mesure où ceux-ci étaient restitués par une bande magnétique pendant l'exécution de l'œuvre. Les instrumentistes devenant de facto prisonniers du temps mécanique de la bande magnétique, toute la vie et la mobilité essentielles à l'exécution musicale étaient perdues. D'où l'intérêt de Boulez pour la technologie du temps réel, développée à l'Ircam à la fin des années 70 et au début des années 80, qui permit à l'exécution musicale de reprendre ses droits dans le monde de la musique infomatique. Dès 1980, au moment où le travail sur *Répons* a commencé, il était possible de transformer le son d'un instrument en temps réel pendant l'exécution musicale. Sur le plan musical, Boulez utilise les transformations du son produites par l'ordinateur comme un moyen de prolonger une idée musicale du monde instrumental dans le monde électronique. Ainsi, la cohérence entre les deux mondes est préservée et le vocabulaire musical globalement élargi.

Andrew Gerzso semble décrire ici, non sans un certain enthousiasme, le temps réel comme une révolution salutaire dans l'histoire de la musique électroacoustique, permettant de dépasser « l'aspect figé d'une musique "préfabriquée" », « d'associer en toute liberté sons synthétiques et sons instrumentaux », d'imaginer une « musique spaciale », et enfin « de prolonger une idée musicale du monde instrumental dans le monde électronique », permettant ainsi de préserver « la cohérence entre les deux mondes ». Au-delà de cette quête de la cohérence entre les deux mondes, de l'instrumental à l'électronique, Jean-Claude Risset expose le lien entre l'avènement du temps réel et la place du monde instrumental dans la musique électroacoustique [Risset, 1999a] :

> Le temps réel réintroduit l'interprète dans la musique électroacoustique : il ouvre de nouvelles possibilités d'interactivité. Déjà l'électronique avait permis l'apparition de nouveaux instruments, mais ces instruments supposaient des interprètes virtuoses. Avec l'informatique, la combinatoire et la présence de mémoires permettent de relayer l'instrumentiste dans ses fonctions de déclenchement et de commande, et donc d'instaurer des modes plus complexes de contrôle en temps réel du rendu musical.

D'abord, on voit ainsi comment le temps réel technologique sert de passerelle entre le monde instrumental et le monde électronique.

Ensuite, ces deux citations (datant de la fin des années 90) montrent nettement un sens plus large de l'expression « temps réel » que le strict sens technologique : dans la

proposition « Le temps réel réintroduit l'interprète dans la musique électroacoustique », la notion historiquement technologique est promue au rang de catégorie, à la fois pratique et musicale. Il ne s'agit plus ici seulement de la puissance de calcul audionumérique, mais bien d'une abstraction qui désigne une pratique musicale (entre un interprète et un système informatique), c'est-à-dire l'usage de l'outil au lieu de l'outil, et plus précisément sa *fonction musicale*.

1.3.3 Le temps différé comme extension de la composition

Le « temps différé » n'est pas « le différé »

Commençons par étudier ce que ne désigne pas l'expression « temps différé », et pour faire référence au sens commun le plus large, prenons la définition du « différé » proposée par une version récente (2005) du dictionnaire Larousse :

> DIFFÉRÉ, E adj. et n.m. Se dit d'un programme radiophonique ou télévisé enregistré avant sa diffusion (par oppos. à *en direct*) *Match retransmis en léger différé.* SYN. : *préenregistré.*

En musique, l'expression « temps différé » ne désigne ni un programme radiophonique, ni un programme télévisé. Néanmoins, le synonyme « préenregistré » indiqué par le Larousse présente un intérêt sur lequel nous reviendrons. Ajoutons, pour abonder dans ce sens, que « différé » se traduit habituellement en anglais par *« recorded »* ou *« pre-recorded »*, sans confusion avec *« deffered time »* pour « temps différé » (parfois aussi *« asynchronous »*), contrairement à la confusion polysémique du mot français. Par ailleurs, pour faire référence à l'opposition sémantique avec « temps réel », nous pouvons indiquer que « temps différé » ne désigne rien d'irréel ni de factice, car il possède une existence dans la réalité au moins au titre de catégorie des logiciels dont le temps de latence est supérieur au seuil s, dans sa définition technologique minimale que nous en avons donné précédemment (définition 1.1 page 25).

Ainsi, ni radiophonique, ni télévisuelle, ni non plus irréelle, la notion de temps différé en musique s'appuie sur la pratique *compositionnelle*, à la fois comme métaphore et comme extension de la composition. Schématiquement, comme nous allons le voir par la suite, les pionniers de l'informatique musicale explorent à partir du milieu des années 50 deux voies différentes du potentiel informatique en musique :

- *composer les notes*, du côté de la CMAO ;
- *composer le son*, du côté de la synthèse sonore.

La CMAO comme activité compositionnelle

Avec son quatuor à corde *Illiac Suite*, Lejaren Hiller ouvre avec Leonard Isaacson[43] dès 1956 la voie de la composition musicale assistée par ordinateur au département de chimie

43. Ils publieront un livre en 1959 pour détailler les procédures du développement des quatres mouvements du quatuor : *Experimental Music : Composition with an Electronic Computer*, aux éditions McGraw-Hill.

de l'université de l'Illinois, sur l'Illiac (ILLInois Automatic Computer), machine faisant partie des tout premiers ordinateurs, issue de la machine IAS de John von Neumann. Pour assister les compositeurs dans leur tâche, l'ordinateur a certainement toute sa place en tant que machine à calculer ; après tout, les dés, les cartes, le nombre d'or, et d'autres outils plus ou moins directement numériques comme le I Ching[44] les accompagnent déjà depuis longtemps ! Le recul historique nous permet toutefois de modérer nos attentes face à l'outil informatique, comme le résume Luc Rondeleux dans son article *Quarante années de représentations numériques au service de la création* [Rondeleux, 1997] :

> Dans cet art technologique nouveau, une connaissance auparavant intuitive devient forcément explicite, même si cette explication ne recouvre pas et ne recouvrira jamais l'ensemble des données de l'art. Les premières déconvenues de l'informatique musicale, précédant de peu celles de la synthèse, proviennent d'une assimilation hâtive entre composition et suite de procédures calculables. Hiller et Isaacson, précurseurs du mythe de la machine à composer s'appuyaient sur des règles ; ils furent souvent imités. Mais la musique n'a pas de caractère déductif. Aux États-Unis l'utilisation du sérialisme par Milton Babbitt, les compositions de Charles Dodge, Paul Lansky, James Tenney, en France la transposition des modèles de la théorie cinétique des gaz par Iannis Xenakis, les oeuvres de Pierre Barbaud, Frank Brown et Geneviève Klein (qui opéraient sous l'acronyme de BBK), celles de Nicole Lachartre, la machine imaginaire (directement calquée des processus informatisables) de Michel Philippot, ou, plus près de nous, les oeuvres d'André Riotte n'ont pas trouvé la théorie universelle. Mais la cherchaient-ils ? Sans doute n'ont-ils jamais eu cette naïveté. Ce dont ont besoin les compositeurs, ce n'est pas d'une machine à composer mais d'un outil qui les guide pour l'exploration du sonore, qui inspire les investigations et soutienne l'intuition musicale.

Bien que le terme ne soit pas très heureux, la fonction « d'assistance » à la composition prodiguée par l'ordinateur s'illustre par ses aptitudes inédites au calcul numérique au sens large. La CMAO prolonge et renforce ainsi le lien intime entre composition et mathématiques, d'une part en accélérant les calculs arithmétiques et logiques, et d'autre part en proposant de nouveaux formalismes et de nouveaux algorithmes, jusqu'aux développements de l'intelligence artificielle.

La synthèse sonore comme activité compositionnelle

De son côté, Max Mathews, ingénieur électronicien aux *Bell Telephone Laboratories*, inaugure en 1957 avec ses collègues la synthèse sonore par ordinateur, en développant le programme Music I sur un IBM 704. En 1997, dans son exposé pour la conférence *Horizons in Computer Music*[45], il revient sur ce moment historique et souligne la rupture technologique qu'il constitue :

44. Il est amusant de remarquer que cette technique ancestrale, utilisée entre autres par John Cage, repose sur 8 symboles (les trigrammes) qui forment une matrice de 64 cases, tant on rencontre fréquemment ces nombres en informatique qui fonctionne en base 2, avec l'octet de 8 bits comme unité et des vecteurs de 64 bits (pour les derniers processeurs personnels AMD ou Intel par exemple).

45. Cette conférence s'est tenue les 8 et 9 mars 1997, au *Simon Recital Center of the School of Music* de l'université de l'Indiana, à Bloomington. http://www.csounds.com/mathews/.

> *Computer performance of music was born in 1957 when an IBM 704 in NYC played a 17 second composition on the Music I program which I wrote. The timbres and notes were not inspiring, but the technical breakthrough is still reverberating. Music I led me to Music II through V. A host of others wrote Music 10, Music 360, Music 15, Csound, Cmix, and SuperCollider. Many exciting pieces are now performed digitally.*

La fin des années 50 a ainsi ouvert la voie à deux branches fondatrices de l'informatique musicale : la composition musicale assistée par ordinateur et la synthèse sonore, avec de nombreux logiciels plutôt dédiés à l'une, comme PatchWork, ou plutôt à l'autre, comme la série Music N. Au delà du fait que ces branches se rejoignent parfois au sein de certains logiciels, elles constituent le socle historique – et surtout le socle *pratique* – de ce qu'on appelle aujourd'hui en informatique musicale « le temps différé ». En effet, les progrès de la synthèse vers le temps réel, avec le contrôle des paramètres par des interfaces plus ou moins complexes, n'ont pas évincé la synthèse en temps différé ni la CMAO qui restent des pratiques compositionnelles, et ne le peuvent intrinsèquement pas, car la pratique de la composition implique la possibilité du retour...

1.3.4 La propriété décisive du retour compositionnel

À notre sens, voilà bien l'enjeu du temps différé, interdit au temps réel : *le retour compositionnel*. Les outils de la composition doivent permettre d'écrire, d'enregistrer et surtout d'*éditer* ce qui a été enregistré, c'est-à-dire d'effacer, de réécrire et d'ajouter ; ceci valait bien sûr pour les outils traditionnels comme le crayon, la gomme et la partition, et vaut toujours pour les outils informatiques. Or le temps réel, en musique, ne permet pas ce retour, par définition, pris dans l'irréversibilité du jeu ; tout au plus pourrait-il permettre d'écrire et d'enregistrer, mais en aucun cas d'éditer. L'édition, finalement une forme de l'écriture dans le domaine numérique, a besoin d'une distanciation temporelle, comme toute écriture vouée à être interprétée :

> [...] dans le domaine de la création musicale, cette évolution technologique [vers le temps réel] a rencontré pendant une période relativement longue, une opposition d'ordre méthodologique. Le *temps réel* oblige l'opérateur à agir et réagir en fonction du résultat adoptant ainsi un comportement proche de l'instrumentiste. Pour beaucoup de compositeurs le temps différé en séparant le moment de la conception de celui de l'écoute créait une distance nécessaire à la réflexion, situation similaire à l'écart temporel existant dans la composition instrumentale entre l'écriture sur papier et l'exécution. [Teruggi, 2005]

En outre, le synonyme « préenregistré » proposé précédemment par le Larousse prend dans le domaine numérique une valeur inédite par rapport à l'écriture : nous avons montré au début de cette thèse que le couple temps réel / temps différé relève d'une ontologie informatique, or l'informatique a ceci de particulier qu'elle enregistre les données de façon absolument fidèle – autrement dit qu'il n'y a pas de différence entre l'original et la copie. Dans ces conditions, le retour compositionnel transposé dans le domaine informatique acquière des propriétés plus poussées que dans le domaine analogique, avec d'abord une édition infiniment plus précise. De plus, comme le fait remarquer Horacio Vaggione, l'informatique permet d'enregistrer non seulement le résultat des opérations mais aussi les

opérations elles-mêmes, ce qui propulse d'un point de vue pratique le retour compositionnel à un niveau d'abstraction inédit.

Ainsi, si nous choisissons de caractériser le temps différé non plus par son incapacité à fournir le résultat d'un calcul instantanément, mais par sa *possibilité de retour* compositionnel, la notion technologique glisse elle aussi, comme le temps réel, vers sa fonction musicale : une extension de l'activité compositionnelle.

* * *

Après l'étude de la section précédente sur la dichotomie historique entre temps réel et temps différé dans différents domaines, cette section a montré plus particulièrement qu'en musique ces deux notions ont subi un glissement sémantique vers leur fonction musicale. Alors, en musique, si le temps réel peut être caractérisé par le modèle de l'instrumentiste et de son rapport quasi immédiat au son, le temps différé quant à lui peut être caractérisé positivement par le modèle du compositeur et de la possibilité du retour compositionnel.

Ces deux caractérisations positives éloignent déjà le couple temps réel / temps différé de la représentation strictement dichotomique, vers une représentation de deux mondes autonomes. Les deux chapitres suivants approfondiront et développeront cette problématique de la représentation du temps réel et du temps différé, vers une troisième forme.

Chapitre 2

Critique épistémologique du cloisonnement des catégories

Résumé du chapitre : Au cours de l'interaction vivante avec les systèmes d'informatique musicale, on ressent avec évidence qu'il y a quelque chose qui relève du « tout de suite » et autre chose qui relève du « plus tard », et, dans ce sens-là, temps réel et temps différé constituent des catégories pertinentes, issues de l'expérience interactive, et délimitées par notre sensation d'immédiateté. La force de cette approche intuitive – la congruence entre notre expérience humaine des sensations d'immédiateté et de délai avec la dichotomie usuelle temps réel / temps différé – semble entériner ce modèle historique, et, ce faisant, dissimule ses faiblesses. De fait, un examen plus approfondi révèle des failles susceptibles de faire vaciller ce modèle robuste en apparence, dont l'évidence intuitive finit par s'évanouir. Plusieurs observations y contribuent : le caractère flou de la frontière entre temps réel et temps différé, la rapidité de l'évolution de l'informatique en général et de l'informatique musicale en particulier, et des contradictions conceptuelles majeures entre les niveaux technologique, pratique et musical des deux catégories.

2.1 Un seuil discutable

2.1.1 Une frontière floue

Comme on l'a vu à la section 1.2.7 page 25, la clé même de la définition des deux catégories du temps réel et du temps différé – le seuil de latence s – repose en général sur des considérations d'ordre perceptif, c'est-à-dire nécessairement variables d'un individu à un autre, d'un contexte d'audition à un autre, ou d'un moment de l'interaction à un autre[1]. Il s'agit toujours de statistiques sur une population particulière, et l'unicité d'une valeur annoncée traduit toujours une *moyenne*. Puisque que ces enquêtes subissent nécessairement une variabilité non négligeable, elles deviennent contestables quand elles décrètent une valeur unique et fixe – par exemple $s = 24\,\text{ms}$. Autrement dit, devrait-on vraiment classer un logiciel ou un système numérique du côté du temps différé lorsqu'on mesure un temps de latence de 25 ms ? De 26 ms ? Ou de 30 ms ?

Ce questionnement, malgré sa légèreté, montre une première limite majeure de l'arbitraire de la définition d'un seuil *fixe* pour départager le temps réel du temps différé. Il suggère préférablement une frontière épaisse, ou bien une frontière floue. Or la solution d'élargir ce seuil ponctuel à un intervalle de la forme $]s_{min}; s_{max}]$ pour formaliser une frontière « épaisse », solution envisagée puis écartée précédemment, complique et fragilise le modèle sans apporter non plus de réponse totalement satisfaisante. D'une part, le problème de l'arbitraire de la définition subsiste, puisque les deux bornes s_{min} et s_{max} de l'intervalle proposé restent fixées et donc contestables en tant que telles, et, d'autre part, des valeurs mesurées dans un contexte particulier ne peuvent être considérées comme pertinentes que dans ce contexte particulier. Pour illustrer ce dernier point, on peut aisément imaginer qu'une latence relativement importante mais acceptable dans un contexte musical où les textures prédominent ne le sera sans doute plus dans un contexte musical où la pulsation et les percussions prévalent.

En substance, le flou « indissipable » de la frontière entre temps réel et temps différé relève ici de deux ordres : l'arbitraire de fixer une valeur ou un intervalle plutôt qu'un autre, et l'impossibilité de généraliser toute valeur ou tout intervalle qui semble pourtant pertinent dans un contexte particulier.

2.1.2 Des définitions multiples

En informatique musicale, plusieurs définitions du temps réel coexistent, tout en différant selon le point de vue adopté. Pour montrer cette multiplicité, nous allons seulement remarquer qu'un même auteur parfaitement légitime dans notre domaine peut proposer des définitions irréductibles l'une à l'autre :

1. Henri Piéron notait déjà en 1913 que « [...] l'allongement des temps de réaction au fur et à mesure qu'on se rapproche du seuil de sensation dépend essentiellement d'une augmentation du temps nécessaire pour que la transformation de l'excitant physique en phénomène cérébral de nature sensorielle s'effectue ; si l'on descend au-dessous du seuil, ce temps devient infini. [...] la loi de décroissance des temps de latence en fonction des intensités varie suivant les sensations [...] » [Piéron, 1913, p. 17–18]

> **Définition 1.** [Roads, 1992, p. 109] « Le matériel rendait impossible de concevoir la synthèse du son haute-fidélité en temps réel. Par temps réel, nous voulons dire qu'on entendra le son en même temps qu'on verra les gestes du musicien. »

> **Définition 2.** [Roads, 1998, p. 69] « Qu'entendons-nous par temps réel ? Dans ce contexte [de la synthèse numérique], le temps réel signifie que nous pouvons achever les calculs pour un échantillon en un temps inférieur à une période d'échantillonnage. »

Précisons que cette multiplicité des définitions n'est pas du tout un phénomène isolé, mais que le temps réel a effectivement reçu de nombreuses définitions hétérogènes. Nous étudierons plus avant les aspects polysémiques des expressions « temps réel » et « temps différé » à la section 2.2 page 42, néanmoins nous pouvons examiner dans cette présente section en quoi ces deux définitions précises divergent.

Du point de vue de la perception

La première définition sous-entend une situation de concert et se fonde sur la perception humaine d'une simultanéité (chez le public qui voit les gestes du musicien et entend les résultats sonores de ces gestes). Cette définition implique clairement deux sens – la vision et l'audition –, or la perception de la simultanéité d'événements multisensoriels reste une question controversée et un sujet de recherche toujours d'actualité[2]. Elle fait appel à des mécanismes complexes, comme l'anticipation, la comparaison, la réinjection et la récursion [Levitin *et al.*, 1999], ainsi qu'à des notions récentes plus spécifiques, comme le « ventriloquisme temporel » ou la « fenêtre déplaçable » pour l'intégration multisensorielle[3].

L'article *The perception of cross-modal simultaneity* [Levitin *et al.*, 1999] rappelle d'abord l'importance des différences interindividuelles et l'asymétrie temporelle de la perception de la désynchronisation[4], puis revient sur plusieurs études qui lui précèdent à propos de la perception de la simultanéité audio/vidéo (cf. figure 2.1 page suivante), et livre enfin les résultats d'une expérience nouvelle. Dans cette expérience, un « acteur » aux yeux bandés *frappe* une surface relativement muette avec le *Radio baton* de Max Mathews, tandis qu'il *entend* un impact préenregistré à travers son casque audio, l'impact étant avancé ou retardé d'une valeur prise aléatoirement entre -200 ms et +200 ms. Un

2. *« An unsolved problem in cognitive science concerns the perception of simultaneous events, particularly when the information impinging on the sensory receptors comes from two different sensory modalities. »* [Levitin *et al.*, 1999] La perception de la simultanéité d'événements multisensoriels constitue d'ailleurs une des plus anciennes questions de la psychologie expérimentale, depuis le « problème de l'observatoire de Greenwich » en 1796 (concernant la vue et l'audition).

3. « *We are rarely aware of differences in the arrival time of inputs to each of our senses. New research suggests that this is explained by a 'moveable window' for multisensory integration and by a 'temporal ventriloquism' effect.* » [Spence et Squire, 2003] Les auteurs notent aussi que la supériorité de la vitesse de la lumière par rapport à celle du son dans l'air (3 000 000 m/s contre 340 m/s) peut être partiellement compensée par la lenteur de la transduction chimique de la lumière à partir de la rétine comparée à la transduction mécanique des ondes sonores dans l'oreille.

4. *« Past investigation of intermodal asynchrony has focused audio/video. There are two general findings. (1) There appear to be large individual differences in perception thresholds. (2) Thresholds are asymmetric : people are more likely to perceive events as synchronous when the audio* precedes *the video than vice versa. »* [Levitin *et al.*, 1999]

« observateur » *regarde* l'acteur depuis une pièce isolée acoustiquement (double vitrage), à une distance de 2 mètres de l'acteur, et *entend* l'impact préenregistré à travers son casque, avec le décalage aléatoire. Après chaque essai, les sujets déclarent si le son a eu lieu en même temps que le geste ou non. Selon la ligne de partage des 75%, l'article conclut que les « acteurs » ont détecté des asynchronies entre le toucher et l'audition à partir de -25 ms et +42 ms, et que les « observateurs » ont détecté des asynchronies entre la vue et l'audition à partir de -41 ms et +45 ms :

> *A simple and intuitive way to consider the data is to plot the percentage of the time subjects judged the stimuli to be synchronous as a function of the sound-contact asynchrony (indicated by responses of "same"). If we consider the 75% line to indicate subjects' detection threshold (i.e., the asynchrony beyond which fewer than 75% of responses incorrectly judge simultaneous), we find that actors detected asynchronies at -25 and +42 msec, and observers detected asynchrony at -41 and +45 msecs.*

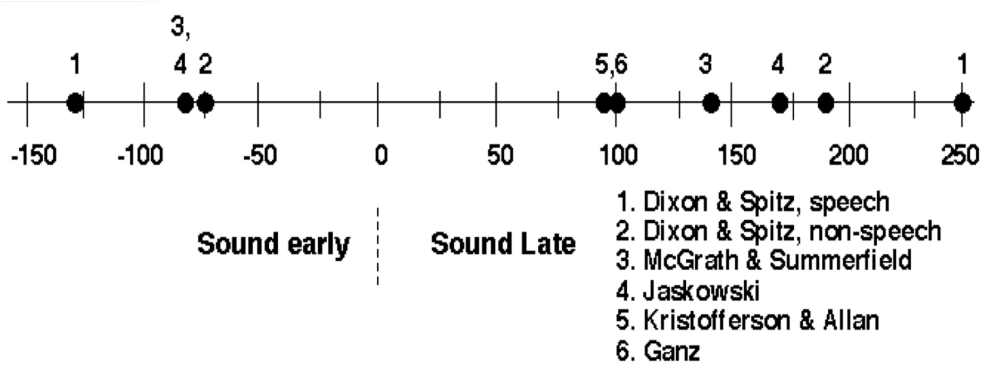


FIG. 2.1 – *Estimations récentes du seuil de simultanéité audio/vidéo, d'après Levitin & al.*

Dans le cas qui nous intéresse (la latence d'un système audionumérique perceptible par un public entre la vue du geste d'un musicien et l'audition du son produit par ce geste), nous pouvons retenir de ces quatre résultats uniquement celui qui concerne le seuil positif (plutôt que le seuil négatif) et perçu par l'observateur (plutôt que par l'acteur) : soit un seuil de +45 ms. Cette valeur, qui correspond à la première définition du temps réel susmentionnée, sera à comparer avec celle qui peut correspondre à la seconde définition.

Du point de vue de la fréquence d'échantillonnage

La seconde définition, liée à la puissance de calcul d'un ordinateur, est issue de l'analyse de Curtis Roads citée précédemment (cf. section 1.2.5 page 21). Cette analyse démontre que pour qu'une synthèse sonore puisse avoir lieu en temps réel, elle doit réaliser les calculs nécessaires à chaque échantillon en un temps inférieur à la période d'échantillonnage. Ainsi, pour un taux d'échantillonnage standard de 44 100 échantillons par seconde, les calculs doivent donc être achevés en moins de 23 *micro*secondes[5][6]...

5. L'auteur de conclure en 1998 : « Ceci signifie qu'il est difficile à un ordinateur d'achever les calculs nécessaires pour plus de quelques oscillateurs simples en *temps réel*. » [Roads, 1998, p. 69]

Or en comparant ces 23 μs (pour la seconde définition) aux 45 ms (admissibles comme limite du temps réel en regard de la première définition), on peut constater une différence colossale[7] d'un facteur 2 000 ! D'où peut venir un tel écart dans la définition du temps réel – une catégorie pourtant si familière en informatique musicale ? Les divergences des études sur la perception, bien qu'elles soient fréquentes, restent toujours bien en deçà d'un tel écart (cf. figure 2.1 page ci-contre). Une telle différence laisse à penser que derrière la même locution, il ne s'agit pas de la même signification : cet écart peut en effet s'expliquer par une confusion dans le langage courant entre les deux niveaux *technologique* et *perceptif* de la définition du temps réel.

Conformément à la démonstration de Curtis Roads, un flux audionumérique doit être émis aussi rapidement que sa fréquence d'échantillonnage le requière pour pouvoir être audible en temps réel, sous peine d'artefacts audionumériques très gênants à l'audition. De fait, au bout de la chaîne de traitement numérique, si le convertisseur numérique-analogique (ou CNA) ne reçoit pas les échantillons à temps, il produira alors un signal erroné, ce qui se traduira typiquement par des saccades sonores intempestives, avec des « clics numériques » plus ou moins perceptibles selon les cas. D'un point de vue technique donc, les échantillons considérés individuellement doivent impérieusement arriver au CNA à une vitesse supérieure ou égale à la fréquence d'échantillonnage.

* * *

En résumé, on peut considérer que la notion de seuil qui est traditionnellement invoquée pour définir le couple temps réel / temps différé est finalement discutable, pour deux raisons principales : le flou de la frontière sensée les séparer et la multiplicité des définitions.

6. Cette période brève est couramment élargie par la technique de vectorisation audionumérique, présentée à la section 2.3.2 page 51.

7. Notons que cette période d'échantillonnage de 23 μs, qui correspond à la « qualité CD », ne vaudrait plus que 10 μs pour les taux professionnels actuels de 96 kHz et seulement 5 μs pour ceux qui travaillent jusqu'à 192 kHz…

2.2 La polysémie du temps réel et du temps différé

Le télescopage de plusieurs valeurs différentes pour définir le problématique seuil de latence censé séparer la catégorie du temps réel de celle du temps différé nous a amené à reconsidérer la définition du temps réel en informatique musicale dans le sens de la *pluralité*, pour proposer de distinguer trois champs sémantiques. Le découpage du sens reste un exercice délicat, car les sens dégagés possèdent évidemment des traits sémantiques communs, mais cette démarche qui, à notre connaissance, n'a pas encore été suivie, devrait permettre de tirer au clair certaines ambiguïtés persistantes.

2.2.1 Le champ technique

Le premier sens, le plus étroit, concerne le plus souvent le traitement numérique du signal (Tns) en rapport avec l'échantillonnage audionumérique, soit un seuil de latence proportionnel à la période d'échantillonnage du signal (de l'ordre de 23 μs pour un signal stéréophonique traditionnel échantillonné à 44 100 Hz, multiplié par la longueur des vecteurs audionumériques, cf. section 2.3.2 page 51). Ce premier sens, impliqué dans les expressions telles que « synthèse (en) temps réel » ou « traitement (en) temps différé », nous proposons de l'indiquer à travers les expressions « temps réel *technique* » et « temps différé *technique* ».

Le champ sémantique le plus ancien

Dans ce champ sémantique, la notion de temps différé se définit en opposition simple avec celle de temps réel, en fonction de la capacité ou de l'incapacité d'un système numérique à *garantir le respect d'une contrainte temporelle donnée*. Il s'agit du sens le plus ancien, proche des définitions généralistes de l'informatique (cf. section 1.2.2 page 14).

Une simple dichotomie technique

Ainsi, en tant que *locution adjective* (par exemple dans l'intitulé « synthèse en temps différé »), les deux expressions « temps réel » et « temps différé » caractérisent les applications et les processus selon qu'ils respectent ou non le seuil de latence fixé. Sous cette forme adjectivale, la préposition « en » peut être présente ou omise, de même qu'un trait d'union entre « temps » et « réel » ou « différé » ; ces deux remarques sur la préposition et le trait d'union semblent s'appliquer en anglais, avec une forme supplémentaire fusionnée qu'on rencontre par exemple dans *« realtime synthesis »*.

En tant que *substantif* (par exemple avec le groupe nominal « le temps réel »), forme plus rare dans ce champ sémantique technique, les deux expressions « temps réel » et « temps différé » désignent simplement le principe technique (de respect ou de non-respect du seuil de latence).

La garantie technique

Les deux expressions « temps réel » et « temps différé » dénotent finalement une qualité technique relative à la *garantie* du respect d'une contrainte temporelle forte et souvent de bas niveau, c'est-à-dire à un niveau proche de la machine. En généralisant, ces expressions relèvent du champ *technique* lorsqu'elles sont situées dans les contextes du Tns (éventuellement à d'autres échelles temporelles) ou de la gestion et du traitement des flux de données (soumis à des contraintes temporelles).

2.2.2 Le champ pratique

Le deuxième sens, relié aux seuils perceptifs de *l'interaction* (souvent indiqués entre 20 ms et 100 ms en fonction des modalités particulières de cette interaction, cf. section 1.2.7 page 25), relève de la psychologie expérimentale tout autant que du langage courant et se retrouve par exemple dans les expressions « installation (en) temps réel » ou « jeu (en) temps réel ». Ce temps caractéristique de l'interaction et des systèmes de contrôle événementiels correspondants, nous proposons de le nommer « temps réel *pratique* ». De l'autre côté, les autres pratiques associées à l'écriture et à la composition acquièrent un statut autonome dans l'expression « le travail en temps différé » par exemple. Ce temps caractéristique de l'écriture, nous proposons de le nommer « temps différé *pratique* ».

Un temps réel pratique interactif

Dans ce champ sémantique, la forme *substantive* « *le* temps réel » prend sa valeur de modalité essentielle de l'interaction, comme temps de l'action ou de la *praxis*, où le geste humain rencontre un retour dans un délai suffisamment court pour permettre son ajustement éventuel. Il s'agit donc du temps intuitif et subjectif du « tout de suite », même si ce temps reste objectivement soumis à un temps de latence du point de vue technique pour permettre la rétroaction dans les faits.

Logiquement, la *locution adjective* (par exemple dans « installation temps réel ») caractérise son sujet en indiquant qu'elle exploite des modalités *interactives*. Cette locution se présente donc comme un synonyme métonymique de l'adjectif *interactif*, désignant avec une légère préciosité la cause – le temps réel technique – pour l'effet – l'interaction. En effet, on a aujourd'hui l'impression que « l'interactivité » a beaucoup perdu de son pouvoir de séduction au profit du « temps réel », dont les racines sémantiques technologiques sont sans doute plus à même de promettre quelque prouesse...

Un temps différé pratique révisable

Une *praxis* du temps différé n'est pas en reste et ne constitue pas cette fois une opposition simple, ni une définition négative (un temps de calcul trop lent). Au contraire, le temps différé, en tant que *substantif*, désigne ici positivement les logiciels et les pratiques musicales qui s'inscrivent dans la durée, typiquement les activités compositionnelles. Cela n'interdit d'ailleurs pas aux logiciels utilisés de répondre à des critères temps réel de bas

niveau (temps réel technique) ni de posséder des capacités interactives (temps réel pratique), seulement l'usage qui en est fait n'est pas destiné à être écouté « tout de suite » par un auditoire. L'activité compositionnelle n'est en effet généralement pas envisagée comme une improvisation, ni comme une interprétation, car pour laisser le temps à l'élaboration de se faire, les résultats de la composition sont réservés à être écoutés « plus tard », voire pas du tout, ou encore en creux[8].

La réversibilité comme frontière

Finalement, dans le champ de la pratique, le critère distinctif majeur abandonne la notion technique du *seuil de latence* au profit de la notion pratique de *possibilité de retour* (cf. 1.3.4 page 34). En effet, *l'impossibilité* de retour caractérise le temps de la performance en tant que pratique, où ce qui est fait est fait, définitivement ; réciproquement, la *possibilité* de retour caractérise de façon toute aussi pertinente le temps de l'écriture en tant que pratique[9]. Il s'agit d'un certain rapport à la flèche du temps et à son irréversibilité : rapport en phase direct pour le « temps réel », et qui lui confère effectivement en ce sens une propriété capitale de la *réalité* avec l'attribut de l'irréversibilité, contre un rapport qui laisse une place à la réversibilité, en différant la diffusion ou l'exécution de l'œuvre, ce qui confère là aussi un sens profond à l'expression « temps différé », sous l'angle d'une *praxis* dont le résultat est remis à « plus tard ». La *possibilité de retour* devient alors une des modalités qu'on s'autorise ou qu'on s'interdit pour l'action musicale, et constitue ainsi un point de rupture valide vis-à-vis des pratiques musicales, entre la performance et la composition (cette dernière distinction sera approfondie au chapitre 5).

Le cas du *live-coding* pourrait apparaître comme un cas-limite (cf. section 2.3.2 page 55), puisqu'il s'agit à la fois d'une écriture (programmation) et d'une performance (en direct). Or, à la lumière du critère de la *possibilité de retour*, ce cas rejoint sans ambiguïté le temps réel (pratique), toute modification validée durant la programmation en direct étant irréversiblement donnée à entendre.

La métaphore du couple signal / code illustre schématiquement la distinction entre temps réel et temps différé dans le champ pratique. D'une part, le *signal* est une information éphémère, prise dans un flux – ce qui correspond au temps réel pratique, de l'interaction à la performance. D'autre part, le *code* est une information inscrite, enregistrée et susceptible d'être consultée, utilisée ou révisée ultérieurement – ce qui correspond au temps différé pratique, du paramétrage à la composition.

Différenciation évolutive des champs technique et pratique

Historiquement, si aux débuts du temps réel (technique et pratique, dans les années 80 pour la musique) les programmes et les usages associés à ces programmes pouvaient se

8. Par exemple, on peut considérer qu'après plusieurs étapes de traitement successives, un son de 1re ou 2e génération qui n'apparaît pas dans l'œuvre finale existe tout de même « en creux » si un autre son de génération suivante, formé à partir de ce son d'origine, y apparaît.

9. La pratique du *work in progress*, chère à Pierre Boulez entre autres, franchit d'ailleurs un pas supplémentaire vis-à-vis de cette possibilité de retour dans l'écriture, entraînant un changement d'échelle temporelle dans la notion d'*œuvre*.

confondre dans une correspondance plus ou moins bijective entre le domaine *technique* et le domaine *pratique*, dès que les programmes dédiés au temps différé (pratique) ont bénéficié de fonctionnalités temps réel (technique et pratique, comme la possibilité d'écouter directement un son), cette bijection relative fut logiquement rompue de fait. Ainsi, la croissance exponentielle de la puissance des systèmes informatiques aidant, l'écart entre le temps différé pratique et le temps différé technique n'a cessé de croître, jusqu'au stade actuel où la plupart des programmes d'écriture (au sens large) sont équipés de nombreuses fonctionnalités temps réel (*Csound* en est un exemple typique[10]). Cette évolution des logiciels traditionnellement considérés « temps différé » vers le temps réel suggère bien d'opérer une distinction entre les niveaux *pratique* et *technique*, puisque le temps réel a tendance à se généraliser au niveau technique. Réciproquement, les pratiques temps réel comportent en réalité presque toujours des phases de *préparation* directes ou indirectes, qui relèvent du temps différé (pratique). Nous reviendrons sur ce point à la section 2.4 page 60.

2.2.3 Le champ musical

Le troisième sens, le plus large, exprime le glissement sémantique vers la fonction musicale en tant que choix esthétique (cf. section 1.3 page 29). Lorsqu'on lit des expressions comme « une œuvre avec temps différé » ou « improvisation et électronique en temps réel », on comprend qu'il ne s'agit pas de caractériser une technologie ni une pratique mais bien *une musique* dans ses choix esthétiques. Nous proposons de nommer cette partie de notre typologie, selon le contexte : « temps réel *esthétique* » et « temps différé *esthétique* », ou bien « temps réel *musical* » et « temps différé *musical* ».

Un temps réel musical non-superposable aux autres champs

Cette dimension *esthétique* ou *musicale* ne s'appuie pas sur la dimension pratique de façon simple ou univoque, de même que la dimension *pratique* identifiée précédemment comporte déjà en elle-même des niveaux d'imbrication entre temps réel et temps différé. Bien sûr, le temps réel *esthétique* ou *musical* implique à la fois les temps réels *pratique* et *technique* (l'expression « une œuvre avec temps réel » fait communément référence aux idées d'interaction, de performance, et aux moyens techniques logiciels et matériels nécessaires à cette interactivité), mais il implique aussi du temps différé ! Et ce à plusieurs niveaux : au niveau *pratique* dans la préparation de la performance, au niveau *technique* aussi selon les programmes utilisés dans cette préparation, voire même au niveau *esthétique*, par exemple lorsque des séquences sonores pré-enregistrées doivent être déclenchées durant la performance ou l'exécution de l'œuvre (ce qui reste très fréquent).

Un temps différé musical « sur support »

Curieusement, le temps différé *esthétique* ou *musical* n'implique pas nécessairement le temps différé *pratique*. Il existe en effet quelques cas extrêmes, mais pas si rares, de

10. *« As fast or realtime software synthesis becomes increasingly possible, the free exchange of* instrument *and* score files *becomes worthwhile. »* [Vercoe et Ellis, 1990, p. 211, c'est nous qui soulignons].

diffusion d'un enregistrement où ce qui a été enregistré provient directement d'une unique improvisation interactive (par exemple avec des magnétophones). Il s'agit là d'une situation musicale apparemment temps différé et le plus souvent déclarée comme telle (diffusion d'une musique « sur support ») alors qu'aucun travail d'écriture, de ré-écriture ou de montage n'a été effectué, ce qui tendrait au contraire à justifier d'une pratique « purement » temps réel (ici, une improvisation)...

Au-delà de ces cas particuliers, le champ *esthétique* de l'expression « temps différé » recouvre souvent l'ensemble des musiques dites « sur support », à comprendre comme les musiques savantes utilisant les technologies numériques et analogiques sous leurs différentes dénominations (concrète, électronique, électroacoustique ou acousmatique), et rejoignant parfois l'idée d'un art des sons *fixés* telle qu'elle a été développée par Michel Chion. L'utilisation de ce sens *esthétique* transgresse de plus en plus l'ontologie informatique[11] du temps différé, en annexant progressivement les technologies analogiques (pour désigner les concerts de musique concrète par exemple). Cette extension du champ sémantique de la notion de « temps différé musical » va parfois même jusqu'à désigner la musique instrumentale qui utilise des partitions, considérant que ces partitions sont effectivement des « supports » de la musique (mais pas directement supports du son, puisque l'interprétation ne se confond jamais avec la transduction).

* * *

La polysémie du temps réel et du temps différé est avérée, il suffit pour s'en convaincre de lire quelques définitions apparemment incompatibles de la notion de temps réel. Aussi, nous avons proposé de distinguer trois champs sémantiques : le champ technique, le champ pratique et le champ musical. Ces trois champs ne peuvent pas se superposer d'une façon simple, car les notions de temps réel et de temps différé y acquièrent un sens suffisamment différent, individuellement et en rapport, à chaque niveau.

2.3 Une multiplicité des temporalités technologiques

Le temps de latence, en tant que paramètre technologique, résulte de la combinaison de plusieurs facteurs distribués tout au long de la chaîne audionumérique, à chaque étape de calcul ou de communication. De plus, la latence s'incarne différemment dans chaque programme, quelle que soit sa hiérarchie dans l'architecture en couche des ordinateurs, et dans chaque interface matérielle. Ce constat sollicite un tour d'horizon des technologies en jeu actuellement dans l'informatique musicale, pour mieux appréhender la notion de « latence » sous ses différentes formes, aux niveaux matériel et logiciel.

2.3.1 Les temps du matériel

Accès disque

11. Cf. section 1.1.1 page 8.

Les programmes et les données de nos ordinateurs sont le plus souvent stockés sur des disques durs magnétiques. Ceux-ci permettent en effet de stocker beaucoup plus d'informations que la mémoire vive, mais le transfert de ces informations se fait à une vitesse bien inférieure, de l'ordre d'un facteur 1 000. Cette différence doit être prise en compte par exemple pour la lecture ou l'enregistrement de fichiers son directement sur disque dur en temps réel. La fragmentation des données sur le disque, c'est-à-dire leur discontinuité sur la surface physique circulaire du disque, constitue un deuxième paramètre important à prendre en compte, le coût temporel du positionnement des têtes de lecture/écriture et celui du positionnement d'un secteur sous ces têtes étant important (jusqu'à 10 ms).

> Aujourd'hui, les temps moyens de positionnement entre deux pistes quelconques s'échelonnent entre 5 et 10 ms, et sont inférieurs à 1 ms entre deux pistes consécutives. Une fois la tête positionnée radialement (le long du rayon du disque), un deuxième délai, appelé temps de latence rotationnel, s'écoule durant lequel le secteur souhaité s'aligne sous la tête. La plupart des disques tournent à 5 400, 7 200 ou 10 800 tours par minute. L'attente moyenne (durée d'une demi-rotation) est donc de 3 à 6 ms. Enfin, la vitesse de transfert d'informations dépend de la densité linéaire et de la vitesse de rotation. Elle est généralement comprise entre 20 et 40 Mo/s. Un secteur de 512 octets requiert donc un temps de lecture de 13 à 26 microsecondes. Le temps moyen de positionnement et le temps de latence rotationnel dominent largement le temps de transfert. C'est pourquoi, il est inefficace de procéder à la lecture des secteurs en les choisissant dans un ordre aléatoire. [Tanenbaum, 2005, p. 88]

Mulot & Co

La réponse de nos périphériques de communication avec l'ordinateur n'est pas instantanée et dépend de facteurs différents en fonction du type des périphériques et des technologies qu'ils utilisent.

Clavier alphanumérique : Signalons d'abord que l'invention du clavier des machines à écrire mécaniques répondait déjà à l'époque à un désir de vitesse[12]. Bien que relativement standardisés, nos claviers d'aujourd'hui ne se valent pas tous : la disposition des touches, leur hauteur, leur largeur, et le seuil de pression pour le déclenchement (considérablement réduit grâce à l'électricité), constituent les premiers facteurs à prendre en compte. En outre, la vitesse effective des claviers informatiques dépend fortement du bus de communication – de sa largeur, de sa fréquence et de son éventuel engorgement (par exemple en USB) –, ainsi que de leur couplage avec le système d'exploitation et donc aussi avec l'unité centrale :

> Lorsque la touche du clavier d'un micro-ordinateur est enfoncée, une interruption est générée et la procédure d'interruption (un logiciel faisant partie du système d'exploitation) du clavier démarre. Cette procédure consulte un registre contenu dans le contrôleur du clavier pour connaître le numéro de la

12. « Une machine aurait été construite dès 1714 par un Anglais. Les premières machines sont à cadran : l'écriture est bien lisible mais peu rapide. [...] L'invention du clavier est décisive car elle permet la vitesse. C'est un imprimeur, Latham Sholes, qui en 1873 aux Etats Unis conçoit la première *"type-writer"*. [...] Les champions tentent de battre des records de nombre de mots tapés à la minute. » http://www.archivesnationales.culture.gouv.fr/camt/fr/se/fiche3/fiche3.html

> touche pressée. Une seconde interruption est générée lorsque la touche est relachée. [...] La gestion des combinaisons de touches incluant SHIFT, CTRL et ALT est entièrement prise en charge par le logiciel. [Tanenbaum, 2005, p.112]

Le clavier reste en général le périphérique d'entrée le plus lent et le moins gourmand, puisqu'il n'envoie qu'un code clavier à la fois en rapport à la vitesse de frappe d'un humain, d'au maximum 20 lettres/secondes, soit 50 ms ; un système de tampon est mis en place pour baisser cette fréquence si nécessaire.

Souris : Complément ou alternative au clavier alphanumérique, la souris subit les mêmes contraintes externes que lui : couplages avec le bus, le système d'exploitation et l'unité centrale. En revanche, les souris émettent un débit d'informations potentiellement plus important, avec des variations significatives selon les technologies employées : mécanique, optique, opto-mécanique, ou laser, sans compter les différentes déclinaisons de pointage comme le pavé tactile, la boule de pointage, la manette de jeu, la tablette graphique avec son stylet, ou l'écran tactile. Ainsi, certaines souris laser de précision ou destinées au jeu montent à une résolution de 2 000 dpi[13] à une fréquence généralement comprise entre 100 Hz et 1000 Hz, soit une latence théorique comprise entre 10 ms et 1 ms, bien inférieure à celle du clavier dans tous les cas.

Écran : Un principe temporel préside au fonctionnement des écrans, qu'ils utilisent une technologie cathodique ou à cristaux liquides, car ils doivent redessiner la totalité de leur surface au moins soixante fois par seconde pour pouvoir tromper la vision, soit une fréquence de rafraîchissement de 60 Hz correspondant à une latence d'environ 17 ms, ou bien 75 Hz, une autre fréquence courante, correspondant à 13 ms. Notons que la puissance de calcul des cartes graphiques devient cruciale pour décharger le processeur principal des tâches graphiques très gourmandes en ressources, comme le rendu 3D, la gestion de la mémoire vidéo, le traitement du signal vidéo, la décompression, etc., grâce au développement des processeurs graphiques[14]. Enfin, mentionnons un dernier phénomène temporel impliqué dans l'affichage vidéo : la rémanence, le temps mis par un point de l'écran pour passer de l'état totalement allumé à l'état totalement éteint ou réciproquement (typiquement, elle peut produire des traînées gênantes si elle est trop élevée).

Suivre le protocole

Les protocoles de communication eux-mêmes possèdent un temps de latence propre et incompressible. L'histoire de ces protocoles reflète d'ailleurs les différents compromis retenus entre le débit des informations à transmettre et le coût des contrôleurs et des câbles associés à un protocole.

Si l'audionumérique a poussé les évolutions technologiques dans le sens de l'augmentation des fréquences d'horloge avec ses taux d'échantillonnage élevés, c'est surtout la vidéo qui les a poussé dans le sens de l'augmentation du *débit*, avec les affichages en plein écran, les vidéos et les jeux multimédias : un déplacement fluide requière un débit de l'ordre de 135 Mo/s sur un écran 1024×768 pixels en pleine couleur, quand un signal stéréophonique standard ne demande que 176 Ko/s (44 100 Hz × 2 octets × 2 canaux), soit 767 fois moins.

13. *dot per inch*, désigne une unité standard de résolution, en nombre de points par pouce.

14. Ou GPU pour *Graphics Process Unit*.

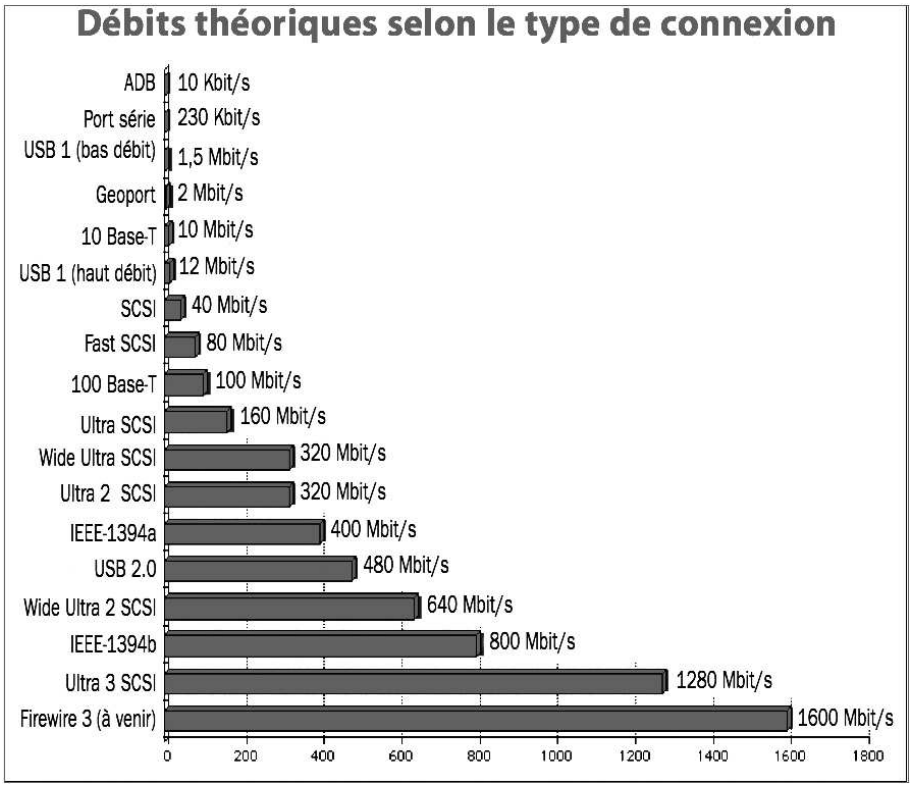


FIG. 2.2 – *Débits théoriques selon le type de connexion*

ADB : L'*Apple Desktop Bus* est un bus série minimaliste et asynchrone conçu par Apple en 1986 pour connecter les périphériques d'entrée bas débit, comme le clavier ou la souris, et remplacé par l'USB à partir de 1998. Son débit théorique de 125 kbit/s descendait plutôt à 10 kbit/s en pratique, soit 1,25 octet/s.

FireWire : Définie en 1995, la norme IEEE[15] 1394 décrit un bus série multiplexé rapide et synchrone pour les périphériques à haut débit, et, selon les déclinaisons, supporte des taux de transferts de 100 Mbit/s à 3200 Mbit/s, le FireWire 400 (IEEE-1394a) et le FireWire 800 (IEEE-1394b) étant les plus répandus. Les paquets sont transmis à une fréquence de 8kHz, soit 0,125 μs.

PCI : Lancé par Intel en 1992 pour répondre à l'évolution vidéo, le bus local et parallèle *Peripheral Component Interconnect* permettait déjà dans sa première version de transférer 133 Mo/s (32 bits × 33 MHz, soit un temps de cycle de 30 ns) afin de pouvoir gérer typiquement les cartes graphiques, les cartes son et les cartes réseau. Si le bus PCI monte actuellement jusqu'à 528 Mo/s (64 bits × 66 MHz) et le PCI-X jusqu'à 2 133 Mo/s, la norme PCI-Express remanie avantageusement l'architecture parallèle traditionnelle[16] au profit de liaisons séries point à point très rapides, avec des connecteurs plus petits et des débits compris entre 250 Mo/s et 8 Go/s.

SATA : À partir de 2003, le bus série *Serial Advanced Technology Attachment* commençait à remplacer le bus parallèle ATA mis au point par l'ANSI[17] en 1994 pour la

15. *Institute of Electrical and Electronic Ingineers ;* « FireWire » est le nom commercial donné par Apple.

16. « L'augmentation maximale de la fréquence d'horloge du bus [parallèle] n'est pas une bonne solution, car elle amplifie les phénomènes de déphasage temporel (*bus skew*), de diaphonie entre les fils et les effets de capacitance. » [Tanenbaum, 2005, p. 227]

connexion de périphériques de stockage à haut débit. Les débits utiles théoriques de 150 Mo/s à 600 Mo/s dépassent pour l'instant largement les débits maximum des disques durs (de l'ordre de 80 Mo/s), ces derniers constituant maintenant les véritables goulots d'étranglement.

SCSI : Aujourd'hui en voie de marginalisation, le bus parallèle *Small Computer System Interface* permet la connexion interne ou externe de nombreux types de périphériques d'entrée ou de sortie. Depuis la première définition de 1986, les différentes versions emploient des largeurs allant de 8 à 32 bits associées à des fréquences allant de 5 à 80 MHz (soit 200 ns à 12 ns), pour des débits allant de 5 à 640 Mo/s (SCSI-3 Ultra-640).

USB : Né de la collaboration entre plusieurs entreprises[18] réunies pour régler les problèmes de connexion des périphériques lents, le standard *Universal Serial Bus*, officiellement publié en 1998, équipe aujourd'hui la plupart des ordinateurs. Respectant les exigences de départ, l'Usb est bon marché, ne nécessite pas de configuration de la part de l'utilisateur et supporte le branchage / débranchage sans éteindre l'ordinateur (le *« Hot Plug-and-Play »*), permet de relier jusqu'à 127 périphériques, de les alimenter, et supporte les équipements temps réel (microphone, téléphone, webcam, etc.). L'Usb actuel, dans sa version 2.0, prend en charge les trames isochrones et gère trois vitesses de transfert : 1,5 Mbit/s, 12 Mbit/s et 480 Mbit/s.

Ethernet : Standardisé en 1980 sous le nom IEEE 802.3, l'Ethernet, un protocole de réseau local, se décline aujourd'hui selon quatre vitesses de transmission : l'Ethernet à 10 Mbit/s, le *Fast Ethernet* à 100 Mbit/s, le *Gigabit Ethernet* à 1 Gbit/s, et le nouvel Ethernet à 10 Gbit/s (pour les gros serveurs).

Réseaux locaux sans fil : Les normes WiFi (IEEE 802.11) et Bluetooth (IEEE 802.15) utilisent toutes les deux la bande de fréquence de 2,4 GHz pour leurs transmissions par ondes radio courte distance, et proposent respectivement des débits de 11 à 248 Mbit/s théoriques pour les différentes déclinaisons de la première, et de 1 à 10 Mbit/s pour la deuxième. Elles remplacent progressivement les liaisons lumineuses infrarouges (115 kbit/s à 16 Mbit/s).

Midi O'Clock

Rare îlot de stabilité dans la tempête informatique de l'obsolescence, la norme Midi, 25 ans après sa naissance, reste d'actualité ! Cette norme, toujours utilisée dans sa version 1.0, n'a pourtant rien d'un foudre de guerre : 31,25 kbaud (soit environ 3,8 Ko/s), à comparer par exemple avec l'Usb (400 fois plus rapide pour l'Usb 1.1, et environ 16 000 fois plus rapide pour l'Usb 2.0). La robustesse du Midi alliée à son omniprésence sur les matériels et les logiciels semblent constituer une inertie insurmontable pour les prétendants à la succession, dont *Distributed MIDI*[19] présenté en 2003 par la très compétente IEEE-SA[20].

17. *American National Standards Institute.*

18. Compaq, DEC, IBM, Intel, Microsoft et Northern Telecom.

19. *« DMIDI will use the current Ethernet-based networking infrastructure to carry MIDI data at transmission speeds as great as 10 Gbit/sec.* [...] *The new standard will increase the number of addressable devices. The original MIDI specification provided 16 channels for devices. DMIDI will allow for nearly 16 million devices, each retaining the existing 16-channel MIDI structure. »* http://standards.ieee.org/announcements/p1639app.html.

Le Midi est tout à la fois la spécification d'une interface matérielle (connecteurs, câbles, voltage), un protocole de communication (structure et ordonnancement des « messages »), un format de fichier (*Standard* Midi *File* et ses variantes), et un standard pour la numérotation d'instruments virtuels (*General* Midi et ses variantes). La latence globale d'un système Midi cumule la latence propre du contrôleur (clavier, contrôleur à vent, pédalier), la latence de la liaison (en tenant compte du nombre d'émetteurs branchés en série), et la latence propre du ou des récepteurs (échantillonneur, module de synthèse, séquenceur, ordinateur). L'unité de base des informations Midi est codée sur 10 bits (1 bit de début + 1 octet utile + 1 bit de fin), les messages Midi sont eux-mêmes constitués d'une concaténation de ces unités (en nombre variable selon les messages), et enfin la sérialisation de la liaison augmente le risque de retard avec l'augmentation du débit.

Dès sa conception, l'enjeu du Midi a été le temps réel, à partir du besoin d'interconnexion entre appareils de marques différentes (essentiellement des synthétiseurs et des séquenceurs numériques) et du besoin de leur contrôle en temps réel. Cet enjeu, transposé plus tard aux ordinateurs, ne s'est pas démenti, comme le montre l'appellation Mros ou *MIDI Real-time Operating System*, un des premiers systèmes d'exploitation destiné aux applications musicales, développé en 1989 sur Atari, ordinateur Midi s'il en est. Notons que le Midi sert à la fois les applications temps réel mais aussi les applications temps différé, grâce à son potentiel solfégique dans la description des hauteurs, des durées, proche de la notion de note musicale, et dans la description des mesures et du tempo. Cet usage temps différé du Midi ne convient toutefois pas bien aux œuvres qui n'intègrent pas ce concept de la note, en tout cas pas sans détournement aux limites de ce qu'offre ce format.

2.3.2 Les temps du logiciel

La vectorisation audionumérique

La « vectorisation audionumérique » est une technique efficace pour accélérer le traitement global d'un signal audionumérique, mais génératrice de latence.

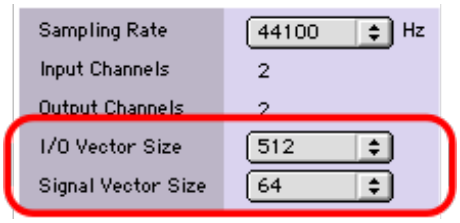


Fig. 2.3 – *Taille des vecteurs audionumériques dans Max/*Msp

En effet, pour diminuer la contrainte temporelle due à la période d'échantillonnage (de l'ordre des 23 μs calculées précédemment), la plupart des systèmes numériques logiciels ou matériels regroupent les échantillons par *vecteurs audionumériques*[21] dont la longueur

20. *Institute of Electrical and Electronics Engineers – Standard Association.*

21. Il s'agit de « vecteurs » au sens informatique : une structure de données de type tableau à une dimension.

est une puissance de 2, comme 64 ou 512 échantillons (cf. figure 2.3). Cette technique exploite le fait que les ordinateurs calculent et transmettent plus rapidement des vecteurs d'échantillons que le même nombre d'échantillons un par un, notamment grâce à la mise en mémoire des données regroupées (*buffering*) et à la technique du pipeline[22]. Ainsi, le gain réalisé par la vectorisation réduit d'autant le temps de calcul global. Cependant – et c'est là le point délicat – cette vectorisation augmente le temps de latence, proportionnellement à la taille des vecteurs. De fait, une fois que les échantillons sont regroupés, il faut attendre qu'un vecteur entier soit calculé avant de pouvoir entendre le son correspondant. Le délai « vectoriel » induit (d_{vect}) vaut alors exactement la longueur des vecteurs (l_{vect}) divisée par la fréquence d'échantillonnage ($f_{éch}$) :

$$\boxed{d_{vect} = \frac{l_{vect}}{f_{éch}}} \tag{2.1}$$

On peut calculer à partir de l'équation 2.1, pour une fréquence d'échantillonnage de 44 100 Hz, les couples courants suivants :

Longueur vectorielle	Délai vectoriel
16 échantillons	0,36 ms
32 échantillons	0,73 ms
64 échantillons	1,45 ms
128 échantillons	2,90 ms
256 échantillons	5,80 ms
512 échantillons	11,61 ms
1024 échantillons	23,22 ms
pour une fréquence d'échantillonnage de 44 100 Hz	

Tab. 2.1 – *Longueurs vectorielles et délais correspondants*

D'abord, ce paramètre de la longueur du vecteur audionumérique, conjugué à la fréquence d'échantillonnage, donne une première mesure technologique du *temps de latence d'un système numérique*. Ensuite, et c'est sans doute le point le plus important, la vectorisation audionumérique constitue un *compromis* entre l'optimisation du traitement du signal (temps de calcul et temps de transfert) et le temps de latence du système. Enfin, comme le montre la figure 2.3), ce compromis est *paramétrable* dans la plupart des logiciels audionumériques : pour le transfert (*« I/O Vector Size »*) et pour les calculs (*« Signal Vector Size »*). De cette façon, l'utilisateur peut lui-même choisir la latence qu'il juge acceptable, en calculant le délai vectoriel induit (le tableau 2.1 page suivante donne les valeurs courantes pour une fréquence d'échantillonnage de 44 100 Hz). Par ailleurs, la technique de vectorisation montre que la puissance de calcul des ordinateurs n'est pas le seul facteur qui conditionne le temps de latence.

22. Les architectures pipelines permettent de gagner du temps à deux niveaux : d'abord en anticipant le chargement des instructions (*prefetching*), car la récupération effective des instructions dans la mémoire est une source de ralentissement majeure, et ensuite en parallélisant l'exécution des instructions sur plusieurs étages, ce qui permet de commencer l'exécution d'une instruction sans attendre la fin de la précédente.

Divergences de politique environnementale

Les programmes ou les langages de programmation qui fonctionnent en temps différé ne rencontrent pas de problème majeur vis-à-vis de la synchronisation ou de la latence, puisqu'ils disposent du temps de calcul nécessaire par définition. Les différences « temporelles » se situent pour eux au niveau de la sémantique et de la syntaxe des langages pour la spécification de l'organisation chronologique des évènements ou des processus, par exemple de manière absolue – avec une ligne chronologique de référence (*timeline*) – ou de manière relative.

De leur côté, les différents environnements de programmation dédiés au temps réel donnent généralement un accès plus ou moins ouvert à la gestion du temps de latence (c'est-à-dire essentiellement à la taille des vecteurs audionumériques unitaires, cf. section 2.3.2 page 51). Il y a donc une certaine équivalence temporelle pour ces environnements temps réel au niveau *technique*. En revanche, au niveau *pratique*, ces environnements se différencient significativement selon le paradigme de programmation adopté[23]. À notre connaissance, actuellement, trois environnements de programmation représentent schématiquement les différentes options pour les applications audionumériques dites « temps réel »[24] : *Max/MSP*, autour de la programmation graphique (les « patchs »), *SuperCollider*, autour de la programmation textuelle orientée objet (entre Smalltalk-80 et C), et *ChucK*, autour de la programmation textuelle « à-la-volée » (orientée vers le *live-coding*).

***Max/MSP* :** *Max* est sans conteste le plus largement utilisé, y compris pour des utilisations qui ne sont pas musicales ou sonores. Cependant, comme l'indique François Déchelle : « La place de Max dans la famille des langages de programmation n'est pas facile à établir : d'une part, Max n'a pas été revendiqué comme un langage de programmation par son inventeur[25], d'autre part, sa caractéristique de langage graphique le rend quelque peu atypique. » Si l'on s'en tient aux propos de Miller Puckette, *Max* n'appartient pas aux langages de programmation, mais relève plutôt d'un langage « d'interconnexion », un *« patching language »* :

> *Max was an attempt to make a screen-based patching language that could imitate the modalities of a patchable analog synthesizer. Many other graphical « patching languages » had been proposed that did not sufficiently address the real-time control aspect ; and many other researchers had by then proposed much more sophisticated real-time control strategies without presenting a clear and fun-to-use graphical interface ; Max was in essence a compromise that got part way toward both goals.* [Puckette, 1996]

23. Au delà de la question temporelle, les différents environnements permettent de programmer à peu près les mêmes choses (synthèse, échantillonnage, interactivité, etc.), mais de façon plus ou moins facile ou difficile du point de vue de la programmation (humaine).

24. Miller Puckette date le début de ces applications en 1981 avec le programme RTSKED de Max Mathews : *« The design of real-time computer music systems has been a subject of active research since the RTSKED program. »* [Puckette, 1996]

25. Miller Puckette, l'auteur principal de Max, précise en effet : *« Max wasn't originally intended as a programming language »* [Puckette, 1996], pour la raison suivante : *« In thinking about Max, I always sought reliability first, simplicity second, expressivity of the language third, and what I call "computer science issues" almost not at all. »* [Desain *et al.*, 1993]

Récemment, Carl Seleborg a proposé un modèle formel de *Max* à partir des *Communicating Sequential Processes* de Hoare[26], éclairant le statut informatique de *Max* du point de vue théorique :

> Un patch Max [...] pourra être qualifié de *réseau d'objets réactifs* ou encore de *système réactif*, ce qui ferait de Max un *environnement de construction de systèmes réactifs musicaux*. [Seleborg, 2004, p. 18]

Dans cette définition, la façon d'envisager le temps dans *Max* commence à transparaître : la *réactivité* s'impose comme modalité structurelle et structurante de la temporalité. Séquentiellement, des messages sont envoyés et reçus entre les objets réactifs qui constituent le réseau (le plus souvent fixé), et la temporalité émerge des réactions en chaîne à travers le réseau. Ainsi, si la métaphore du synthétiseur analogique reste la clé principale du succès de *Max*, elle constitue aussi son point faible en ce qui concerne le contrôle temporel[27] et les grands systèmes[28]. Le signal audionumérique est calculé vecteur par vecteur pour toute la chaîne de traitement, alors que les messages utilisent un système de double pile selon leur priorité – haute ou basse. Ce système de priorité à deux vitesses oblige parfois à employer des objets comme `defer` ou `deferlow`[29] lorsque certains messages à priorité haute[30] perturbent l'ordonnancement général.

***SuperCollider* :** Le statut informatique de *SuperCollider* est beaucoup plus facile à définir : c'est à la fois un environnement de programmation – avec une interface-utilisateur graphique[31] – et un langage de programmation orienté objet dans sa structure, proche de Smalltalk-80, combiné avec une forme de langage C :

---

26. C. A. R. Hoare. *Communicating sequential processes*. Communications of the ACM, volume 21(8), pages 666–677, New York, 1978.

27. *« I feel that the main reason for the problems of structure in the code is that Max is evidently based on an analog synthesizer metaphor. One can regard the analog synthesizer as a quasiparallel structure, where outputs and inputs seem to be available at any time, and simultaneously. There needn't be a single signal flow path, and you can have feedback between arbitrary points. The main problem with analog synthesizers is that you know what, but you don't know when. It is OK for voltages to behave this way, but not for a computer program that is handling musical data of any complexity. Data-flow and signal flow are two different animals ; while watching someone create analog synthesizer patches, I often feel that if a program were structured in a similar fashion, it would either never work, or be too hard to debug. »* George Lewis dans [Desain *et al.*, 1993].

28. *« I think my central criticism is that Max does not scale well. It is fine for protoyping small systems, and great for control panels, but it becomes unmanageable in large systems. »* Roger Dannenberg dans [Desain *et al.*, 1993].

29. J'utilise régulièrement l'objet `deferlow` pour garantir l'ordonnancement des messages dans mes patchs. Le manuel de référence précise à ce sujet : *« The deferlow object places all incoming messages at the tail of the low priority queue. [...] The deferlow object is useful to preserve message sequencing that might otherwise be reversed with the defer object and/or guarantee that an incoming message will be deferred to a future servicing of the low priority queue even if that message is low priority itself. » [Zicarelli et al., 2006b, p. 138]*

30. *« Examples of high priority messages are those generated by a MIDI object (such as notein) or a timing object (such as metro or seq), and examples of low priority message are those generated in response to user events (such as clicking a button). » [Zicarelli et al., 2006b, p. 138]*

31. *Graphical User Interface* ou *GUI* en anglais. Sous *SuperCollider*, la classe *GUI* permet de choisir entre Cocoa (Macintosh) ou Swing (Java) ; elle pourvoit en outre une large collection d'objets graphiques plus spécifiques aux us de l'interaction musicale, comme des boutons, des curseurs, des oscilloscopes, des visualiseurs de formes d'onde ou d'enveloppes, etc.

> *SuperCollider is an environment for real time audio synthesis which runs on a Power Macintosh with no additional hardware. SuperCollider features : a built in programming language with real time incremental garbage collection, first class functions/closures, a small object oriented class system, a mini GUI builder for creating a patch control panel, a graphical interface for creating wave tables and breakpoint envelopes, MIDI control, a large library of signal processing and synthesis functions, and a large library of functions for list processing of musical data. The user can write both the synthesis and compositional algorithms for their piece in the same high level language.* [McCartney, 1996]

Dans *SuperCollider*, la gestion du temps informatique se scinde en deux modalités : le taux audio (*audio rate*) `.ar` et le taux de contrôle (*« kontrol » rate*) `.kr`, comme dans l'exemple suivant.

```
{ SinOsc.ar(200 * MouseX.kr(1, 100)); }.play;
```

Par rapport à la performance, le fait que la version actuelle (« SC3 ») s'articule autour d'une architecture client-serveur avec deux applications séparées (l'application-serveur, qui représente le moteur de synthèse audio, et l'application-client qui permet d'écrire le code et de l'envoyer au serveur, via le protocole OSC) permet entre autres le *live-coding*[32]. Enfin, par rapport à la composition, James McCartney, en précisant ses motivations, induit la possibilité formelle de transposer la notion d'héritage, propre aux langages de programmation objets, aux œuvres musicales, au-delà du processus de l'interprétation musicale, ce qui modifie potentiellement la notion du temps de la composition :

> *I wrote SuperCollider because I was interested in listening to music that was different each time it was played. I also wanted to be able to specify a class of compositions and then listen to instances of the class.* [McCartney, 2003]

ChucK : *ChucK* se distingue immédiatement des autres langages par son parti pris tranché pour la programmation en direct, et, dans ce sens, il s'agit du langage le plus interactif des trois. *ChucK* est en effet un langage de programmation audio *concurrent* et *à-la-volée* :

> *On-the-fly programming is a style of programming in which the programmer/performer/composer augments and modifies the program while it is running, without stopping or restarting, in order to assert expressive, programmable control for performance, composition, and experimentation at run-time. Because of the fundamental powers of programming languages, we believe the technical and aesthetic aspects of on-the-fly programming are worth exploring.* [Wang et Cook, 2007, p. 2]
>
> *In fact, coding, composing, and performing are identical in ChucK.* [Wang et Cook, 2003]

Cette activité de « programmation à-la-volée » s'inscrit manifestement parmi celles qui renouvellent la dichotomie traditionnelle entre temps réel et temps différé, et celle encore plus ancienne entre composition et interprétation… Il nous semble que la programmation renoue ici avec l'improvisation, à ceci près qu'il s'agit de *codes informatiques* improvisés – donc d'une écriture – et non pas seulement de

32. L'expression *« live-coding »* ne possède pas encore de traduction officielle ; nous proposons « programmation en direct ».

gestes physiques sur un corps vibratoire. D'ailleurs, la programmation *à-la-volée* (« programmation interactive », « programmation en direct » ou *« live-coding »* sont des synonymes) ne se confond pas exactement avec le temps réel, car l'étape de l'écriture du code dans une ligne de commande reste inaudible, seule la validation du code d'un processus donne à entendre ce processus programmé en temps réel, par le truchement de la machine virtuelle. En outre, ces travaux postulent implicitement que la programmation est éligible au statut de performance musicale :

> *Indeed, we are motivated in the design of ChucK to investigate new ways of thinking about programming sound and music, particularly by looking at it from a human-centric perspective (e.g., as opposed to a machine-centric one). As we posited above, a programming language is a highly general and yet highly intimate human-computer interaction (HCI) device. If that is the case, then perhaps we can think of the task of programming language design as HCI design – loosely speaking.* [Wang, 2008, p. 4]
>
> *This style of development has led to applications in prototyping, teaching, and live musical performance where the audience observes the « live code » as musical gestures.* [Wang, 2008, p. 6]

ChucK se distingue aussi par *l'unification temporelle* du langage qu'il propose. Contrairement à *Max/MSP* et *SuperCollider*, qui opèrent une séparation entre deux fréquences d'échantillonnage pour l'audio et pour le contrôle, *ChucK* instaure un continuum logique non-borné, éventuellement inférieur à l'échantillon :

> *[...] there are no restrictions (other than underlying floating point precision) on how much time is advanced. So it is possible to advance time by a microsecond, a samp, 2 hours, or 10 years. The system will behave accordingly and deterministically. This mechanism allows time to be controlled at any desired rate, according to any programmable pattern. With respect to sound synthesis, it is possible to control any unit generator at literally any rate, even sub-sample rate. The power of the timing mechanism is extended by the ability to write parallel code [...]* [Wang et Cook, 2007, p. 73]

Les derniers développements ont permis d'intégrer l'analyse en temps réel [Wang *et al.*, 2007].

Du câblage virtuel

L'architecture multi-couche des ordinateurs ne simplifie pas la prise en charge de la latence. En particulier, au niveau logiciel, deux types de communications audionumériques peuvent avoir lieu : la communication entre une application et un périphérique matériel, au travers d'un pilote ou d'un gestionnaire de pilote, et la communication entre deux applications. Le premier type est souvent géré par une couche système (CoreAudio, DirectX, Jack) étoffée par les pilotes des cartes matérielles, tandis que le deuxième type voit s'affronter plusieurs technologies concurrentes (ASIO, ReWire, SoundFlower, etc.) devant le double problème de la latence et de la synchronisation (du son et des événements de contrôle).

Le foisonnement des plugiciels

Certaines pratiques musicales font un usage extensif des plugiciels ou *plug-ins*[33]. Ces composants logiciels réduits s'ajoutent à une application hôte, et remplissent généralement une fonction isolée et identifiable parmi les effets audionumériques traditionnels (compression, égalisation, réverbération, écho, chorus, etc.), simulent des instruments virtuels (synthétiseurs, boîtes à rythmes, arpeggiateurs, échantillonneurs, etc.), présentent des analyseurs (d'amplitude ou de spectre), ou proposent d'autres fonctions plus originales[34]. Les plugiciels dits « natifs » utilisent directement l'unité centrale de l'ordinateur pour effectuer les calculs, tandis que les autres utilisent des processeurs dédiés intégrés sur des cartes externes, les DSP[35]. Plusieurs formats de plugiciels audio existent, selon la marque à l'origine de leur développement : AU (*Audio Unit* d'Apple), DirectX (API multimédia de Windows), MAS (*MOTU Audio System*), RTAS (*Real Time Audio Suite*), TDM (*Time Division Multiplexing* utilisent la puissance de calcul des cartes DSP Digidesign), et VST (*Virtual Studio Technology* développée en 1996 par Steinberg[36]).

On distingue traditionnellement deux catégories de plugiciels : les plugiciels « en temps réel », qui permettent d'écouter sans délai perceptible toutes les modifications du son qui interviennent au fur et à mesure, et les plugiciels « en recalcul » ou « en temps différé », qui permettent d'économiser la puissance du processeur de l'ordinateur hôte en substituant au fichier son original le nouveau fichier son traité.

Finalement, pour apprécier le temps de latence global lors de l'utilisation de plugiciels, il faut prendre en compte les facteurs suivants : le format, le mode natif ou DSP additionnel, la largeur des mémoires tampon, la rapidité du système de fenêtrage d'analyse ou de synthèse, la charge de calcul et les accès au disque dur.

2.3.3 Les « méta-temps » informatiques

Latence, siamoise du quartz

On a montré précédemment comment le temps de latence intervient dans la définition du temps réel et du temps différé dans le champs *technique*. Cependant, de même que le temps réel, la latence revêt plusieurs significations selon le contexte. Le temps de latence responsable du délai lors des interactions homme-machine provient de plusieurs facteurs :

> La latence d'un système numérique de traitement du son a plusieurs origines : on peut identifier le système matériel d'entrées-sorties (temps de transfert par accès

33. Pour des précisions terminologiques, voir la section 9.2.1 page 189.

34. le système *Pluggo* de Cycling'74 permet de réaliser ses propres plugiciels avec *Max/MSP*.

35. *« Digital Signal Processor »* ou processeur de signal numérique. Le traitement du signal faisant essentiellement appel à des instructions mathématiques, les DSP peuvent se borner à un jeu d'instructions adapté pour lequel ils seront efficaces, alors que le processeur central d'un ordinateur doit rester polyvalent.

36. Steinberg proposa un kit de développement gratuit sur internet, favorisant ainsi son adoption rapide par un très grand nombre de logiciels audio, et en fit un standard de fait, ainsi que des instruments virtuels appelés VSTi.

> direct mémoire), le système d'exploitation (temps de prise en compte d'une interruption matérielle) et enfin l'implantation de l'algorithme de traitement. [Déchelle, 1999, p. 86]

Les cartes son introduisent leur propre temps de latence en fonctions de la taille de leurs tampons internes, ainsi que chaque interface électronique et informatique en réalité : une lettre tapée au clavier n'est pas affichée instantanément, à cause de la mécanique et de l'électronique du clavier, du protocole de communication, de la période de la boucle d'écoute du système d'exploitation, de la gestion des priorités par ce même système d'exploitation, et de l'électronique de l'écran. De façon analogue, une information MIDI qui arrive à l'ordinateur ne se traduit pas immédiatement en résultat sonore, en fonction de chaque étape du système global et de son câblage ; même les haut-parleurs possèdent un temps de réponse propre, en fonction de l'inertie des parties mécaniques.

Pour mesurer le temps de latence d'un système global, on doit prendre en compte tous les paramètres susceptibles d'introduire leur temps de latence propre, qui s'ajoute en général aux autres[37]. Dans le cas des logiciels, d'autres sources de latence que la taille du tampon audio peuvent apparaître, en particulier avec les effets sonores qui utilisent des fenêtres temporelles pour de l'analyse-synthèse, comme la FFT, que ce soit sous forme de logiciels autonomes ou de modules additionnels (*plugiciels*).

Les auteurs du système de programmation événementiel MidiShare en proposent la suivante :

> Nous désignons ici la latence comme étant le délai qui sépare la date d'échéance d'une tâche de sa réalisation effective. Ces délais peuvent intervenir à tous les niveaux dans la chaîne de traitement d'un message musical : quand l'ordonnanceur du système d'exploitation est en jeu (particulièrement dans le cas des langages non temps réel tels que Lisp ou Java) ou encore quand les événements sont délivrés via un canal de communication dont les temps de transmission ne sont pas déterministes (par exemple : cas des transmissions sur un réseau). [Fober *et al.*, 2004]

Il faut souligner qu'il n'y a pas de numérique sans latence, même lorsqu'on ne la perçoit pas. Le numérique compte des avantages certains par rapport à l'analogique – la reproductibilité à l'identique, la mémoire des opérations – mais la latence du numérique reste en quelque sorte « le prix à payer », pour des raisons intrinsèques. Pourtant, ces deux technologies n'accusent pas une différence de nature fondamentale au niveau matériel, puisqu'elles puisent toutes les deux à la même source de l'électronique. Au-delà de la taille des composants électroniques généralement utilisés, la différence subtile réside principalement dans un unique composant électronique : l'horloge, composant au nom révélateur vis-à-vis de la latence.

> L'horloge est un circuit qui émet une suite d'impulsions de durée précise, l'intervalle entre deux impulsions consécutives étant lui aussi très précis. [...] Les fréquences

37. La société allemande Ploytec commercialise depuis 2007 un instrument qui permet de mesurer avec une bonne précision la latence d'un système audio (le *Latenc-o-meter*). Cet appareil génère un clic qu'il envoie dans l'installation afin de mesurer le délai temporel entre son émission et sa réception, le résultat s'affichant alors sur l'écran en millisecondes, avec une précision de deux chiffres après la virgule grâce au taux d'échantillonnage interne de 100 kHz.

> d'impulsion sont généralement comprises entre 1 et 500 MHz, soit des cycles d'horloge de 1000 à 2 ns. Afin de garantir une excellente précision, les fréquences des horloges sont le plus souvent contrôlées par des oscillateurs à quartz. [Tanenbaum, 2005, p.168]

Un strabisme des champs

Les questions temporelles en général, du fait de leur résistance à la définition, font émerger des regards divergents sur elles selon *qui* les regarde...

D'abord, nous l'avons vu, le temps de latence des logiciels recèle plusieurs facettes, matérielles et logicielles. Ensuite, nous pensons qu'il existe plusieurs approches de ce temps de latence selon l'activité concernée, et en particulier que les ingénieurs et les scientifiques n'en partagent pas la même vision.

- Les premiers, pris dans un opératoire *technologique*, doivent respecter un cahier des charges de type « industriel », avec des objectifs programmés et déterminés du point de vue quantitatif ; par exemple, un seuil de latence sera déterminé par une valeur précise, finie et suffisante. La latence est ici une contrainte parmi d'autres.
- Les seconds, pris dans une quête de vérité *scientifique*, évoluent avec un cahier des charges plutôt « récursif », renouvelant leurs recherches dès qu'un palier est atteint. La latence devient un sujet de recherche en soi.
- En art, il s'agit encore d'autre chose : d'un opératoire *esthétique*, et, passant, d'un cahier des charges « flou ». L'élaboration d'une œuvre se fait avec la latence du système utilisé, éventuellement en la détournant... La latence reste là principalement une contingence, au mieux un ressort créatif.

Ainsi, la multiplicité et l'hétérogénéité des domaines concernés – art, science, et technologie – engendrent une multiplicité des points de vue et des définitions du temps de latence, du temps réel, et du temps différé.

2.3.4 Une impermanence caractéristique

La « loi » de Moore et sa croissance inédite (doublement tous les 18 mois) justifie le phénomène de migration progressive de certains logiciels musicaux réputés temps différé vers le temps réel, à l'instar de *Csound*, partiellement ou complètement. Cette loi, ou plus exactement cette prédiction qui ne s'est pas vue démentie pendant 30 ans, s'applique rétrospectivement à la fois à la croissance de la fréquence d'horloge des ordinateurs et à la croissance des capacités de stockage[38].

Cette croissance exponentielle et vertigineuse est responsable d'une impermanence caractéristique des frontières entre les catégories informatiques de haut niveau. Ainsi,

38. « Tout comme les gains en nombre de transistors par puce, les progrès des autres aspects technologiques de l'informatique ont été phénoménaux. [...] Ces gains énormes en termes de performances de disque dur, associés au fait que le volume financier représenté par les disques en provenance de la Silicon Valley a dépassé celui des puces des unités centrales, font dire à Al Hoagland que la "vallée du silicium" a été incorrectement nommée : on devrait plutôt parler de la "vallée de l'oxyde de fer" (nom du support d'enregistrement utilisé dans les disques). » [Tanenbaum, 2005, p. 29]

techniquement, le champ d'application du temps réel n'a cessé de s'étendre au fur et à mesure que la puissance de calcul lui permettait d'effectuer des calculs d'une méthode de synthèse appartenant auparavant au champ du temps différé. Miller Puckette évoque aussi l'éventualité de la disparition future d'une frontière entre deux autres catégories, jusque là fondamentales en l'informatique musicale :

> *Related to this, perhaps even a case of the trend toward interoperation, is the growing involvement of CAC*[39] *in manipulating sounds directly (not through the mediation of a score). Such work lies simultaneously within CGM*[40] *and CAC, and in one possible future the distinction between the two will simply disappear. This is in high contrast to the early days of CAC in which the output was a stack of punched cards, or even to the situation only ten years ago, in which Patchwork users needed MIDI hardware to hear their musical ideas.* [Puckette, 2006]

De plus, si la distinction entre la synthèse sonore et la CMAO risque de disparaître, la distinction entre le temps réel et le temps différé a déjà muté plusieurs fois : d'abord lors du passage du champ *technique* au champ *pratique*, puis lors du passage du champs *pratique* au champ *musical* (ou *esthétique*). Ces mutations sémantiques semblent correspondre à des sauts quantitatifs importants dans l'évolution de l'informatique.

Ainsi l'impermanence caractéristique de l'informatique joue-t-elle un rôle majeur dans l'obsolescence technologique et dans l'instabilité du vocabulaire qui en dépend, entraînant dans le cas du temps réel et du temps différé une remise en cause de la possibilité d'une frontière clairement identifiable et pérenne.

2.4 Des contradictions croisées

De plus en plus d'observateurs relèvent des contradictions dans différents discours sur le temps réel ou le temps différé, et en particulier sur les logiciels d'informatique musicale. Cette section rappelle donc en premier lieu la valeur paradigmatique qu'ont pris certains logiciels pour le temps réel ou pour le temps différé ; puis elle revient séparément sur des contradictions fréquemment évoquées à propos du temps réel et du temps différé, des idées typiquement proches des énoncés « le temps réel nécessite du temps différé » ou « le temps différé s'opère de plus en plus en temps réel » ; elle propose enfin une explication de ces contradictions à travers l'enchevêtrement des niveaux polysémiques.

2.4.1 La valeur paradigmatique de certains logiciels

Comme on peut le constater d'après les sections précédentes, les notions de temps réel et de temps différé restent relativement floues en général, jusqu'à ce qu'elles s'incarnent dans des logiciels particuliers, dont certains prennent alors une valeur paradigmatique. Ainsi, *Max/MSP* et ses cousins *jMax* et *PureData* – ou plus simplement le paradigme « Max[41] » – viennent souvent s'opposer à *OpenMusic*, son ancêtre Patchwork ou encore la série des Music N dans le discours, précisément par rapport à la notion de temps réel :

39. *Computer Aided Composition.*
40. *Computer Generated Music.*

> *The semantic of OpenMusic (and Patchwork) is one of demand-driven dataflow. Each object is essentially a function call, which recursively evaluates its inputs, precisely as a Lisp form is evaluated. Compared to Pd, the relationship between data and process is reversed. There is no notion of real-time events or even of real time itself; rather, the contents of a patch are static data.* [Puckette, 2004]

Notons qu'il en va de même à propos des logiciels du commerce. Ainsi, *Live* est régulièrement associé au temps réel dans le discours, à commencer par son nom qui fait évidemment référence au *« live »* tel que ce mot s'est intégré dans le vocabulaire, y compris en France, à partir des expressions *« live music »* et *« live electronic(s) »*. En opposition, les logiciels de montage (comme ProTools) ou encore d'édition de partition (comme Finale) sont volontiers associés au temps différé.

2.4.2 Le temps réel nécessite du temps différé

L'utilisation du logiciel *Live* mise en avant est incontestablement la situation de concert, même si ses capacités de montage sont parfois évoquées ; *Live* serait en quelque sorte « le » logiciel temps réel par excellence. Or, des observateurs de plus en plus nombreux font malicieusement remarquer que la part du temps différé dans *Live* est *primordiale*, puisque son fonctionnement repose essentiellement sur l'utilisation de fichiers audio pré-enregistrés ou de séquences MIDI... Ce cas particulier souvent repris reflète une critique bien plus large, qui dénonce en réalité un abus de langage quant au « temps réel » :

> Ce qu'on appelle alors « temps réel » dans la composition musicale serait un abus de langage puisqu'une part des composants musicaux est souvent déjà fixée, et n'a pas pour vocation à varier d'une interprétation à l'autre. [Manoury, 2007, p. 8]

Cette critique de Philippe Manoury se fonde sur son analyse du temps réel à partir de l'interprétation musicale, et pointe la *fixité* des éléments pré-enregistrés, qui entre en contradiction avec l'interprétation comme *variabilité*.

On peut ajouter à cette remarque le fait que ces logiciels dits « temps réel » demandent paradoxalement un temps de préparation souvent important. Dans le cas de *Live*, il s'agit de la préparation des configurations. Dans celui de *Max/MSP*, il ne s'agit de rien de moins que de la *programmation* d'un patch [42], ce qui peut demander un temps tout à fait

41. *« I have worked for many years on a computer environment for realizing works of live electronic music, which I named in honor of Max Mathews. Three currently supported computer programs – Max/MSP, Jmax, and Pd – can be considered as extended implementations of a common paradigm I'll refer to here as "Max." [...] The Max paradigm can be described as a way of combining pre-designed building blocks into configurations useful for real-time computer music performance. »* [Puckette, 2002]

42. Le préjugé selon lequel programmer en *Max* serait facile se révèle souvent faux : cela dépend du type d'application à réaliser, et surtout de la taille du patch. Miller Puckette, s'il ne reconnaît pas *Max* comme un véritable langage informatique, reconnaît néanmoins que « faire un patch » s'apparente bien à une activité de programmation ; seul le niveau d'abstraction change : *« a patch is itself a program, and the job of connecting simple functions together to make larger ones can be thought of as programming. In this sense, the trend toward patch-based software can be seen as shifting the level on which the user programs the computer away from the C or Lisp code itself and into the "language" of patches. It may be that this is fundamentally a better level at which to operate a computer, than either that of code or that of the user of a large, monolithic program such as a database application. »* [Puckette, 2006]

considérable... Or, de toute évidence, cette préparation relève du temps différé (au sens pratique), en se rapprochant de la composition : le choix de séquences et la constitution de configurations verticales potentielles dans le cas de *Live*, comme la programmation de circuits sonores réactifs et la mise en correspondance des paramètres dans le cas de *Max/MSP*, sont autant de choix compositionnels et d'écriture, c'est-à-dire de codes enregistrés. Surtout, le résultat de ces codes préparés n'est pas destiné à être entendu sur-le-champ, mais bien *ultérieurement*, ce qui les rend révisables et constitue une marque indubitable du temps différé.

2.4.3 Le temps différé s'opère de plus en plus en temps réel

Concernant la synthèse sonore, il ne fait aucun doute que la synthèse en temps différé se raréfie au profit de la synthèse en temps réel. De fait, *Csound*, l'héritier des langages *Music* N à forte valeur paradigmatique temps différé, est déjà passé à sa version temps réel en 1990. Les bénéfices de cet accès au son en temps réel ont d'ailleurs été largement plébiscités :

> *As performance increases, things that were impossible become possible. [...] When you can create sound in real time you can interact with it, and interaction makes possible things that never could have been achieved by typing in lists of commands.* [McCartney, 2003]

Mais le temps différé ne se limite pas à la synthèse. Par exemple, Miller Puckette a expliqué récemment comment, contrairement à la synthèse, la CMAO[43] n'est pas devenue temps réel :

> *In general, the cost of CGM per audio sample has not remained constant, but has not grown quickly. The best CGM of the seventies, thirty years ago say, probably cost less than ten thousand arithmetic operations per sample of output. The speedup in computing in the intervening years has allowed us the luxury of running the classic CGM algorithms in real time, thus changing the nature of the pursuit fundamentally. The algorithms themselves have grown somewhat more complex as well, but the universal preference for real-time synthesis and processing has put a lid on that growth.*
>
> *CAC has not become real-time at all. Since it appeals to the composer in us (whereas CGM appeals to the performer in us), it seems reasonable to expect that CAC software will continue to absorb all the computing resources that can possibly be brought to bear on it. Anything allowed to grow will naturally do so.* [Puckette, 2006]

Si actuellement les ressources de calcul ne permettent pas à la CMAO de se faire en temps réel, comme l'indique Miller Puckette, il n'en reste pas moins qu'un degré d'interaction de plus en plus important apparaît dans les applications temps différé. Signalons par exemple les travaux de l'équipe « Représentations musicales » de l'IRCAM sur l'intercommunication entre *OpenMusic* et *Max/MSP*, dont la bibliothèque OSC *Moscou* pour

43. Miller Puckette emploie *CGM* (*Computer Generated Music*) pour « synthèse sonore » et *CAC* (*Computer Aided Composition*) pour CMAO.

OpenMusic [Seleborg, 2004, p. 36, avec Carlos Agon] et le projet *OMax*[44], qui fait collaborer *Max/MSP* et *OpenMusic* pour former un « co-improvisateur » par « réinjection stylistique ».

2.4.4 La confusion des niveaux polysémiques

Deux types de contradictions ont été relevées : un temps réel qui comporte en réalité une grande part de temps différé, et réciproquement un temps différé qui comporte à son tour une part importante de temps réel. Or ces contradictions peuvent être levées après un examen des niveaux sémantiques.

Les reproches avancés à l'encontre de *Live* par exemple, c'est-à-dire la fixité des séquences employées et le temps important de préparation, appartiennent au niveau du temps différé *pratique*, ce qui n'empêche pas le logiciel *Live* d'appartenir au niveau du temps réel *musical*, en accord avec son utilisation musicale de type « performance ».

De la même façon, les fonctionnalités interactives d'*OpenMusic*, appartenant au niveau du temps réel *pratique*, n'invalident pas la catégorisation consensuelle de cet environnement dans le niveau du temps différé *musical*, en accord avec son utilisation musicale de type « composition ».

En réalité, à bien y regarder, un logiciel destiné à la performance aura d'autant plus de chances d'être opérationnel et efficace en situation de direct (temps réel technique, pratique et musical) qu'il aura bénéficié d'une préparation importante au préalable (temps différé pratique). Inversement, un logiciel destiné à la composition aura d'autant plus de chances d'être opérationnel et efficace dans une situation d'écriture (temps différé pratique et musical) qu'il offrira des fonctionnalités interactives (temps réel technique et pratique)...

* * *

On peut retenir deux points importants de cette analyse des contradictions. D'une part, les contradictions apparentes entre le temps réel et le temps différé proviennent souvent d'une confusion croisée entre les différents niveaux polysémiques de ces deux notions. D'autre part, on peut constater que c'est le niveau *musical* qui fait finalement le plus consensus dans notre domaine de l'informatique musicale, c'est-à-dire le niveau le plus abstrait.

44. *« OMax is a software environment which learns in real-time typical features of a musician's style and plays along with him interactively, giving the flavor of a machine co-improvisation. OMax uses OpenMusic and Max. It is based on researches on stylistic modeling carried out by Gerard Assayag and Shlomo Dubnov and on researches on improvisation with the computer by G. Assayag, M. Chemillier and G. Bloch (Aka the OMax Brothers) in the Ircam Music Representations group. »* http://recherche.ircam.fr/equipes/repmus/OMax/

Chapitre 3

Vers un axe ordonné, tendu entre deux pôles idéaux...

Résumé du chapitre : Temps réel et temps différé recouvrent aujourd'hui à la fois des notions *technologiques*, pour classer les logiciels d'informatique musicale, et des notions *musicales*, par exemple pour distinguer certains courants esthétiques par rapport à l'usage de l'ordinateur en musique. Cette superposition des sens technologiques et esthétiques, mouvante, ajoute à la relative ambiguïté de ces deux catégories. Aussi, afin de préciser la signification des concepts de « temps réel » et de « temps différé », nous essaierons dans ce chapitre de nous dégager de la définition ensembliste dichotomique traditionnelle, certes historique et consensuelle, mais sûrement étroite, en construisant un autre modèle, axial.

3.1 Deux idéaux qui renouvellent la réflexion

Dans ce chapitre, « l'idéalité » des deux pôles « temps réel » et « temps différé » n'est pas à considérer dans son acception commune de perfection, mais dans celle d'*abstraction*, c'est-à-dire en tant qu'elle échappe à un certain niveau concret de la réalité. Nous introduisons ce caractère d'idéalité pour pouvoir formaliser et proposer un nouveau modèle, visant *in fine* à dépasser la taxonomie binaire traditionnelle temps réel / temps différé et à stimuler l'élaboration d'une taxonomie plus riche. Le trait pourra paraître un peu forcé sur ce caractère idéal, mais il ne s'agit que de montrer *la possibilité* d'un tel point de vue, qui sera ensuite enchassé dans un travail de formalisation axiale plus large.

3.1.1 Le temps réel comme idéal

Nous proposons que la notion de temps réel accède au statut d'*idéal*, au triple titre d'idéal technique, pratique et musical.

La latence nulle : un idéal technique

Nous avons vu dès le début de ce mémoire (section 1.1 page 8) que les technologies informatiques restent dépendantes de la vitesse de l'électricité dans les métaux conducteurs, comme première limite, et au mieux de la vitesse de la lumière comme seconde limite (assez proche de la précédente d'ailleurs). En réalité, la puissance des ordinateurs connaît déjà d'autres limites en amont, et elle se déduit habituellement à partir du nombre de transistors par puce et de la fréquence d'horloge de l'unité centrale.

Parmi tous les facteurs qui entrent en ligne de compte pour évaluer les aptitudes d'un système audionumérique au temps réel technique, le facteur de la durée d'un cycle d'horloge reste le plus petit intervalle temporel. Or, cette durée ne pouvant être nulle, l'instantanéité absolue du point de vue *technique* est nécessairement inaccessible, tout en constituant un objectif technologique de « latence nulle » – nous dirons alors que le temps réel technique est un *idéal* technique.

L'instantanéité : un idéal pratique

La latence, dans sa dimension technologique pratique, constitue un point de séparation factuel entre temps réel et temps différé : la caractéristique *interactive* s'évanouit aussitôt que le temps de latence devient perceptible. Autrement dit, le temps réel signe une condition nécessaire d'un territoire pratique, ou praticable. Ainsi, du point de vue des pratiques instrumentales comme de celui des pratiques compositionnelles, le temps réel semble faire consensus depuis longtemps :

> S'il s'agit d'opposer synthèses en *temps réel* et en *temps différé*, je préfère comme tout le monde, le temps réel – dans la mesure du possible. [Murail, 1992, p. 28]

L'incise « dans la mesure du possible » ne laisse ici aucun doute sur le caractère *idéal* du temps réel pratique.

La richessse de la rétroaction : un idéal musical

De surcroît, nous avons montré précédemment le glissement de la notion de temps réel vers la fonction musicale (section 1.3 page 29). Depuis ce glissement, c'est-à-dire depuis que des compositeurs et des musiciens envisagent *musicalement* une idée du temps réel (parfois même qualifié de « pur »[1]), la notion de temps réel a glissé vers un *idéal musical*, esthétique ou esthétisant.

> Quoi qu'il en soit, les désavantages de la synthèse logicielle différée sont évidents. On perd du temps à attendre que les échantillons soient calculés. Le son est déconnecté des gestes humains en temps réel – nous ne pouvons pas modeler le son tout en l'entendant se transformer. Le style guindé de certaines musiques informatiques provient de cette situation fâcheuse. L'avantage de la programmabilité devient un désavantage lorsque nous devons coder des phrases musicales simples avec autant d'efforts que les plus compliquées. Même une enveloppe banale peut demander de précalculer et d'entrer des douzaines de nombres. La synthèse logicielle différée est un « chemin difficile » vers la musique. [Roads, 1998, p. 72]

Cette citation montre à nouveau combien le temps réel a été perçu comme un *idéal* à atteindre, à la fois freiné par les limites de la puissance de calcul, et libérant, à travers la rétroaction, les possibilités expressives de la création musicale et de la musique elle-même.

3.1.2 Le temps différé comme idéal

À l'instar de la notion de temps réel, la notion de temps différé recèle plusieurs points de fuite susceptibles de la faire basculer du statut de catégorie à un statut d'idéal, dans les trois champs, *technique*, *pratique* et *musical*.

La latence infinie : une représentation technique idéale

Une réflexion semble indiquer pertinemment une idéalité du temps différé technique, à partir de l'opposition logique avec le temps réel technique idéal, défini précédemment comme une latence nulle. En effet, par symétrie, le temps différé technique idéal devient alors une « latence infinie »... Si ce concept peut paraître saugrenu de prime abord, il fait cependant sens du point de vue de l'ingénierie informatique.

Typiquement, les critères d'évaluation de la vitesse d'un algorithme employés dans la programmation par contraintes projettent les temps de calcul « trop » importants (par exemple issus des méthodes de recherche exhaustives) sur une représentation ou un symbole *unique*, comme la mention « $>$1h » du tableau 3.1 page suivante[2]. Or, cette re-

1. « Ainsi, au terme de sa "déduction" [de la notion de temps réel à partir d'une analyse de la notation musicale dans ses rapports avec l'interprétation], Manoury dégage, pour la composition, *un concept musical du temps réel pur*. [...] C'est toutefois, pour Manoury, *au suivi de partitions* qu'il convient de réserver le qualificatif de temps réel, dans son acception compositionnelle la plus *pure*. » [Szendy, 1998, c'est l'auteur qui souligne]

2. Ce tableau est reproduit d'après l'article *CAO et contraintes* [Truchet *et al.*, 2001, p. 90]. Il compare les temps de calcul de la recherche adaptative avec ceux de la méthode traditionnelle par *backtracking*

présentation, de par son unicité, révèle une *idéalité* : celle d'un temps considéré comme infiniment long – une « latence infinie ».

	Backtracking	*Adaptative*	
Nombre de voix	Temps	Itérations	Temps
3	>1h	350	12 s
4	>1h	463	22 s
5	>1h	923	58 s
6	>1h	1208	108 s

Tab. 3.1 – *Temps de calcul comparés pour un quasi-canon rythmique contraint*

Le retour perpétuel : une idéalité de la pratique

Sur le plan pratique, plusieurs réflexions peuvent conduire à considérer un aspect idéal dans la notion de temps différé. Le fait que l'interaction ait aujourd'hui largement investi les pratiques du temps différé, introduisant en quelque sorte du temps réel dans le temps différé systématiquement, tend à montrer qu'il n'existe plus de temps différé « pur » qu'à le considérer comme *idéal*.

Mais au niveau du processus de création, on peut se demander ce que deviendrait la possibilité de retour qui caractérise le temps différé (cf. section 1.3.4 page 34) si on imagine un temps différé « absolu » ? Elle deviendrait probablement quelque chose de comparable au mythe de Sisyphe, où le compositeur reviendrait indéfiniment sur la même tâche (dans les deux sens du terme, travail et trace). Or pour qu'il y ait œuvre, même dans le cas d'une pensée de type *work in progress*, il faut nécessairement sortir du retour perpétuel pour entrer à un moment donné dans la *présentation*, qu'elle soit diffusée, interprétée, copiée ou gravée.

Pouvoir prendre le temps : un idéal musical

En réaction à l'engouement pour le temps réel, certains auteurs n'ont pas hésité à brocarder ses lacunes ou ses limites. Ainsi, comme le préconise Luc Rondeleux, « dans l'absolu, il faut pouvoir prendre le temps » :

> Les progrès des composants électroniques plus performants – accroissement de la vitesse de calcul, diminution du dégagement thermique, de l'encombrement... – sont à l'origine d'une évolution marquante du matériau musical et de la standardisation des instruments de synthèse : c'est l'apport principal du temps réel. La technique passe de l'enregistrement à la musique mixte puis des systèmes de synthèse hybride aux systèmes temps réel. Au fur et à mesure, l'exigence de souplesse et de rapidité se traduit surtout par une impatience devant l'instrument de production. Car le temps réel se révèle être un leurre : en limitant le traitement du matériau au contrôle de quelques paramètres, il limite la marge de manœuvre du musicien. Le processus de synthèse demeure immuable et il ne permet pas d'orienter les formalismes.

(le « retour arrière »), sur un problème de rythmes sans simultanéité dans un quasi-canon rythmique, problème posé par le compositeur Mauro Lanza.

> La composition se situe hors du temps linéaire ; elle n'a pas à se soucier des réponses immédiates. Seule une limitation de la mémoire auditive impose des écoutes immédiates pour mieux jauger, estimer, comparer… Dans l'absolu, il faut pouvoir prendre le temps de spécifier dans les moindres détails de la microstructure les caractéristiques des sons recherchés. Par conséquent le compositeur se méfie d'un temps figé. [Rondeleux, 1999, p. 7]

Cependant, comme toute œuvre doit sortir de son cycle d'élaboration un jour pour exister en tant qu'œuvre, il n'est pas possible de prendre le temps indéfiniment pour « spécifier dans les moindres détails de la microstructure les caractéristiques des sons recherchés » : c'est seulement un *idéal*.

* * *

Cette section avait pour but de montrer qu'il est possible d'envisager le temps réel et le temps différé comme deux *idéaux*, inaccessibles, afin de pouvoir proposer dans les sections suivantes un nouveau modèle pour appréhender différemment ces notions. Ainsi, nous avons déployé une réflexion qui projette le temps réel et le temps différé sur les trois champs sémantiques associés, résumée dans le tableau 3.2.

	Temps réel	Temps différé
Champ technique	latence nulle	latence infinie
Champ pratique	instantanéité	retour perpétuel
Champ musical	richesse de la rétroaction	pouvoir prendre le temps

Tab. 3.2 – *Synthèse des statuts idéaux du temps réel et du temps différé*

3.2 Un modèle axial ordonné

Est-ce qu'un autre modèle que le réflexe ensembliste permettrait de reconsidérer le couple temps réel / temps différé sous un jour intéressant ? Existe-t-il une alternative plus fine à cette classification dichotomique rudimentaire ?

3.2.1 Une représentation axiale…

On peut en effet effectuer une opération conceptuelle de basculement, à partir de la possibilité de changement de statut développée précédemment (cf. section 3.1 page 66) : il suffit d'envisager le couple « temps réel » / « temps différé » non plus comme *deux sous-ensembles* complémentaires parmi l'ensemble des logiciels d'informatique musicale, mais comme *deux pôles idéaux* tirés à chaque extrémité d'un axe.

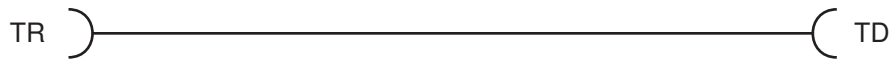


FIG. 3.1 – *Un axe, tendu entre deux pôles idéaux*

L'exclusion des extrémités

Les deux pôles doivent être exclus de l'axe qu'ils définissent, étant donné la différence de nature inconciliable entre les pôles et les logiciels ; par conséquent, aucun logiciel ne pourra strictement se confondre avec l'un des deux pôles, qui sont de toute façon définis comme *idéaux*. Sous cette condition d'exclusion, le pôle « temps réel » apparaît comme un zéro inaccessible, c'est-à-dire une latence idéalement nulle, tandis que le pôle « temps différé » apparaît comme un infini inaccessible lui aussi, c'est-à-dire comme une latence idéalement infinie.

Deux questions peuvent alors se poser à ce stade de la réflexion :

1. est-il possible de normer cet axe ?
2. ne serait-il pas préférable d'utiliser un plan à deux dimensions, une pour le temps réel et une pour le temps différé ?

Pas de norme pour le non-trivial

La première question revient à se demander s'il existe *une unité* pour mesurer une grandeur pertinente entre temps réel et temps différé, et si cette grandeur existe. On peut admettre pour l'instant que la notion de latence représente suffisamment bien cette grandeur (ce point sera étudié à la section 3.2.1 page ci-contre) ; reste alors la question de l'unité de mesure.

Pour le niveau technique, la durée d'un échantillon audionumérique, de l'ordre de 23 μs pour une fréquence d'échantillonnage de 44 100 Hz pourrait convenir. De même, pour le niveau pratique, il est possible de trouver une unité, par exemple 1 ms ou 10 ms, en fonction des besoins. En revanche, pour le niveau musical ou esthétique, il semble impossible de trouver une unité de mesure valable, car ce niveau relève davantage de conventions sociales ou de représentations subjectives que d'un temps mesurable.

Or, pour le classement des logiciels, ce niveau musical nous intéressera au premier chef parce qu'il est le seul niveau non-trivial, les niveaux technique et pratique relevant du simple test de performance (ou *benchmark*). En effet, nous verrons que dans la plupart des utilisations, la taille des vecteurs audionumériques n'est pas un critère discriminant entre les logiciels (niveau technique), ni, du même coup, la latence globale d'un logiciel (niveau pratique).

Par conséquent, la réponse à la première question doit être « non » : dans les cas qui nous intéressent, il n'est pas possible de normer cet axe. Cela implique notamment que, faute de repères *absolus*, tout positionnement sur l'axe devra nécessairement se faire de manière *relative*.

Une intégration unidimensionnelle

Le plan bidimensionnel suggéré dans la seconde question est tout aussi formalisable *a priori* qu'un axe unidimensionnel : on peut par exemple imaginer un quart de plan formé par deux demi-droites, une pour le temps réel et une pour le temps différé. Dans ce modèle bidimensionnel, chaque élément aurait deux coordonnées, ce qui présenterait pour les niveaux *musical* et *pratique* l'avantage de signifier automatiquement que chaque logiciel possède intrinsèquement *les deux propriétés* du temps réel et du temps différé.

Pourtant, d'une part, ce modèle ne correspond pas à l'unidimensionnalité du niveau *technique*. D'autre part, ce modèle sous-entend à nouveau une séparation entre le temps réel et le temps différé, alors que nous pensons qu'il existe une possibilité intéressante de dissolution de cette frontière (le chapitre 2 critique en détail le cloisonnement de ces catégories), notamment à travers le concept élargi de la « latence » (ou de « latentialité »).

Finalement, un axe, donc un espace unidimensionnel, constitue à nos yeux la représentation la mieux adaptée, à la fois pour son intégration des trois niveaux (technique, pratique et musical) et pour l'absence de dichotomie.

Une extension sémantique de la latence

Le terme « latence » semble pouvoir bénéficier de la même variété polysémique que « temps réel » et « temps différé », avec toutefois une extension particulière pour le niveau « musical ».

En effet, le niveau « technique » est évident, car il s'agit simplement de la latence audionumérique, proportionnelle à la fréquence d'échantillonnage et à la vectorisation audionumérique (cf. section 2.3.2 page 51). De même, le niveau « pratique » relève simplement du sens commun, soit le délai perceptible dans une interaction (cf. section 2.1.2 page 39).

En réalité, seul le niveau « musical » appelle une extension sémantique de l'idée de « latence », pour désigner finalement une abstraction qui puisse englober le temps réel et le temps différé « musicaux ». Cette latence « musicale » ou « esthétique » devrait alors indiquer *le rapport à la durée dans les processus de création musicale*, au sens large, c'est-à-dire entre la « performance » et la « composition » en tant qu'archétypes[3].

3.2.2 . . . munie d'un ordonnancement relatif

Une relation d'ordre terme à terme

On a vu que l'absence de repères absolus implique un positionnement relatif sur l'axe. Cet axe peut alors devenir un support pour le positionnement relatif des logiciels (ou d'autres éléments), c'est-à-dire les uns par rapport aux autres. Cette démarche relative

3. Cf. section 2.2.3 page 45 pour une présentation du champ musical, ainsi qu'au chapitre 5 pour une discussion sur la performance et la composition en tant qu'archétypes, les deux sections 5.4 page 109 (performativité) et 5.5 page 112 (composabilité).

permet d'ordonner un ensemble sans faire appel à des données numériques quantitatives (nous avons vu qu'il n'existe pas de norme à un niveau non-trivial), mais seulement en comparant ses éléments terme à terme.

Si l'on postule que deux logiciels peuvent toujours être ordonnés entre eux selon un critère dit « latentiel », alors il doit exister un opérateur qui permet de représenter la relation d'ordre entre deux logiciels quelconques ; cet opérateur sera lui-même dit « latentiel ».

L'utilisation pratique de cet opérateur latentiel implique de définir sous quels critères il agit. En particulier, dans le cas du classement de logiciels d'informatique musicale, il faut préciser le ou les niveaux concernés parmi les trois niveaux « technique », « pratique » et « musical » identifiés précédemment (section 2.2 page 42) : dans les analyses qui seront présentées ultérieurement (section 3.4 page 77), le niveau technique sera écarté à chaque fois que l'ensemble des logiciels à ordonner partageront la même latence audionumérique.

Un barycentre en première approximation

Si l'on admet un axe de localisation des logiciels d'informatique musicale défini par deux pôles opposés, le pôle « temps réel » et le pôle « temps différé », alors la localisation de ces logiciels peut se définir à l'aide du concept de *barycentre* (dans un sens simple, proche de celui de centre de gravité d'Archimède).

Proposition 1 (Barycentre latentiel relatif)**.** Nous appelons « barycentre latentiel relatif » le point qui représente le mieux un logiciel d'informatique musicale sur un axe destiné à classer ces logiciels entre les deux pôles opposés « temps réel » et « temps différé », les uns par rapports aux autres.

Corollaire 1 (Relativité des barycentres latentiels entre eux)**.** Les deux pôles « temps réel » et « temps différé » étant considérés tous deux comme idéaux, les barycentres latentiels se placent relativement, les uns par rapport aux autres. Placer un barycentre unique sur cet axe n'apporte donc aucune information.

Un segment caractéristique en deuxième approximation

Un point unique, le barycentre latentiel relatif, pourrait ne pas toujours suffire à représenter correctement un logiciel sur l'axe, attendu que certains logiciels cumulent des propriétés temporelles variées, donc étalées sur l'axe.

Proposition 2 (Segment caractéristique de dispersion)**.** Nous appelons « segment caractéristique de dispersion » la représentation longitudinale de l'écart type, comme dispersion autour du barycentre latentiel relatif.

Corollaire 2 (Ponctualité signifiante)**.** Le barycentre latentiel relatif est un point qui peut résumer de façon acceptable la position d'un logiciel sur l'axe.

* * *

En définitive, la possibilité d'ordonner des logiciels entre eux sur un axe « latentiel » annonce *une finesse d'analyse* immanquablement supérieure à la simple dichotomie traditionnelle du temps réel et du temps différé : par exemple sur la prise en compte de plusieurs des temporalités émanant d'un même système d'informatique musicale, ou bien sur l'histoire de l'évolution de ces systèmes au cours des années ou de leur développement, ou encore sur l'éventuelle constitution de familles de logiciels en fonction de leur rapprochement longitudinal sur l'axe.

3.3 Un outil de classement relatif

Afin de donner plus de réalité à notre proposition de basculement conceptuel de la dualité temps réel / temps différé, depuis une vision ensembliste dichotomique vers une vision axiale ordonnée, nous avons développé un outil logiciel d'aide au classement relatif. Ce logiciel a pris la forme d'une application en ligne[4], utilisable depuis l'adresse suivante : http://karim.barkati.online.fr/Universite/Doctorat/Classement/classement.html.

Cette démarche s'inspire des aspects *subjectif* et *relationnel* du protocole utilisé par David Wessel pour recueillir les jugements de dissemblance dans son travail d'analyse multidimensionnelle du timbre :

> *Though there exist a variety of ways to collect perceptual dissimilarities, we have found simple ratings of the extent of dissimilarity to be the most efficient and least tedious way to make the judgments. At IRCAM we have been using direct estimates of the dissimilarity between two tones. Our listening judge, using a program written by Bennett Smith called ESQUISSE, sits before a CRT display terminal and an audio system fed by the computer's digital-to-analog converters. The two sounds in a pair are related to the terminal keys "m" and "n", allowing the listener to play the sounds at will. After listening, the judge enters a rating with one of the keys "0" through "9" on the terminal. Immediately after the judgment is entered, the next pair of tones is ready to be played from the keyboard. The sequencing of the judgment trials is random, and all of the n(n - 1)/2 pairs are used, where n is the number of sounds under study. After all the pairs have been judged, the random sequence is unscrambled and a matrix of dissimilarities is formed.* [Wessel, 1979]

Il s'agit cependant ici d'un outil individuel, non statistique, ni audio : notre logiciel conserve principalement les aspects subjectif (un « juge » face au logiciel de classement) et relationnel (un principe de jugement par paire entre les éléments à classer).

3.3.1 La page d'accueil

Une symétrie visible, de part et d'autre des boutons de classement centraux « Classer avant » et « Classer après », structure la page d'accueil, présentée sur la figure 3.2 page suivante. Ces boutons sont d'ailleurs désactivés à l'ouverture, tant que les listes sont vides et qu'elles ne possèdent pas chacune un élément sélectionné.

4. Cet outil générique a été programmé en combinant plusieurs langages : Html, javascript, Php, Svg et Css, dont les codes sources sont présentés en annexe C page 281.

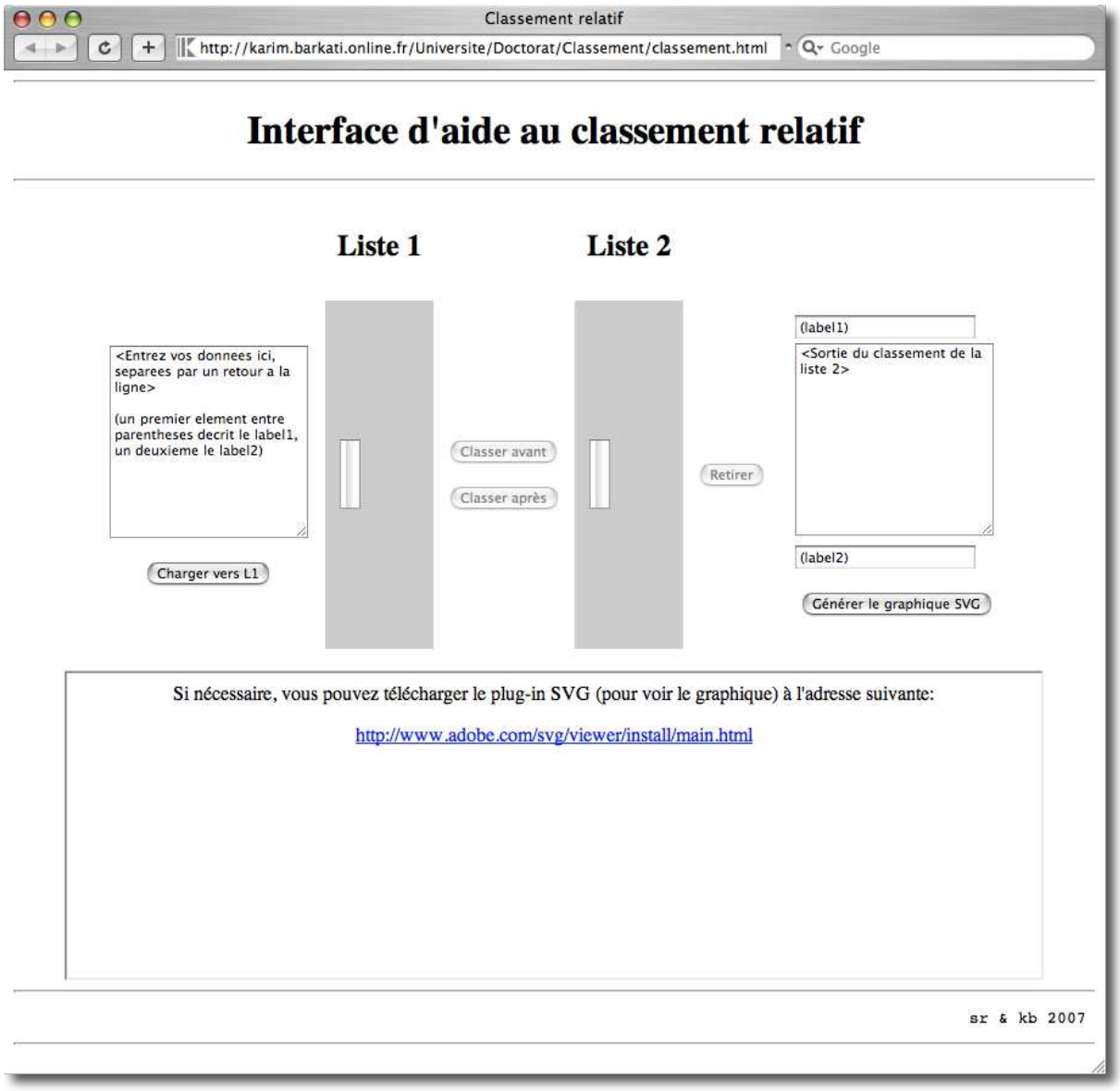


FIG. 3.2 – *Accueil de la page d'aide au classement relatif*

3.3.2 Le chargement

Le chargement des données procède très simplement par copier-coller ou par saisie directe, dans le champ textuel tout à gauche, puis en cliquant sur le bouton « Charger vers L1 », situé au-dessous de ce champ. On peut remarquer sur la copie d'écran 3.3 page ci-contre que les lignes entre parenthèses « (TR) » et « (TD) » n'apparaissent pas dans *Liste1* ; ces deux lignes sont captées dans les champs des labels 1 et 2, tout à droite de la fenêtre, et serviront à donner un nom aux deux extrémités de l'axe graphique.

3.3.3 Le classement à l'œuvre

Le processus global opère de gauche à droite : on a vu qu'il faut d'abord remplir le champ de texte situé tout à gauche. Une fois la liste L1 chargée, les deux boutons

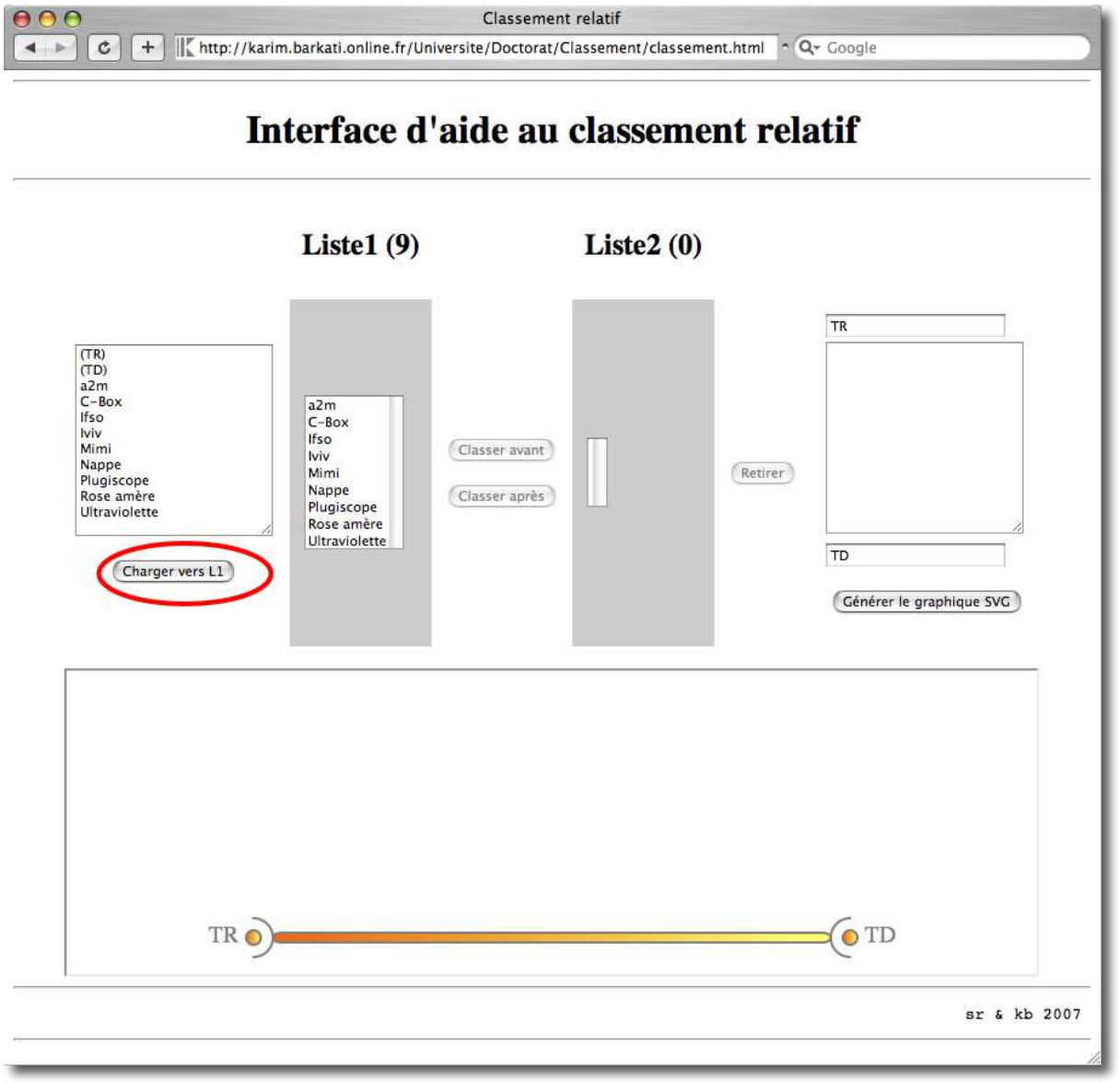


FIG. 3.3 – *Chargement des données*

centraux permettent de classer un par un les éléments de L1 dans L2, par rapport à l'élément sélectionné dans L2 ; le bouton « Retirer » permet de retirer un élément de L2 à tout moment, en le remettant dans L1. Le champ de texte situé tout à droite reflète la liste L2 afin de permettre les opérations de copier-coller sur le texte classé.

Sur la copie d'écran en figure 3.4 page suivante, l'utilisateur s'apprête probablement à cliquer sur l'un des deux boutons « Classer après » (ce qui aura pour effet de classer *Rose amère* après *Mimi*) ou « Classer avant » (inversement). Dès que deux éléments sont sélectionnés, un dans chaque liste, il est possible de classer celui de la liste 1 *relativement* à celui de la liste 2.

Cette approche *relative* permet en fait d'utiliser l'efficience humaine en matière de réduction multidimensionnelle. En effet, il apparaît aujourd'hui infiniment complexe d'essayer de formaliser en détail une analyse multidimensionnelle de ce type... Il faudrait

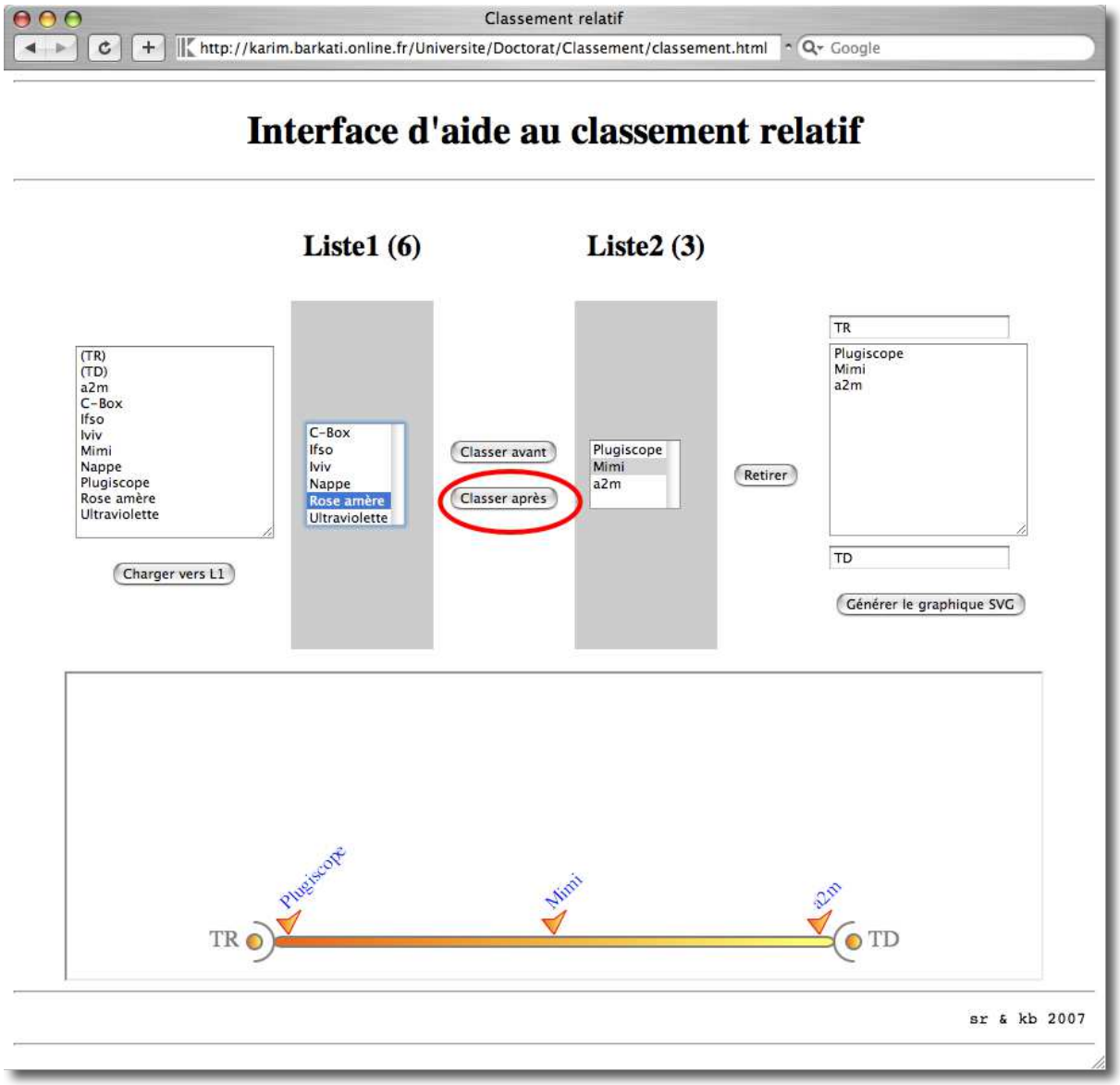


FIG. 3.4 – *Classement relatif en cours*

isoler tous les paramètres qui seraient nécessaires à un tel tri sur un axe temps réel – temps différé, puis renseigner des bases de données sur ces paramètres, dont certains resteraient de toute façon abstraits ou subjectifs (comme l'ergonomie). Partant de ce constat qu'un humain réussit à condenser une impression d'ensemble sur un axe unidimensionnel, nous avons développé cette interface d'aide au classement *relatif*, c'est-à-dire classer un élément « simplement » avant ou après un autre.

Le champ textuel à droite permet de récupérer le fruit du travail de classement, par simple copier-coller, mais il permet aussi d'opérer des modifications *a posteriori*, puis de les valider graphiquement par le bouton « Générer le graphique SVG ». Cette souplesse peut se révéler très pratique pour revenir sur un tri déjà effectué, en évitant de devoir tout reclasser.

3.3.4 Le résultat graphique

La zone inférieure, représentant l'axe graphique, se redessine à chaque modification de la liste 2, en recalculant un fichier SVG (cf. annexe C.3 page 285) dont le code est récupérable depuis la plupart des navigateurs actuels et utilisable dans de nombreux logiciels de dessins vectoriels. Voici un exemple du résultat graphique du classement sur la figure 3.5.

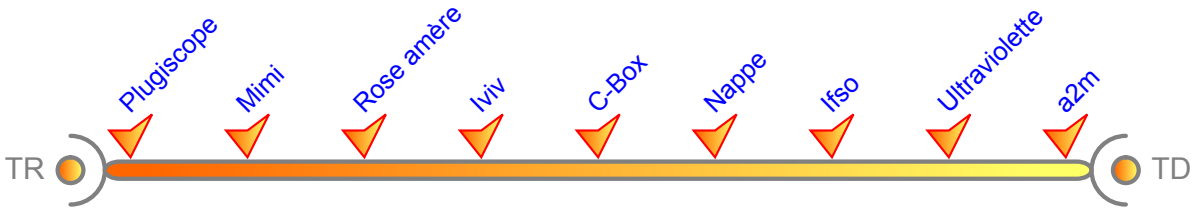


FIG. 3.5 – *Résultat graphique d'un classement relatif*

Le résultat graphique présente clairement un axe tendu entre deux pôles, et surtout les différents éléments classés, les uns par rapports aux autres, régulièrement espacés, entre ces deux pôles. C'est précisément ce graphique qui doit permettre le classement relatif de façon interactive, puisqu'il se met à jour automatiquement à chaque élément ajouté ou retranché de la liste 2.

Les effets graphiques peuvent sembler superflus au premier abord, cependant ils expriment, à leur niveau, des idées développées précédemment. Le dégradé de couleur entre les deux pôles souligne l'idée d'une certaine continuité sur l'axe, sans frontière qui pourrait marquer une ligne de démarcation *a priori* entre deux camps à l'intérieur de l'axe. Les deux crochets extrêmes, dirigés vers l'extérieur, montrent quant à eux l'exclusion des deux pôles par rapport à l'axe lui-même, pour rendre compte de leur caractère *idéal*, inaccessible (aucun élément sur l'axe ne saurait s'identifier à l'un des deux pôles tels que nous les avons défini à la section 3.1 page 66 : idéaux).

Enfin, outre l'activité de classement qui reste inévitablement un exercice d'analyse mais principalement personnel, ce graphique peut devenir un outil d'analyse *communicable* et donc potentiellement collectif : comme support, représentation, fixation d'une pensée, d'une façon d'envisager les rapports terme à terme au sein d'un ensemble fini d'éléments.

3.4 Quelques analyses axiales

Cette section présente quelques classements réalisés à partir de notre outil de classement relatif, sur un axe de localisation entre les deux pôles « temps réel » et « temps différé ». Le premier exemple présente une analyse axiale d'après mes logiciels musicaux développés dans le cadre de cette thèse, en tentant d'expliciter *l'intégration* de la polysémie du temps réel et du temps différé. Le deuxième exemple examine la robustesse du modèle en confrontant le classement de mes logiciels au classement de leurs fonctionnalités. Enfin, le troisième exemple interroge les limites de la pertinence d'un classement sur un corpus hétérogène, extrait d'un annuaire de logiciels.

3.4.1 Classement de mes logiciels

Présentation générale de mes logiciels

Essentiellement, j'ai développé neuf logiciels au long de ma thèse, toujours en collaboration avec un compositeur ou un instrumentiste. La liste suivante décrit brièvement ces logiciels[5], dans l'ordre chronologique de développement :

1. C-Box, un logiciel circulaire à 3 canaux stéréophoniques, jouant sur la réinjection, sur le croisement entre canaux, et sur le retardement de prélèvements consécutifs à la détection d'attaques, tous ces paramètres étant contrôlables par une partition Midi ;
2. Mimi, un auto-échantillonneur monophonique interactif pour pédalier Midi, permettant de contrôler la vitesse de lecture, la taille de sélection, le point de départ, le bouclage et l'aléatoirité ;
3. Plugiscope, une interface généraliste et interactive d'exploration des plugiciels Vst, par visualisation collective et par sélection individuelle des programmes et des paramètres d'une liste de plugiciels ;
4. Iviv, un auto-échantillonneur « anamorphiste » interactif pour pédalier Midi, à 3 voies paramétrables : délai, boucles avec Lfo[6], et « fleurs » (enveloppes dynamiques) ;
5. a2m, (*« audio to material »*) un outil de génération de matériau précompositionnel par transcription des pics d'amplitude d'un fichier audio en une séquence mélodique Midi, en fonction de l'intensité des pics.
6. Nappe, un générateur de textures à 4 voies, selon des traitements semi-aléatoires programmables : stéréophonie aléatoire, vitesses de lecture oscillantes, convolution, granulation, et enveloppes d'amplitude globales ;
7. Ifso, un lecteur polyphonique interactif pour pédalier Midi, permettant de contrôler plusieurs traitements sur 70 fichiers audio et sur l'entrée audio : vitesse, panoramique, volume, transposition, granulation, bouclage, inversion ;
8. Rose amère, un auto-échantillonneur interactif pour pédalier Midi, à 4 voies paramétrables : boucles, « impacts » (enveloppes percussives), délai, et effets (transposition et distorsion), orienté vers l'improvisation ;
9. Ultraviolette, un générateur de flux rythmique interactif à partir d'une collection de cellules rythmiques prédéfinies et d'un dossier de fichiers audio pré-enregistrés, permettant l'interpolation interactive du tempo et de la vitesse de lecture.

La latence audionumérique hors jeu

Ces logiciels partagent tous la même latence, celle du moteur audio de *Max/Msp* associée à celle du dispositif matériel (essentiellement ordinateur et carte son). En conséquence, la seule dimension *technique* du couple temps réel / temps différé, telle que nous

5. Pour plus de détails, consulter le tableau 3.3 page 82.

6. *Low frequency oscillator*, soit « oscillateur basse fréquence » en français.

l'avons identifiée précédemment (section 2.2 page 42), s'avère inutilisable pour ordonner ces logiciels entre eux sur un axe. Partant, le classement relatif de mes logiciels présenté à la figure 3.6 repose sur les deux autres dimensions du couple temps réel / temps différé : la dimension *pratique* et la dimension *musicale*, l'une permettant souvent de comparer deux logiciels lorsque l'autre semble y échouer.

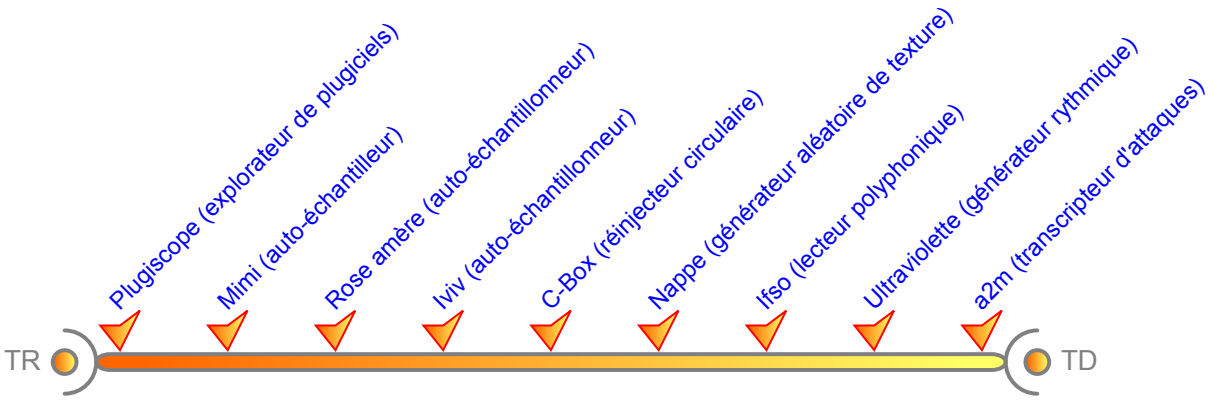


FIG. 3.6 – *Classement relatif de mes logiciels*

Démonstration d'un classement particulier

Commençons le classement[7] : parmi ces logiciels, deux d'entre eux se positionnent sur l'axe avec une certaine évidence, à savoir celui qui se trouve le plus proche du pôle « temps réel » ainsi que celui qui se trouve le plus proche du pôle « temps différé ». En effet, Plugiscope constitue un archétype du logiciel temps réel tant au sens pratique, puisqu'il s'agit d'une chaîne de traitement du signal complètement destinée à l'interaction, via MIDI et sans programmation préalable des paramètres, qu'au sens musical, puisque la visée concerne *a priori* la performance instrumentale bien plus que le travail compositionnel de studio.

À l'opposé, a2m constitue un archétype du logiciel temps différé au sens pratique puisqu'il ne permet pas d'interaction sonore de type instrumentale (même s'il propose une interaction visuelle, de type compositionnelle), mais surtout au sens musical puisque la visée concerne justement le travail compositionnel de studio - ici la génération de matériau pré-compositionnel. À ce titre, la fonction de déclenchement interactif de la lecture audible de la séquence MIDI générée, fonction disponible dans a2m, ne constitue pas un argument décisif pour éloigner a2m de sa proximité avec le pôle temps différé, tant la visée de son usage se concentre sur la composition et non sur la performance. Autrement dit, bien que ce logiciel soit capable de déclencher et de faire entendre des séquences, son identité reste d'abord attachée à un usage *compositionnel*.

Ensuite, les autres logiciels peuvent se positionner relativement à Plugiscope et a2m, qui constituent donc les extrêmes *concrets* (par opposition aux pôles *idéaux*) de ce classement. En considérant que Mimi possède peu de préréglages et vise presqu'exclusivement

7. La figure 3.4 page 76 présentait une session de classement en cours de réalisation.

le jeu en direct (sur l'auto-enregistrement bouclé), on peut placer ce logiciel immédiatement après[8] Plugiscope. Rose amère et Iviv visent aussi le jeu en direct, mais possèdent davantage de préréglages, ce qui les place juste après Mimi ; ils se départagent d'après leur interface : celle de Rose amère s'adresse plus directement à l'interprète tandis que celle d'Iviv s'adresse plus directement au compositeur.

En reculant depuis a2m, on peut placer d'abord Ultraviolette, qui, parmi les logiciels qui peuvent être joués en direct, contient d'une part des fichiers audio, nécessairement préparés en temps différé, et d'autre part le plus de préréglages à fournir : la liste des fichiers audio, celle des motifs rythmiques, et celle des *tempi* et des bornes du tirage aléatoire de la vitesse de lecture. Ifso se place juste avant Ultraviolette, parce qu'il contient aussi des fichiers audio et à peine moins de préréglages : la liste des fichiers audio et les préréglages de l'entrée audio.

Les derniers cas sont un peu plus complexes. D'un côté, Nappe contient de nombreux préréglages (pour spécifier les bornes de chaque paramètre aléatoire) et deux fichiers audio (pour la convolution), mais de l'autre côté, Nappe s'utilise finalement en direct, appliquant ses traitements aléatoires sans aucune interaction avec l'utilisateur, d'où un classement avant Ifso. Enfin, C-Box se place à la fois après Iviv, pour sa *composabilité* importante, et avant Nappe, pour ses quelques possibilités d'interaction (au clavier alphanumérique), la possibilité d'absence de fichiers audio, la présence d'une entrée audio, et le traitement du signal omniprésent.

3.4.2 Classement des fonctionnalités de mes logiciels

Présentation du classement des fonctionnalités

Résultat d'un processus de classement relatif analogue au précédent, la figure 3.7 présente vingt-sept fonctionnalités ordonnées sur un axe temps réel / temps différé, retenues parmi celles que comprennent mes logiciels.

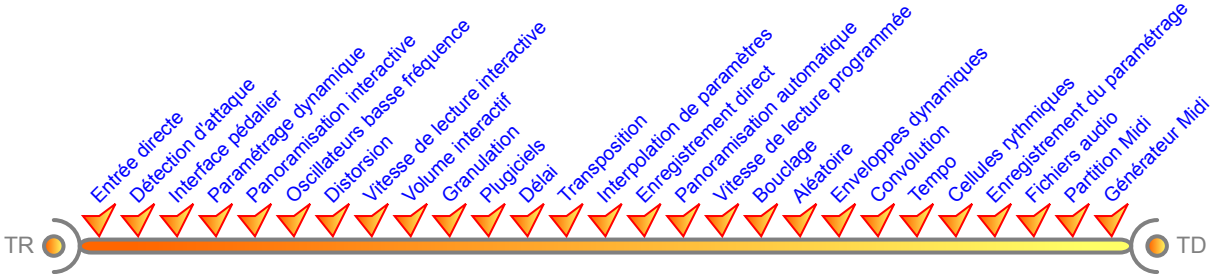


FIG. 3.7 – *Classement des fonctionnalités de mes logiciels*

8. Les indications « avant » et « après » n'ont pas ici d'autre signification que celle du positionnement relatif sur l'axe, de gauche à droite (c'est-à-dire dans le sens de l'écriture) ; elles correspondent aussi aux indications « Placer avant » et « Placer après » de l'interface d'aide au classement relatif.

Cette section fait volontairement l'économie d'une explicitation fastidieuse de chaque étape du classement de tous ces éléments, au profit d'une confrontation entre le classement précédent (sur les logiciels) avec le classement présent (sur les *fonctionnalités* de ces mêmes logiciels). Cette confrontation prend la forme du tableau 3.3 page suivante, sur lequel on peut grossièrement imaginer une diagonale qui partirait du coin supérieur gauche (logiciel le plus temps réel / fonctionnalité la plus temps réel) pour descendre jusqu'au coin inférieur droit (logiciel le plus temps différé / fonctionnalité la plus temps différé).

Un écart révélateur du polymorphisme

Cette première approximation reste valable en tant que telle, montrant un certain degré de cohérence du modèle de l'axe de classement relatif lorsqu'on compare un ensemble d'éléments avec un ensemble de leurs propriétés, à partir de pôles identiques (ici, « TR » et « TD »). Cependant, au-delà de cette première approximation, certaines zones observables s'éloignent de la diagonale, dont nous retiendrons les deux plus proches des coins opposés :

1. la détection d'attaque (une fonctionnalité plutôt TR) présente dans Ultraviolette et dans a2m (deux logiciels plutôt TD relativement aux autres),
2. et l'enregistrement du paramétrage (une fonctionnalité plutôt TD) présent dans Mimi, Rose amère et Iviv (trois logiciels plutôt TR relativement aux autres).

Ces zones éloignées de la diagonale indiquent-elles une faille dans le modèle ? Un examen plus approfondi de ces zones est nécessaire pour pouvoir répondre à cette question.

Concernant a2m, même si la détection d'attaque (incluse dans le processus de transcription) est une fonctionnalité plutôt temps réel, la visée du programme reste fondamentalement la génération de matériau précompositionnel ; autrement dit la technologie ne fonde pas nécessairement l'usage. Symétriquement, une fonctionnalité *technique* plutôt TD comme l'enregistrement du paramétrage ne nuit pas nécessairement aux logiciels orientés temps réel, c'est-à-dire vers des *pratiques* temps réel : intuitivement, on sent bien que plus un programme est paramétré (ou paramétrable), plus il sera facilement utilisable en situation de direct, une fois paramétré (cf. section 2.4.2 page 61).

Concernant Ultraviolette, l'analyse diffère sensiblement : sa visée est essentiellement le jeu en direct… En revanche, son « empan fonctionnel » s'étale largement, de la détection d'attaques aux fichiers audio, et surtout son « barycentre fonctionnel » se situe plutôt vers le temps différé. Si on ne remet pas en question les deux classements, il faut admettre qu'un même logiciel peut s'étaler largement lorsqu'on étudie ses fonctionnalités, et, conséquemment, posséder une ambiguïté intrinsèque irréductible. Dans ces deux cas, a2m et Ultraviolette, la présence d'une fonctionnalité plutôt TR dans des logiciels plutôt TD illustrerait l'adage « qui peut le plus peut le moins » : un logiciel sans contrainte *technique* temps réel peut incorporer des aptitudes temps réel sans que cela nuise à sa nature ou à sa visée *pratique* temps différé.

Somme toute, les deux zones éloignées de la diagonale, apparemment contradictoires, constituent à nos yeux moins une faille dans le modèle qu'un biais rélévateur de la complexité de toute évaluation basée sur des critères polymorphes (ou polysémiques), comme ici lorsque le temps réel et le temps différé sont enchevêtrés dans des domaines disjoints (techniques, pratiques et musicaux).

TR	Plugiscope	Mimi	Rose amère	Iviv	C-Box	Nappe	Ifso	Ultraviolette	a2m	TD
Entrée directe	✔	✔	✔	✔	✔		✔			
Détection d'attaque					✔			✔	✔	
Interface pédalier	✔	✔	✔	✔			✔	✔		
Paramétrage dynamique	✔	✔	✔	✔	✔		✔	✔		
Panoramisation interactive							✔			
Oscillateurs basse fréquence			✔	✔		✔				
Distorsion			✔							
Vitesse de lecture interactive		✔	✔				✔			
Volume interactif	✔	✔	✔	✔			✔	✔		
Granulation						✔	✔			
Plugiciels	✔		✔							
Délai			✔	✔	✔					
Transposition			✔				✔			
Interpolation de paramètres								✔		
Enregistrement direct		✔	✔	✔	✔					
Panoramisation automatique					✔	✔				
Vitesse de lecture programmée				✔		✔		✔		
Bouclage		✔	✔	✔	✔		✔			
Aléatoire		✔	✔	✔		✔		✔		
Enveloppes dynamiques			✔	✔	✔	✔				
Convolution						✔				
Tempo								✔		
Cellules rythmiques								✔		
Enregistrement du paramétrage		✔	✔	✔	✔	✔	✔	✔		
Fichiers audio					✔	✔	✔	✔	✔	
Partition Midi					✔				✔	
TD Génération de séquences Midi									✔	
Polyphonie interne	2	1	4	3	3x2	4	9+	128+	1	
Polyphonie de sortie	2	1	1	1	2	2	2	1	1	

TAB. 3.3 – *Tableau fonctionnel de mes logiciels, classé TR/TD*

3.4.3 Classement dans un annuaire

Notre outil d'aide au classement relatif ne s'applique pas seulement à nos logiciels ou à leurs fonctions : il peut s'appliquer par exemple à d'autres ensembles de logiciels. Nous présentons donc dans cette section un classement sur un corpus volontairement vaste, issu d'internet, aux limites de la classabilité. Dès lors que le classement ne relève pas du trivial, il trahit nécessairement une vision des choses non neutre de la part de la personne qui a effectué un tel classement. La figure 3.8 représente ainsi *mon* analyse relative synchronique des logiciels que j'ai retenus pour représenter vingt-neuf des catégories proposées par l'annuaire du site macmusic.org[9].

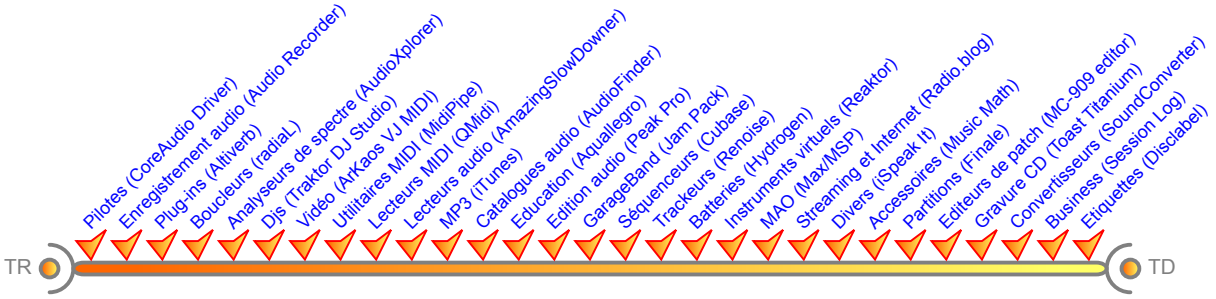


FIG. 3.8 – *Classement subjectif des représentants d'un annuaire de logiciels*

Cet exemple montre d'une part la subjectivité qui intervient lorsqu'il s'agit d'une population de logiciels particulièrement hétérogène, en quelque sorte la limite de la pertinence d'un classement, et d'autre part que classer reste finalement possible même dans ce cas extrême, dès lors que celui qui opère ce classement attribue un sens à ce classement (ce qui est le cas ici puisque j'ai pu trouver pour chaque logiciel des raisons subjectives de les classer à un endroit plutôt qu'à un autre[10]).

Un tel outil se révélera tout aussi précieux pour des analyses diachroniques, par exemple l'évolution d'un même logiciel au cours de son histoire, ou bien pour des analyses de développement, par exemple les fonctionnalités d'un même logiciel à un instant donné.

9. Leur annaire, qui évolue fréquemment, se trouve à l'adresse http://www.macmusic.org/software/?lang=fr. J'ai choisi cet annuaire plutôt qu'un autre essentiellement pour deux raisons : d'une part pour ma connaissance plus approfondie de l'environnement Macintosh dans ma pratique de l'informatique musicale – c'est utile pour le classement –, et d'autre part pour la fréquence et l'importance du recensement des logiciels dans cet annuaire.

10. Ces raisons ne seront pas exposées ici à cause de leur nombre important et de la grande similarité avec la démarche exposée en détail pour le classement de mes logiciels à la section 3.4.1 page 78.

3.5 Conclusion : une représentation intégrative

Finalement, cet axe renverse la vision traditionnelle du temps réel et du temps différé (ou d'autres catégories présentées inopportunément comme dichotomiques), vus comme deux ensembles disjoints. Il la dépasse aussi en quelque sorte, en présentant des éléments (des logiciels, des fonctions, des usages, etc.) sur un axe continu, comme si chacun de ces éléments possédait intrinsèquement ces deux propriétés, à des degrés divers, vers une vision potentiellement plus riche et plus nuancée. Cette représentation relative intègre ainsi, indirectement, le temps réel dans le temps différé et réciproquement.

Sans doute l'intégration réciproque du temps réel et du temps différé est-elle évidente pour chacun, car il est difficile d'être dupe du trait dichotomique énoncé dans la plupart des discours sur le temps réel et le temps différé, aussi la contribution essentielle de cette représentation alternative réside-t-elle moins dans la critique de la dichotomie que simplement dans son existence. En quelque sorte, nous considérons que cette représentation s'offre comme un nouveau symbole, à disposition, apte à être employé dès qu'il s'agira de nuancer de façon explicite la dichotomie classique là où elle devrait l'être.

Précisons enfin que l'effet d'intégration provient de la conjonction entre *l'idéalité* posée des deux pôles et la méthode *relative* de l'opération de classement, laquelle permet d'éviter tout recours quantitatif numérique, puisque dans cette méthode de classement seule la relation de voisinage importe. De fait, cette relation stipule formellement qu'un élément est *à la fois* plus temps réel *et* moins temps différé que son voisin (ou réciproquement). La notion de l'« entre » temps réel et temps différé prend d'ailleurs sous cet effet d'intégration toute sa valeur : une valeur nécessaire, apodictique.

Autrement dit, si le temps réel et le temps différé n'existent que comme *idéaux*, inaccessibles par nature, alors tout ce dont nous pouvons faire effectivement l'expérience se trouve nécessairement *entre* ces deux idéalités.

Deuxième partie

Les logiciels musicaux développés en collaboration

Chapitre 4

Nos collaborations musicales

Résumé du chapitre : Après les remarques préliminaires d'usage, ce chapitre présente quelques jalons prodromiques parmi nos recherches prédoctorales, puis situe brièvement les différentes collaborations musicales éprouvées dans le cadre du doctorat. Il relève ensuite et détaille les principaux traits communs de ces collaborations : les choix musicaux, pratiques et techniques.

4.1 Remarques préliminaires

4.1.1 À la première personne

L'organisation de ce mémoire s'articule en deux parties : une première partie théorique puis cette seconde partie pratique. Cette dernière expose en effet des collaborations étroites, dont je fais partie prenante. Aussi, pour éviter les risques de confusion, l'emploi de la première personne du singulier semble préférable dans l'ensemble de cette seconde partie, et y sera donc appliqué.

En outre, certaines sections adopteront ponctuellement un ton volontairement narratif lorsqu'il s'agira de situer les collaborations dans leur contexte, essentiellement pour deux raisons : d'une part, cela me permet d'exprimer que ces exposés ne relatent qu'une version *subjective* de ces collaborations – la mienne –, et d'autre part de réinsérer la programmation dans un processus humain et social, et non pas seulement technique, ce qui me paraît important dans notre contexte artistique. D'une manière plus générale, Pierre Lévy faisait remarquer l'importance de cette dimension dans son livre *De la programmation considérée comme un des beaux-arts* :

> L'écriture d'un logiciel, comme sans doute toute écriture, est un processus social, elle s'élabore en interaction avec un très grand nombre d'acteurs humains, techniques et sémiotiques. [...] Le dispositif technique est un analyseur social. [Lévy, 1992, p. 9]

Évidemment, une véritable analyse sociale sortirait du cadre de ce mémoire, et n'aura donc pas lieu ici.

4.1.2 Une curiosité circonstancielle pour le temps réel

Dans plusieurs de nos collaborations musicales, on peut observer un engouement pour l'expérimentation d'un temps réel qualifiable de « pur », en considérant d'une part l'inclusion de pratiques associées au temps réel – la captation en direct du jeu d'un instrumentiste, le traitement en direct de cette captation par l'ordinateur et ses logiciels, le contrôle en direct de ces traitements (si possible par l'instrumentiste lui-même) –, et d'autre part l'exclusion de pratiques tournées vers le temps différé – l'usage de séquences préenregistrées, certaines techniques de synthèse, etc. Toutefois, il faut préciser que cet engouement provient davantage de notre curiosité (celle des compositeurs, des instrumentistes, et la mienne) et du souci d'évaluer les possibilités et les limites du temps réel, que d'une quelconque idéologie ; tout au plus, on pourrait envisager de considérer chez certains une poétique de l'instant présent...

D'abord, les compositeurs avec qui j'ai travaillé possédaient souvent une expérience bien plus grande avec le temps différé qu'avec le temps réel, sur leurs ordinateurs personnels ou bien en studio, leur laissant intacte une certaine curiosité. Puis il faut noter que les dispositifs temps réel sont plus difficiles à réaliser techniquement, à cause de leur intolérance foncière à l'erreur, puisque par définition seul le temps différé jouit de la possibilité d'un temps de correction ; d'ailleurs, le temps réel est par nature une forme logicielle critique quelque soit la discipline concernée.

Ensuite, l'objectif primitif et partagé de pousser jusqu'au concert le résultat des collaborations, associé à l'implication d'instrumentistes dans les projets, orientait assez naturellement nos travaux vers des problématiques liées à l'interaction en direct.

Enfin, je m'étais présenté à eux comme un programmeur *Max/MSP*[1], environnement nettement orienté vers le temps réel. Dans ce contexte, on comprend que ces compositeurs se sont révélés plus intéressés à explorer du temps réel « pur » à l'occasion d'une collaboration privilégiée avec un programmeur, en quelque sorte là où il est profitable de s'unir pour aborder un domaine délicat.

En résumé, la prédominance notable de l'aspect temps réel dans nos collaborations ne résulte pas d'une idéologie, ni d'une utopie, mais plutôt d'un désir d'exploration de certains territoires ardus de l'informatique musicale : ceux d'une musique mêlant instrumentistes et informatique temps réel et qu'on pourrait appeler « musique interactive »[2].

4.1.3 Précisions typographiques informatiques

Une typographie *ad hoc* sera appliquée au vocabulaire technique de *Max/MSP* afin d'en faciliter la lecture et la compréhension :

- les `objets` figurent dans une police de type machine à écrire (*typewriter type*, avec empattement et espacement fixe) ;
- les messages figurent dans une police sans empattement (*sans serif*) ;
- les *noms-de-variable* figurent en italique ;
- les `[patchs]` figurent dans une police de type machine à écrire comme pour les objets, mais encadrés par des crochets ;
- le mot « patch » lui-même est considéré ici comme francisé[3]. Il est donc écrit normalement, pour faciliter la lecture en évitant que trop de mots par paragraphe soient écrits penchés, tant son usage est fréquent. Il prend en outre la marque du pluriel sans le *« e »* anglo-saxon, soit « patchs » au lieu de *« patches »*, qui ne correspondrait pas au genre de la traduction usuelle en français.

4.2 Brève situation de nos recherches actuelles

4.2.1 Une formation double ...

1. Pour « programmeur *Max/MSP* », on lit parfois « maxeur » (*maxer* en anglais).

2. En toute rigueur, cette forme musicale ne s'identifierait pas à ce qu'on appelle la « musique mixte », qui s'entend généralement comme le rapprochement d'interprètes et de la diffusion de sons fixés.

3. Les traductions les plus proches de ce qu'est un « *patch Max* » que nous pouvons proposer – « module », « câblage » ou « matrice » – restent trop éloignées de la prononciation et de la syntaxe du mot « patch » pour le remplacer dans ce texte, sachant que ce mot a été totalement adopté en France dans le domaine de l'informatique musicale. Surtout, dans l'environnement de programmation *Max*, un patch est finalement le nom d'un objet technique spécifique, mais aussi d'un concept central, renforcé par le concept de « sous-patch » (*subpatch*). Ainsi, cet objet technique spécifique et tout à fait central dans le paradigme de programmation *Max* nous semble difficilement pouvoir recevoir un équivalent français viable.

Avant le doctorat, j'ai suivi un double cursus en musicologie et en informatique, dans les deux départements idoines de l'université Paris 8, au cours duquel j'ai pu réaliser plusieurs logiciels en rapport avec l'informatique musicale, préparant mes recherches doctorales avec une certaine ouverture :

- K-lecteur, un lecteur-boucleur de fichiers audionumériques en temps réel, programmé dans l'environnement *SuperCollider* [Barkati, 1999] ;
- K-Sampler, un échantillonneur virtuel stéréophonique et MIDI programmé dans l'environnement *Max/MSP* [Barkati, 1999] ;
- K-Box, une extension logicielle pour instrument percussif développée en collaboration avec le compositeur Mario Lorenzo [Barkati, 2000b] ;
- kediteur et les bibliothèques kwav et kfft, bibliothèques C compatibles C++ pour la gestion du son sous Linux [Barkati, 2000a] ;
- MusicXML2SVG, un moteur expérimental d'application de feuilles de styles XSLT sur des partitions au format MusicXML vers un rendu SVG. [Barkati et Saint-James, 2003]

Parallèlement à ce cursus universitaire, j'ai pu suivre des études musicales en conservatoire, notamment en clarinette[4], qui ont évidemment facilité le dialogue au cours des collaborations musicales.

4.2.2 . . . pour un travail à l'interface

Au départ de ma thèse, mon interdisciplinarité informatique et musicale me situait favorablement « à une interface » plutôt que dans un champ en particulier. Aussi, j'ai choisi pour ce travail de thèse d'orienter mes recherches précisément sur cette interface entre la musique et l'informatique, et plus concrètement entre des musiciens et l'ordinateur : comme programmeur, autour de la collaboration avec des compositeurs et des improvisateurs dont je pourrais être un interlocuteur privilégié, capable d'entendre leurs souhaits musicaux. Ma démarche semble, *a posteriori*, avoir suivi l'appel lancé par Jean-Baptiste Barrière en 1992 dans son article *Le sensible et le connaissable* :

> Une logique de continuité entre tradition et création doit dominer les efforts de développements présents et à venir. Cette logique doit s'alimenter aux sources de la création telle qu'en train de se jouer aujourd'hui, auprès des compositeurs eux-mêmes, avec leur implication directe, à partir de la substance même des problèmes musicaux, des idées musicales, en considérant la musique comme source toujours renouvelée de connaissances. [Barrière, 1992, p. 86]

S'alimenter aux sources de la création, auprès des compositeurs, des interprètes et des improvisateurs eux-mêmes, voilà une des préoccupations majeures qui a présidé à ma recherche et à mes développements logiciels tout au long de la thèse, avec la conviction que la musique constitue effectivement une source toujours renouvelée de connaissances. Chaque collaborateur a d'ailleurs soulevé des problèmes musicaux différents et variés, en rapport avec ses aspirations esthétiques du moment, à la fois stimulé et limité par notre collaboration et ses moyens.

4. J'ai obtenu un 1er prix en clarinette avec Nicolas Baldeyrou comme professeur, en musique de chambre avec Jean-Claude Bouveresse, en analyse musicale avec Cécile Gilly, et en solfège avec Jacques Dauchy.

4.2.3 Deux périodes principales

Deux périodes successives se distinguent nettement au cours des collaborations musicales entreprises dans le cadre de mon doctorat.

Dans la première période, forts de la complicité acquise en DEA, Mario Lorenzo et moi avons exploré certaines applications musicales et sonores de l'idée de *circularité*, à travers la conception et la réalisation d'un logiciel relativement complexe – la CBox – dont le développement nous a permis de nous interroger concrètement sur les notions de temps réel et de temps différé. En effet, ces réflexions ont d'abord donné lieu à un article, publié dans les actes des *Journées d'informatique musicale* de 2005 et co-écrit avec Mario Lorenzo (cf. annexe A page 243), puis elles ont motivé le développement théorique exposé dans la première partie de la présente thèse. Le chapitre 5 (page 103) développera différents points de cette collaboration, ainsi que les spécificités techniques et algorithmiques déployées dans le logiciel CBox.

La seconde période répondait au vœu personnel d'aller, avec des pièces mixtes dont j'aurais programmé les logiciels, jusqu'au concert. De nouvelles collaborations ont alors débuté avec six musiciens : trois compositeurs étudiant à l'université Paris 8 – Santiago Quintáns, Mauricio Meza et Pedro Castillo-Lara – ainsi que trois instrumentistes – Stéphanie Réthoré, altiste, Clarissa Borba, percussionniste, et Iván Solano, clarinettiste et compositeur. Ainsi, les autres logiciels ont été développés durant cette seconde période, et sont présenté dans la deuxième partie de la thèse, organisée selon les collaborations, avec un chapitre par collaboration.

4.3 Traits communs et positions

Chaque collaboration diffère des autres, mais plusieurs des collaborations entreprises pour ce doctorat partagent aussi des traits communs, surtout celles de la seconde période. Cette section présente ainsi les traits communs qui m'ont parus importants, pour introduire les chapitres suivants, qui se consacreront chacun à une collaboration en particulier.

4.3.1 Le choix de la confrontation musicale

Chacune des collaborations m'a placé dans une relation directe avec des compositeurs et des instrumentistes, sur des problématiques exclusivement *musicales* ; aucun d'entre eux n'étant à proprement parler scientifique ni informaticien. Cette prise directe avec des musiciens rejoint la préoccupation essentielle de confrontation musicale évoquée précédemment avec Jean-Baptiste Barrière.

D'abord, l'activité de l'informaticien amène souvent à être en contact plus ou moins prolongé avec d'autres champs, car l'informatique en tant que discipline cache une propriété singulière : elle est fortement exogène, voire « exophage ». De fait, l'intérêt de l'informatique réside souvent moins dans l'informatique elle-même que dans les ouvertures depuis et vers d'autres disciplines. D'une part, l'informatique se nourrit directement ou indirectement des besoins et des structures des autres disciplines, scientifiques ou artistiques (dont les neurosciences et les arts graphiques sont des exemples flagrants), et d'autre part

elle se nourrit et contamine directement ou indirectement de très nombreuses disciplines, par sa puissance de calcul numérique, ses moyens de communication informationnels, ses facilités de stockage, et sa propriété de reproductibilité sans dégradation.

Ensuite, l'informatique contient en elle-même la notion de « service *à* », suite à sa fonction d'outil, fonction qui ne se construit qu'en capturant les savoirs-faire de l'« autre » discipline, par modélisation, formalisation et intégration. Ainsi, par exemple, les logiciels de notation musicale phagocytent proprement le métier de graveur au fur et à mesure de leur perfectionnement, dans une double opération d'absorption plus ou moins progressive des savoirs-faire et de destruction du métier dans son état initial, à plus ou moins court terme. Dans cette boucle d'exploitation réciproque, rien n'interdit à un métier « phagocyté » par l'informatique de muter en réaction : en conservant les tâches difficilement automatisables, en reprenant à son compte l'outil « gourmand », et en investissant finalement des niveaux plus *créatifs*, difficilement automatisables par définition. Voilà une spirale typique de l'informatique se nourrissant de disciplines qu'elle nourrit, et où finalement, l'automatisable étant automatisé, ne subsiste que le non-automatisable, soit essentiellement ce qui relève de la créativité[5].

Précisément, la composition, l'improvisation et l'interprétation musicales, trois activités hautement créatives, restent hors de portée de l'ordinateur en tant que telles, rendant vaines les tentatives de composition totalement automatiques. En revanche, le couplage homme/machine présente alors de nombreux intérêts, comme en témoigne la diversité des champs d'investigation de l'informatique musicale. Nos collaborations s'inscrivent dans cette démarche d'exploration des couplages homme/machine, cherchant à servir la créativité et tout en se servant d'elle, sans velléités généralistes mais au contraire en partant des individualités, des subjectivités, pour tenter d'arriver à des expériences et des résultats originaux. Le choix de la confrontation musicale nous a donc conduit vers une pratique du « sur-mesure »...

Cette expression du « sur-mesure » résume bien l'esprit de ce travail, en binôme strict la plupart du temps, avec ma recherche d'une compréhension fine de mon interlocuteur, de son esthétique, de ses souhaits ou de ses attentes, afin de proposer une programmation qui puisse devenir pertinente. La métaphore empruntée à la couture semble aussi seoir à cette pratique de la programmation en collaboration *à deux*, d'après les différentes étapes de la confection d'un vêtement sur mesure, et particulièrement à son cycle de conception et de production en aller-retour entre « l'essayeur » et « le confectionneur ». Les deux protagonistes communiquent ensemble d'abord sur un projet, puis sur des esquisses, puis sur l'ébauche du vêtement lui-même, selon une alternance caractéristique essayage/ajustement, et où les rôles sont bien définis dans ce processus de création. Enfin, l'idée du « sur-mesure » évoque le fait qu'à chaque projet correspond un logiciel unique (ou un ensemble de logiciels uniques), qui lui est entièrement et strictement dévolu.

5. Cf. *What computer can't do* et *What computer still can't do* de Hubert Dreyfus, et *Why people think computers can't* de Marvin Minsky pour une discussion sur les territoires comparés de l'ordinateur et de l'humain. Par ailleurs, Edgar Morin relève certains points de disjonction à partir de la biologie : « Une des conquêtes préliminaires dans l'étude du cerveau humain est de comprendre qu'une de ses supériorités sur l'ordinateur est de pouvoir travailler avec de l'insuffisant et du flou ; il faut désormais accepter une certaine ambiguïté et une ambiguïté certaine (dans la relation sujet/objet, ordre/désordre, auto-hétéro-organisation). Il faut reconnaître des phénomènes, comme liberté ou créativité, inexplicables hors du cadre complexe qui seul permet leur apparition. » [Morin, 2005, p. 50]

Ainsi, une pratique « sur-mesure » m'intéressait comme travail de recherche pour plusieurs raisons : rencontrer au plus près possible des esthétiques singulières, un rapport complexe à la musique et au son, l'humanité de la collaboration, l'essence véritablement musicale des discussions, le plaisir de participer à quelque chose de nouveau, et la satisfaction du projet quand il finit par fonctionner !

4.3.2 Un concert comme objectif

Suite à la première phase de mon doctorat, orientée toute entière vers des aspects théoriques interrogés via l'expérimentation régulière en studio, une nouvelle aspiration, motrice, a fait basculer le cours de mes recherches dans une deuxième phase : je voulais mettre mon expertise informatique et musicale à l'épreuve du concert. En effet, pour le programmeur ou le chercheur en informatique musicale, le concert en tant qu'expérience constitue une épreuve de consistance du projet musical et de stabilité du dispositif informatisé, autant qu'un moyen généralement efficace de prévenir ou de contrecarrer les risques dus aux pratiques de construction intellectuelle, risques qu'Edgar Morin désigne crûment comme le « délire de cohérence absolue[6] ».

De fait, une fois la décision prise, l'objectif du concert s'est mis en place à la manière d'un *dispositif*, au sens stratégique que Giorgio Agamben met en évidence dans son récent essai *Qu'est-ce qu'un dispositif ?* : « Le terme dispositif nomme ce en quoi et ce par quoi se réalise une pure activité de gouvernement sans le moindre fondement dans l'être. » [Agamben, 2006, p. 26][7]. Même si cette référence peut sembler trop forte, il faut reconnaître que l'échéance d'un concert agit la plupart du temps comme une tension impérieuse et motrice pour chaque collaboration impliquée, mobilisant par lui-même les ressources des uns et des autres, basculant les recherches dans un double impératif qualitatif et temporel : il faut que ça marche, et que ce soit prêt à temps ! Le concert contraint d'ailleurs aussi bien les activités de programmation que celles de composition, mais aussi d'organisation, de préparation, de répétition, et que de toutes celles qui seront jugées nécessaires, remplissant ainsi cette fonction *stratégique* à tous les niveaux.

Le niveau des contingences matérielles n'échappant pas à la règle, il faut par exemple résoudre les problèmes d'organisation, de lieu, de matériel (la liste du matériel utilisé est détaillée en annexe D page 291) et de communication (cf. figure 4.3.2 page suivante). Je tiens ici à remercier tout particulièrement la direction et l'équipe du conservatoire de Gennevilliers, qui ont fait preuve d'une ouverture et d'une générosité dont je leur sais gré.

6. « L'homme a deux types de délire. L'un évidemment est très visible, c'est celui de l'incohérence absolue, des onomatopées, des mots prononcés au hasard. L'autre est beaucoup moins visible, c'est le délire de la cohérence absolue. Contre ce deuxième délire, la ressource est dans la rationnalité autocritique et l'expérience. » [Morin, 2005, p. 97]

7. Agamben développe cette première définition quelques pages plus loin : « En donnant une généralité encore plus grande à la classe déjà très vaste des dispositifs de Foucault, j'appelle dispositif tout ce qui a, d'une manière ou d'une autre, la capacité de capturer, d'orienter, de déterminer, d'intercepter, de modeler, de contrôler et d'assurer les gestes, les conduites, les opinions et les discours des êtres vivants. » [Agamben, 2006, p. 30-31].

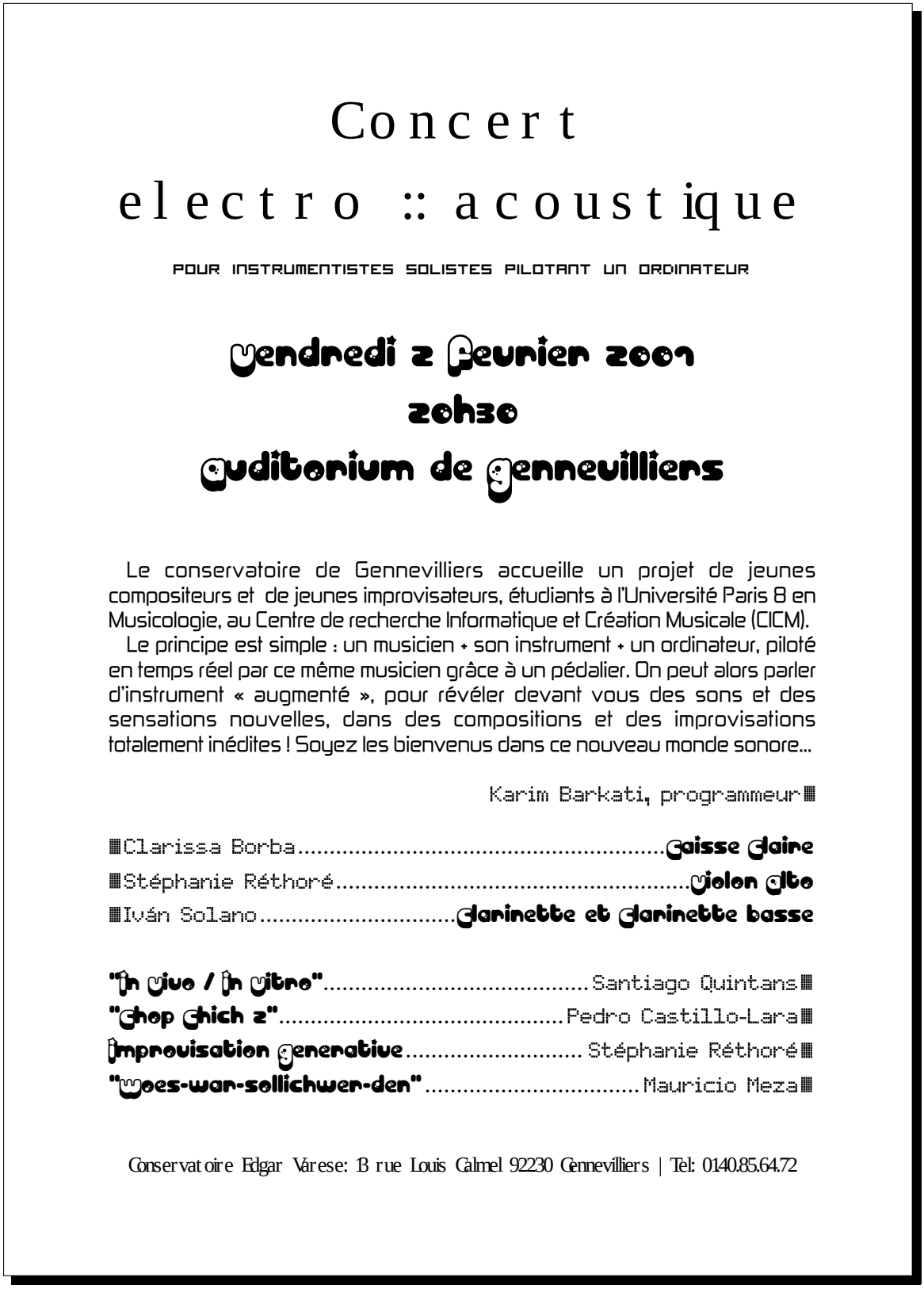


FIG. 4.1 – *Affichette du concert du 2 février 2007*

4.3.3 Le choix de l'interaction

Les utilisations, les apports ou les contributions de l'informatique dans le domaine de la musique sont multiples et l'interaction – au sens fort, l'interaction « instrumentale » – ne concerne qu'une partie de ceux-ci. Notre position sur ce sujet se rapproche fortement de la position manourienne, tranchée mais dotée d'un fort potentiel fondateur :

> La musique doit être, pour moi, soumise à une interprétation. Le concept d'interactivité est le seul qui réunisse les possibilités sonores de l'électroacoustique avec celles de l'interprétation. [Manoury, 1992, p. 44]

Depuis les débuts de la musique électroacoustique, les modalités du concert et de la diffusion ont bien changé, au fil des progrès technologiques, des réussites et des échecs. En particulier, Frédéric Durieux rappelle de façon critique l'éviction de l'interprète pratiquée par nombre d'œuvres, son impact et le type de prise de conscience associée :

> La solution adoptée [par les premiers studios électroacoustiques et les instituts de recherche pour faire connaître les œuvres électroacoustiques] fut de plonger la salle dans une lumière tamisée, pour créer l'ambiance, et de placer les auditeurs au milieu d'une armada de haut-parleurs aussi amènes et expressifs qu'un distributeur de boisson. On s'aperçu alors que le concert ne sollicite pas seulement l'oreille mais qu'on a plaisir à voir qui fait quoi et comment. [...] l'absence d'événements concrets rappelle que le concert est un échange d'énergie entre celui qui joue et celui qui écoute. [Durieux, 1992, p. 92]

La conscience de l'importance de cet échange et de l'importance de la présence de l'interprète – comme *médiateur* et comme *médiation* entre l'œuvre et le public –, constitue la raison première de notre choix pour l'interaction dans notre travail avec l'informatique. Car nous n'ignorons pas que l'informatique musicale présente une inclination pour l'auto-suffisance au moins équivalente aux moyens de l'époque prénumérique, dès qu'on considère le potentiel de l'algorithmique entre autres, comme en témoigne l'efficacité des installations sonores informatisées qui fleurissent avec bonheur.

De fait, l'algorithmique ne saurait davantage se substituer à l'humain qu'un mur de haut-parleurs diffusant une bande magnétique seule. Pour reprendre les propos d'Antonia Soulez[8], un programme pourrait être défini comme « intentionnalité gelée ». Ce « gel » constitue la deuxième raison pour laquelle nous avons choisi d'explorer des chemins de l'interactivité, pour échapper au *déterminisme* de l'algorithmique pure, « gelée ». En revanche, il faut reconnaître que ce gel de l'intentionnalité possède une propriété singulière : celle de *formaliser* son objet dans un langage de programmation. Ceci représente l'avantage considérable de constituer une mémoire *symbolique* et donc robuste des opérations, et de rendre ainsi l'objet programmé parfaitement accessible à la réflexion, la prospection, la recherche et à la composition dans la récursivité propre à l'écriture[9].

8. Professeur de philosophie du langage et d'épistémologie, séminaire du 4 avril 2007 à la Maison des Sciences de l'Homme – Paris Nord.

9. Bien sûr, l'intentionnalité qui est formalisée ici ne se confond pas avec les raisons profondes ni esthétiques de l'acte compositionnel, mais offre un matériel technique formalisé donc exploitable à l'analyse musicologique. « Tout comme un système de notation, un langage de programmation est porteur d'une ergonomie cognitive qui conditionne, d'une part, la facilité avec laquelle on va passer d'une idée à sa

Néanmoins, les raisons qui nous ont poussé à choisir l'interaction et à la placer au cœur même de ces projets ne sont pas strictement négatives (pour contrer l'éviction de l'interprète et le gel de l'intentionnalité évoqués plus haut) mais aussi positives. En effet, j'ai pu constater *a posteriori* que chacune des collaborations de la deuxième période a débuté par une discussion sur nos convictions personnelles à propos du rôle de l'interprète dans la musique électroacoustique, toujours dans le sens d'une revalorisation. Un résumé de ma position transparaît dans mon journal de thèse, dans une note datée du 28 octobre 2006 :

> Je souhaite construire et défendre une implication du musicien dans le résultat sonore du dispositif électronumérique : c'est important de ne pas le déposséder de cet élément, pour le résultat musical et pour l'évolution des mentalités.

C'est pourquoi j'ai décidé de ne pas adopter la solution traditionnelle, consistant à utiliser une pédale de déclenchement unique qui incrémente systématiquement un compteur d'états préprogrammés et préordonnés par rapport à la chronologie de la pièce (qu'il y ait répétition d'états ou non). En effet, même si cette solution possède des avantages certains – dont la légèreté du dispositif, un moindre risque d'erreur de la part de l'instrumentiste, ainsi qu'un temps d'apprentissage minimum –, la *conscience* de cet instrumentiste quant à la partie électronique est nécessairement entravée ou tout du moins ralentie par l'opacité de ce système, alors qu'un pédalier plus complet permet d'assigner beaucoup plus clairement un état (ou une fonction) identifiable par pédale. Il s'agit donc ici de permettre à l'instrumentiste de faire la relation entre son geste avec le dispositif matériel de contrôle et les conséquences de ce geste, soient les opérations logicielles et le résultat sonore qui en découlent.

Si l'instrumentiste est une interface privilégiée entre l'œuvre et les auditeurs, doué d'un pouvoir qu'on qualifie d'interprétation, alors je crois que la musique a beaucoup à y gagner dans la clarification des relations sonores de cause à effet qui concernent cet instrumentiste, spécialement avec l'électronique, précisément pour donner les moyens à cette interprétation d'avoir lieu. Pour le dire d'une façon brutale, il me semble que priver l'instrumentiste des moyens de comprendre ce avec quoi il interagit mutile la qualité de son action, et par suite le résultat musical de l'ensemble, puis la musique elle-même.

4.3.4 La prémisse acoustique

La collaboration avec des compositeurs aurait pu aboutir à des projets orientés vers de l'électronique « pure », que ce soit en temps différé – le temps pour ainsi dire le plus naturel aux compositeurs –, ou même en temps réel mais sans la présence d'instrument acoustique. Le choix de l'interaction exposé précédemment n'implique d'ailleurs pas *a priori* l'emploi d'instrument acoustique, or chacun des projets entrepris ici implique d'une manière ou d'une autre l'intervention d'un instrument acoustique, pourquoi ? De façon plus radicale encore, la plupart de ces projets placent l'instrumentiste et son instrument acoustique au cœur du système, et beaucoup vont jusqu'à renoncer à tout déclenchement

formulation dans le langage, et d'autre part, celle avec laquelle on va être capable de comprendre l'idée sous-jacente à sa formulation. C'est en cela qu'il est une interface plus ou moins efficace entre le monde des idées et celui de leurs réalisations concrètes. » [Orlarey *et al.*, 1999]

de séquences préenregistrées, adoptant une forme de « prémisse acoustique » à l'intérieur de la démarche ; à nouveau, on peut se demander pourquoi.

D'abord, l'idée de *prémisse* voudrait traduire le fait que la présence d'un instrumentiste et de son instrument acoustique n'a jamais été discutée ni remise en question, pour aucune des collaborations. Cette présence allait de soi en quelque sorte, elle était à la fois admise et invisible, à la manière des *prémisses* qui s'insinuent dans les raisonnements logiques sans qu'on en ait conscience et qui imposent de se hisser à un niveau *méta* pour pouvoir les déceler[10]. Après examen, cette « prémisse acoustique » s'avère cependant ni utopique ni incidieuse, mais seulement « souterraine », au regard son apparition historique consciente tardive. Une adhésion si unanime dans nos collaborations laisse entendre des raisons fortes à cette position, parmi lesquelles je suppose que se trouvent des idées comme la causalité, l'humanisation et la qualité sonore.

La causalité du son, traditionnellement depuis un geste instrumental, disparaît évidemment dans l'absence d'interprète, et cette disparition entraîne mécaniquement des problèmes d'intelligibilité à la majorité des auditeurs. Jean-Baptiste Barrière éclaircit ce point pour arriver à la question du « communicable » :

> La musique contemporaine, surtout lorsqu'elle met en jeu l'électronique, a déstabilisé nos habitudes culturelles, nos systèmes de références, aussi bien vis-à-vis des matériaux sonores que des processus d'organisation formelle. Les instruments traditionnels disparaissent ou changent de fonction. Ils sont transformés, voire munis de prothèses (cf. : les hyper-instruments). Ils deviennent parfois méconnaissables.
>
> Les instruments virtuels de l'électronique, sont, quant à eux, *inconnaissables*, d'autant qu'ils n'ont pour seule dimension tangible que l'écoute : ils ne sont pas visibles sur scène, et leurs manifestations sensibles (les sons !) apparaissent et disparaissent, produits d'une volonté compositionnelle souvent manichéenne pour l'auditeur. En fait, ce ne sont pas des instruments. Ce ne sont même pas des machines (tels les instruments traditionnels) : ce sont des instruments virtuels, des abstractions, des concepts réalisés à partir du mélange – improbable pour le néophyte – de matériels et de logiciels. Avec ces abstractions, tout est imaginable ; mais tout n'est pas nécessairement réalisable, et, peut-être plus important encore, tout n'est pas forcément communicable. [Barrière, 1992, p.77-78]

Ainsi, on peut faire l'hypothèse qu'un souci de communication, ou plus exactement de *communicablilité*, se tapit sous la prémisse acoustique observée. Même transformé et augmenté, l'instrument acoustique fournit un niveau de causalité acoustique auquel l'auditeur-spectateur peut toujours se raccrocher, sans compter le pédalier multiple qui fournit quant à lui une prise de causalité supplémentaire, pour la partie électroacoustique. L'instrumentiste lui-même intervient fortement dans ce processus de communication en contrebalançant les abstractions, rendant aussi cette musique plus communicable.

Ensuite, sur le plan de la qualité sonore, la confrontation de la sphère acoustique avec la sphère électronique présente quelques risques : l'électronique ne jouit généralement

10. Dans leur essai brillant sur la structure du changement [Watzlawick *et al.*, 1975], les auteurs distinguent le changement 1 (celui qui ne change pas véritablement) et le changement 2 (celui qui est susceptible d'apporter une solution) en rapport avec l'examen des prémisses : « lorsqu'on se place à l'extérieur du système, [le changement 2] n'apparaît comme rien de plus qu'un changement des prémisses qui gouvernent le système *en tant que totalité*. [...] il suffit, pour effectuer un changement dans le système de référence, d'agir seulement au niveau *méta*. »

pas de la même qualité que l'acoustique et, symétriquement, l'acoustique pâtit souvent d'un volume insuffisant devant celui de l'électronique, justement poussé pour combler son manque éventuel de richesse sonore. J'imagine pourtant qu'un désir de qualité sonore a participé à la constitution de la prémisse acoustique, en tirant les développements du dispositif électronumérique « vers le haut », à travers la confrontation permanente avec le son de l'instrument acoustique.

Enfin, je pense que pour notre génération, l'heure est à la synthèse (et donc à la mixité) : schématiquement, l'instrument traditionnel porte les références qui parlent à la mémoire collective, tandis que l'électronique permet de se jouer de ces références, en les déstabilisant. De même, concernant la structuration du discours musical, les dogmes semblent progressivement laisser la place à une intégration libre et cultivée, empruntant tantôt au sérialisme, tantôt à la musique concrète, au spectralisme ou encore au collage. On peut penser que l'espace de création offert par la mixité est potentiellement plus vaste que ce qui reste en musique instrumentale et que ce que permet effectivement l'informatique actuelle[11]. Dans cette intégration globale, technologique et esthétique, l'instrument acoustique assure un pont générationnel et permet d'ouvrir un dialogue entre les anciennes et les nouvelles technologies, et, passant, d'éventuellement questionner la place de l'humain dans le monde numérique.

4.3.5 Un pédalier Midi multiple

Le choix d'explorer l'interaction entre un instrumentiste et un logiciel contrôlable par le même instrumentiste nous a assez rapidement orienté vers le travail avec un pédalier Midi multiple.

De fait, beaucoup d'instruments de musique occupent largement les mains tout en laissant les pieds relativement libres, dont mon propre instrument, la clarinette, ainsi que les autres instruments joués lors de nos collaborations – l'alto, la caisse claire, la guitare, la clarinette basse et le vibraphone (qui comporte déjà une pédale). À titre de contre-exemple, une collaboration potentielle avec une harpiste intéressée ne s'est pas faite à cause du pédalier, la harpe occupant déjà beaucoup les pieds et surtout l'espace à portée des pieds ; en revanche, étonnamment, la contrebassiste Charlotte Testu n'a pas éprouvé de difficulté majeure lors d'une improvisation avec le pédalier et le logiciel Mimi, malgré la taille de son instrument, ni le pianiste et compositeur Alexandros Markeas, malgré l'encombrement des pieds et du pédalier du piano à queue. Ce pédalier permet donc un jeu simultané avec de nombreux instruments de musique : la plupart des instruments à vents, des instruments à cordes, certaines percussions, et certains claviers. La souplesse et la coordination varie naturellement d'un instrumentiste à l'autre, mais après quelques répétitions, une familiarisation tout à fait perceptible a clairement émergé chez chacun.

D'autres protocoles de communication plus performants existent mais le Midi représente un standard de fait pour les interfaces matérielles et assure la compatibilité du pédalier avec la carte son et l'ordinateur. De plus en plus de musiciens acquièrent ce type

11. À nouveau, les critiques renouvelées de Hubert Dreyfus donnent une idée des désillusions historiques quant à l'informatique et plus particulièrement à l'intelligence artificielle, ainsi que les critiques de Terry Winograd et Fernando Flores dans leur livre *Understanding computers and cognition* (1986).

d'équipement grâce à la disponibilité croissante dans les magasins spécialisés (le FCB1010 date de 2001), au coût raisonnable[12], et à la compatibilité MIDI pour contrôler tout type d'appareils, de la boîte d'effet à l'ordinateur (via la carte son). En outre, les environnements de programmation musicaux prodiguent en général moult facilités sur ce type de données standardisées et largement propagées depuis 25 ans (cf. section sur le MIDI page 51).

FIG. 4.2 – *Le pédalier Midi FCB1010 de Behringer*

Nous avons choisi le pédalier FCB1010 pour la quantité de pédales matérielles et virtuelles proposées, sa disponibilité (au cas où certains collaborateurs souhaiteraient l'acquérir), sa robustesse, sa stabilité sur scène (son poids d'environ 3,5 kg associé aux patins en caoutchouc donne à ce pédalier une adhérence au sol tout à fait correcte et rassurante pour les utilisateurs assis ou debout) et pour son prix bien inférieur à celui de ses concurrents[13]. Le FCB1010 possède au total 14 pédales matérielles, réparties en trois parties sur une surface de 68,7 cm par 22,1 cm (pour 6 cm de hauteur au plus haut), soit de gauche à droite :

- 10 pédales de déclenchement sont distribuées sur 2 lignes en quinconce (pédales numérotées de 1 à 5 pour la première ligne, et de 6 à 10 pour la seconde). Par défaut, un message de type *Program Change* est déclenché aussitôt que la pédale est enfoncée, sans tenir compte ensuite du relâchement, mais les dernières versions du programme interne sont capables, lorsque l'appareil est placé en mode *note*, d'envoyer aussi un message lorsqu'elle est relevée (*note-off*).
- 2 pédales imprimées « UP » et « DOWN » et alignées verticalement permettent de passer d'une banque à une autre. Ce système décimal des 10 banques multipliant les 10 pédales unitaires se révèle très pratique à l'usage, naturellement mnémotechnique. Techniquement, les pédales de déclenchement envoient par défaut un message MIDI *Program Change* de 0 à 99 selon la banque sélectionnée, ce qui procure 100 pédales virtuelles ou *presets*.

12. 170 € généralement constatés pour un appareil neuf.

13. À titre indicatif, le FC-200 de Roland coûte le double, avec une seule pédale d'expression.

– 2 pédales dites « d'expression[14] » transmettent le mouvement des pieds, encodé sur une échelle de 0 à 127. La pédale de gauche est nommée « A » et celle de droite « B », se distinguant ainsi des pédales de déclenchement et se conformant au sens de la lecture occidentale ; par défaut, la pédale A émet des messages MIDI de type *Control Change n° 27*, non précisé par la norme MIDI, et la pédale B des messages MIDI de type *Control Change n° 7*, défini par la norme comme le contrôle *Volume*.

Le dispositif de visualisation comporte un afficheur numérique à trois chiffres, qui indique lisiblement le numéro du dernier *preset* déclenché, et dix voyants individuels, placés au-dessus des pédales de déclenchement. Un seul de ces voyants individuels s'allume dès que la pédale correspondante est enfoncée et aussi longtemps qu'aucune autre ne l'est, nous renseignant distinctement sur la dernière pédale actionnée.

FIG. 4.3 – *Vue arrière du FCB1010*

Enfin, ce pédalier possède de nombreuses autres possibilités, comme deux sorties *jack* programmables de commutation par relais pour les changements de canal d'amplificateurs par exemple (*switches* 1 et 2 sur la figure 4.3, à gauche des prises MIDI), différents modes de déclenchement, et surtout un système de programmation des configurations très poussé, avec le réglage des canaux MIDI, des types de messages MIDI associés, la transmission simultanée de 5 *Program Changes* et 2 *Control Changes* par *preset*, une mémoire des configurations dont trois configurations d'usine, et un mécanisme MIDI *System Exclusive* pour la sauvegarde et le rappel à distance des configurations[15].

4.3.6 Un dispositif « podophonique » interactif

La traduction *en acte* de nos différents choix – la confrontation musicale, la causalité acoustique, l'objectif du concert, l'interactivité autonomisante – passe par la mise en place d'un dispositif matériel et logiciel opérationnel concret.

En particulier, nous l'avons vu, l'autonomie de l'instrumentiste dans l'interprétation musicale de la partie dite « électronique » trouve un vecteur de sa réalisation à travers l'emploi d'un pédalier *multiple*. Cette observation justifie un néologisme pour désigner de façon concise la forme de cette solution pratique au problème de l'autonomisation de l'instrumentiste doublement interprète : le dispositif « podophonique » interactif lui permet de contrôler du logiciel tout en jouant de son instrument. Une telle condensation, sévère, laisse évidemment quelques caractéristiques essentielles de côté ; par exemple, l'abstraction du dispositif n'apparaît pas clairement, « podophonique » pouvant même laisser entendre

14. Ne pas confondre avec le paramètre MIDI d'expression (*Control Change n° 11*).

15. Pour plus d'informations, se référer aux documents disponibles directement sur le site de Behringer ; ils sont assez complets et traduits dans plusieurs langues.

que le son proviendrait directement du pédalier… Cependant, l'adjonction du simple qualificatif « interactif » indique en creux cette abstraction informatique, car il relève d'une ontologie *informatique*, tout comme le couple temps réel / temps différé (cf. section 1.1 page 8), et permet ainsi l'économie d'un préfixe supplémentaire (« métapodophonique », quoique sans doute plus juste, nous paraît trop lourd).

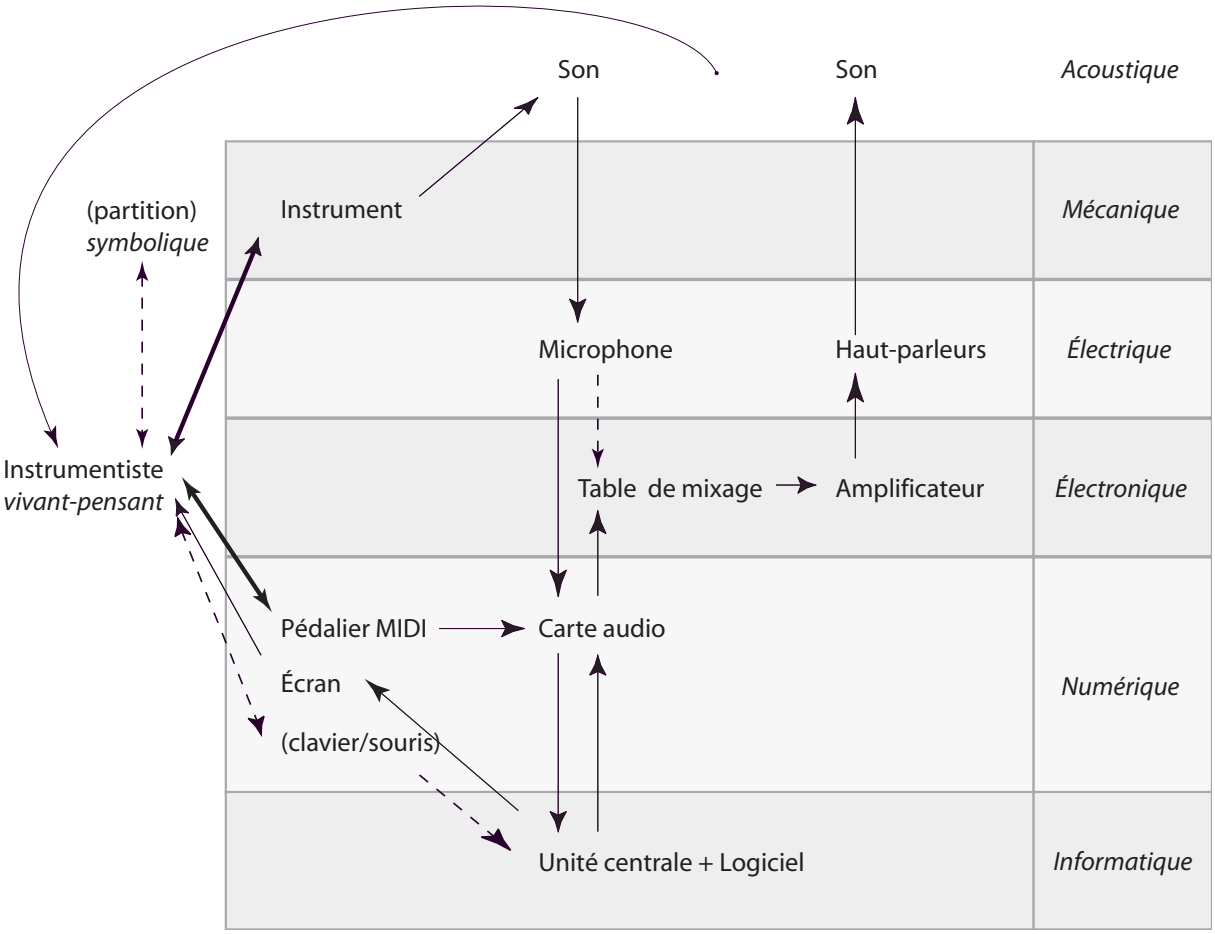


Fig. 4.4 – *Schéma relationnel du dispositif électro-informatique*

Le schéma de la figure 4.4 représente les relations entre les différents éléments de ce dispositif, pour tenter d'éclairer plusieurs aspects. Le premier de ces aspects, la double responsabilité de l'interprète, se lit avec les deux flèches *en gras*. Ces flèches relient l'interprète à la fois à son instrument acoustique et au pédalier Midi, qui constituent les deux portes d'entrée à l'ensemble du circuit.

Le deuxième aspect fondamental se lit au carrefour représenté par la carte audio : elle accueille à la fois les informations provenant du pédalier et le son de l'instrument ; cette duplicité confère à l'ordinateur – et donc au logiciel – la double compétence du traitement du signal entrant (et de son contrôle interactif) et de la synthèse (en temps réel). Ce dispositif peut donc se comporter alternativement comme une boîte à effet, un échantillonneur, un synthétiseur ou comme un hybride de ces archétypes, selon les fonctionnalités implémentées effectivement dans les logiciels.

Le troisième aspect important se lit en creux : le carrefour stratégique de la table de mixage n'est pas directement accessible par l'instrumentiste, or on sait que ce poste

contrôle est absolument capital en situation de concert… En pratique, le travail d'une personne dédiée au mixage reste l'idéal, mais quelques astuces ont dû être trouvées pour les autres cas : essentiellement des fonctionnalités de mixage contrôlables depuis le pédalier au sein même des logiciels, l'utilisation d'un boîtier anti-larsen, une bonne balance avant le concert et la proximité de la table de mixage avec la place de l'interprète (déterminée par celle du pédalier).

Le dernier aspect important se rapproche du précédent et concerne le retour sonore à l'instrumentiste, représentée par la flèche *courbe*. Une difficulté d'un tel dispositif réside dans les compromis pratiques à trouver entre la qualité de la sonorisation de la salle et la qualité du retour, indispensable à toute performance musicale.

Les autres aspects revêtent une importance bien moindre ; signalons simplement que les flèches *en pointillé* indiquent des connexions facultatives (par exemple la partition qui devient facultative dans le cas d'une improvisation) et que l'architecture en couche du schéma vise principalement à en faciliter la lecture.

Chapitre 5

La CBox / Mario Lorenzo

Résumé du chapitre : Ce chapitre relie entre elles la première partie théorique et la seconde partie pratique de ce mémoire, en examinant les notions de temps réel et de temps différé à travers le prisme de la CBox, un logiciel développé en collaboration avec Mario Lorenzo. Après une contextualisation du projet et des précisions typologiques à propos des interactions logicielles, deux concepts seront développés : la *performativité* et la *composabilité*, au titre de modalités archétypales observables dans la CBox. Enfin, nous formulerons quelques remarques au sujet de la circularité telle que nous l'avons expérimentée avec la CBox.

5.1 Introduction

La CBox ayant déjà fait l'objet d'un article (publié lors des JIM 2005) et cet article étant joint en annexe de ce mémoire (annexe A page 243), nous présenterons dans ce chapitre soit des traits plus généraux, soit des traits plus détaillés, afin d'éviter les redites. Ainsi, ces écarts par rapport à l'article nous donneront l'occasion d'approfondir certaines notions de la partie théorique de la thèse, à l'aide du cas concret que constituent à la fois ce logiciel et l'histoire particulière de son développement.

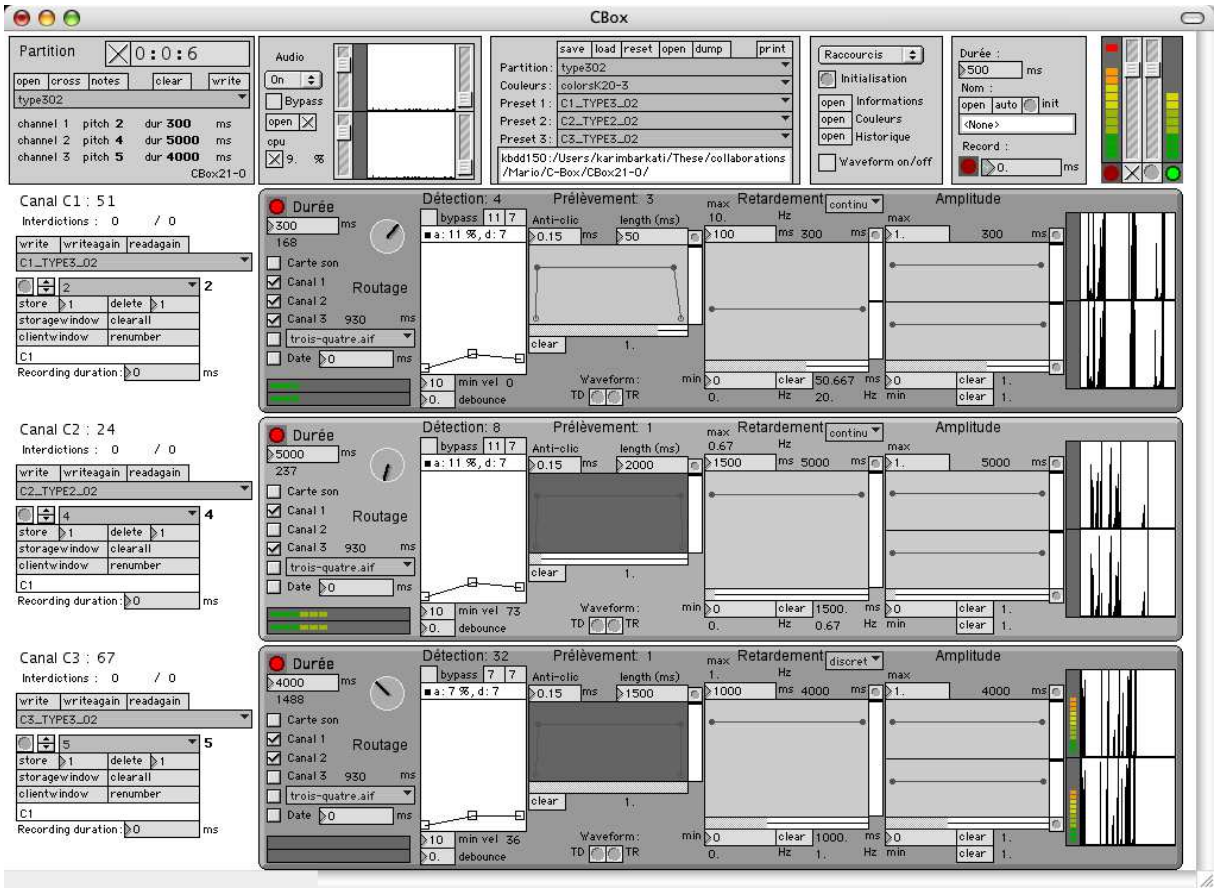


FIG. 5.1 – *Capture de l'interface visuelle de la CBox*

Nous avons commencé à travailler ensemble avec Mario dès mon DEA de musicologie (dont l'intitulé *Programmation d'outils logiciels pour les compositeurs* préfigurait l'orientation de ce doctorat), sur le projet K-Box – une extension logicielle pour instrument percussif [Barkati, 2001]. Ce logiciel, qui prenait en compte la sensibilité du jeu de l'interprète, fut expérimenté avec un résultat positif dans la composition *Erre-Erre*, spécialement écrite par Mario pour son utilisation[1].

Après cette première expérience en collaboration sur un projet interactif, concret et fonctionnel, l'idée d'un nouveau projet s'est formée pour se concentrer sur une réflexion commune : donc un projet davantage destiné à la recherche et à l'expérimentation, à partir de l'intérêt grandissant de Mario pour le microtemps (1) (2), de sa déception quant aux

1. Avec le soutien de l'association Densité 93 et créé par l'ensemble Soli-Tutti.

granulateurs ne proposant que des contrôles macrotemporels (3), et enfin, parallèlement, à partir de la lecture de Francisco Varela, notamment autour de la question de l'autopoièse[2]. Trois citations de Mario, extraites de mes notes de travail, étayent les deux premiers points :

- (1) C'est ce qui m'attire le plus en ce moment : l'incorporation du microtemps dans la composition musicale.
- (2) Je ne connais pas encore de travaux en temps réel où le microtemps a un degré d'autonomie fort.
- (3) Du point de vue de la composition, le problème majeur des granulateurs, c'est qu'ils partent d'une vision plus large, coincée dans des valeurs statistiques. Les grains restent alors esclaves du temps de la note ou de celui de la phrase[3].

Ainsi, la question des possibilités d'une articulation microtemporelle dans l'interaction en temps réel fonde une des orientations majeures du projet, à partir de l'idée d'une granulation non statistique, qui renouerait avec une forme de manualité dans la composition. La deuxième orientation importante reprend à son compte métaphoriquement la circularité varelienne[4] (d'où le choix du nom en extensif *CircularBox*), pour justement tenter de bâtir un modèle fonctionnel d'articulation temporelle multi-échelle et interactive à partir de l'idée originale des « vertus créatrices » de certaines circularités :

> Habituellement, de telles éventualités s'appellent des cercles vicieux ; ils sont précisément considérés comme ce qu'il faut éviter. Je suggère, au contraire, que *ces cercles ont des vertus créatrices*[5]. [Varela, 1987, p. 19] (c'est nous qui soulignons).

2. Dans son ouvrage *Autonomie et connaissance*, Varela place l'autopoièse au cœur de son raisonnement, à partir de l'examen du fonctionnement des cellules vivantes : « nous affirmons que *la notion d'autopoièse est nécessaire et suffisante pour définir l'organisation des êtres vivants.* » [Varela, 1987, p. 48, c'est l'auteur qui souligne] « Ces systèmes produisent leur identité ; ils se distinguent eux-mêmes de leur environnement : c'est pourquoi nous les nommons autopoiétiques, du grec *autos* (soi) et *poiein* (produire). Un système autopoiétique est organisé comme un réseau de processus de production de composants qui (a) régénèrent continuellement par leurs transformations et leurs interactions le réseau qui les a produits, et qui (b) constituent le système en tant qu'unité concrète dans l'espace où il existe, en spécifiant le domaine topologique où il se réalise comme réseau. » [Varela, 1987, p. 45] Du point de vue musical, ces idées peuvent se révéler stimulantes à de nombreux titres – notamment sous les aspects de cohérence et d'identité de l'œuvre – mais surtout concernant les modes d'articulation entre les niveaux du global et du local, sur lesquels elles apportent un éclairage nouveau : « Le rapport entre le global et le local est fourni par la nature cyclique et autoréférentielle des interactions entre les cellules et leur environnement, c'est-à-dire par leur clôture. » [Varela, 1987, p. 103]

3. Cette remarque a été complétée par Mario dans un entretien postérieur : « Ce qui est paradoxal dans l'incorporation du microtemps et plus particulièrement dans la synthèse granulaire, c'est qu'on part d'une vision macrotemporelle, et pour trouver une certaine souplesse, on fait appel à des valeurs statistiques. Les grains restent alors esclaves des échelles supérieures et cela s'entend très vite, si complexe soit la loi qui les gouverne. Autrement dit, on renonce à la possibilité d'accéder directement au microtemps. Quoi qu'il en soit, une alternative à l'approche aléatoire pourrait consister, par manipulation directe ou pas, à travailler sur l'interaction de plusieurs échelles différentes, c'est-à-dire de façon à ce que ce qui se passe dans le niveau le plus petit ait une répercussion dans les autres échelles et inversement. Pour cela, la mise en place d'un système de réinjection, ou d'un réseau de circuits fermés (ou comme dit Varela par "clôture opérationnelle") pourrait s'avérer plus adéquat que par linéarité. »

4. Le concept de circularité développé par Varela s'inscrit en fait dans un mouvement beaucoup plus large, lequel a comme origine les travaux de Norbert Wiener sur le *feedback* et tout le mouvement multidisciplinaire qui s'est constitué par la suite sous le thème *« Circular Causal and Feedback Mechanisms in Biological and Social Systems »* dès la première conférence Macy en 1946 à New York [Segal, 2003, p. 187].

Comme le montrent les quatre autres remarques suivantes de Mario, notre recherche d'un modèle circulaire composable et interactif s'appuie sur l'imbrication de deux postulats forts – le son comme *thesis* (4) et le retardement comme processus compositionnel (5) (6) – et sur la prévalence de la composition sur le hasard (7) :

- (4) Tout son est déjà une thèse ; il n'y a pas de neutralité. [6]
- (5) Le retard pensé non pas comme un effet, mais comme un processus compositionnel.
- (6) La façon dont on répète est à composer : répéter juste après (effets sonores), bien après (réexpositions), ou en même temps qu'un autre élément… tout doit être spécifiable.
- (7) Je n'aime pas trop le *random* ; pour moi, la composition reste une affaire de choix.

5.2 Deux attracteurs d'échange

En empruntant une métaphore dynamique de Pierre Boulez, le travail en collaboration à deux, sur un projet musical avec informatique, peut se comparer à une partie de ping-pong :

> La collaboration de Pierre Boulez et de son assistant musical Andrew Gerzso, débutée, en 1980, avec *Répons*, s'apparente selon le compositeur à une partie de ping-pong. « Comme je ne vais pas journellement en studio, nous parlons longuement du projet. Pas dans l'abstrait mais à partir de mes réalisations antérieures. Je fais des propositions musicales qu'Andrew Gerzso, musicien, comprend. Il cherche et me propose une solution que j'étudie pour voir si elle correspond à ce que je veux ou s'il faut encore l'élargir. Et ainsi de suite [7]. »

Or pour le développement de la CBox, la « partie de ping-pong » entre Mario et moi ne s'est pas déroulée de la même façon que dans les précédents projets, celui de la K-Box par exemple. Le cahier des charges comportait des points de flottement quant à la finalité du logiciel. D'un point de vue « ontogénétique », on peut dire très schématiquement qu'un conflit d'intérêts musicaux divergents a travaillé la CBox tout au long de son développement : d'un côté, Mario souhaitait toujours plus de fonctionnalités pour la composition, tandis que de l'autre, je cherchais à orienter l'utilisation de ce logiciel vers la performance musicale, vers la situation de concert. Aujourd'hui, avec le recul, on peut observer que loin de nuire au projet, cette opposition active a considérablement stimulé nos échanges, autour de débats régulièrement renouvellés par les avancées du logiciel, et

5. Le chapitre dont cette citation est issue, *L'histoire naturelle de la circularité*, reprend un précédent article de Varela au titre évocateur – *The creative circle : sketches on the natural history of circularity* [Varela, 1984].

6. Sous ces termes, on peut reconnaitre l'influence d'Adorno, mais le travail de micromontage qu'Adorno ne pouvait pas connaître leur donne une résonance toute particulière, une valeur renouvelée : « Tout ce qui peut apparaître en musique comme immédiat et naturel est déjà – pour parler le langage de la logique dialectique – médiatisé : résulte déjà d'une "thesis" ; le son isolé n'échappe pas à cette règle. » [Adorno, 1963, p. 319]

7. Propos recueillis par Pierre Gervasoni, dans un numéro hors-série du *Monde* de juin 2000, consacré au festival Agora.

que, en définitive, la divergence de nos intérêts a fonctionné à la manière de deux attracteurs étranges, favorisant le développement tantôt dans un sens, tantôt dans l'autre, vers une forme résultante singulière.

Les premiers effets de cette lutte entre les deux types d'usage visés sont apparus dès l'établissement du cahier des charges. En particulier, nous avons dû choisir entre la possibilité de programmer deux logiciels séparés et celle de programmer un seul logiciel qui intégrerait ces deux types de contraintes : après réflexion, il nous a semblé que l'intégration serait possible, car *Max/MSP* est certes un environnement orienté vers le temps réel, mais il permet aussi de programmer des fonctionnalités pour l'écriture.

Niveau	**Application**	**Pôle**
1. Paramètres	Ajustement des paramètres	C
	Tâtonnement paramétrique	P
2. Préréglages	Ajustement des préréglages	C
	Improvisation de préréglages	P
	Interprétation de préréglages	P
3. Partition	Organisation du rappel des préréglages	C

TAB. 5.1 – *Les trois niveaux de contrôle de la CBox*

En effet, comme le montre le tableau 5.1, les deux pôles – composition (C) et performance (P) – ont été ménagés, à travers une architecture à trois niveaux informationnels et un interfaçage dédoublé, c'est-à-dire adapté aux deux usages visés sur chaque niveau (cf. section suivante pour une discussion sur la notion d'*usage*).

Finalement, après avoir été façonnée par un processus alternatif, la CBox répond effectivement aux deux exigences, elle est devenue « bicompétente » en quelque sorte, elle a pris un pli ou une forme « bipolaire », à la façon d'un aimant qui se retourne pour présenter son pôle nord ou son pôle sud selon la nature de ce qui entre dans son champ...

5.3 Interactivités

L'exercice de l'étude de cas pourrait nous permettre d'éclaircir ou de généraliser certains aspects des logiciels d'informatique musicale, par exemple à propos de l'interactivité. Ce terme dont la signification semble bien claire – un principe d'action réciproque – est cependant utile à la description de tant de situations, qu'il s'est fixé dans de nombreux discours incompatibles entre eux ; en quelque sorte, il bénéficie et pâtit d'une éloquence trop vaste... Il ne distingue en effet ni entre la nature des éléments en interaction – objets, processus, humains – ni entre les échelles temporelles véritablement concernées. Yann Orlarey évoque ce double caractère indispensable et flou [Orlarey, 2006, p. 3] :

> Bien entendu, c'est une banalité que de rappeler que nous interagissons en permance avec le monde qui nous entoure. Le monde « est » interactif. Mais la technologie nous permet de changer ces interactions, d'en inventer de nouvelles, de les écrire, de les décrire et de les « mettre en œuvre ».

Même en ne considérant que les aspects qui concernent le logiciel[8], on peut discerner plusieurs niveaux d'interaction, dont un niveau technique et deux niveaux pratiques dans le cas de la CBox :

- une interaction intra-numérique, entre deux processus informatiques, que nous avons dénommée « circularité de type 2 » [Barkati et Lorenzo, 2005] ;
- un premier type d'interaction homme / machine, *pour la composition*, c'est-à-dire la spécification et l'enregistrement des paramètres et des différents fichiers (MIDI, audio, ou de configuration) ;
- un second type d'interaction homme / machine, *pour la performance musicale*.

Il nous semble important de distinguer ces deux dernières interactions même si elles partagent la même structure apparente du dialogue homme / machine, car nous pensons que c'est précisément cette inaptitude du terme « interaction » seul à discriminer entre ces deux usages – fondamentalement différents et majeurs en informatique musicale – que le couple d'expressions « temps réel » et « temps différé » a colonisé progressivement le champ pratique, depuis son champ d'origine, technique ; mais nous avons vu dans la première partie (cf. section 2.2 page 42) qu'à leur tour ces expressions recouvrent alors une polysémie à démêler, au fur et à mesure de leur expansion. Dès lors, comment trouver une terminologie adéquate pour désigner sans ambiguïté ces deux usages interactifs différents ? Car la question des *usages* reste tout à fait centrale, c'est elle qui détermine les choix de programmation, ce que confirme le statut de « l'Utilisateur » dans le vocabulaire des informaticiens[9]… et derrière cette figure commode de *l'utilisateur*, référentielle et « allative[10] », se retrouve la question de *l'usage* du logiciel, de sa finalité supposée (ou plutôt de l'espace de ses finalités supposées), de sa visée.

Autrement dit, l'interactivité ne concerne pas seulement l'instrumentiste, mais aussi le compositeur, et même d'une façon primordiale, comme le fait remarquer Tristan Murail :

> La façon de concevoir l'interface utilisateur est primordiale : c'est presque plus important que le programme lui-même ! L'interface doit permettre d'interagir librement et rapidement avec la machine [...] [Murail, 1992]

Dans ce sens pratique de l'interface-utilisateur, l'interactivité dénote donc la souplesse des échanges homme / machine, conditionnant en dernière instance la créativité, au-delà de la création. Cette vision éloigne l'ergonomie de la superficialité de la cosmétique, précise leurs frontières, et, surtout, revient sur la primauté de la puissance « pure » en informatique – essentiellement la puissance de calcul et la capacité de stockage – en plébiscitant

8. J'écarte ici les aspects compositionnels de l'interaction comme « modalité interne à l'œuvre » décrits par Vaggione : « Je vais employer abondamment, dans ce qui suit, le mot "interaction". Ce mot ne désigne pas seulement une flexibilité logicielle au niveau de l'interface-utilisateur, ni le fait d'un dépassement de la dichotomie temps réel/temps différé ; plus profondément, il fait référence à une inter-relation postulée entre divers paliers temporels et échelles de représentation. L'interaction correspond ici à une modalité interne à l'œuvre : une façon d'aborder la complexité du jeu de dimensions qui véhicule la composition musicale. » [Vaggione, 1998b, p. 171]

9. Pierre Lévy justifie qu'il s'agit d'une abstraction, non sans humour : « Dans les milieux de l'informatique, on annonce toujours avec emphase que le fameux "utilisateur" du logiciel est un personnage capital. [...] Personne n'a jamais rencontré *l'utilisateur* des informaticiens, parce que *les utilisateurs* de la plupart des logiciels sont légions. » [Lévy, 1992, p. 17]

10. Nous importons le nom donné en linguistique au cas de la déclinaison qui dans certaines langues sert à indiquer *ce vers quoi tend une action* (notion opposée à celle de l'ablatif), pour désigner ici *ce vers quoi tend une programmation* (en tant qu'action).

en quelque sorte la forme plutôt que le fond[11]. Je ne crois pas que cette vigueur de la forme traduise une quelconque escroquerie en faisant passer le fond au second plan, mais simplement que l'informatique, et plus généralement les nouvelles technologies, sont arrivées à un stade de leur évolution qui leur permet d'envisager de s'adapter plus sérieusement à l'humain – à sa perception, son fonctionnement, son corps, sa manualité et sa cognition. Ce changement de paradigme – l'ergonomie contre la puissance – réaffirme donc l'humain dans son rapport avec la machine et appelle davantage de finesse quant à la définition des usages, pour vouloir y répondre de façon la plus adaptée possible. L'ergonomie devient ainsi un critère décisif vis-à-vis de la programmation, permettant d'appréhender la *qualité* des interactions homme / machine, leur quantité étant déjà comprise dans la notion d'interactivité.

Ainsi, nous tenterons de distinguer dans les sections suivantes les notions de *performatif* et de *compositionnel* dans le souci d'affiner une grille d'analyse des usages pour les logiciels musicaux.

5.4 Performativité

Au sein d'un logiciel mixte comme la CBox, en vue de la programmation, la question devient la suivante : quels sont les usages visés et quelles propriétés doit acquérir le logiciel pour pouvoir répondre à ces usages ?

Nous avons dénombré deux usages essentiels – la composition et la performance. Ceux-ci correspondent respectivement à la figure du compositeur et à celle de l'interprète (ces figures constituant deux profils d'*utilisateur* différents du point de vue du programmeur, sachant que dans les faits ces rôles peuvent évidemment se fondre en une seule personne ou se distribuer sur plus de deux) et appellent deux propriétés à implémenter dans le logiciel : la *composabilité* et la *performabilité*. Si le terme « composabilité » est relativement univoque[12], le choix de celui de « performabilité » demande quelques explications (qui devraient incidemment éclaircir la notion d'interactivité).

D'abord, notons que Miller Puckette, mathématicien et programmeur, indiquait déjà des termes proches dans un article de 2004 : *A divide between 'compositional' and 'performative' aspects of Pd*. Les guillemets sont de l'auteur et dénotent sans doute un certain flou, d'autant que ces termes ne seront pas directement éclaircis dans l'article, néanmoins les deux catégories sont clairement posées.

Ensuite, relevons que le compositeur Agostino Di Scipio, à l'occasion du travail sur son projet *Ecosistemico Udibile*, a proposé récemment un éclaircissement de la signification de la performance :

11. Le succès récent des ordinateurs à bas coût, utilisant des éléments moins puissants, est un phénomène en rupture avec les anciens schémas et illustre en tant que tel le recul de la puissance comme critère dominant. Parallèlement, la récente fortune de l'ergonomie peut être illustrée par les investissements dans l'innovation qui font effectivement le succès radical de certains appareils plutôt que des concurrents plus puissants (comme une molette ergonomique sur des baladeurs numériques).

12. cf. *Entre le décomposé et l'incomposable* [Courtot, 1993], *L'espace composable : sur quelques catégories opératoires dans la musique électroacoustique* [Vaggione, 1998a] et *Espaces composables - essais sur la musique et la pensée musicale d'Horacio Vaggione* [Solomos *et al.*, 2007].

> « Forme », on le sait, est clôture, distinction, perception de la différence.
> « Formance » est prédisposition à la clôture, à la distinction, à la différence perceptible.
> « Performance » est le processus dans lequel la clôture a lieu, la conclusion qui consiste à clore le temps avec le lieu. [Di Scipio, 2008, p. 244]

Enfin, nous proposons les définitions suivantes :

Définition (Performable)**.** Est « performable » tout logiciel ou dispositif pouvant être joué musicalement, notamment en situation de concert.

Propriété (Performabilité)**.** La « performabilité » sanctionne le caractère « performable » d'un logiciel ou d'un dispositif.

À la lecture de ces définitions, on peut se demander si le substantif « jouabilité » n'eût pas été plus simple ou plus judicieux que « performabilité », étant donné la présence du verbe jouer, d'autant qu'une référence potentielle au *jeu* instrumental semble le rapprocher du champ musical. Cependant, deux raisons s'y opposent. D'une part, le terme « jouabilité » désigne déjà en informatique une autre idée : le degré de confort et de rapidité d'utilisation *d'un jeu vidéo*, et, pour le coup, nous éloigne de la musique. D'autre part, contrairement à notre idée de la « performabilité », cette notion vidéoludique de « jouabilité » se place en fait au-delà du *statut* (prédicatif) d'un logiciel pour indiquer un *degré* (relatif)[13].

Le point le plus délicat consiste à qualifier les types d'interactivités alors en jeu : comme pour le couple composabilité / performabilité, on comprend clairement l'expression « interactivité *compositionnelle* » pour désigner la partie de l'interface-utilisateur destinée à l'activité de composition et sans doute moins clairement l'expression « interactivité *performative* » pour désigner la partie de l'interface-utilisateur destinée à la performance musicale. En effet, la notion de « performatif » a déjà été développée dans un autre domaine, celui de la linguistique, par plusieurs auteurs – en particulier par Austin[14] et Chomsky[15] – dans des sens différents entre eux et différents de celui que nous proposons. Cependant, on peut considérer, d'une part, qu'un mot déjà polysémique supporte mieux un sens supplémentaire qu'un mot monosémique et, d'autre part, qu'on peut affilier ce nouveau sens métamusical au sens linguistique chomskien en substituant l'idée de la mise en acte du langage par celle de la mise en acte du programme, en premier lieu parce qu'il s'agit dans les deux cas de *l'actualisation de compétences* (linguistiques dans un cas et informatiques dans l'autre) et en second lieu parce que ces deux notions partagent la propriété de *l'irréversibilité* dans cette actualisation (comme le temps réel *pratique*

13. Formulé autrement : dire d'un logiciel qu'il est « non-jouable » aurait plus difficilement une signification (un jeu est toujours « jouable », même mal) que de dire qu'un logiciel est « non-performable » (un logiciel de CMAO n'émet pas forcément du son par exemple).

14. John Langshaw Austin a développé la notion de *performatif*, distinguée de la notion de *constatif*, dans son ouvrage *Quand dire, c'est faire* (l'édition originale *How to do things with words* date de 1962). Elle caractérise certaines expressions qui *font* littéralement ce qu'elles énoncent, « Je vous marie » (prononcée dans la situation qui convient) constituant un exemple canonique.

15. Dans sa théorie de la grammaire générative transformationnelle, Noam Chomsky a développé la notion de *performance* comme activité linguistique réelle d'un sujet, à distinguer de la *compétence* comme savoir implicite de la langue, notamment dans son ouvrage *Aspects de la théorie syntaxique* (l'édition originale *Aspects of the theory of syntax* date de 1965).

défini dans la première partie à la section 2.2 page 42). Incidemment, on peut noter que l'interprétation musicale peut elle aussi se comprendre assez bien à partir cette analyse chomskienne – entre compétence et performance –, ainsi que l'improvisation.

Définition (Performatif)**.** Est « performative » toute interactivité susceptible d'être utilisée musicalement en direct, notamment en situation de concert.

Propriété (Performativité)**.** La « performativité » désigne la qualité « performative » d'une interactivité.

Pour acquérir cette qualité *performative*, un logiciel doit répondre aux exigences des échelles temporelles impliquées lors de la performance ainsi qu'à celles de la contrainte d'irréversibilité de la performance. Typiquement, il s'agit de déployer des mécanismes de contrôle (déclenchement, régulation), de rétrocontrôle (affichage des valeurs importantes, arrêt d'urgence) et d'anticipation (prévisualisation, compteurs), dans une échelle temporelle intégrable dans le temps de la performance, c'est-à-dire dans le temps de l'instrumentiste en situation d'interprétation ou d'improvisation. Pour définir cette échelle temporelle propre, nous nous appuyons sur la catégorie de la note telle que l'envisage Horacio Vaggione comme niveau de référence entre macrotemps et microtemps :

> Il me semble cependant pertinent de faire une distinction générale, en deçà des micromondes postulés dans des œuvres particulières, entre deux grands domaines opératoires : le *macro-temps* (qui englobe toutes les échelles possibles s'étalant vers le « grand », à partir de celle de la *note*) et le *micro-temps* (qui englobe toutes les échelles possibles s'étalant vers le « petit », en-dessous de celle de la *note*). La catégorie de note (dans sa signification conventionnelle) est ici postulée comme niveau de référence, comme ligne d'horizon entre macro-temps et micro-temps, afin de cerner intuitivement l'enjeu de cette distinction.

En effet, c'est principalement cette échelle qui permet d'orienter les développements du logiciel vers une interactivité performative. Notons pour finir que la notion de *performativité* permet de formuler une hypothèse à propos de la différence entre « le direct » et « le *live* » : car il n'y a pas nécessairement *performance* dans le premier, qui reste malgré tout attaché à la retransmission d'émissions (donc à une modalité de communication synchronisée mais délocalisée), alors que dans le second, l'idée de performance est prépondérante, comme on la retrouve dans l'idée du « spectacle vivant » (c'est-à-dire, à l'origine, sans écran interposé, localisée, dans un *ici-ensemble* – quitte à devoir ré-imaginer la situation de concert pour les albums dits *« live »*). On peut supposer que cette idée de la performativité permet d'expliquer en partie la persistance de l'expression *« live electronic »* dans les pays anglo-saxons comme dans les pays francophones (où l'expression « musique électronique en temps réel », plus lourde, reste toutefois employée).

5.5 Composabilité

Pour un logiciel d'informatique musicale, les modalités d'interaction ne se réduisent pas à la performabilité, bien que le raccourci soit souvent employé sous les intitulés métonymiques « dispositif interactif » ou « musique interactive ». Comme nous l'avons mentionné précédemment avec la remarque de Tristan Murail (cf. section 5.3 page 107), l'interface-utilisateur intéresse tout autant l'usage compositionnel que l'usage performatif, à des niveaux différents cependant. En outre, une partie importante des problématiques de la CMAO peut se condenser en cette qualité particulière de l'interaction homme / machine – la composabilité.

La composabilité se distingue constitutivement de la performabilité dans son rapport au temps, car l'un des intérêts majeurs de la composition – c'est d'ailleurs un de ses fondements – réside précisément dans la décorrélation entre le temps de l'écriture et celui de l'exécution musicale. Ainsi, le travail d'écriture devient possible à condition de quitter l'irréversibilité du temps de la performance : c'est-à-dire à condition de bénéficier de la *possibilité du retour* telle que nous l'avons présentée dans la première partie, pour montrer le rapport entre la composition et le temps différé comme catégorie (cf. section 1.3.3 page 32), en *différant* la performance.

Premièrement, l'enregistrement, au sens le plus large, représente le vecteur fondamental du processus qui permet de différer l'actualisation de la musique portée par l'écrit au sens large – par de *l'inscrit* –, qu'il s'agisse d'une partition manuscrite, d'une bande magnétique ou d'un disque dur (des compétences complémentaires idoines, humaines, mécaniques ou informatiques doivent ensuite assurer l'actualisation musicale, soit respectivement l'interprétation, la transduction et la traduction pour ces trois cas particuliers). Ainsi, plus un logiciel possédera de compétence pour l'enregistrement et la gestion de fichiers, plus il pourra être qualifié de *composable*.

Deuxièmement, la prise en compte de la décorrélation temporelle libre propre à l'écriture implique au niveau logiciel, d'une part l'abandon de la contrainte forte sur la latence – les temps réels technique et pratique ne sont plus tenus d'être garantis – et, d'autre part, l'implémentation de fonctionnalités différentes de celles de la performativité, dans un cadre de pensée différent. Car en sus de la réversibilité temporelle, qui appelle des fonctionnalités *d'édition* (cf. figure 5.2 page suivante), l'appréhension des échelles temporelles doit être multiple : typiquement, cela peut se traduire par des fonctions de *zoom* (sur des formes d'ondes, des enveloppes dynamiques, des partitions, etc.) mais surtout, plus généralement, par la prise en charge des différentes échelles composables au moyen de représentations et d'opérations adaptées à ces échelles, à l'image de la nécessité exprimée par Horacio Vaggione pour les échelles microtemporelles :

> [...] en changeant d'échelle, on change également de système de représentation. Les symboles macroscopiques n'ayant pas de pertinence au-delà de leur dimension spécifique, il nous faut trouver d'autres systèmes de symboles relevant d'une catégorie opératoire valable au niveau du micro-temps. [Vaggione, 1998b, p. 176]

Troisièmement, puisque stipuler les paramètres de l'interaction *performative*, c'est participer à la définition et *in fine* à la construction de l'instrument de musique, on peut considérer que le travail de composition électroacoustique s'enrichit de plus en plus souvent d'une partie du métier du luthier numérique : celle qui consiste en *l'ajustement*

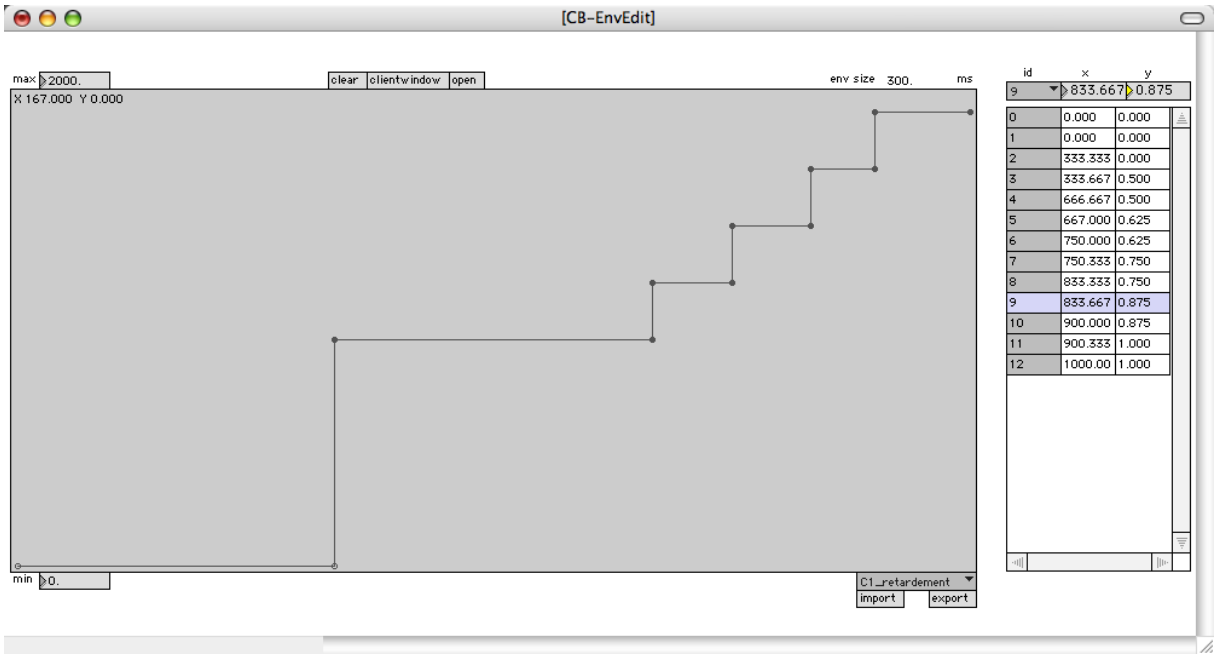


FIG. 5.2 – *L'éditeur d'enveloppe de la CBox*

paramétrique. Ce cas se produit en fait avec chaque logiciel qui appartient à la classe des logiciels *performables*. À l'inverse de la vision traditionnelle qui considère que les méta-instruments sont premiers et s'adjoignent des logiciels secondairement, les logiciels deviennent premiers dès lors qu'ils sont capables d'être joués par différents dispositifs de contrôle – qu'il s'agisse de gants extrêmement élaborés ou d'une vulgaire souris – comme autant de méta-instruments de fait. Or la plupart des logiciels performables répondent à des protocoles standardisés (MIDI, ASCII, OSC, etc.) et cette conformité protocolaire, qui leur permet aisément d'être contrôlés par une multitude d'appareils, entérine donc ce renversement. Quoi qu'il en soit, un logiciel *performable* récupère généralement la partie du traitement des informations (l'interface logique) provenant du dispositif de contrôle (l'interface physique) et figure ainsi une fonction de transfert complexe et *composable* du méta-instrument déjà jeté dans le symbolique par son ontologie informatique[16].

Enfin, il faut souligner que les interactivités *compositionnelle* et *performative* ne s'opposent pas – ce que prouve l'intégration des deux types au sein d'un unique logiciel comme la CBox. Ces deux types d'interactions homme / machine forment une bipolarité pratique, qui peut guider les développements logiciels, mais pas technique : ce sont seulement des *usages* visés qui ne clivent pas les aspects techniques, ni en termes de langage de programmation ni en termes de traitement du signal. Cette absence de clivage fonde d'ailleurs l'idée de *régions fractionnaires* développée dans l'article.

16. « Le méta-instrument ne joue plus le rôle de la lutherie acoustique ou électroacoustique, qui pourraît être définie comme la résonance sonore du geste, mais bien celui d'une *interface*. Ce changement d'état est fondamental en ce qu'il détermine une rupture : le méta-instrument n'agit plus dans le domaine physique, mais dans celui, symbolique, de la représentation. » [Battier, 1999b, p. 311]

5.6 Circularités

Dans la CBox, les possibilités de spécification de l'organisation temporelle sont de deux ordres : une organisation (facultative) directe sous la forme bidimensionnelle traditionnelle du rouleau musical (*piano roll*), avec l'objet `detonate`, et une organisation indirecte sous la forme des processus circulaires qui possèdent une durée propre et qui sont déclenchables par des raccourcis-clavier prévus à cet effet. Ensuite, le retardement et la réinjection croisée sont promus de fait principes compositionnels majeurs. Ainsi, un système basé sur un nombre restreint de fonctions atomiques – cinq opérations dynamiques simples qui s'enchaînent en série à l'intérieur d'un canal stéréophonique – tente de proposer une expérience sonore particulière des circularités de types 1 et 2 : soit, respectivement, la réinjection de la sortie d'un circuit à sa propre entrée (le canal 1 possède une entrée Canal 1) et la réinjection de la sortie d'un circuit à l'entrée d'un autre circuit. La figure 5.3 montre l'organisation structurelle du parcours du signal audionumérique à travers les trois canaux qui composent la CBox.

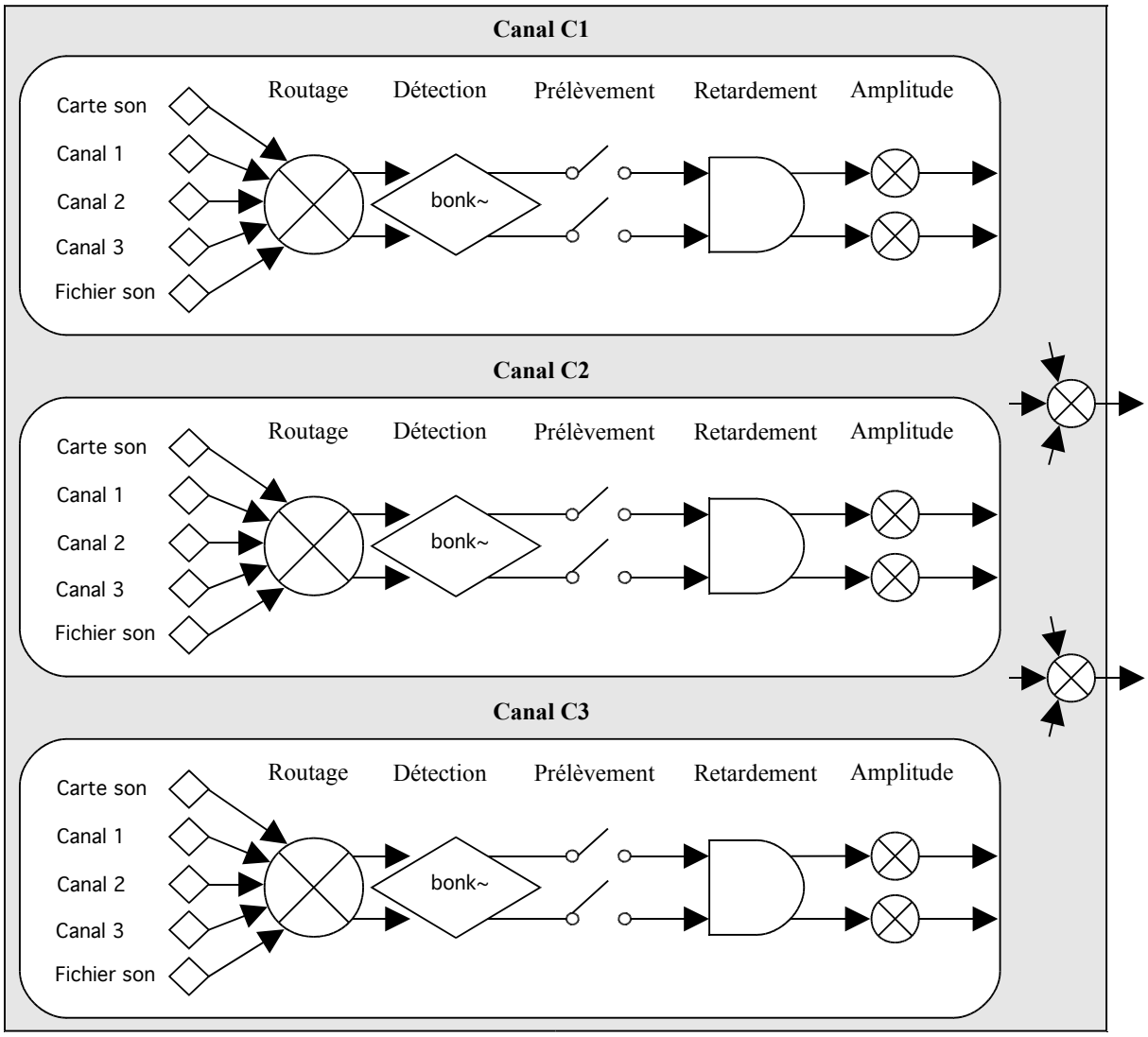


FIG. 5.3 – *Schéma logique du parcours du signal audionumérique dans la CBox*

La complexité de l'écriture provient essentiellement de la circularité de type 2, c'est-à-dire de la réinjection croisée entre deux canaux différents. Pour accompagner l'écriture de cette circularité, un module de visualisation zoomable (cf. figure 5.4) a donc été développé (avec l'objet `lcd`), et représente en couleur les entrées effectivement sélectionnées au niveau du routage de chaque canal, dans l'ordre de rappel indiqué dans la partition Midi. En particulier, ce module permet ainsi de se faire une idée plus précise des trajets possibles du son à travers les croisements inter-canaux.

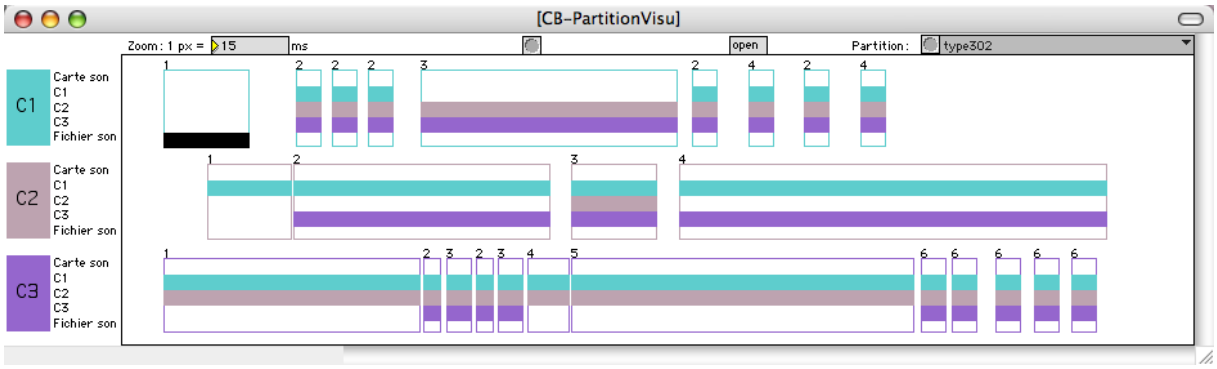


Fig. 5.4 – *Visualisation des routages croisés de la partition*

Parmi les cinq opérations élémentaires mentionnées précédemment, il en est donc une qui se démarque pour s'articuler à la circularité : le retardement. En effet, d'un point de vue formel, on peut considérer que c'est la relation entre ces deux opérations – la circularité et le retardement – qui est susceptible de fonder un support cohérent à la structuration temporelle au sein du logiciel. À une certaine échelle temporelle, disons « méso-temporelle », le phénomène de l'écho illustre cette relation ; or, d'une part, l'écho possède une prégnance incontestable, par sa nature répétitive, et d'autre part, selon les échelles, les ressorts musicaux du retardement changent, depuis la réverbération et la granulation dans le microtemps, jusqu'au canon et à la réexposition dans le macrotemps, en passant par l'écho ou le hoquet.

Finalement, le couplage de la circularité et du retardement intègre de nombreux procédés musicaux, qui relèvent tantôt de la qualité acoustique tantôt de l'articulation discours musical, ou encore de la structuration de la macro-forme. Néanmoins, il faut reconnaître que la beauté formelle de la concision opératoire de ce simple couplage a un prix, d'une part parce que la complexité bascule alors du côté du compositeur, qui doit faire l'effort de se figurer lui-même quels types de résultats perceptifs se cachent derrière les valeurs numériques et les enveloppes dynamiques qu'il écrit, et d'autre part parce que, comme le dit Mario, « le plus difficile avec la circularité, c'est de ne pas l'entendre. »

Chapitre 6

Mimi et Rose Amère / Stéphanie Réthoré

Résumé du chapitre : Ce chapitre présente Mimi et Rose amère, deux logiciels interactifs développés en collaboration avec Stéphanie Réthoré dans l'objectif d'« augmenter » un instrument acoustique – ici un alto – à l'aide d'un microphone, d'un ordinateur et d'un pédalier multiple, principalement pour de l'improvisation. Mimi, un logiciel relativement rudimentaire mais tout à fait fonctionnel, a servi à démontrer la stabilité du dispositif dans les conditions pratiques du temps réel, tandis que Rose amère, logiciel tout aussi fonctionnel mais beaucoup plus complexe, a déjà eu plusieurs occasions concluantes d'être joué en concert, lui dessinant un début de « carrière » intéressant.

6.1 L'improvisation à la corde

6.1.1 Une altiste improvisatrice

Stéphanie Réthoré suit un double cursus au Cnsm de Paris : elle étudie l'alto dans la classe de Bruno Pasquier, ainsi que l'improvisation générative dans la classe d'Alain Savouret. Nous nous sommes rencontrés en 2002 lors d'un stage de musique pour préparer nos concours d'instrument, mais notre collaboration sur ce projet est beaucoup plus récente et s'est inscrite au départ dans le cadre de son cursus d'improvisation générative au Cnsm, où elle a obtenu le Prix d'improvisation générative en juin 2008, avec le logiciel Rose amère pour son projet libre. En effet, elle avait choisi pour cette partie d'improviser avec son alto « augmenté », c'est-à-dire augmenté par le dispositif podophonique interactif[1] comprenant essentiellement le pédalier, l'ordinateur et ce dernier logiciel que nous avions établi ensemble.

6.1.2 Des contraintes pour l'improvisation générative

Assez tôt dans la collaboration, nous avons entrepris d'inventorier ensemble les contraintes majeures qui nous paraissaient liées à l'improvisation générative, dans son exercice collectif ou individuel, puis de les hiérarchiser ; ce recensement nous a guidé durant le développement des deux projets, Mimi et Rose amère. En outre, la spécificité de l'improvisation bascule radicalement la conception d'un tel dispositif musical vers le point de vue de l'interprète.

L'équilibre sonore

Dans un contexte d'improvisation à plusieurs musiciens, les besoins musicaux peuvent être de plusieurs ordres, mais celui de l'équilibre sonore est primordial. L'amplitude sonore naturelle de l'alto acoustique par rapport à celle de la batterie, du saxophone ou du piano, par exemple, souffre d'un déficit flagrant. Il s'agit donc dans un premier temps d'« augmenter » l'instrument au sens de l'amplifier, l'agrandir, le multiplier, de lui donner une plus grande portée sonore, davantage de place dans le collectif.

Par contre, il faut éviter, lors de cette opération, de tomber dans un excès inverse, où cette extension prendrait trop d'importance au point de soumettre et contraindre en permanence les autres improvisateurs. Autrement dit : comment augmenter l'instrument tout en évitant que l'improvisation ne s'organise seulement autour du logiciel ?

Ainsi, paradoxalement, cette question de l'équilibre sonore au sein d'un groupe est à la fois une motivation historique importante pour l'augmentation de l'alto et reste la réserve majeure quant à cette augmentation, car toute amplification a tôt fait de bouleverser l'équilibre entre les instruments, à plus forte raison si certains instruments ne sont pas amplifiés, ou bien si le dispositif emploie de l'informatique en temps réel.

1. Cf. section 4.3.6 page 100.

La variété musicale

La deuxième contrainte relève du domaine directement musical : l'extension logicielle doit pouvoir s'adapter à ce qui se passe *musicalement*, par exemple pouvoir alternativement accompagner, se taire, répondre, proposer, etc. La variété expressive devient ici une nécessité absolue, sous peine de se voir rapidement exclu du jeu musical collectif, ou pire, d'appauvrir les possibilités expressives du groupe.

On peut légitimement envisager de déployer deux catégories musicales dans ces situations : l'« horizontalité », les déroulements étales, continus et plutôt arythmiques, et la « verticalité », rythmique, accentuée, et éventuellement périodique. Ces deux catégories rejoignent d'ailleurs la formalisation duale entre *voix* et *percussion* proposée par Martin Laliberté dans son analyse *Archétypes et paradoxes des nouveaux instruments* :

> Elle [la *percussion*] recherche plutôt les objets sonores ambigus, tels les composites timbre/harmonie, que les objets sonores mélodiques simples. Il s'en suit une nature verticale, harmonique ou massique, contrastant avec la nature horizontale de la *voix*. [Laliberté, 1999, p.125]

> [...] la richesse de telle nouvelle forme instrumentale se mesure dans sa réalisation convaincante des grands courants de fond du développement organologique, par un succès dans la réalisation des tendances *vocales* secrètes d'un instrument de percussion ou par la floraison de l'aspect *percussif* d'un instrument en apparence *vocal*. [Laliberté, 1999, p.137]

Il y a bien sûr d'infinies gradations entre ces deux archétypes musicaux et, précisément, la réussite ou l'échec du logiciel dépend en grande partie de sa capacité à proposer un éventail le plus large possible entre ces deux extrêmes, idéaux.

La promptitude réactionnelle

La troisième contrainte concerne surtout la rapidité, voire l'instantanéité (perceptive) des transitions entre les différents modes de réponse musicaux de l'extension logicielle. C'est en effet souvent sur cette promptitude réactionnelle que repose la dramatisation du jeu collectif et la construction des évolutions du discours musical ensemble. Martin Laliberté rappelle la nécessité de prendre en compte les aspects concrets dans la réalisation et pas seulement les aspects musicaux :

> La grande majorité des nouveaux instruments furent des échecs à cause de lacunes fondamentales dans un des aspects essentiels propres aux instruments de musique. Non seulement les nouveaux instruments de musique ont-ils besoin d'une ouverture sonore, d'une richesse des approches et des solutions proposées aux différents problèmes musicaux mais aussi doivent-ils s'incarner de façon probante. [Laliberté, 1999, p.129]

En particulier, dans le contexte musical de l'improvisation, qui se caractérise par une versatilité potentiellement importante, une extension logicielle insuffisamment réactive constituerait de toute évidence un frein inacceptable.

* * *

L'ensemble de ces contraintes – l'équilibre sonore, la variété expressive et la promptitude réactionnelle – laisse encore une grande marge de créativité logicielle, à partir du dispositif informatique et électroacoustique choisi. Notre collaboration a ainsi donné lieu à deux auto-échantillonneurs interactifs pour pédalier MIDI multiple : d'abord Mimi, monophonique et relativement simple, puis Rose amère, à quatre voies et beaucoup plus complexe.

6.2 Mimi : un auto-échantillonneur prototype

Globalement, Mimi est un simple auto-échantillonneur monophonique, envisagé comme l'extension logicielle d'un instrument acoustique en situation d'improvisation. Son fonctionnement interne repose sur un enregistrement et une relecture dans une même mémoire tampon, éventuellement en simultané, tandis que son intérêt réside essentiellement dans ses capacités interactives (notamment l'interfaçage avec le pédalier multiple), sa simplicité et sa robustesse.

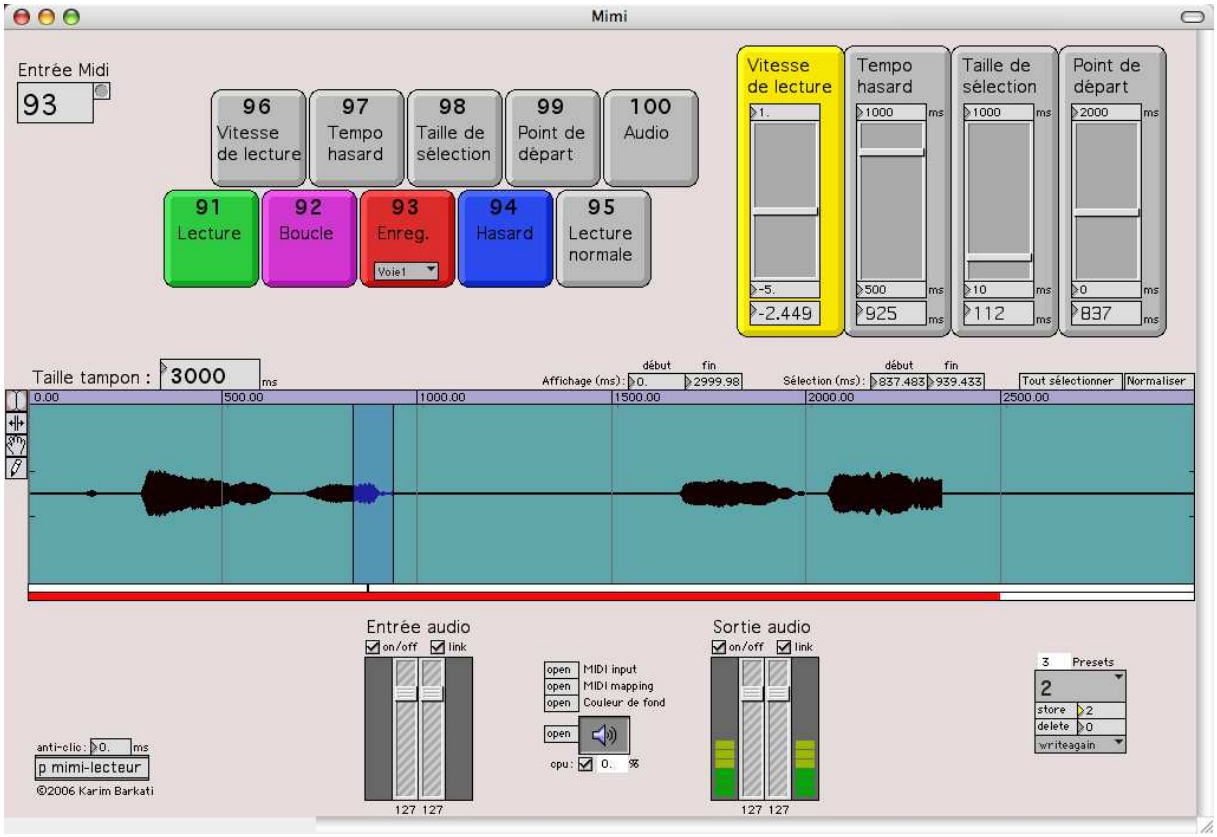


FIG. 6.1 – *Capture de l'interface visuelle de Mimi*

6.2.1 Un prototype de démonstration

Si aujourd'hui Mimi n'est pas le logiciel le plus intéressant parmi ceux développés au cours du doctorat, ce logiciel tient néanmoins une place toute particulière : il est à l'origine de toutes les collaborations de la deuxième période de ce doctorat – avec Stéphanie, Santiago, Mauricio, Pedro et Iván –, période dirigée vers l'objectif d'un concert. Avec Mimi, il s'est donc agi d'essayer de faire la démonstration de la stabilité d'un dispositif interactif (temps réel pratique), de faire sentir ses potentialités musicales et de présenter mes compétences sur *Max/MSP*. Ainsi, l'accent a été mis sur les aspects techniques et pratiques du temps réel plutôt que du temps différé, en écartant par exemple l'emploi de fichiers audio préenregistrés.

Le nom « Mimi » témoigne d'ailleurs de l'intrication des deux acteurs – l'humain et le dispositif podophonique interactif – en situation de direct : il provient de la boutade *« me, myself and I »*... Non pas pour signifier une quelconque tendance égocentrique, mais comme une façon d'indiquer la triple implication du musicien avec cet instrument augmenté : *« me »* pour le musicien qui joue de son instrument acoustique, *« myself »* pour l'enregistrement de ce son acoustique en temps réel, *« I »* pour le même musicien qui contrôle le logiciel avec le pédalier. Cette triple implication décrit en quelque sorte le rapport instrumental du musicien à son intrument « augmenté », c'est-à-dire augmenté par les possibilités sonores et musicales apportées par le pédalier, l'ordinateur et le logiciel.

En tant que prototype, Mimi intègre déjà les deux principes principaux qui sous-tendent la plupart des logiciels suivants : la performabilité[2] et la composabilité.

6.2.2 Un premier modèle d'interaction pédestre

Du côté de la performabilité du dispositif podophonique, Mimi a fait office de pionnier pour la série de mes logiciels développés dans ce doctorat. En effet, la majorité des techniques d'interfaçage complexe sont déjà explorées dans Mimi : en particulier la différenciation fonctionnelle des banques d'interrupteurs et la permutation fonctionnelle des pédales progressives. D'ailleurs, la plupart des logiciels postérieurs à Mimi restreindront l'une ou l'autre de ces techniques expertes, afin de réduire le temps d'apprentissage de l'instrumentiste, ainsi que les risques d'erreur lors du jeu en direct.

Différenciation fonctionnelle des banques

L'interface graphique de Mimi (cf. figure 6.1 page ci-contre) montre distinctement dix boutons dans sa partie supérieure, curieusement (de prime abord) numérotés de 91 à 100... Cette numérotation étrange annonce en fait une « différenciation fonctionnelle des banques » : toutes les banques ne sont pas associées aux mêmes fonctions. Ainsi, dans Mimi, les dix pédales de déclenchement du pédalier n'agissent sur les fonctions indiquées par les boutons du logiciel que lorsque la dixième banque du pédalier est activée (émettant des valeurs de *control change* comprises entre 90 et 99), tandis qu'elles rappellent des

2. Cf. section 5.4 page 109.

préréglages dans le cas contraire, c'est-à-dire lorsqu'une autre banque est active (émettant des valeurs de *control change* comprises entre 0 et 89)[3].

Le principe de la différenciation fonctionnelle des banques étant posé, la répartition dans Mimi se fait comme suit :

- pour les neuf premières banques, chaque pédale de déclenchement (de 1 à 90) rappelle le préréglage dont le numéro correspond s'il existe, et ne fait rien sinon ;
- pour la dixième banque, les quatre premiers boutons (91, 92, 93, 94) ainsi que le dernier (100) se comportent comme des interrupteurs *locaux*, respectivement pour la lecture, le bouclage, l'enregistrement, le tirage aléatoire et le moteur audio principal, tandis que les cinq boutons intermédiaires (95, 96, 97, 98, 99) se comportent comme des commutateurs *généraux*, le bouton 95 faisant « disjoncter » la vitesse de lecture en la ramenant à sa valeur normale (1.), et les quatre autres boutons se partageant la permutation fonctionnelle de la pédale progressive A (cf. section suivante).

Cette répartition entre les banques traduit simplement une préférence mnémotechnique : la numérotation des préréglages suit ici exactement celle des pédales de déclenchement. Du reste, l'accès à la dixième banque se fait aisément puisqu'il suffit pour passer de la première à la dixième banque d'appuyer une seule fois sur la pédale DOWN, selon un principe circulaire (il n'y a pas besoin de remonter les neuf banques intermédiaires).

Rigidification fonctionnelle de la pédale B

Nous avons décidé, en consultation avec plusieurs instrumentistes, que la pédale progressive de droite du FCB1010, la pédale B, située à l'extrémité du pédalier donc facilement repérable, servirait systématiquement à contrôler le volume[4] de sortie générale des logiciels pour toutes les collaborations.

De fait, l'abstraction d'une interface ou d'un système MIDI comporte des limites pratiques : si tout est toujours permutable, alors aucune association réflexe n'est possible… Or, du point de vue de l'instrumentiste en situation de répétition ou de concert, le contrôle du volume général du dispositif doit être le plus instinctif possible. Conséquemment, cette association volontairement « rigidifiée » de la pédale de droite au même paramètre du volume général, salutaire, a été adoptée et bien accueillie par l'ensemble des collaborateurs – compositeurs comme interprètes.

En effet, si l'on souhaite que l'instrumentiste puisse s'approprier le dispositif électroacoustique en tant qu'instrument supplémentaire et/ou augmentant, nous pensons que c'est à lui qu'il faut confier la responsabilité du volume général du dispositif. Toutefois, cela n'empêche pas un mixage en aval par quelqu'un d'autre, entre la sortie sonore du dispositif et le système de diffusion ; le mixage est même recommandé en situation de concert,

3. Notons que ce système décimal (10 pédales × 10 banques) s'est révélé se prêter particulièrement bien à la mémorisation pour l'instrumentiste ; le changement de banque se fait par incrémentation ou par décrémentation via les deux pédales notées UP et DOWN. Par ailleurs, on peut noter qu'il y a un décalage d'une unité entre la numérotation des pédales (qui commence à 1) et celle de la valeur des messages MIDI *control change* envoyés (qui commence à 0). Ce décalage unitaire provient de la différence entre *l'ergonomie* des indications du pédalier, notées entre 1 et 10, et de la *normalisation informatique* des messages MIDI codés de 0 à 127 (correspondant au codes binaires allant de 0000000 à 1111111).

4. Incidemment, il se trouve que cette pédale B envoie par défaut un message MIDI *volume*…

à la fois pour la qualité de la diffusion, comme toute sonorisation, et pour prévenir tout risque de *larsen* ou d'explosion sonore, risque rarement nul dans ce type d'installation électroacoustique[5].

Permutation fonctionnelle de la pédale A

Musicalement, les pédales progressives possèdent un avantage majeur sur les pédales interruptrices : une *course*, qui permet de varier, de doser, de moduler, de se positionner progressivement à l'intérieur d'un intervalle (de 0 à 127), et pas seulement de déclencher un évènement ponctuel. En cela, les pédales progressives sont tout à fait précieuses à l'instrumentiste, à l'improvisateur et à la composition, comme accès à un geste instrumental perceptivement continu, c'est-à-dire à un geste qui relève de l'archétype *vocal*, par opposition à l'archétype *percussif* dont relèvent les pédales interruptrices, pour reprendre l'analyse de Martin Laliberté[6]. Aussi, plus le nombre de paramètres contrôlables *progressivement* est important, plus le dispositif est susceptible d'être intéressant musicalement.

Après qu'on a décidé de réserver systématiquement la pédale B pour le volume général du logiciel, il ne reste donc que la pédale A de disponible pour le contrôle d'un second paramètre, mais on devine que le contrôle progressif de deux paramètres seulement – le volume et un seul autre – peut rapidement se révéler insuffisant à renouveler l'intérêt musical. Comment donner la possibilité de contrôler plus de deux paramètres avec seulement deux pédales progressives, dont l'une est rigidifiée ?

Une première réponse consiste à permuter les fonctions qui sont associées à la pédale A : la « A-permutation » Ainsi, dans Mimi, les boutons 96 à 99 permettent de permuter la fonction associée à la pédale progressive A parmi quatre fonctions :

- la vitesse de lecture,
- le tempo du hasard,
- la taille de sélection,
- le point de départ.

De cette façon, tandis que le volume général du logiciel reste toujours accessible via la pédale B, l'instrumentiste peut changer le paramètre associé à la pédale A.

Pour signaler visuellement le paramètre sélectionné, nous avons choisi de faire passer la couleur du curseur du gris au jaune (cf. figure 6.2 page suivante). Ce code couleur respecte ainsi la signalétique courante de *Max/MSP*, en copiant les changements de couleur

5. La machine seule ne peut pas répondre aux situations trop complexes, comme les situations de concerts amplifiés ; le couplage homme-machine reste ainsi la solution la plus judicieuse dans de nombreux cas, dont celui du mixage et de la diffusion, mais aussi celui de la programmation…

6. « Le Theremin, les ondes Martenot, la guitare électrique, les synthétiseurs analogiques, le "Lightning" de Don Buchla, voir les systèmes interactifs à base d'ondes cérébrales (Gordon Mumma) ou d'électricité corporelle (Atau Tanaka) ou le spectaculaire "Méta-Instrument" de Serge de Laubier, sont tous des réalisations des nouveaux luthiers cherchant à créer l'instrument "parfait", du point de vue de l'interprète. Les attributs de celui-ci sont essentiellement *vocaux* : continuité sonore des hauteurs, des dynamiques et du timbre, apesanteur matérielle et base technologique invisible. On recherche ainsi une sensibilité instrumentales idéales, favorisant une immédiateté et une entière efficacité de la réalisation de l'Idée musicale. » [Laliberté, 1999, p. 132].

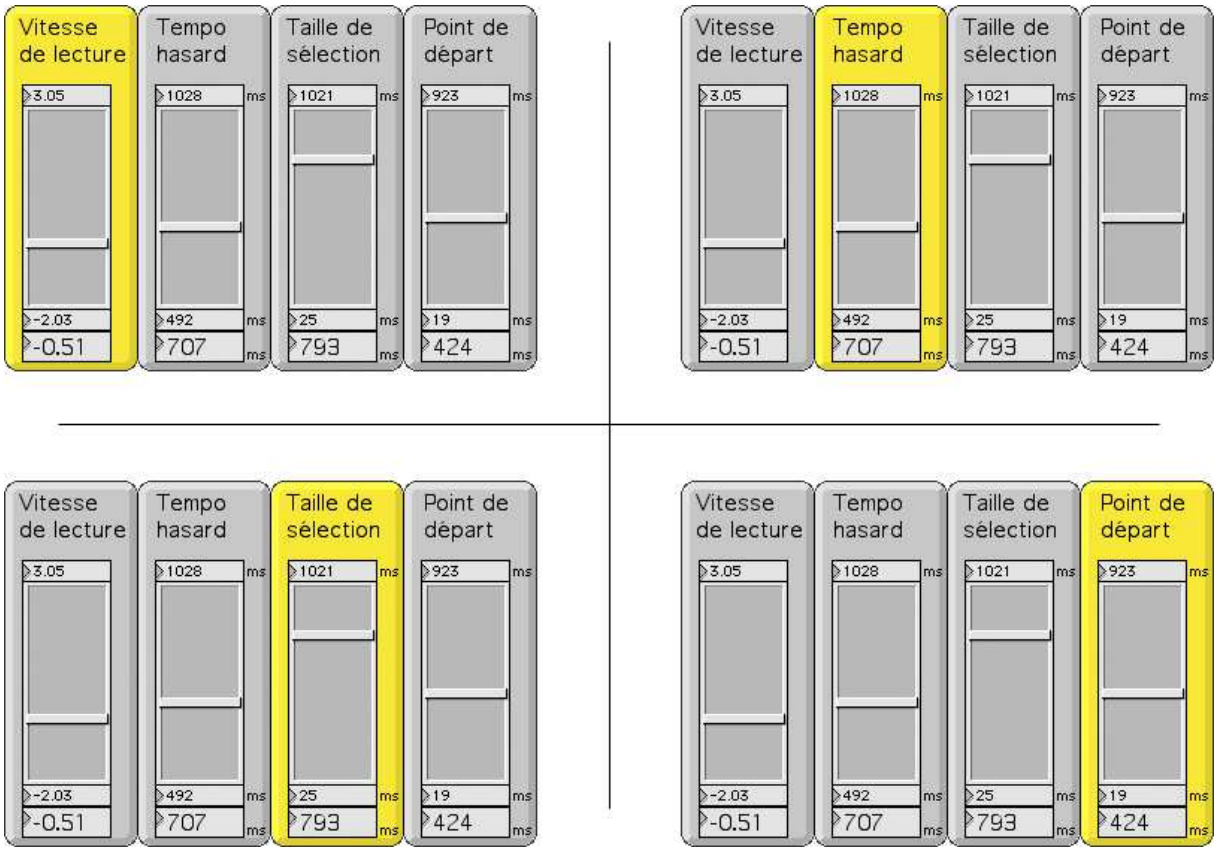


FIG. 6.2 – *Permutation fonctionnelle dans Mimi*

de l'objet `button` (qui visualise les messages `bang` en jaune par défaut). Cette convention graphique sera reprise dans la plupart de mes logiciels utilisant la permutation fonctionnelle.

6.2.3 Composabilité de l'interaction

Une relecture interactive...

En quoi consiste la relecture d'un *auto*-échantillonneur comme Mimi ? Par définition, l'échantillonneur ne peut proposer qu'une *relecture* de ce qui est enregistré, d'une façon éventuellement modifiée (traditionnellement par un algorithme de transposition plus ou moins élaboré et des filtres fréquentiels). Cette fonction de relecture est tout aussi première dans le cas d'un auto-échantillonneur, cependant le préfixe *auto* indique que l'instrumentiste a non seulement accès aux fonctions de relecture mais aussi la fonction d'enregistrement : c'est-à-dire qu'il peut enregistrer *lui-même* (avec le dispositif[7]) le son qu'il aura produit *lui-même* (avec l'instrument).

Au-delà de l'auto-échantillonnage, Mimi permet à l'instrumentiste de contrôler les paramètres de la relecture, soit principalement huit fonctions en plus de l'enregistrement.

7. Ici, le dispositif podophonique permet à un instrumentiste dont l'instrument laisse les pieds libres (cf. section 4.3.6 page 100) de déclencher de telles fonctions.

Ces possibilités de modification interactive du processus de relecture sont indiquées sur l'unique fenêtre de Mimi (cf. figure 6.1 page 120) par des objets graphiques représentant soit des boutons, soit des curseurs, ces deux types d'objets correspondant respectivement soit aux dix pédales interruptrices, soit aux deux pédales progressives du pédalier :

- la lecture (marche/arrêt),
- le bouclage (marche/arrêt),
- le hasard (marche/arrêt),
- la vitesse de lecture (progressive[8]),
- le tempo hasard (progressif),
- la taille de la sélection (progressive),
- le point de départ de la sélection (progressif),
- le changement de configuration (direct).

Il s'agit donc tout à la fois pour l'instrumentiste de *relire* ce qu'il a lui-même décidé *d'enregistrer* à un moment de sa performance avec son instrument, et de *modifier* interactivement les paramètres de la relecture proposés par le logiciel.

. . . et mouvante

Lors de la découverte de Mimi, les fonctions associées aux objets graphiques se comprennent en général facilement grâce aux noms qui sont relativement explicites, sauf la fonction « Hasard », qui mérite des explications.

En effet, cette fonction a été ajoutée pour prendre en charge *plusieurs* paramètres à la fois, car même si la permutation fonctionnelle permet à l'instrumentiste d'avoir accès à tous les paramètres progressifs, elle ne lui permet d'accéder qu'à *un seul* de ces paramètres à la fois (avec la pédale A). Ainsi, le bouton 94 nommé « Hasard » active ou désactive un processus de tirage aléatoire pour trois paramètres temporels à la fois :

- le tempo hasard,
- la taille de la sélection[9],
- le point de départ de la sélection.

Ce sont là tous les paramètres progressifs de Mimi associés à la pédale A, à l'exception de la vitesse de lecture qui n'a pas été jugée probante à l'écoute. Le tirage aléatoire se fait entre les deux bornes minimum et maximum pour ces trois paramètres, ces bornes étant placées à l'écran respectivement au-dessous et au-dessus des objets curseurs.

8. J'entends par « progressif » une opposition avec le comportement binaire de type « marche/arrêt ». Dans Mimi, ces paramètres « progressifs » évoluent de manière discrète de 0 à 127, donc de manière non continue, et c'est pourquoi j'emploie le mot « progressif » : pour désigner un comportement discret au niveau de sa réalité informatique mais qui peut cependant sembler continu au niveau de la perception. En effet, il n'est pas exclu dans le langage courant qu'un progrès puisse se faire par paliers successifs. . . Ces paramètres correspondent en outre aux pédales *progressives* du pédalier, à distinguer des pédales *interruptrices*.

9. La sélection peut être vue comme un segment, définissable de deux manières. On pourrait définir l'intervalle sélectionné comme un bipoint, avec un point de départ et un point d'arrivée, mais la définition qui utilise un point et une *longueur* (*i. e.* le point de départ et la taille de la sélection) nous a paru mieux adaptée pour la situation du direct et de l'auto-enregistrement, car la notion de longueur est plus intuitive.

Techniquement, le bouton 94 envoie le message `1` à l'objet `metro` qui déclenche à son tour le tirage aléatoire de trois objets `random` : celui de « Tempo hasard » détermine la durée avant le prochain tirage aléatoire, celui de « Taille de sélection » détermine la taille de la sélection à relire dans le tampon audio et celui du « Point de départ » détermine le point de départ de cette sélection. Évidemment, lorsque la fonction Hasard est active mais qu'un des trois paramètres est sélectionné pour la pédale A, un mécanisme d'inhibition automatique du tirage aléatoire a été programmé, de façon à ce que l'instrumentiste reprenne le contrôle de ce paramètre sans qu'il soit modifié par la fonction aléatoire, tandis que les autres continuent d'y être soumis.

Enfin, pour éviter la brutalité de ces changements aléatoires, les deux paramètres « Taille de sélection » et « Point de départ » ne sautent pas directement aux valeurs tirées aléatoirement mais s'y rendent progressivement avec l'objet `line`, pendant la durée donnée par « Tempo hasard », ce qui donne le caractère mouvant, glissé, aux variations de la sélection.

Une interaction à composer

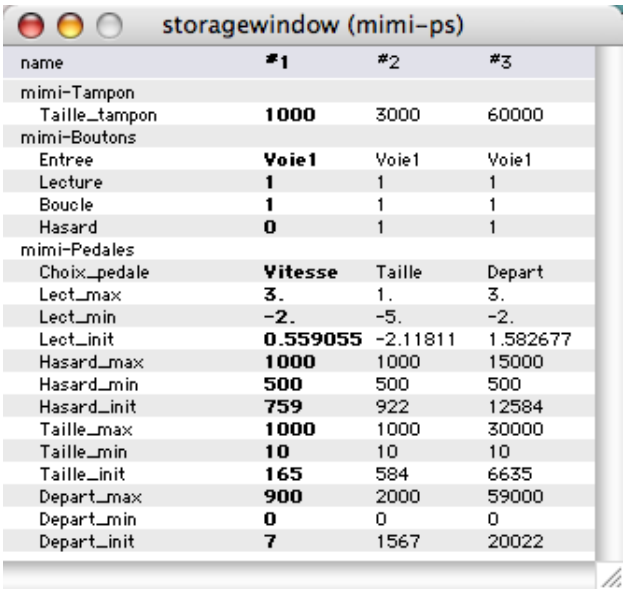
Programmer des dispositifs interactifs ressemble à un compromis avantageux à la fois du point de vue de la vivacité potentielle de l'œuvre lors de son interprétation, et du point de vue de l'écriture. Au-delà de la programmation, il s'agit d'ailleurs ici d'une écriture dans un sens élargi pour le compositeur, comprenant l'écriture de la partition pour l'instrumentiste et la spécification précise et enregistrable du paramétrage du dispositif interactif, comme le note Bruno Bossis :

> L'interaction remet en cause le processus même de la composition musicale. Traditionnellement, la musique est composée avant d'être interprétée sur scène. La phase de composition ne réside plus seulement dans l'écriture d'une partition, mais également dans la définition des caractéristiques du dispositif qui produira la musique, y compris les modes d'interaction entre les machines et les musiciens. [Bossis, 2005]

De ce point de vue, dans le cas précis des collaborations de ce doctorat, il faudrait ajouter une « méta-écriture » : celle des logiciels, pour lesquels les compositeurs ou les musiciens ont pleinement participé à l'établissement du cahier des charges associé et ont évalué régulièrement les versions successives tout au long du développement, apportant des directions décisives à la programmation, conformément à la confrontation musicale choisie sur le mode du sur-mesure (cf. section 4.3.1 page 91).

Pour en revenir à l'écriture élargie, Mimi offre de préprogrammer les modalités de l'interaction pour l'instrumentiste, à partir des dix-huit paramètres montrés en figure 6.3 page suivante : la taille du tampon audio, l'entrée audio, trois interrupteurs[10], le choix de la fonction associée à la pédale A, et enfin les quatre triplets de valeurs (la valeur initiale et les deux bornes minimum et maximum des curseurs) pour les paramètres progressifs.

10. L'interrupteur d'enregistrement a été exclu afin d'éviter tout écrasement involontaire du tampon audio.



name	#1	#2	#3
mimi-Tampon			
Taille_tampon	**1000**	3000	60000
mimi-Boutons			
Entree	**Voie1**	Voie1	Voie1
Lecture	**1**	1	1
Boucle	**1**	1	1
Hasard	**0**	1	1
mimi-Pedales			
Choix_pedale	**Vitesse**	Taille	Depart
Lect_max	**3.**	1.	3.
Lect_min	**-2.**	-5.	-2.
Lect_init	**0.559055**	-2.11811	1.582677
Hasard_max	**1000**	1000	15000
Hasard_min	**500**	500	500
Hasard_init	**759**	922	12584
Taille_max	**1000**	1000	30000
Taille_min	**10**	10	10
Taille_init	**165**	584	6635
Depart_max	**900**	2000	59000
Depart_min	**0**	0	0
Depart_init	**7**	1567	20022

FIG. 6.3 – *Les paramètres de Mimi*

Tout préréglage de cet ensemble de paramètres constitue une configuration *composée*, à l'intérieur de laquelle l'instrumentiste peut développer son jeu.

De fait, plusieurs de ces paramètres ne sont pas directement accessibles à l'instrumentiste, comme la taille du tampon et les bornes des curseurs, mais il peut les changer indirectement en changeant de configuration, de façon globale. Mimi permet d'utiliser jusqu'à 90 configurations (ou préréglages), ce qui correspond aux 9 premières banques du pédalier multiple. Tous ces réglages sont enregistrés sous forme d'un fichier XML[11] grâce au système des attributs de patch de *Max* (les *« patcher attributes »*[12]) :

Code 6.1 – *Fichier XML des préréglages de Mimi*

```
1  <?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
2
3  <pattrstorage name = "mimi-ps">
4    <slot number = "1">
5      <patcher name = "mimi-Tampon">
6        <pattr name = "Taille_tampon" value = "1000" />
7      </patcher>
8      <patcher name = "mimi-Boutons">
9        <pattr name = "Entree" value = "Voie1" />
10       <pattr name = "Lecture" value = "1" />
11       <pattr name = "Boucle" value = "1" />
12       <pattr name = "Hasard" value = "0" />
13     </patcher>
14     <patcher name = "mimi-Pedales">
15       <pattr name = "Choix_pedale" value = "Vitesse" />
16       <pattr name = "Lect_max" value = "3." />
```

11. XML (*eXtensible Markup Language*), un langage à balise issu de SGML (*Standard Generalized Markup Language*), possède de nombreux avantages, en particulier pour l'archivage et l'échange de contenus : il permet de structurer les données (il est ordonné et hiérachique), il est normalisé (par le Web Consortium en 1996), interopérable, public (et libre de droits), extensible, modulaire, lisible (textuel), indépendant des plates-formes (il utilise Unicode) et bénéficie aujourd'hui de nombreux outils dédiés et d'une grande compatibilité avec les logiciels courants.

12. Mimi utilise les objets `pattrstorage` et `autopattr`, comme la plupart de mes logiciels qui enregistrent des préréglages, en conjonction avec mon sous-patch `[kb-Storage-mng]` pour l'interface d'édition, d'enregistrement et de rappel des préréglages.

```
17        <pattr name = "Lect_min" value = "-2." />
18        <pattr name = "Lect_init" value = "0.559055" />
19        <pattr name = "Hasard_max" value = "1000" />
20        <pattr name = "Hasard_min" value = "500" />
21        <pattr name = "Hasard_init" value = "759" />
22        <pattr name = "Taille_max" value = "1000" />
23        <pattr name = "Taille_min" value = "10" />
24        <pattr name = "Taille_init" value = "165" />
25        <pattr name = "Depart_max" value = "900" />
26        <pattr name = "Depart_min" value = "0" />
27        <pattr name = "Depart_init" value = "7" />
28      </patcher>
29    </slot>
30    <slot number = "2">
31      <patcher name = "mimi-Tampon">
32        <pattr name = "Taille_tampon" value = "3000" />
33      </patcher>
34      <patcher name = "mimi-Boutons">
35        <pattr name = "Entree" value = "Voie1" />
36        <pattr name = "Lecture" value = "1" />
37        <pattr name = "Boucle" value = "1" />
38        <pattr name = "Hasard" value = "1" />
39      </patcher>
40      <patcher name = "mimi-Pedales">
41        <pattr name = "Choix_pedale" value = "Taille" />
42        <pattr name = "Lect_max" value = "1." />
43        <pattr name = "Lect_min" value = "-5." />
44        <pattr name = "Lect_init" value = "-2.11811" />
45        <pattr name = "Hasard_max" value = "1000" />
46        <pattr name = "Hasard_min" value = "500" />
47        <pattr name = "Hasard_init" value = "922" />
48        <pattr name = "Taille_max" value = "1000" />
49        <pattr name = "Taille_min" value = "10" />
50        <pattr name = "Taille_init" value = "584" />
51        <pattr name = "Depart_max" value = "2000" />
52        <pattr name = "Depart_min" value = "0" />
53        <pattr name = "Depart_init" value = "1567" />
54      </patcher>
55    </slot>
56  </pattrstorage>
```

On peut constater avec ce fichier qu'un travail de spécification des paramètres doit fatalement avoir lieu au préalable pour pouvoir permettre, ultérieurement, l'interaction. Sans ces préréglages, il serait impossible d'utiliser les curseurs, puisqu'ils nécessitent des bornes définies, ni de changer d'un coup toute une configuration : les bornes des curseurs doivent être choisies et enregistrées, ainsi que les valeurs initiales de chaque paramètre (tant pour les curseurs que pour les interrupteurs).

Ainsi, pratiquement et formellement, l'interaction ne peut pas avoir lieu sans que la correspondance entre les instruments de contrôle et les processus du logiciel n'ait été précisément déterminée. Si « composer c'est choisir » comme me confiait un jour Mario, alors ces choix nécessaires du paramétrage, qui permettent finalement l'interaction instrumentale et sonore, appartiennent en toute légitimité au travail de composition musicale, au titre d'une « composition de l'interaction ».

6.2.4 Une poétique de l'instant présent

En guise de conclusion, on peut rapprocher l'auto-échantillonnage couplé à un instrument acoustique d'une certaine poétique de l'instant présent. Ce couplage particulier correspond effectivement à quelque chose pour Stéphanie, au sein d'une pensée musicale imprégnée d'une pratique soutenue de l'improvisation : dans *Le plaidoyer pour l'improvi-*

sation dans l'apprentissage instrumental, Volker Biesenbender affirme que « la musique, c'est le pur présent »[13] [Biesenbender, 1992, p. 92].

De fait, si la reproductibilité *stricto sensu* est l'apanage du numérique, l'*in*-reproductibilité caractérise l'analogique en corollaire et précipite cet analogique dans le présent. La prémisse acoustique évoquée précédemment (cf. section 4.3.4 page 96) prend ici toute sa force : l'instrument acoustique, joué en direct et par un être humain, se porte garant de l'éphémère, de l'unicité du concert, du moment partagé, à travers l'inreproductibilité intrinsèque de ce qu'il promet de produire. En quelque sorte, l'auto-échantillonnage – numérique par nature – vient capturer un instant du présent tel que le garantit l'instrument – acoustique pour l'alto.

6.3 Rose amère : présentation et mode d'emploi

6.3.1 Présentation générale

Un auto-échantilloneur interactif confirmé

Dans son principe, Rose amère est un auto-échantillonneur interactif pour pédalier Midi, comme Mimi, orienté vers l'improvisation. Néanmoins, Rose amère est beaucoup plus complexe : ce logiciel comporte quatre voies paramétrables : les boucles, les impacts (enveloppes percussives), le délai, et les effets (transposition et distorsion).

La conception et le développement se sont déroulés en collaboration avec Stéphanie Réthoré, dans un aller-retour fréquent entre la programmation et les essayages. De plus, Rose amère étant chronologiquement le dernier logiciel développé pour ce doctorat, il hérite directement de plusieurs modules de Mimi, mais aussi de FeedItBack et de Iviv, développés avec Santiago Quintáns (cf. chapitre 7 page 147), et de Plugiscope et de Ifso, développés avec Iván Solano (cf. chapitre 9 page 187).

Globalement, le résultat sonore relève à la fois de l'augmentation et de l'anamorphose. L'augmentation provient de l'amplification mais aussi de la polyphonie qui démultiplie véritablement l'instrument ; par exemple une boucle s'entend clairement comme un discours musical parallèle à celui de l'instrument acoustique, donnant nettement la sensation d'un dédoublement. L'anamorphose provient quant à elle des opérations audionumériques programmées dans chacun des modules sonores, par exemple les modifications de vitesse de relecture ou d'enveloppe dynamique ; certaines déformations sont trop importantes pour reconnaître la source, ce qui accentue l'effet de démultiplication.

Le logiciel « Rose amère », joué par Stéphanie sous la forme d'une extension logicielle de l'alto au sein du dispositif podophonique interactif, a fait ses preuves lors de plusieurs occasions :

13. Volker Biesenbender fait alors immédiatement le lien entre *présent* et *présence* : « Si nous considérons l'improvisation musicale comme une possibilité de relation artistique créatrice avec l'ici et maintenant, nous devons mobiliser toutes les facultés qui contribuent à établir en nous une présence vigilante. [...] Le présent n'est pas un point, c'est un flux. » [Biesenbender, 1992, p. 92]

FIG. 6.4 – *Capture de l'interface visuelle de Rose amère*

- lors du Prix d'improvisation générative du CNSMDP, en trio[14] alto augmenté / contrebasse / piano, le 13 juin 2008 ;
- lors de l'émission « À l'improviste » d'Anne Montaron au studio 106 de la Maison de la Radio, en trio puis en tutti[15], enregistrée le 21 juin 2008 ;
- pour un CD « Rose amère » constitué d'improvisations solo enregistrées en mai 2008, réalisé en juillet 2008 ;
- aux Journées de la profession organisées au CNSMDP, en trio, le 11 septembre 2008 ;
- au vernissage de l'exposition de la photographe Geneviève Hofman au Scriptorial d'Avranches « Le Serpent, le Dragon et les Ailes », en solo, le 19 septembre 2008.

Les sections suivantes se présentent comme un mode d'emploi de Rose amère, à la fois pour briser la monotonie qui pourrait apparaître dans un texte qui détaille neuf logiciels, et pour inviter à utiliser ce logiciel, assurés de la stabilité qu'il a acquise au cours de ces expériences réussies.

Quatre modules sonores

Les quatre modules sonores de Rose amère sont superposables, constituant une polyphonie de quatre voies pour le logiciel. Les deux premiers modules sont basés sur des

14. Un trio stable s'est constitué avec Charlotte Testu à la contrebasse et Laurent Durupt au piano, qui restent tous deux sans informatique dans ce trio, mais amplifiés.

15. Alto augmenté, clarinette, violoncelle, batterie, deux contrebasses, piano, saxophone.

relectures déformantes à partir de l'auto-enregistrement, alors que les deux derniers relèvent du traitement du signal audionumérique entrant, en temps réel :

1. le module « Boucle » (en bleu) permet de déclencher trois processus macro-temporels au choix, variés sur la vitesse de lecture ;
2. le module « Impacts » (en rose, fuchsia et violet) génère trois enveloppes dynamiques de type percussif, à partir du son enregistré en direct, avec des paramètres semi-aléatoires bornés ;
3. le module « Délai » (en vert) retarde le son entrant selon deux paramètres contrôlables : la durée du délai et le taux de réinjection ;
4. le module « Effets » (en orange) applique deux effets en série : la transposition du son, et la distorsion numérique.

Chacun de ces quatre modules est relié à la pédale d'expression gauche (A), en respect du principe de permutation fonctionnelle[16], ce qui permet au musicien de réaliser lui-même le mixage des modules, de paramétrer chaque fonction et de doser l'équilibre avec son instrument. Quant à la pédale d'expression droite (B), elle permet simplement de gérer le volume global sortant du logiciel, en respect de sa rigidification fonctionnelle[17], ce qui revient à gérer la dynamique générale de l'instrument augmenté.

6.3.2 L'organisation de l'interface graphique

Présentation synthétique : quatre étages

Étages	Représentations	Fonctions
1	8 boutons textuels et 3 boîtes numériques	Configuration générale des entrées et informations numériques mineures
2	2 formes d'ondes et 5 barres de progression	Visualisation de l'enregistrement et de la lecture
3	12 boutons et 9 curseurs	Visualisation des fonctions performables à partir du pédalier
4	4 curseurs, 4 VU-mètres, 4 historiques déroulants et 4 boutons textuels	Visualisation de l'activité sonore de chaque voie

TAB. 6.1 – *Organisation de l'interface graphique de Rose amère*

L'interface graphique de Rose amère s'organise[18] en trois étages bien visibles, pour le jeu en direct, surmontés d'un étage plus discret, pour la configuration des entrées. Ces quatre étages sont synthétisés dans le tableau 6.1.

16. Cf. section 6.2.2 page 123.

17. Cf. section 6.2.2 page 122.

18. Traditionnellement, les indications graphiques sont données du haut vers le bas en informatique, car l'origine du repère d'une fenêtre se trouve en haut, à gauche, et non en bas, à gauche.

Premier étage : configuration des entrées

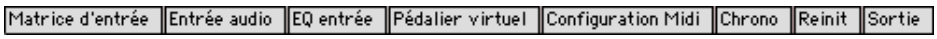


FIG. 6.5 – *Le premier étage visuel de Rose amère*

Le premier étage reste le plus discret visuellement, parce qu'il est dédié essentiellement à la configuration et non à la performance, avec 8 petits boutons textuels qui ouvrent chacun une fenêtre de configuration en rapport avec les entrées/sorties ou MIDI :

- « Matrice d'entrée », qui permet de configurer le nombre d'entrées externes et de les prémixer ;
- « Entrée audio », qui permet de sélectionner l'entrée effective parmi Silence, Carte son et Fichier son, et de régler son niveau ;
- « EQ entrée », qui ouvre un éditeur graphique de 5 filtres fréquentiels ;
- « Pédalier virtuel », qui affiche le pédalier virtuel, soit pour la visualisation, soit pour l'interaction directe (à la souris ou au clavier alphanumérique[19], cf. tableau 6.2) ;
- « Configuration Midi », qui permet de rappeler et d'enregistrer ses propres réglages de correspondance MIDI avec les trois entrées logiques Déclencheurs, Pédale A et Pédale B, ainsi que de visualiser les différents messages MIDI entrants.
- « Chrono », qui permet de visualiser l'écoulement du temps depuis le premier clic[20] dans sa fenêtre ;
- « Reinit », qui réinitialise l'ensemble du patch `[Rose_amere]` ;
- « Sorties », qui ouvre le sous-patch de routage des sorties audio.

a	z	e	r	t	y	⟼	1	2	3	4	5	UP
q	s	d	f	g	h		6	7	8	9	10	DN

TAB. 6.2 – *Les raccourcis-clavier du pédalier virtuel*

Deuxième étage : visualisation de l'enregistrement et de la lecture

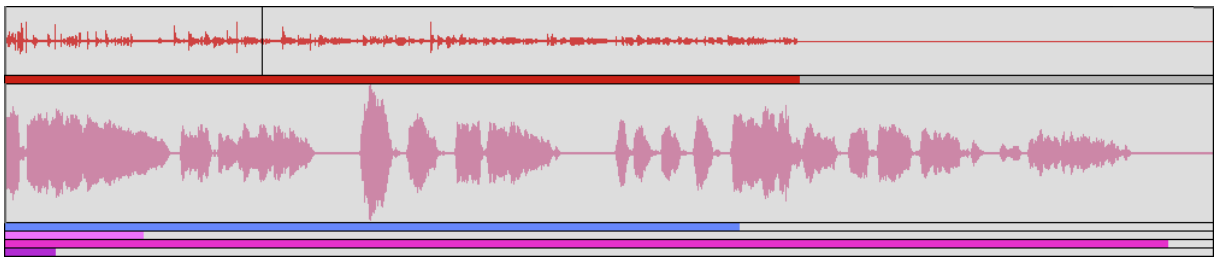


FIG. 6.6 – *Le deuxième étage visuel de Rose amère*

19. Les raccourcis-clavier sont prévus par défaut sur un clavier azerty et pour la main gauche, la droite pouvant ainsi gérer les curseurs avec la souris.

20. Un double clic réinitialise le chronomètre.

Le deuxième étage montre clairement deux formes d'ondes occupant toute la largeur, sous lesquelles se trouvent des barres de progression :

– la forme d'onde supérieure, en rouge et plus petite, représente le tampon d'enregistrement ;
– la forme d'onde inférieure, en rose et plus grande, représente le tampon de lecture.

En cours d'enregistrement, la barre de progression correspondante, en rouge, avance de gauche à droite. Le principe reste le même pour la lecture, sauf qu'il y a quatre barres de progression différentes, soit de haut en bas :

– le pointeur de lecture du module « Boucles », en bleu ;
– le pointeur de lecture de l'impact « Ti » (le plus aigu), en rose ;
– le pointeur de lecture de l'impact « Ka » (médium), en fuchsia ;
– le pointeur de lecture de l'impact « Pou » (le plus grave), en violet.

Troisième étage : visualisation des fonctions performables

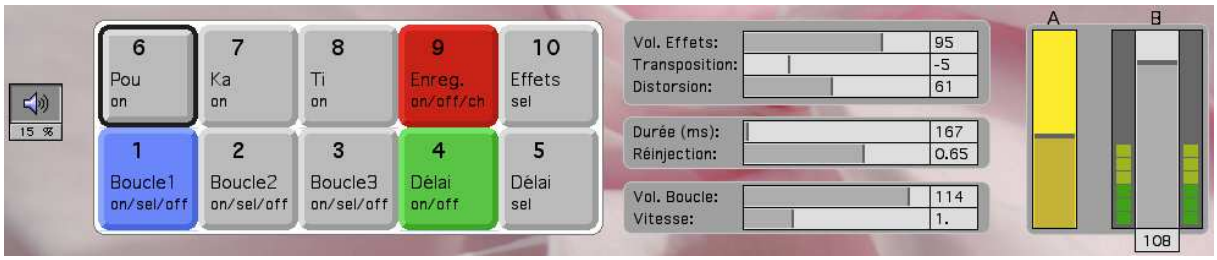


FIG. 6.7 – *Le troisième étage visuel de Rose amère*

Le troisième étage s'organise horizontalement en quatre parties, soit de gauche à droite :

– 2 boutons pour les statuts audio (marche/arrêt et *DSP Status*) ;
– 10 boutons correspondant aux 10 pédales interruptrices ;
– 7 curseurs horizontaux pour le paramétrage interactif ;
– 2 curseurs verticaux correspondant aux 2 pédales progressives A et B.

Cette disposition horizontale reprend donc la métaphore visuelle du pédalier, en intercalant les 7 curseurs de paramétrage entre les 10 boutons et les 2 curseurs verticaux. Cependant, les pédales UP et DOWN n'apparaissent pas ici, car Rose amère n'utilise que la première banque, afin d'optimiser la rapidité de l'utilisation du dispositif en supprimant la navigation dans les banques.

Cette restriction d'une banque unique implique en contrepartie une certaine complexité pour l'accès aux différentes fonctions, avec une diversité de mécanismes comme des doubles clics et des permutations circulaires, ces comportements différant d'un bouton à l'autre.

Quatrième étage : visualisation de l'activité polyphonique

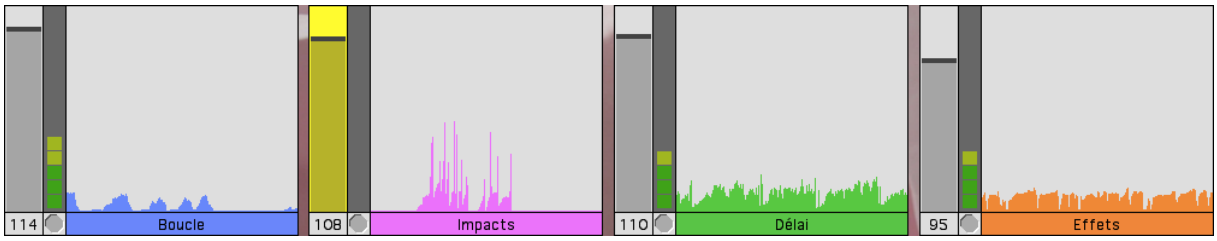


FIG. 6.8 – *Le quatrième étage visuel de Rose amère*

Le quatrième étage s'organise horizontalement en quatre modules identiques, regroupant chacun :

- un curseur vertical pour le volume (de 0 à 127), assorti d'un VU-mètre ;
- un historique déroulant du signal, pour voir une trace de 10 secondes[21] ;
- un bouton textuel qui ouvre la fenêtre de préréglage du module idoine[22].

Ainsi, cet étage se révèle tout à fait précieux pour le mixage entre les quatre voies en situation de direct.

6.3.3 L'enregistrement

L'enregistrement demande de maîtriser différents types de déclenchement avec la pédale 9, selon les transitions d'états représentées figure 6.9 :

- le simple clic (« 1c ») qui déclenche l'enregistrement (états « enregistrement » e_0 et e_1) ;
- le simple clic (« 1c ») qui arrête l'enregistrement (états « prêt » p_0 et p_1), le tampon d'enregistrement devenant prêt à charger ou à réenregistrer ;
- le double clic (« 2c ») qui stoppe d'abord les éventuelles lectures en cours (boucle et impacts) et qui charge ensuite l'enregistrement dans le tampon de lecture (état « chargé » c_1).

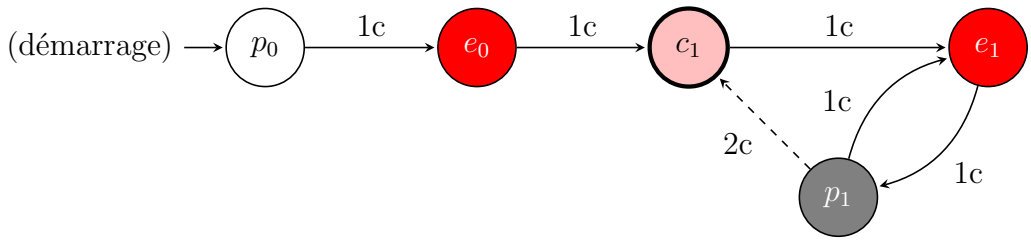


FIG. 6.9 – *Diagramme performatif de l'enregistrement dans Rose amère (bouton 9)*

21. La trace dure exactement 195 pixels × 50 millisecondes = 9 750 millisecondes.

22. Ces fenêtres ne sont pas montrées ici, par souci de concision, à part la fenêtre du module « Effets », figure 6.17 page 142.

Ce mécanisme qui utilise deux tampons différents – un pour l'enregistrement et un pour la lecture – permet de ne pas interrompre les boucles en cours de lecture pendant les enregistrements, de façon à rendre l'opération d'enregistrement plus discrète à l'écoute, sans rupture. Ainsi, l'arrêt de l'enregistrement (un simple clic) est dissocié du chargement effectif dans le tampon de lecture (un double clic) ; sauf au départ, où l'arrêt de l'enregistrement (un simple clic) suffit à charger aussitôt le tampon de lecture (pas de double clic), puisque ce dernier est nécessairement vide. Cette dissociation permet en outre de réenregistrer sans avoir chargé (retour p_1 vers e_1) ; ceci permet donc de charger uniquement ce que l'on souhaite vraiment, et seulement au moment voulu (chargement p_1 vers c_1).

Par ailleurs, la fenêtre temporelle de détection du type de clic (simple ou double) a été fixée à 500 ms pour tous les boutons qui font appel au double clic, les autres boutons ne subissant pas ce retard. Cette durée nous a semblé raisonnable pour réaliser un double clic avec les pieds ; en revanche, les simples clics d'un tel bouton subissent aussi un retard équivalent de 500 ms, à cause de la fenêtre temporelle de détection.

Enfin, la durée maximum d'un enregistrement à été fixée à 1 mn : si cette durée est atteinte, alors l'enregistrement s'arrête automatiquement, sinon, si l'enregistrement est arrêté avant cette durée, alors le tampon est redimensionné automatiquement jusqu'à ce point d'arrêt (mais le prochain enregistrement se verra à nouveau allouer 1 mn).

6.3.4 Les trois boucles

Présentation générale des trois boucles

Une seule des trois boucles peut être lue à la fois, car elle partagent le même moteur de lecture – techniquement le même objet `groove~` (partiellement repris du logiciel Iviv[23]). Cependant, elles procèdent chacune d'un fonctionnement différent :

- la boucle 1 déclenche une lecture lente et grave ;
- la boucle 2 déclenche une lecture aléatoire « agitée » dans les médiums/aigus ;
- la boucle 3 déclenche une lecture dont la vitesse est contrôlable avec la pédale A.

storagewindow (loop-ps)

name	#1	#2	#3	#4	#5	#6
loop-bp-VitesseLecture						
Vitesse_lecture_min	0.15	1.	5.	-3.	-7.	*-0.98*
Vitesse_lecture_max	0.16	4.	8.	-0.91	-2.	*-1.02*
Vitesse_lecture_init	0.156214	2.978718	5.383405	-0.927099	-2.379416	*-1.02*
loop-bp-DureeOscillations						
Duree_oscillation_min	10.	0.36	1.	0.36	0.36	*0.8*
Duree_oscillation_max	40.	2.	2.	1.	1.47	*1.5*
Duree_oscillation_init	29.266001	1.086	1.336	0.36	1.165	*0.8*

FIG. 6.10 – *Préréglages des boucles de Rose amère*

La figure 6.10 présente le fichier *loop-ps* qui contient les préréglages retenus pour la boucle 1 (préréglage n° 1) et la boucle 2 (tous les autres préréglages, ici n^{os} 2 à 6) ; la

23. Cf. le module « Loop », section 7.3.8 page 164.

boucle 3 n'utilise pas ces préréglages puisque la vitesse de lecture est confiée à la pédale A (dans un intervalle compris entre 0.25 et 3).

Un mécanisme supplémentaire intervient pour l'interaction avec les boucles : la « A-sélection », qui correspond à la permutation fonctionnelle de la pédale A étudiée précédemment avec Mimi[24]. Ce mécanisme, qui sera repris pour le délai et les effets, permet d'affecter à la pédale A le premier paramètre du module *sélectionné* (le volume le plus souvent). Afin de visualiser cette sélection pédestre à l'écran, d'une part un épais contour noir est dessiné sur le dernier bouton sélectionné (ainsi que sur les diagrammes performatifs[25] des figures 6.11 et 6.12), et d'autre part le curseur concerné et celui de la pédale A s'allument en jaune à l'écran.

La boucle 1

La boucle 1 applique le préréglage n° 1 (cf. figure 6.10 page précédente) qui correspond à une vitesse de lecture comprise entre 0.15 et 0.16, soit une lecture ralentie plus de six fois[26] qui sonne plus grave de deux octaves et une sixte mineure ou majeure[27], modulée par un oscillateur à très basse fréquence, dont la période varie aléatoirement entre 10 s et 40 s[28].

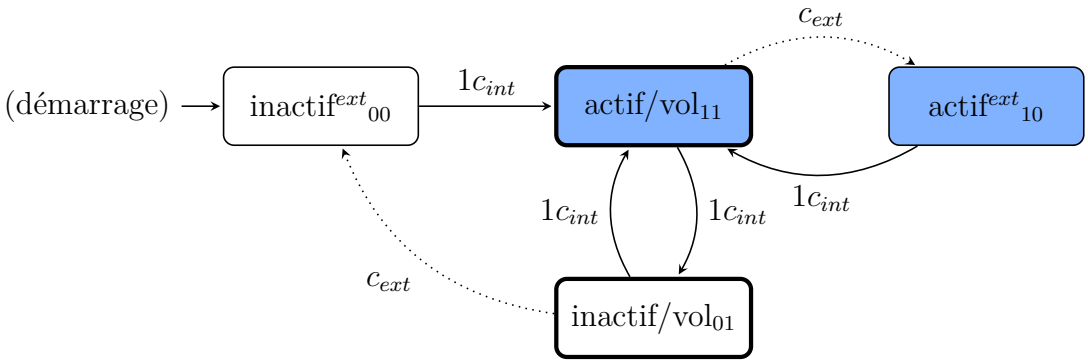


FIG. 6.11 – *Diagramme performatif des boucles 1 et 2 dans Rose amère (boutons 1 et 2)*

À cause du mécanisme de A-sélection, il faut distinguer 4 états possibles[29] et 2 types de clics pour la boucle 1 (cf. figure 6.11) :

– le clic interne « $1c_{int}$ », correspondant à un appui sur la pédale 1, qui permet alternativement d'activer et de désactiver la lecture ;

24. Cf. section 6.2.2 page 123.

25. Ces diagrammes sont dits « performatifs » parce qu'ils relèvent de la performance, conformément au vocabulaire proposé en section 5.4 page 109.

26. $1/0,15 = 6,25$ et $1/0,16 \approx 6,67$.

27. Une vitesse de lecture de 0,15 correspond environ à 33 demi-tons ($2^{\frac{33}{12}} \approx 0,15$) et 0,16 à 32 demi-tons ($2^{\frac{32}{12}} \approx 0,16$) ; soit à une transposition comprise entre deux octaves inférieures et une sixte mineure (32) ou une sixte majeure (33).

28. Soit une fréquence comprise entre 0,1 Hz et 0,025 Hz.

29.

	processus inactif	processus actif
non A-sélection	00	10
A-sélection	01	11

– le clic externe « c_{ext} », correspondant à un appui sur toute autre pédale que la pédale 1, qui n'interrompt pas nécessairement la lecture.

Ainsi, la lecture de la boucle 1 n'est pas non plus interrompue quand on « arrive dessus » (*i. e.* dans le cas où la boucle est active sans être sélectionnée, et qu'on effectue un clic interne) : dans ce cas, ce clic se contente de A-sélectionner la boucle 1, de façon à pouvoir ajuster son volume sans rupture même en venant d'ailleurs. Autrement dit : un *premier* clic interne (*i. e.* depuis une A-sélection externe) n'interrompt jamais la fonction, c'est un clic « d'entrée » ; il faut donc généralement appuyer deux fois de suite pour pouvoir désactiver une fonction quand on vient d'une autre pédale. En outre, un premier clic interne implique toujours une A-sélection interne, que la fonction soit active ou non (sauf pour l'enregistrement, soit le bouton 9).

La boucle 2

La boucle 2 applique un préréglage tiré au hasard à partir du deuxième préréglage dans le fichier idoine (donc ici du n° 2 au n° 6, cf. figure 6.10 page 135). Chaque tirage aléatoire est renouvelé lorsqu'un cycle de l'oscillateur basse fréquence se termine ; soit, avec ce fichier de réglage, dans un laps de temps compris entre 0,36 s et 2 s, donc assez fréquemment, ce qui donne son côté « agité ». De plus, les vitesses de lectures sont beaucoup plus variées, éventuellement à l'envers et beaucoup plus aiguës : jusqu'à 8 fois plus vite (préréglage n° 3) et à peine moins que la vitesse normale pour le minimum ($-0,91$ pour le préréglage n° 4).

En dehors de ce tirage aléatoire récurrent, le fonctionnement de la boucle 2 reste identique à celui de la boucle 1 (cf. figure 6.11 page ci-contre).

La boucle 3

La boucle 3 permet à l'instrumentiste de piloter lui-même la vitesse de lecture à partir de la pédale A, en plus du volume, sur un intervalle $[0.25, 3.]$ fixé. Elle se distingue des deux premières boucles par son absence de préréglage dans le fichier et par la permutation fonctionnelle de la pédale A sur deux paramètres (une « 2A-permutation ») : le volume et la vitesse de lecture.

Le diagramme performatif de la boucle 3 comporte alors 6 six états différents[30] et 3 types de clics :

30. Deux cas ne sont pas autorisés parmi les 8 cas théoriques (2^3) formulables à partir des trois conditions binaires – la lecture, la A-sélection du volume et la A-sélection de la vitesse. De fait, puisque la sélection du volume et de la vitesse procède par permutation circulaire (permutation fonctionnelle de la pédale A), ces deux paramètres s'excluent mutuellement, éliminant les deux cas « 011 » et « 111 » du tableau suivant :

lecture	Asel-vol	Asel-vit	
0	0	0	inactif
0	0	1	inactif/vit
0	1	0	inactif/vol
0	1	1	(n'existe pas)
1	0	0	actif
1	0	1	actif/vit
1	1	0	actif/vol
1	1	1	(n'existe pas)

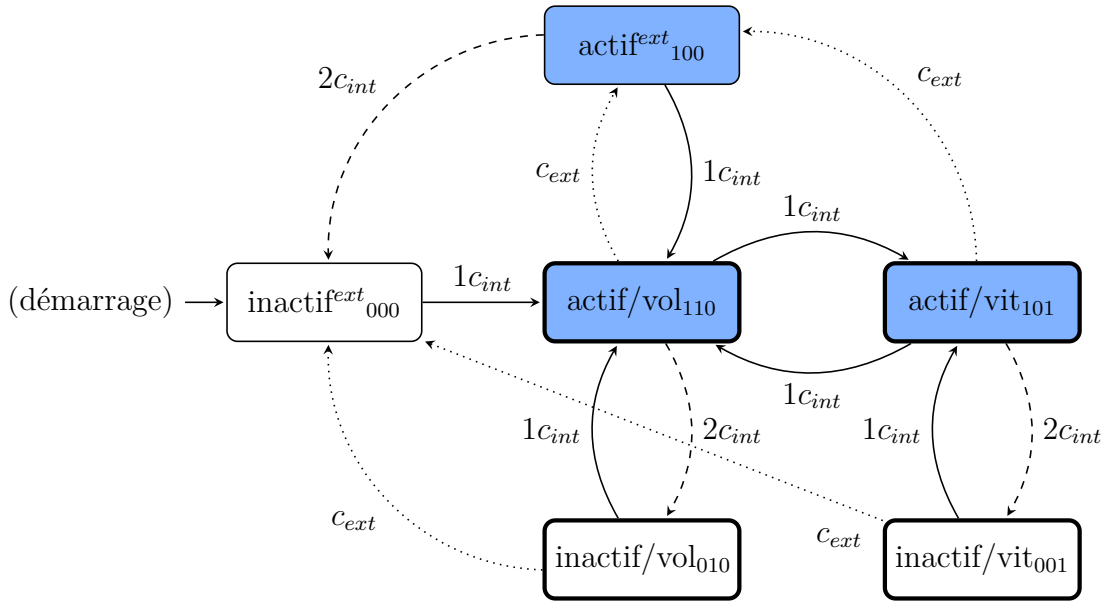


FIG. 6.12 – *Diagramme performatif de la boucle 3 dans Rose amère (bouton 3)*

- le clic interne « $1c_{int}$ », correspondant à un appui sur la pédale 3, qui permet cette fois-ci de permuter alternativement la A-sélection entre le volume et la vitesse de lecture, soit d'effectuer une A-permutation interne ;
- le double clic interne « $2c_{int}$ », correspondant à un double appui sur la pédale 3 (sous 500 ms), qui permet de désactiver la lecture ;
- le clic externe « c_{ext} », correspondant à un appui sur toute autre pédale que la pédale 3, qui permet de passer sur une autre fonction (externe par rapport à celle-ci) mais qui n'interrompt pas la lecture si elle est en cours.

Par rapport aux boucles 1 et 2, il faut donc apprendre un nouveau comportement du simple clic interne – la A-permutation –, anticiper le retard de détection de 500 ms, et intégrer le double clic pour stopper la lecture. De plus, visuellement, il y a deux curseurs à guetter lors des paramétrages en direct : « Vol. boucle » (de 0 à 127) et « Vitesse » (de 0.25 à 3.), en bas de la zone des sept curseurs horizontaux.

6.3.5 Les trois impacts

Présentation des impacts

La création des « impacts » répond à une certaine frustration due à l'absence de sons percussifs et potentiellement puissants à l'alto – le volume des *pizzicati* naturels restant très modéré –, et réalise en quelque sorte ce que Martin Laliberté appelle « la floraison de l'aspect *percussif* d'un instrument en apparence *vocal* » (cf. section 6.1.2 page 119).

Techniquement, les impacts appliquent une enveloppe dynamique de type ADSR[31]. Même si Curtis Roads notait déjà en 1998 l'aspect anachronique de ces enveloppes[32],

31. *« Attack, decay, sustain, release »*, soit « attaque, chute, maintien, extinction » en français.

32. « [...] pour la spécification d'une enveloppe musicale, une limite de quatre étapes est anachronique.

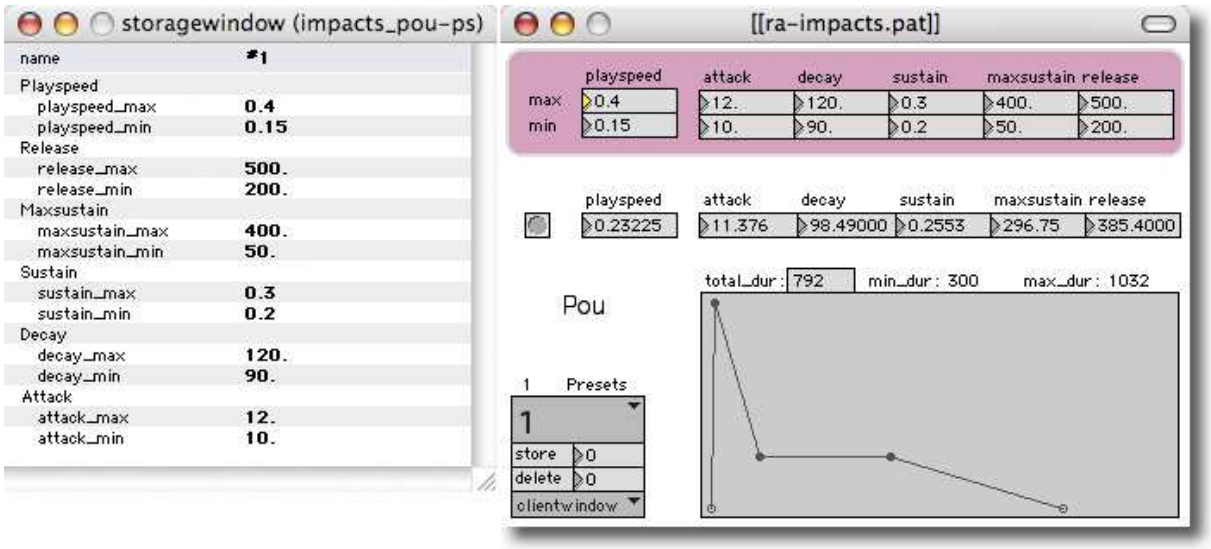


FIG. 6.13 – *Édition des impacts dans Rose amère*

elles peuvent trouver une application pertinente dans un processus aléatoire sur de l'enregistrement en direct. En effet, il ne s'agit pas de synthèse : le son provient du tampon de lecture, donc préenregistré en direct, et possède déjà sa propre richesse sonore. D'abord, ce son est renormalisé au maximum pour chaque impact ; ensuite, une vitesse de lecture est tirée aléatoirement (entre les bornes minimum et maximum spécifiées dans le fichier de préréglage) et déclenche la lecture ; enfin, une enveloppe ADSR tirée aléatoirement de la même façon module l'amplitude au cours du temps.

Fonctionnement des impacts

Ainsi, chaque type d'impact – Pou, Ka et Ti – possède son propre fichier de préréglage (dont seul le premier préréglage est pris en compte par le logiciel, cf. figure 6.13 pour Pou et figure 6.14 page suivante pour Ka et Ti), pour un résultat globalement différent entre les types d'impacts qui deviennent souvent identifiables :

- Pou sonne plutôt grave et un peu mou ;
- Ka sonne plutôt médium et plus franc, avec une lecture inverse ;
- Ti sonne plutôt aigu et parfois fluet.

Au niveau du jeu musical, deux déclenchements successifs d'un même impact donnent un résultat différent pour le deuxième impact : si le deuxième déclenchement a lieu après la fin de l'enveloppe, alors le deuxième impact ressemblera au premier impact car le pointeur de lecture revient à zéro dans ce cas ; sinon, si le deuxième déclenchement a lieu avant la fin de l'enveloppe, alors le deuxième impact sonnera différemment du premier impact car le pointeur de lecture continue sa progression dans ce cas (de façon circulaire si la fin du tampon est atteinte). Ainsi, les déclenchements rapprochés donnent une plus grande variété spectrale.

Le façonnage d'une amplitude est une opération délicate, et c'est ainsi que des éditeurs d'enveloppe plus précis permettent au musicien de tracer des courbes arbitraires. » [Roads, 1998, p. 64]

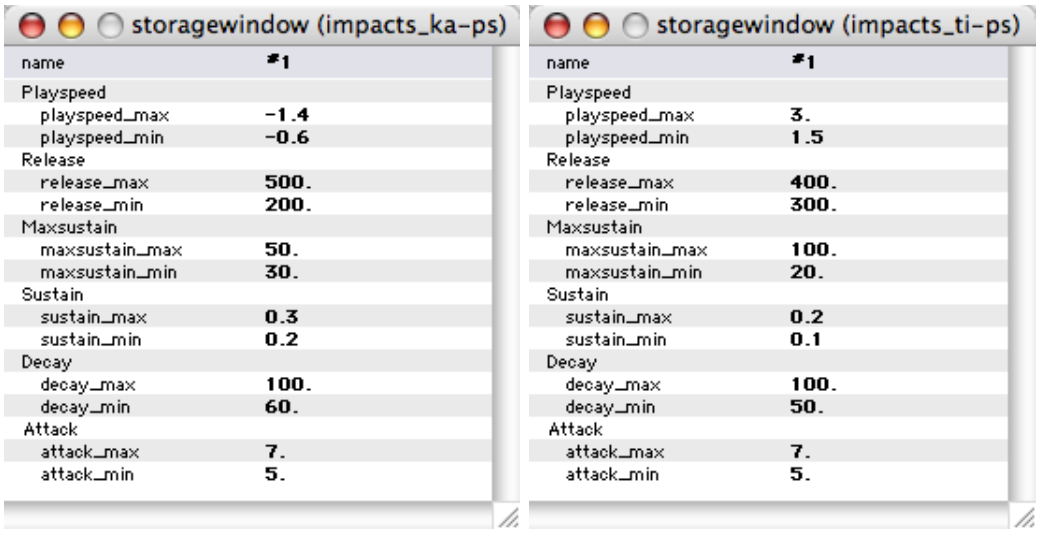


FIG. 6.14 – *Préréglages des impacts Ka et Ti dans Rose amère*

Enfin, il y a une astuce pour l'édition des 12 valeurs à saisir pour chaque impact : la touche « tabulation » permet de passer d'une boîte numérique à l'autre sans devoir passer par la souris.

6.3.6 Le délai

Présentation du délai

À la différence des boucles et des impacts, le délai n'utilise pas le tampon de lecture, car il s'applique directement sur l'entrée audio, par exemple sur la captation de l'alto en direct. En revanche, le délai comporte ici trois paramètres – le volume, la durée et la réinjection –, au lieu de deux pour la boucle 3 et un seul pour les boucles 1 et 2.

Paramètre	Minimum	Maximum	Évolution
Volume	0	127	exponentielle
Durée (ms)	40.	1000.	exponentielle
Réinjection	0.	1.	linéaire

TAB. 6.3 – *Les trois paramètres du délai de Rose amère*

Les préréglages sont enregistrés dans le fichier *fib-ps* présenté figure 6.15 page ci-contre, où seul le premier préréglage est pris en compte par Rose amère. Le tableau 6.3 synthétise l'ensemble des informations pratiques sur les trois paramètres du délai, y compris sur ceux qui n'appartiennent pas au fichier de préréglage.

Le module du délai reprenant en grande partie les développements de FeedItBack et d'Iviv, on pourra se reporter à la section 7.3.7 page 162 pour plus de détails sur la programmation du délai.

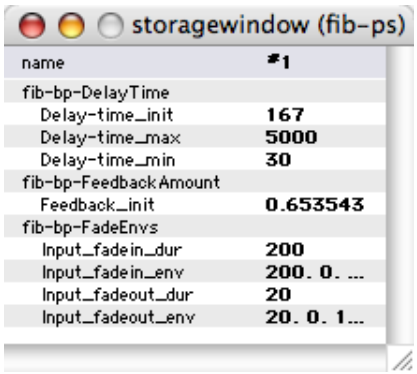


FIG. 6.15 – *Préréglages du délai dans Rose amère*

Fonctionnement du délai

On a vu avec la boucle 3 que l'augmentation du nombre de paramètres combinée à la gestion de l'activation entraîne une complexité croissante. Ainsi, pour pouvoir gérer le délai correctement, Rose amère utilise deux boutons au lieu d'un seul, en reprenant toutefois les principes de fonctionnement des boutons des boucles :

- le bouton 4 reprend exactement le double mécanisme d'activation/désactivation et de A-sélection du volume présenté dans le diagramme performatif des boucles 1 et 2 (cf. figure 6.11 page 136) ;
- le bouton 5 reprend le principe de A-permutation entre deux paramètres exposé pour la boucle 3, mais de façon bien plus simple (cf. figure 6.16), puisque la gestion de l'activation est déportée sur un autre bouton (le bouton 4 en l'occurence).

Ainsi, il existe une véritable indépendance entre le bouton 4 (la commutation du délai et la A-sélection du volume) et le bouton 5 (la A-permutation entre la durée et la réinjection) ; en particulier, si le délai est inactif, le bouton 5 fonctionne quand même et permet de préparamétrer discrètement la durée ou la réinjection.

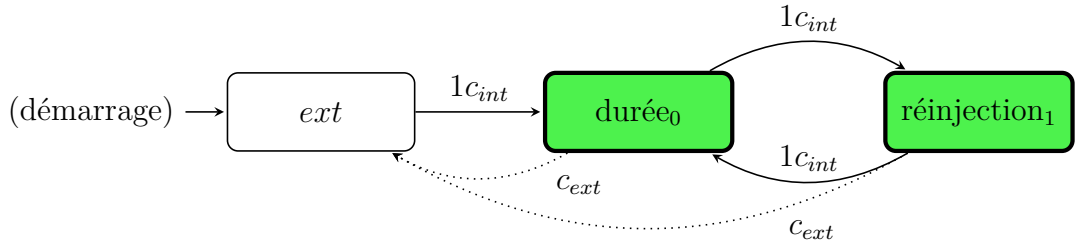


FIG. 6.16 – *Diagramme performatif de la A-permutation des paramètres du délai dans Rose amère (bouton 5)*

Il y a d'autres avantages à cette répartition sur deux boutons. D'une part, le double clic n'étant pas nécessaire, les 500 ms de retard dues à l'analyse n'ont pas lieu. D'autre part, pour permettre à l'instrumentiste de faire l'économie d'une mémorisation supplémentaire, un premier clic interne A-sélectionne toujours le paramètre de la durée (jamais celui de la réinjection), quelque soit la dernière A-sélection quittée (la durée ou la réinjection).

Le résultat sonore dépend d'abord de la durée, et celle-ci peut varier dans des proportions considérables : depuis 30 ms – pour des effets de filtre en peigne – jusqu'à 5 s – pour

une répétition retardée au-delà de l'écho traditionnel. Rose amère ne propose que 126 valeurs intermédiaires, un nombre finalement restreint en regard de la durée importante à parcourir, mais cette quantification garde ici une pertinence musicale grâce à l'échelle exponentielle[33] utilisée pour mettre en correspondance les valeurs Midi de la pédale A avec le contrôle de la durée.

De fait, une échelle linéaire aurait progressé régulièrement par pas de 40 ms, alors qu'il est nettement plus intéressant musicalement de progresser finement pour les durées courtes que pour les durées longues : une palette large dans les durées courtes donne une grande variété d'effets psychoacoustiques jusqu'à 100 ms environ – *phasing*, *flanging*, réverbération, écho – pour devenir ensuite un effet essentiellement rythmique où la précision demeure moins cruciale.

Le paramètre de réinjection se contrôle quant à lui de façon linéaire, entre 0. et 1. ; le fait que le maximum ne dépasse pas 1. permet de poser une première limite aux risques de larsen, même si ce risque n'est jamais nul pour du délai, surtout avec des durées courtes.

6.3.7 Les effets

Présentation des effets

Le quatrième et dernier module sonore offre deux traitements en série sur l'entrée audio : une transposition et une distorsion, auxquelles il faut ajouter le volume, ce qui porte à trois le nombre de paramètres de ce module.

La transposition progresse par demi-ton entre −12 et +12, en colorant le son à cause de l'algorithme de transposition[34] : le son subit un léger retard et les attaques sont un peu gommées ; le résultat sonne donc assez doux, comme « poli » par le traitement.

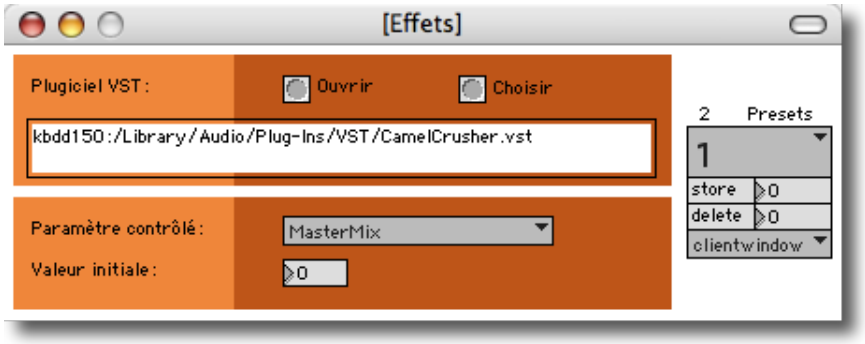


Fig. 6.17 – *Choix du plugiciel à contrôler avec Rose amère*

Pour la distorsion, Rose amère utilise par défaut le plugiciel *CamelCrusher*[35] et contrôle le mixage de l'effet par rapport au signal entrant (le paramètre *MasterMix*).

33. Il s'agit de l'objet `scale` paramétré comme suit : `scale 0 126 50. 5000. 1.06`, dont le cinquième argument indique un mode exponentiel lorsqu'il est supérieur à 1.

34. Cet algorithme utilise une combinaison des objets `pfft~` et `gizmo~`, sur une fenêtre d'analyse de 4096 points avec un facteur de recouvrement égal à 8.

35. Ce plugiciel doit alors se trouver dans le répertoire `/Library/Audio/Plug-ins/VST/`.

Cependant, Rose amère accepte n'importe quel plugiciel VST, et permet de sauvegarder son chemin, le paramètre à contrôler, ainsi que la valeur initiale de ce paramètre (cf. figure 6.17 page précédente).

Lorsque le volume est supérieur à 0 et que les autres paramètres sont à 0, le circuit des effets devient simplement un amplificateur de l'entrée audio.

Fonctionnement des effets

D'abord, le bouton des effets n'a pas de statut pour l'activation, car il est activé en permanence ; il suffit que le volume soit à zéro pour ne pas l'entendre, et le traitement de la transposition se désactive automatiquement si elle est réglée sur zéro. Ainsi, grâce à cette économie du statut d'activation, un seul bouton suffit.

Ensuite, ce bouton unique fonctionne sur le principe de la A-permutation circulaire, comme pour le bouton 5 (du délai), mais avec trois paramètres – le volume, la transposition et la distorsion. La A-sélection d'entrée se fait toujours par le volume, et dans cet ordre.

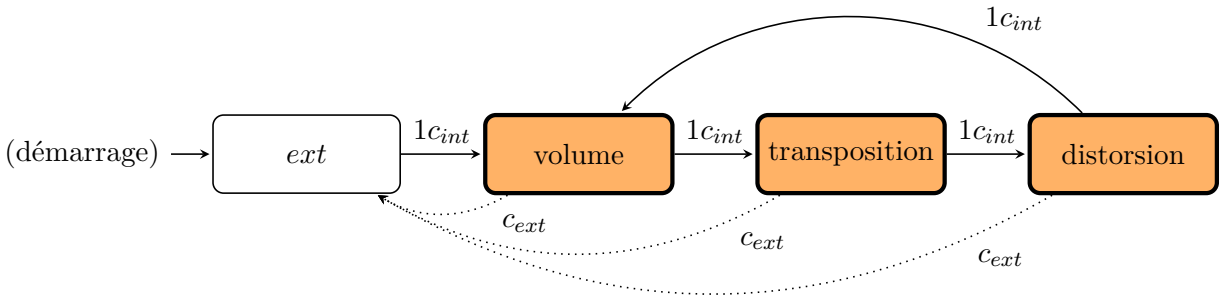


FIG. 6.18 – *Diagramme performatif de la A-permutation des paramètres des effets dans Rose amère (bouton 10)*

Enfin, il y a une astuce pour remettre à la fois la transposition et la distorsion à zéro : un double clic ; cela évite de devoir répéter l'opération de remise à zéro avec la pédale A pour les deux paramètres, surtout que cette opération reste délicate pour la transposition (le zéro se trouvant au milieu de la course). Cependant, ce double clic continue la A-permutation en parallèle des deux simples clic équivalents qui déplacent la A-sélection, ceci afin d'éviter le retard de 500 ms de l'analyse des clics. En effet, comme la distorsion est susceptible d'entraîner des larsens, la priorité a été donnée à la vitesse de réaction.

6.3.8 Le mixage

Dans la polyphonie à quatre voies que propose Rose amère, le mixage apparaît rapidement comme une fonctionnalité indispensable : pour que le résultat sonore global ait un sens, il faut que les différents plans puissent être ajustés en direct, et de la façon la plus commode possible.

Cependant, sachant que la pédale B est strictement réservée au volume global, il ne reste qu'une seule pédale pour mixer les quatre voies. Le mixage se fait donc par A-sélection :

Module	*Boutons*	*Exceptions*	*Pré-normalisation*	*Sortie séparée*
Boucle	1, 2, 3	3 (2 paramètres)	oui	`dac 3`
Impacts	6, 7, 8	–	oui	`dac 4`
Délai	4	5 (2 paramètres)	non	`dac 5`
Effets	10	10 (3 paramètres)	non	`dac 6`

Tab. 6.4 – *Récapitulatif des accès au mixage de Rose amère*

Le niveau (pré-fadeur) des boucles et des impacts est approximativement prévisible, grâce à la normalisation automatique du tampon de lecture lors de son chargement (objet `normalize~`). En revanche, le niveau du délai et des effets n'est pas prévisible, car il dépend de l'entrée audio. Tous les niveaux sont à 0 au démarrage, sauf les impacts qui sont à 70, déjà disponibles.

L'accès au volume d'un module (*i. e.* l'accès à la A-sélection du volume) se fait toujours dès le *premier* clic entrant sur le bouton correspondant (cf. tableau 6.4), ce qui répond à l'aspect prioritaire du mixage en situation de direct. Excepté donc pour les boutons qui utilisent la A-permutation entre plusieurs paramètres (les boutons 3, 5 et 10), cet accès direct au volume est maintenu le plus souvent possible. De cette façon, le volume peut être ajusté même si le module n'est pas actif, par exemple pour préparer un fondu entrant sur une boucle à venir.

La sortie audio générale, monophonique pour l'instant, est dupliquée sur les sorties 1 et 2, tandis que les quatre modules Boucle, Impacts, Délai et Effets sortent respectivement en 3, 4, 5, et 6 ; enfin, l'entrée pré-traitement (égalisation et volume) sort en 7 et l'entrée post-traitement sort en 8.

6.3.9 L'accrochage

Un mécanisme d'accrochage, repris du logiciel Ifso[36], pare à un problème dû à la A-permutation. En effet, lorsqu'on change le paramètre associé à la pédale A, celle-ci n'étant pas motorisée, il y a de grandes chances pour que la valeur actuelle du nouveau paramètre soit en décalage avec la position effective de la pédale progressive. Ce décalage peut se révéler rédhibitoire à l'usage, par exemple sur le volume qui prendrait une valeur tout à coup très éloignée, au moindre mouvement de la pédale…

L'accrochage empêche précisément ces sauts brutaux d'arriver, en bloquant la transmission tant qu'une valeur envoyée par la pédale A n'a pas suffisamment approché la valeur actuelle du paramètre nouvellement sélectionné. La zone d'accrochage a été fixée par défaut entre -10 et $+10$ autour de la valeur actuelle du paramètre, sur l'intervalle des 128 valeurs possibles, à travers mon objet `kb-accrocheur127`.

La notification visuelle du bon accrochage à l'utilisateur se fait par un basculement de la couleur de deux curseurs : le curseur concerné et le curseur de la pédale A, qui passent tous deux du jaune – la couleur de première A-sélection, donc en instance d'accrochage – à la couleur du module concerné – soit bleu, rose, vert ou orange. Tant que l'accrochage

36. Cf. figure 9.9 page 202.

n'a pas eu lieu, le curseur du module concerné reste immobile et jaune, indiquant la valeur que la pédale doit approcher pour le débloquer.

6.3.10 Les fichiers de préréglages

Rose amère compte en tout 8 fichiers de préréglages[37], rassemblés et commentés dans le tableau 6.5.

Entrée	`entree_matrice-ps.xml` `filtergraph5-ps.xml`	détermine le nombre d'entrées et leur routage stocke les coefficients des 5 filtres d'égalisation
Boucles	`loop-ps.xml`	paramètre les boucles 1 et 2 pour la vitesse de lecture et le LFO

Impacts	`impacts_pou-ps.xml`
	`impacts_ka-ps.xml`
	`impacts_ti-ps.xml`
		contiennent les bornes du tirage aléatoire pour la vitesse de lecture et les segments de l'enveloppe ADSR

Délai	`fib-ps.xml`	spécifie la course de la durée et du taux de réinjection
Effets	`plugiciel-ps.xml`	mémorise le plugiciel à utiliser et le paramètre à contrôler

TAB. 6.5 – *Les fichiers des préréglages de Rose amère*

L'édition de ces fichiers de préréglages est unifiée par le bpatcher `[kb-Storage-mng]` qui permet d'accéder aux principaux messages adressables à l'objet `pattrstorage` : store, delete, clientwindow, storagewindow, read, readagain, write, writeagain, getslotlist, renumber, et clear.

FIG. 6.19 – *Capture visuelle du bpatcher kb-Storage-mng*

À nouveau, la prolifération des préréglages dans des logiciels comme Mimi et Rose amère, logiciels clairement destinés à la performance en direct jusqu'à l'improvisation – soit au temps réel musical –, tend à confirmer une des conclusions de la première partie de ce mémoire : pour fonctionner efficacement en temps réel (musical ou pratique), un logiciel tire avantage de l'utilisation d'éléments temps différé (pratiques ou techniques).

* * *

Après avoir relevé l'épreuve du concert à plusieurs reprises, Rose amère nous semble répondre de façon satisfaisante aux souhaits et aux contraintes formulés par rapport à la situation du direct, à la pratique de l'improvisation et à l'augmentation de l'alto. En effet,

37. Le suffixe « `-ps` » fait référence à l'objet `pattrstorage` qui gère ces fichiers.

ce logiciel s'insère dans le dispositif podophonique interactif en proposant une certaine variété sonore – à travers les quatre modules et leurs potentiels *vocal* et *percussif* –, et en opérant les compromis nécessaires à la fluidité de l'interaction pédestre, avec la restriction à une banque unique. Cette restriction à dix pédales interruptrices seulement engendre des conséquences pratiques : le recours à plusieurs mécanismes locaux, comme la A-sélection et la A-permutation, selon différents schémas de transition d'états, parfois complexes.

Rose amère devient ainsi une extension logicielle qui permet à l'instrumentiste de jouer de l'ordinateur en même temps qu'il joue de son instrument et qu'il s'auto-enregistre. Néanmoins, ce double jeu – acoustique et électronique – demande un temps d'apprentissage conséquent et surtout une virtuosité supplémentaire importante : au niveau du jeu pédestre bien sûr, mais aussi pour l'écoute démultipliée par les quatre modules, ainsi que pour la visualisation des retours graphiques ; quant à l'improvisation collective, elle exige une disponibilité aux autres qui rend évidemment l'exercice encore un peu plus virtuose. . .

Chapitre 7

FeedItBack et Iviv / Santiago Quintáns

Résumé du chapitre : Ce chapitre retrace ma collaboration avec le compositeur Santiago Quintáns et la percussionniste Clarissa Borba. La genèse de cette collaboration s'est opérée autour de FeedItBack, un logiciel d'écho pour pédalier MIDI. Ensuite, sa maturation s'est incarnée dans la programmation du logiciel « Iviv » parallèlement à l'écriture de la pièce *In Vivo / In Vitro* (pour caisse claire et pédalier). Enfin, son dénouement s'est produit en concert, avec l'exécution de cette pièce et de deux improvisations, l'une au vibraphone et l'autre à la guitare.

7.1 Introduction

7.1.1 Rencontre

José-Manuel Lopez-Lopez, qui enseigne la composition à l'université Paris 8, m'a permis de passer une annonce avant l'un de ses ateliers : j'ai alors proposé aux étudiants qui le souhaitaient une collaboration informatique en vue d'un concert, avec le pédalier MIDI et la clarinette, ou tout autre instrument laissant les pieds suffisamment libres pour jouer du pédalier. C'est à cette occasion que j'ai rencontré Santiago Quintáns, intéressé par ce projet de musique mixte et interactive, appliqué à la caisse claire.

7.1.2 Deux projets musicaux progressifs

Santiago, compositeur et guitariste, possédait déjà une grande expérience des pédaliers pour guitare, ce qui a immédiatement orienté les débuts de notre collaboration : il m'a proposé de reproduire les fonctionnalités d'une pédale d'écho dont il a l'habitude[1].

Ainsi, une fois que le premier logiciel, nommé « FeedItBack », s'est révélé fonctionnel, le second projet, nettement plus ambitieux mais tout à fait réalisable, a pu démarrer... Nous avons alors établi un nouveau cahier des charges en concertation, et Santiago a pu écrire la pièce *In Vivo / In Vitro*, au fur et à mesure de mes avancées sur le nouveau logiciel, nommé « Iviv ».

Remarque : L'anglais s'est imposé comme langage pour l'interface des logiciels de ce projet afin que l'on se comprenne rapidement, car Santiago revenait en 2006 de plusieurs années d'études universitaires aux États-Unis, au cours desquelles il a hérité d'un vocabulaire technique que je partage. Cela nous a aussi permi d'échanger les logiciels avec ses amis qui habitent là-bas. Espagnol, il maîtrise évidemment parfaitement sa langue maternelle qui aurait pu être celle des interfaces, mais malheureusement je ne suis pas du tout hispanophone.

7.1.3 In Vivo / In Vitro – Quelques notes sur la pièce

In Vivo / In Vitro est une pièce de Santiago Quintáns pour caisse claire solo et dispositif podophonique interactif interfacé par le logiciel Iviv, écrite pour la percussionniste Clarissa Borba qui l'a créée le 2 février 2007 à Gennevilliers, et qui dure environ trois minutes. Une structure tripartite apparaît clairement sur la partition de Santiago (jointe en annexe E.1 page 293), qui indique trois sections nommées A, B et C.

La section A utilise les balais sur deux modes de jeu principaux qui se répondent : les frottements rapides en crescendo, suivis d'un enchaînement de quelques cellules rythmiques accentuées en decrescendo. L'électronique est utilisée de plusieurs façons, en rap-

1. L'aspect concret et efficace de cette proposition, qui fixe un objectif tout à fait accessible, préfigurait le pragmatisme de la démarche de Santiago, et plus tard le succès global du projet en situation de concert, tant pour la pièce écrite que pour les improvisations.

		Delay	*Rec*	*Loop*	*Fleur*	
	1	1 on				écho des frottements
	1	1 off				
	5	1 on				*idem*
	5	1 off				
	9	1 on				*idem*
	10	1 off				
	14		2 on			1er enregistrement (frottements)
	18		2 off			
A	21			6 on		début d'un « long souffle grave »
	22	1 on				écho de frottements
	22	1 off				
	24	1 on				*idem*
	24	1 off				
	30				3 on fleur	fleur brève en réponse (2 temps)
	30				4 off fleur	
	31				3 on fleur	*idem*, moins bref (4 temps)
	31				4 off fleur	
	36				3 on fleur	enchaînement *on/rev*, plus épanoui
	38				5 rev fleur	
	42				3 on fleur	*idem*, en avance sur la c.c.
	43				5 rev fleur	
B	45				8 zoom	synchrone avec la c.c.
	46				8 zoom	*idem*
	47				3 on fleur	*on/rev*, en retard sur la c.c.
	49				3 rev fleur	
	52			7 off		silence
	59		2 on			2e enregistrement (roulement)
	60		2 off			
	62			6 on		boucle de « pseudo-gong étouffé »
	64				3 on fleur	ouverture brève
C	65				3 on fleur	*idem*
	66				4 off fleur	
	68				3 on fleur	enchaînement *on/rev*
	68				5 rev fleur	
	70			7 off		
	76				10 final fleur	geste final ensemble

FIG. 7.1 – *In Vivo / In Vitro – Relevé des déclenchements*

port avec ces deux modes de jeu de la caisse claire, mais seuls les frottements servent pour l'entrée audio du logiciel dans cette première partie. En effet, ils sont repris plusieurs fois par le délai (mes. 1, 5, 9, 22, 24), pour prolonger cette texture des balais sur les réponses percussives. De même, ce sont des frottements qui entrent pour l'auto-enregistrement (mes. 14–17), nourrissant ainsi une longue boucle très ralentie (mes. 21–52), et toutes les « fleurs[2] » jusqu'à la section C. Deux fleurs très brèves terminent la section A en suspens, tandis que la boucle continue son souffle grave.

La section B troque les balais contre les baguettes, inaugurant des textures granuleuses produites par les roulements et articulées aux réponses rythmiques, comme dans la section A (des subdivisions ternaires font cependant leur apparition). L'électronique

2. Les « fleurs » peuvent être décrite en première approximation comme des événements sonores saillants, déterminés par une enveloppe d'amplitude croissante ou décroissante. La section 7.3.9 page 166 présente en détail le module qui permet de les générer.

emploie ici un jeu différent, faisant appel aux trois déclinaisons des fleurs : à l'endroit (*on fleur*), à l'envers (*rev fleur*), ou brèves (*zoom*). Trois figures constituées par un enchaînement à l'endroit puis à l'envers sont exposées en décalage par rapport à la partie de caisse claire : d'abord solo (mes. 36–37), puis en avance sur un roulement globalement continu (mes. 42), et enfin en retard sur un roulement en decrescendo (mes. 47). Les *zooms*, quant à eux, interviennent en synchronie avec deux répliques rythmiques (mes. 45 et 46). Une fin abrupte clôt cette partie, en arrêtant simultanément le jeu de la caisse claire et la lecture de la boucle grave d'Iviv.

La section C reprend ainsi sur un solo de caisse claire de sept mesures, dont les deux dernières – une longue série de doubles croches jouées avec une gestuelle circulaire tout autour de la partie métallique du fût [3] – doivent être enregistrées. Ce nouveau tampon renouvelle complètement la sonorité des boucles et des fleurs : par exemple, la boucle sonne comme une série de coups de gong étouffés et lointains (mes. 62–70), au lieu de sonner comme un souffle continu. Encore quelques fleurs (mes. 64, 65, 68), un dernier solo *forte* sur des réminiscences des cellules rythmiques du début, et le geste final rassemble les deux instruments sur un long roulement en decrescendo *al niente* avec une *final fleur*…

Finalement, cette pièce déploie un dialogue élaboré entre la caisse claire et ses propres déformations informatiques, que ce soit à travers le délai, qui ne fait que répéter le jeu de la caisse claire, ou à travers les boucles et les fleurs, qui relisent bien un enregistrement réalisé en direct de cette même caisse claire, même si la reconnaissance perceptive est rendue plus difficile par le changement de vitesse de lecture. En quelque sorte, la caisse claire dialogue avec une autre elle-même, impression qui peut être renforcée par le fait que c'est le même instrumentiste qui joue des deux.

7.2 FeedItBack : notes de programmation

7.2.1 Mimique d'une pédale de délai par un pédalier Midi

La pédale d'écho à imiter représentait un modèle standard dans sa forme, relativement simple et plutôt robuste : il s'agit de la Boss RV-3 reproduite figure 7.2 page suivante. Elle propose à la fois de la réverbération (4 modes) et du délai (3 modes), individuellement ou ensemble, ces effets étant numériques mais sans rappel de préréglage ; précisons que l'exercice consistait à imiter le module de délai uniquement, sans le module de réverbération.

Ce boîtier ne comporte qu'une seule pédale, dédiée au volume, l'accès aux autres paramètres devant se faire à la main sur les potentiomètres rotatifs, pour doser le délai (*delay time*), la réinjection (*feedback*) et le mixage de l'effet (*balance* ou *dry/wet*). Par rapport à la pédale d'origine, le pédalier FCB1010 possède deux pédales progressives, ce qui l'autorise à se prêter à l'exercice de mimique envisagé.

À la différence de la pédale RV-3 qui émet un signal analogique, les deux pédales du FCB1010 transmettent en sortie un signal numérique MIDI, allant de 0 à 127. La pédale B

3. *« Gradually move around all metallic parts of snare »*, mes. 59.

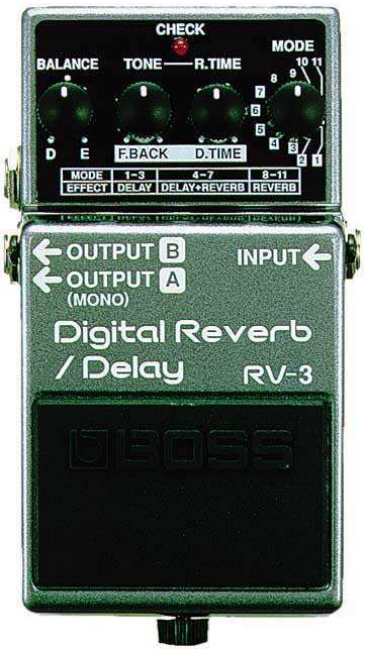


FIG. 7.2 – *Pédale de réverbération et de délai Boss RV-3*

(celle de droite) reprend la convention de rigidification que nous avons adoptée pour toutes les collaborations avec le FCB1010 – à savoir l'accès persistant au volume général du logiciel (cf. section 6.2.2 page 122) –, tandis que la pédale A reprend le mécanisme de la permutation fonctionnelle sur d'autres paramètres, différents du volume général (cf. section 6.2.2 page 123).

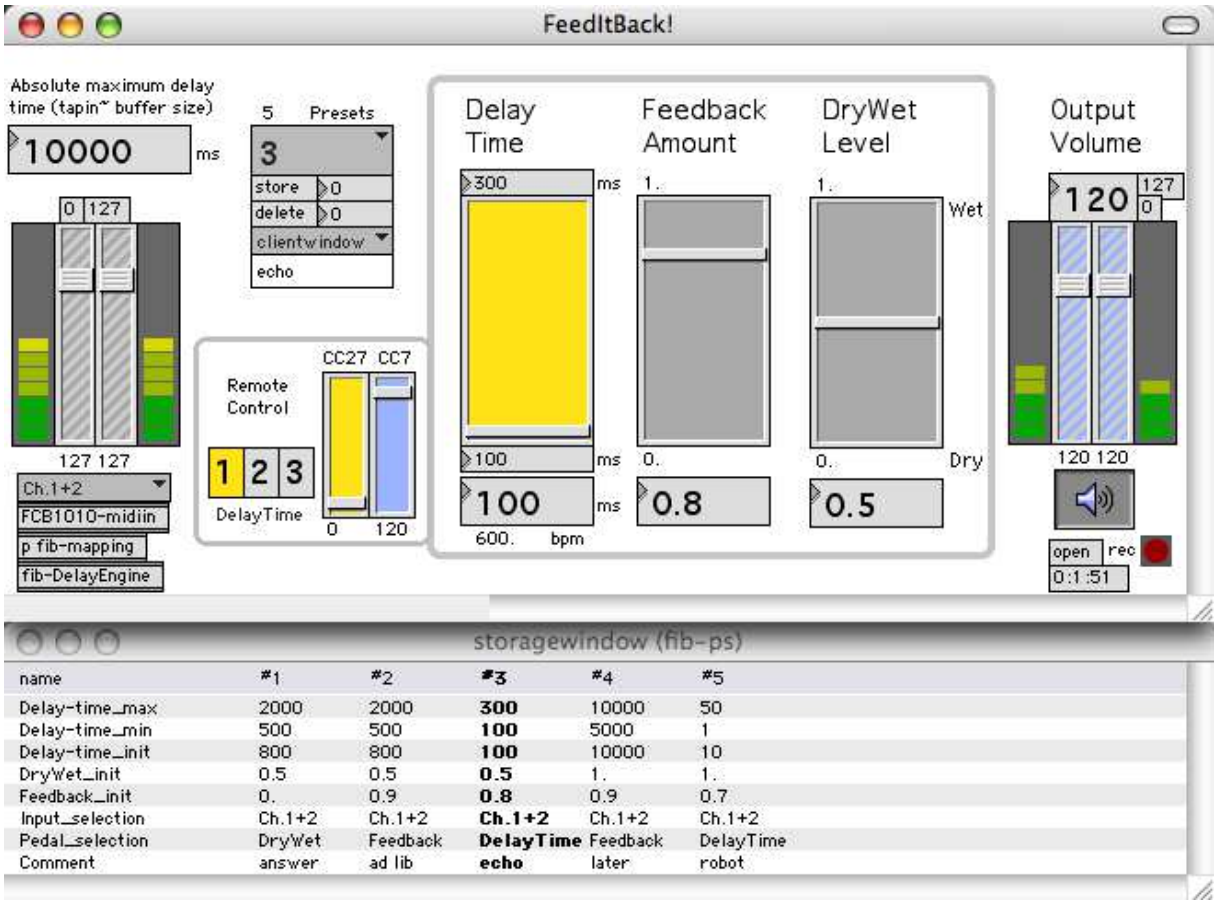
7.2.2 La souplesse mnésique du numérique

Les trois paramètres principaux de la pédale d'origine s'appellent donc *« Delay Time »*, *« Feedback Amount »* et *« Output Volume »*, auxquels j'ai ajouté *« DryWet Level »* pour doser le mixage entre la source et son traitement (cf. figure 7.3 page suivante). Les bornes de la plupart des paramètres peuvent être fixées *a priori* :

- pour *Feedback Amount*, la quantité de réinjection est un facteur nul au minimum et égal à un au maximum (sous peine de rapidement faire exploser le système) ;
- pour *DryWet Level* plusieurs solutions étaient possibles, mais j'ai gardé le même intervalle [0.; 1.][4] qui correspond au dosage du son traité, le son de la source parcourant simultanément l'intervalle inverse [1.; 0.] (0.5 correspond à un mixage à égalité) ;
- pour *Output Volume*, le principe d'un coefficient multiplicateur identique s'applique, aménagé d'une part avec une échelle apparente exprimée selon l'habitude MIDI.

Quant au paramètre *Delay Time*, si la pédale d'origine subissait des bornes fixes de 0,5 s à 2 s, soit un écho de 120 à 30 battements par minute, un logiciel peut facilement ne pas se limiter à ces bornes. En effet, la fenêtre *storagewindow* reproduite sur la figure 7.4

4. Les nombres décimaux, normalement écrits avec une virgule en typographie française, peuvent être notés ici avec un point pour rester cohérent avec la notation de *Max/MSP*, par ailleurs comme dans la plupart des langages de programmation.



name	#1	#2	#3	#4	#5
Delay-time_max	2000	2000	**300**	10000	50
Delay-time_min	500	500	**100**	5000	1
Delay-time_init	800	800	**100**	10000	10
DryWet_init	0.5	0.5	**0.5**	1.	1.
Feedback_init	0.	0.9	**0.8**	0.9	0.7
Input_selection	Ch.1+2	Ch.1+2	**Ch.1+2**	Ch.1+2	Ch.1+2
Pedal_selection	DryWet	Feedback	**DelayTime**	Feedback	DelayTime
Comment	answer	ad lib	**echo**	later	robot

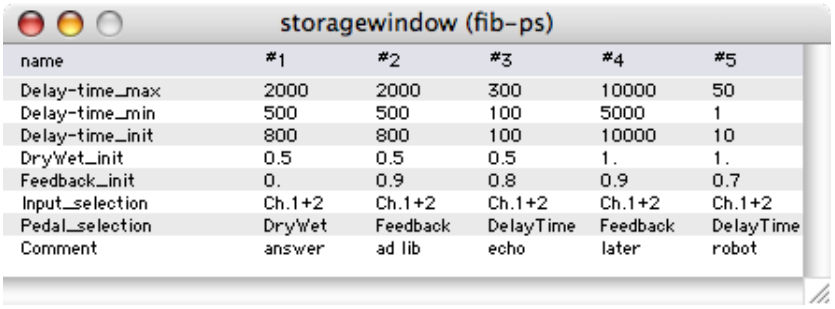
FIG. 7.3 – *Capture de l'interface visuelle de FeedItBack*

page suivante montre cette variété possible pour le logiciel, à partir des cinq préréglages que j'ai établis grâce au système `pattr` intégré à *Max*[5]. Les deux bornes du délai, nommées ici *Delay-time_max* et *Delay-time_min*, reprennent les valeurs de la pédale d'origine pour les deux premiers préréglages, soit respectivement *2000* et *500* (en millisecondes), puis des valeurs tout à fait différentes pour les trois autres préréglages, allant jusqu'à 10 000 ms pour le maximum (#4) et jusqu'à 1 ms pour le minimum (#5).

En deçà d'une milliseconde, on trouve avec les nombres entiers un délai nécessairement nul, ce qui ne présente pas d'intérêt. En revanche, la borne supérieure n'admet pas de limite théorique, et le jeu musical avec des délais longs (*i. e.* bien au-delà des 2 secondes proposées au départ) fait parfois émerger des trésors de dialogue avec la réminiscence ainsi « matérialisée ». La limite technique de cette borne supérieure ne dépend ici que de la quantité de mémoire vive disponible, une fois qu'on a pris soin d'ajuster la taille des lignes à retard (avec l'objet `numberbox` tout en haut à gauche de la figure 7.3, initialisé par défaut à 10 000 ms).

Au total, chaque préréglage mémorise ici la valeur de huit paramètres différents, qui sont rappelés dans l'ordre où ils sont affichés dans les fenêtres *storagewindow* ou *clientwindow*, de haut en bas[6] :

5. Pour les (*patcher attributes*), cf. section 6.2.3 page 126.



name	#1	#2	#3	#4	#5
Delay-time_max	2000	2000	300	10000	50
Delay-time_min	500	500	100	5000	1
Delay-time_init	800	800	100	10000	10
DryWet_init	0.5	0.5	0.5	1.	1.
Feedback_init	0.	0.9	0.8	0.9	0.7
Input_selection	Ch.1+2	Ch.1+2	Ch.1+2	Ch.1+2	Ch.1+2
Pedal_selection	DryWet	Feedback	DelayTime	Feedback	DelayTime
Comment	answer	ad lib	echo	later	robot

FIG. 7.4 – *Fenêtre des préréglages de FeedItBack*

1. *Delay-time_max* : valeur maximum du délai variable, en millisecondes ;
2. *Delay-time_min* : valeur minimum du délai variable, en millisecondes ;
3. *Delay-time_init* : valeur d'initialisation[7] du délai variable, en millisecondes ;
4. *DryWet_init* : valeur d'initialisation du rapport de mixage entre la source et le traitement, entre 0. et 1. (correspond en fait au coefficient $c_{traitmt}$ de mixage du traitement, sachant que $c_{source} = 1 - c_{traitmt}$) ;
5. *Feedback_init* : valeur d'initialisation du taux de réinjection du signal retardé, entre 0. et 1. ;
6. *Input_selection* : sélection de l'entrée parmi les trois possibilités Ch. 1+1, Ch. 2+2, et Ch. 1+2 pour mettre en correspondance la première, la deuxième, ou bien les deux voies d'entrées avec les deux voies internes du traitement ;
7. *Pedal_selection* : sélection de la connexion entre les données MIDI de type Control Change n° 27, qui correspondent à la pédale A du pédalier, et l'un des trois paramètres assignables parmi DelayTime, Feedback, et DryWet ;
8. *Comment* : commentaire textuel (facultatif).

7.2.3 Une pédale « détriplée »

Comme la pédale d'origine, la pédale B reste dédiée au volume général de sortie (Ouput Volume), cependant, contrairement à la pédale d'origine, le logiciel FeedItBack permet de contrôler au pied trois paramètres supplémentaires à partir de la pédale A :

– la durée du délai (DelayTime),
– le taux de réinjection (Feedback),
– et le mixage (DryWet).

Ainsi, ces trois paramètres qui devaient être ajustés à la main avec les potentiomètres rotatifs de la RV-3, inaccessibles lorsque l'on joue déjà d'un instrument avec les mains, deviennent directement accessibles aux pieds dans notre dispositif, par une A-permutation[8] sur trois paramètres.

6. Le message priority envoyé à l'objet `pattrstorage` permet de modifier de cet ordre, calqué par défaut sur l'ordre alphabétique, en associant un nombre entier au nom du paramètre (les valeurs négatives sont autorisées).
7. L'initialisation a lieu lors de chaque rappel de préréglage.
8. Cf. section 6.2.2 page 123.

En outre, dans la fenêtre principale du logiciel (figure 7.3 page 152), la petite zone carrée nommée *« Remote Control »* permet à la fois de visualiser et de simuler le pédalier :

- les deux petits curseurs jaune et bleu (objets `slider`) représentent respectivement les pédales d'expression A et B, c'est-à-dire les données Control Change n° 27 et n° 7 ;
- et les trois boutons numérotés 1, 2 et 3 (objets `message`), capables de prendre individuellement la couleur jaune lorsqu'ils sont activés, représentent les pédales 1, 2, et 3 (données Program Change) qui permettent de sélectionner le paramètre qui sera effectivement contrôlé à distance par la pédale A, « détriplée » par le logiciel.

7.2.4 Lignes à retard avec réinjection

Le sous-patch `[fib-DelayEngine]`, reproduit sur la figure 7.5 page 156, montre à lui seul l'essentiel de la chaîne de traitement audionumérique, largement inspirée du tutoriel MSP *« 28. Delay line feedback »*. Néanmoins, j'ai modifié ce tutoriel une première fois pour le connecter aux variables du reste du logiciel, mais surtout je l'ai augmenté de mécanismes anti-clic.

Le choix du couple d'objets `tapin~` et `tapout~` plutôt que de l'objet `delay~` repose sur l'impossibilité de la réinjection chez ce dernier, comme le précise le manuel de référence [Zicarelli *et al.*, 2006a, p. 114] :

> *The differences between* `delay~` *and* `tapin~/tapout~` *are as follows : First, delay times with* `delay~` *are specified in terms of samples rather than milliseconds, so they will change duration if the sampling rate changes. Second, the* `delay~` *object can reliably delay a signal a number of samples that is less than a vector size. Finally, unlike* `tapin~` *and* `tapout~`*, you cannot feed the output of* `delay~` *back to its input. If you wish to use feedback with short delays, consider using the* `comb~` *object.*

Ainsi, deux circuits audio stéréo principaux traversent la fenêtre de haut en bas : la source directe, à gauche de la fenêtre, et le signal retardé, passant par les objets `tapin~` et `tapout~`, au milieu de la fenêtre. Dans ce dernier circuit, le branchement cybernétique typique de la réinjection se reconnaît avec les deux boucles formées par les connexions audio jaunes autour des objets `tapin~` et `tapout~`.

7.2.5 Parades anti-clics

Dans l'interaction physique avec les pédales d'expression, les transitions entre les valeurs restent le plus souvent progressives, évitant ainsi les risques de clics audionumériques caractéristiques des changements brusques. Or cela n'est plus le cas dès lors qu'on introduit un système de rappel de configuration interactif, dont l'intérêt, en termes de gestion d'une certaine complexité, est pourtant indiscutable. Il faut alors employer des mécanismes « anti-clic » à chaque endroit de la chaîne de traitement audionumérique susceptible de subir un changement de valeur brusque, le plus souvent en imposant le parcours d'une rampe transitionnelle durant un délai bref, réglé ici à 15 ms par défaut. Cette rampe, initiée à l'aide d'un message adressé à l'objet `line~`, génère autant de valeurs intermédiaires que nécessaire pour le laps de temps défini, et atténue ainsi systématiquement la brutalité des sauts sur plusieurs paramètres.

À propos des trois mécanismes anti-clic mis en œuvre ici, les dénommés *anti-clic feedback* et *anti-clic dry-wet* restent relativement simples : les messages reçus par `r fib-feedback`[9] et `r fib-dry-wet_level` génèrent une brève rampe (de 15 ms par défaut) au lieu de modifier abruptement le facteur d'amplitude des multiplicateurs audio `*~`. En revanche, le troisième mécanisme, *anti-clic delay*, fait appel à un algorithme plus complexe, avec trois boucles fonctionnelles qui sont visibles dans la partie droite de la fenêtre (cf. figure 7.5 page suivante). Il s'agit de contrer les clics qui peuvent survenir lors des changements de valeur du délai, car ces changements provoquent un saut d'adresse dans la mémoire de la ligne à retard, entraînant la plupart du temps une discontinuité pendant la lecture. Voici le schéma général de cet algorithme :

1. réception des nouvelles valeurs avec l'objet `r fib-delay_time` ;
2. limitation de la durée minimum entre l'arrivée de deux nouvelles valeurs, ici par défaut à 500 ms, pour éviter les artefacts dûs aux changements rapides de délai pendant la lecture d'une même mémoire sonore (effets de granulation, ou même de *glissendi* pour des changements continus suffisamment lents[10]) ;
3. passage de la nouvelle valeur de délai à traiter, puis blocage de l'accès à la suite de l'algorithme pendant la durée de son exécution ;
4. fermeture en fondu rapide du son provenant de la ligne à retard (15 ms par défaut), avant la modification effective du délai ;
5. positionnement de la nouvelle valeur de délai à la fin de la fermeture en fondu, de façon à ce qu'un clic éventuel ne puisse pas être audible ;
6. réouverture en fondu rapide (15 ms par défaut) du son provenant de la ligne à retard, une fois la nouvelle valeur appliquée ;
7. réouverture de l'accès à l'algorithme de fermeture-positionnement-ouverture (points 4, 5 et 6), après la fin de la rampe de réouverture du son (l'objet `line~` émet en effet un *bang* sur sa sortie droite lorsqu'une rampe est arrivée à son terme).

* * *

FeedItBack interface le pédalier Midi multiple avec un algorithme de délai avec réinjection, afin de simuler une pédale de délai standard. Il donne ainsi un accès pédestre à quatre paramètres, rassemblés dans le tableau 7.1 page suivante.

Il offre en outre la possibilité d'enregistrer et de rappeler ses propres préréglages, et prévient les risques de clics numériques. Enfin, ce logiciel a servi de passerelle vers un projet plus important, Iviv, et y sera réutilisé.

9. Les variables sont ici préfixées par `fib-` pour réduire les risques de conflits de nommage.

10. *« If a signal is connected to an inlet of `tapout~`, the signal coming out of the outlet below it will use a continuous delay algorithm. Incoming signal values represent the delay time in milliseconds. If the signal increases slowly enough, the pitch of the output will decrease, while if the signal decreases slowly, the pitch of the output will increase. »* [Zicarelli *et al.*, 2006a, p. 414]

Durée du délai	*à prérégler*	pédale A
Taux de réinjection	[0., 1.]	pédale A
Niveau du mixage	[0., 1.]	pédale A
Volume général	[0., 1.]	pédale B

TAB. 7.1 – *Les quatre paramètres de FeedItBack*

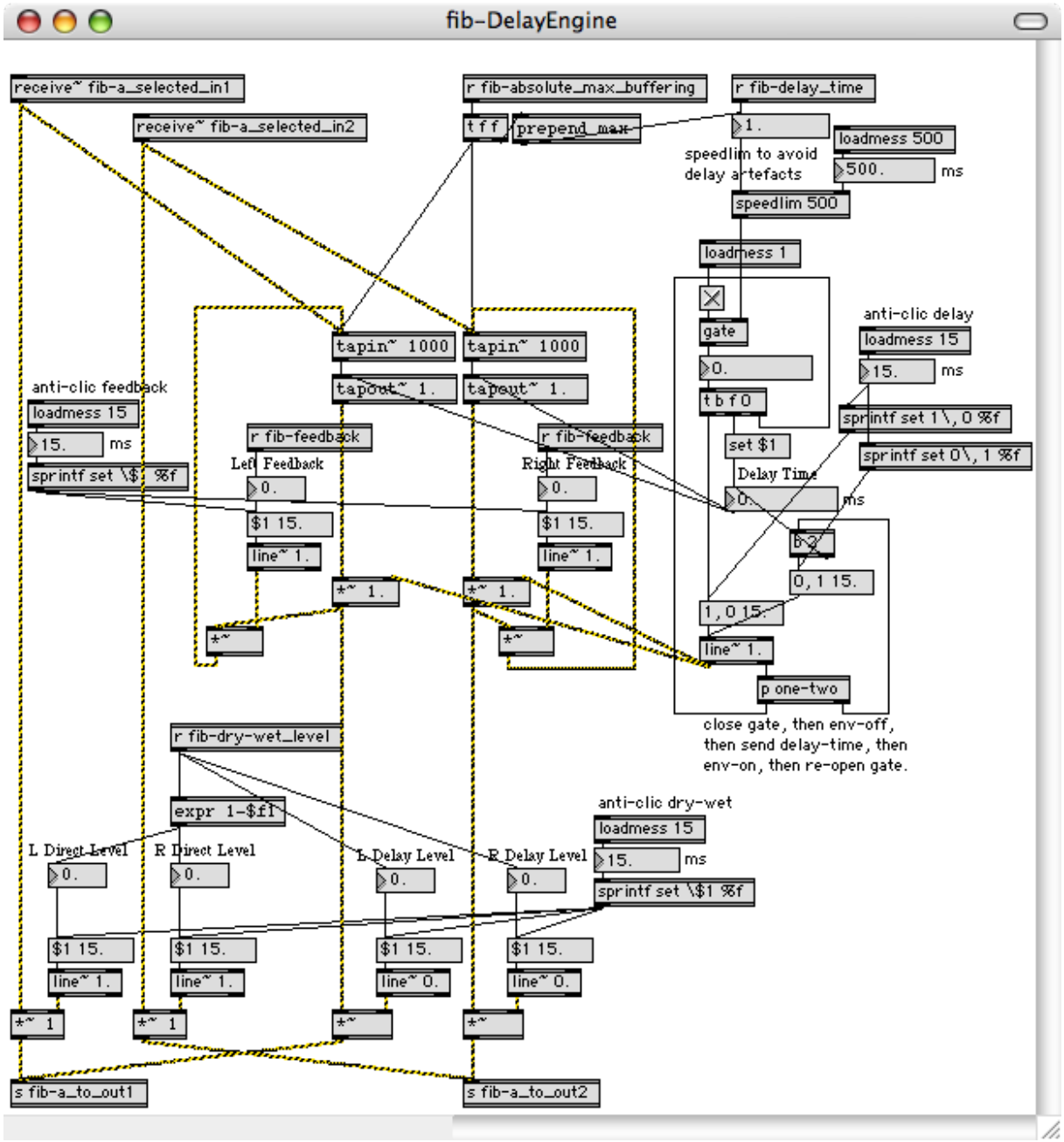


FIG. 7.5 – *Fenêtre des lignes à retard de FeedItBack*

7.3 Iviv : notes de programmation

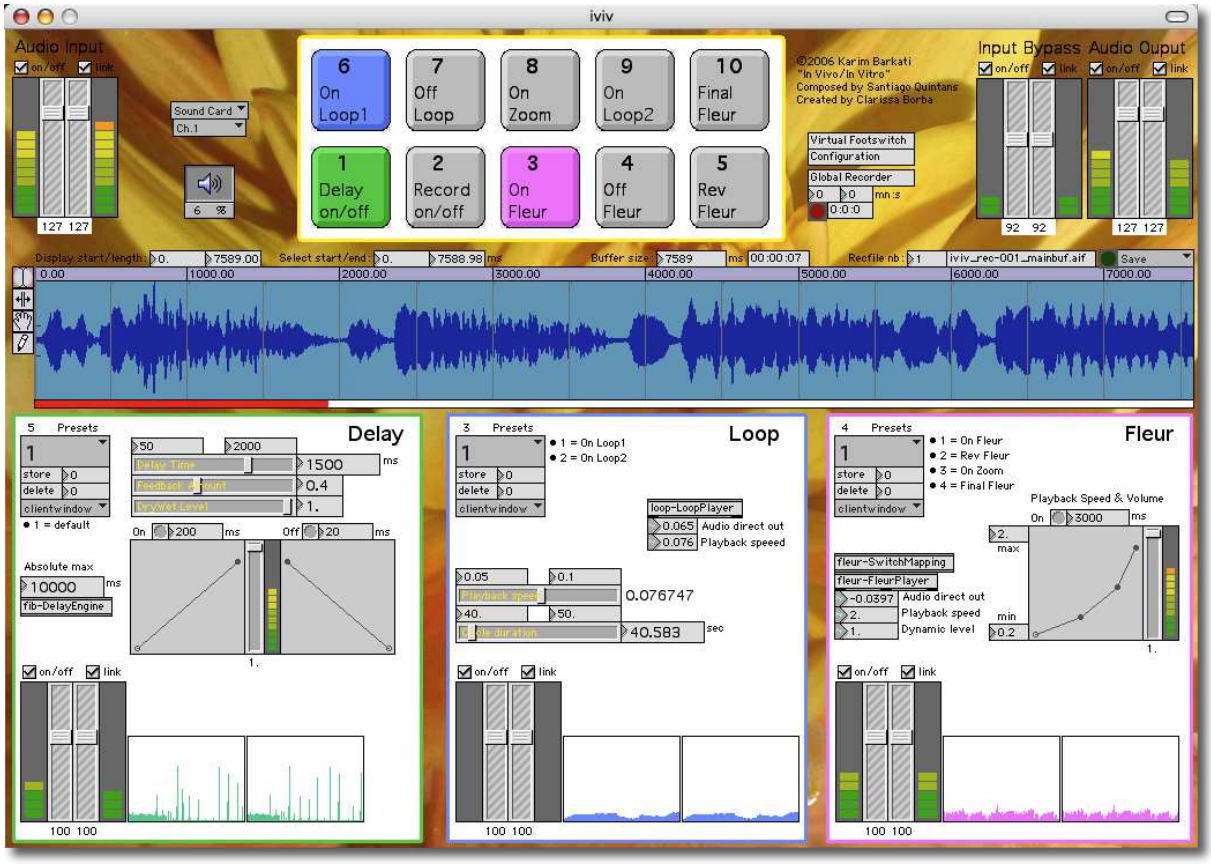


FIG. 7.6 – *Capture de l'interface visuelle de Iviv*

7.3.1 Présentation générale d'Iviv

Après les essais plutôt concluants du logiciel FeedItBack, Santiago a rapidement eu une idée assez précise de la pièce qu'il voulait écrire avec la caisse claire et le pédalier. Nous avons alors défini ensemble les différentes modalités de l'interaction avec le logiciel, en fonction de ses souhaits compositionnels et de mes possibilités en programmation, autour de trois modules :

1. le module *« Delay »*, inspiré des travaux précédents sur FeedItBack,
2. le module *« Loop »*,
3. et le module *« Fleur »*[11], ces deux derniers modules étant de lointains parents du logiciel Mimi dont j'avais fait une démonstration à Santiago au début de notre collaboration.

11. La dimension poétique de ce terme au milieu du vocabulaire technique ambiant m'a conduit à placer une photographie de fleur en fond de la fenêtre principale. Ce terme, choisi par Santiago, possède en outre une efficacité métaphorique surprenante tant il s'est révélé explicite pendant le travail, avec moi comme avec la percussionniste.

Ces trois modules répondent à un souhait d'équilibre sonore déployé sur différents plans. En particulier, le module *Loop* est envisagé comme un arrière-plan, une sorte de générateur de textures graves et lentes (ce qui justifie l'emploi de vitesses de lecture inférieures à 1), et le module *Fleur* est davantage envisagé comme un avant-plan, un générateur d'événements sonores plus saillants et fugaces (ce qui justifie l'emploi d'enveloppes dynamiques rapidement croissantes ou décroissantes). À cet égard, la partition corrobore cette dichotomie : les boucles sont rares (il y en a deux en tout) et peuvent durer très longtemps (p. ex. *loop 1* de mes. 21 à mes. 52), tandis que les fleurs restent brèves (exception faite de *Final Fleur*) mais beaucoup plus nombreuses (j'en dénombre quatorze : *On Fleur*, *Rev Fleur* et *Zoom* compris).

Une autre caractéristique du logiciel Iviv réside dans l'absence de son *a priori*, c'est-à-dire l'absence de synthèse et l'absence de sons préenregistrés (cf. section 4.1.2 page 88), et donc dans la contrainte de capture du son à partir du jeu en direct : *Iviv* ne produit du son qu'à partir de ce qu'il reçoit, sans toutefois se réduire à un module d'effet. Cette caractéristique, tout à fait volontaire, traduit notre souhait commun d'explorer, à travers ce projet, le temps réel en tant qu'un *espace des possibles* musical.

Les sections suivantes présentent d'abord les éléments communs aux trois modules – leur aspect visuel général, les objets de manipulation, le système de préréglage, les outils de surveillance –, puis les trois modules eux-mêmes – *Delay*, *Loop* et *Fleur*.

7.3.2 Aspect visuel général et encapsulation

Les trois modules partagent un certain nombre d'éléments, et d'abord leur aspect visuel général :

- un rectangle de fond blanc délimité par un contour coloré (objet `panel`) ;
- le nom du module en haut à droite avec une grande police (*Geneva* 18 pt) ;
- le système de gestion des préréglages, en haut à gauche (nommé *Presets*) ;
- un même système de visualisation de l'activité sonore, en bas.

En outre, ils partagent la structure d'encapsulation fenêtrée offerte par l'objet `bpatcher`, ce qui facilite le masquage des objets et des connexions dans la fenêtre principale du logiciel. Chaque module utilise cette structure en profondeur 2, avec un `bpatcher` principal qui contient lui-même plusieurs autres objets `bpatcher` (autant que de sous-groupes d'interface graphique, pour les préréglages, la visualisation de l'activité sonore, etc.).

7.3.3 Visualisation de l'activité sonore des modules

Tout en bas de la fenêtre principale d'Iviv (cf. figure 7.6 page précédente) se retrouve donc trois fois le même système de visualisation de l'activité sonore, individualisé par la couleur selon le module concerné (vert, bleu ou rose), en concordance avec le contour du module et surtout avec la couleur des dix boutons qui représentent les pédales interruptrices (en haut de la fenêtre principale). Cette coloration des modules permet de les identifier d'un coup d'œil (par exemple lorsque le regard doit faire des aller-retours avec la partition), mais on souhaite aussi voir à l'écran une trace de l'histoire récente de l'activité sonore…

Si les traditionnels objets `meter~` permettent de visualiser le niveau instantané du signal, c'est une astuce d'emploi de la propriété de défilement des objets `multislider` qui permet de visualiser l'histoire récente de ce niveau d'amplitude, en gardant visible une trace de l'activité sonore de chacun des trois modules.

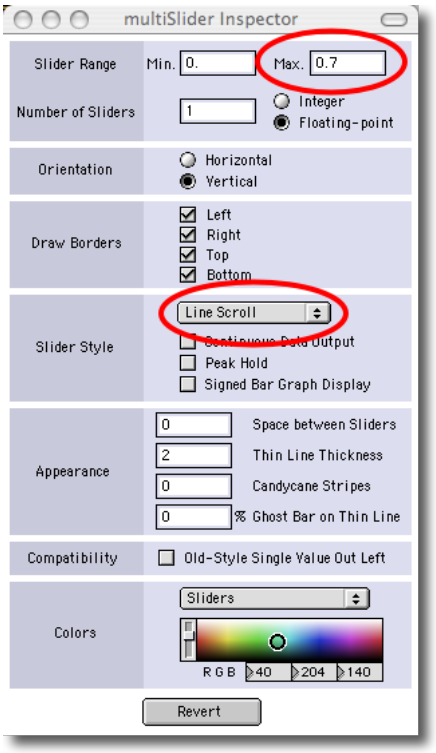


FIG. 7.7 – *Inspecteur d'un multislider en mode défilement*

Cette astuce consiste à utiliser le mode *Line Scroll* de ces objets, conjugué à un intervalle rétrécit de [0.; 0.7] pour l'amplitude (cf. figure 7.7), où le maximum inférieur à 1. accroît la lisibilité de la trace en augmentant relativement la hauteur des segments unitaires verticaux, segments qui correspondent aux crêtes du signal relevées à intervalle régulier par les objets `meter~` (ici toutes les 100 ms), et dont la succession régulière des mesures provoque un défilement régulier, de gauche à droite[12].

Les 102 pixels de largeur des objets `multislider` représentent exactement les 10 dernières secondes de l'histoire du signal, une fois déduits les 2 pixels de contours [13], soit une information potentiellement fort utile, au compositeur comme à l'interprète. Enfin, ces objets permettent d'individualiser efficacement ces modules les uns des autres grâce à la possibilité de colorisation des segments de l'objet `multislider` conjuguée à la surface non négligeable qu'ils occupent à l'écran (102 px × 75 px par objet).

12. « Quelles qu'en soient les origines, on ne peut que constater la convergence dans la culture occidentale des représentations du temps associées à un déplacement de gauche à droite sur un axe horizontal, qui sont également la règle générale dans les interfaces graphiques. » [Vinet, 1999, p. 107]

13. 100 px × 100 ms = 10 000 ms.

7.3.4 Objets de manipulation

Ce logiciel visant à pouvoir n'être utilisé que via un pédalier – idéal premier au cours de son développement –, on ne devrait logiquement y trouver que des objets de *« pédi-pulation »* et non de *mani*pulation, or ce serait oublier le compositeur. C'est-à-dire que le même logiciel doit prendre en compte non pas un utilisateur mais deux utilisateurs, aux profils bien différenciables : l'instrumentiste et le compositeur (il s'agit de la *fonction* utilisateur, non de deux personnes nécessairement distinctes, cf. section 5.3 page 107). La distinction de ces deux profils entraîne la dissociation de deux usages, créant ainsi une certaine tension dans la définition fonctionnelle du logiciel : l'interprétation demande une interface à la fois expurgée des moyens de réglage et un contrôle pédestre, au pédalier, tandis que la composition, au contraire, demande à la fois une interface riche de moyens de réglage et un contrôle manuel, au clavier et à la souris. En effet, pour composer les réglages, les choisir et les mémoriser, il faut pouvoir tâtonner commodément, dans une boucle action/perception fluide avec l'interface du logiciel. Bien sûr, ces objets de manipulation chargent d'autant l'interface graphique, mais sur la fenêtre principale de ce programme, moyennant quelques efforts de mise en page[14], il y a finalement de la place à la fois pour l'instrumentiste et pour le compositeur.

Ces objets de manipulation – ou objets de commande virtuels[15] – sont, dans ces modules, au nombre de quatre : la `numberbox`, le `slider`, l'objet `function`[16] et le `button`. Tous les quatre permettent d'interagir avec les paramètres de traitement, mais seulement certains d'entre eux sont ici mémorisés par les systèmes de préréglage. D'abord, les objets `button` sont exclus d'emblée, puisqu'ils ne servent qu'au déclenchement. Ensuite, les « doublons » d'interface, destinés à diversifier les modalités d'interaction, ne sont pas retenus : par exemple, pour un même paramètre, une `numberbox` permet d'imposer une valeur numérique précise, tandis qu'un `slider` permet d'explorer rapidement et intuitivement un domaine de variation ; dans ces cas précis de redondance d'information, j'ai choisi de mémoriser la valeur de la `numberbox` plutôt que le `slider`, pour prendre en compte les cas où la résolution est supérieure au 128^{e} (valeur entière standard de l'objet `slider`).

7.3.5 Systèmes de préréglage

Concernant les systèmes *Presets* de chaque module, leur gestion s'appuie sur le même sous-patch générique `[kb-Storage-mng]`, associé aux objets `autopattr` et `pattrstorage` qui sont placés dans les trois modules et individualisés par un préfixe différent (*fib-*, *loop-*, et *fleur-*). Néanmoins, ces trois sous-patchs de préréglage ne sont pas connectés de la même façon au programme principal, et en particulier aux événements provenant du pédalier : les préréglages du module *Delay* sont indépendants, tandis que pour le module *Loop*, le préréglage 1 est déclenché par la pédale 6 et le préréglage 2 par la pédale 9, et que pour

14. Voir par exemple l'astuce de superposition d'un objet `comment` sur un objet `hslider`, qui permet de placer du texte en sur-impression d'un glisseur horizontal, texte en jaune sur la figure 7.6 page 157.

15. « Cette catégorie [des objets de commande virtuels] regroupe les interfaces graphiques associées à des métaphores d'objets, dont la manipulation, par l'intermédiaire de la souris, repose sur une interaction gestuelle simulant une commande réelle : déplacement d'un curseur ou d'une source sonore, clic correspondant à la pression d'un bouton. » [Vinet, 1999, p. 101]

16. Métonymie de *« graphical breakpoint function editor »*.

le module *Fleur* les préréglages 1, 2 et 3 correspondent respectivement aux pédales 4, 5 et 6, et le préréglage 4 à la pédale 10 (cf. tableau 7.2).

Module	Préréglage	Pédale	Étiquette
Loop	1	6	*Loop 1*
	2	9	*Loop 2*
Fleur	1	3	*On Fleur*
	2	5	*Rev Fleur*
	3	8	*On Zoom*
	4	10	*Final Fleur*

TAB. 7.2 – *Correspondance entre les préréglages et les pédales dans Iviv*

En conséquence, l'ordre des préréglages revêt une importance capitale pour les modules *Loop* et *Fleur*, certaines pédales étant directement reliées à leurs préréglages. Il faut en tenir compte lors de chaque opération d'enregistrement d'un préréglage.

7.3.6 Outils de surveillance

Les deux activités de composition et de performance instrumentale (en concert ou en répétition) se déroulent d'autant mieux que les retours offerts par le logiciel sont adaptés, clairs et diversifiés. Le retour sonore, au sens de la « sonification » des programmes informatiques, n'est pas de mise dans ce type de logiciel : il est exclu de faire entendre un quelconque son de confirmation lorsqu'on a appuyé sur la bonne pédale, en concert par exemple. Parallèlement, le son produit par le logiciel constitue déjà une certaine forme de sonification complexe des fonctions, entachée cependant d'un retard invalidant dès lors qu'on utilise des algorithmes de délai, comme c'est le cas pour Iviv. Au total, une double rétroaction s'exerce ici, sonore naturellement, et visuelle, déployée à partir de plusieurs stratégies.

D'abord, on a vu que les systèmes de visualisation de l'activité sonore permettent un retour d'information à la fois diachronique et synchronique grâce au couplage des objets `meter~`, indiquant clairement les pics de saturation d'amplitude en rouge, et des objets `multislider`, montrant une trace graphique déroulante sur les 10 dernières secondes de l'histoire du signal (cf. section 7.3.3 page 158).

Ensuite, les objets de manipulation évoqués précédemment (`numberbox`, `slider`, `function`, `button`) affichent tous leur état courant, même lorsqu'il est modifié de l'extérieur, c'est-à-dire autrement que par leur manipulation directe[17] à la souris. Pour tous ces objets, le retour visuel a bien lieu s'ils sont modifiés par exemple par le pédalier, ou par la représentation à l'écran du pédalier (le bpatcher `[iviv-FcbVisuSwitch]`), ou encore par le rappel global d'un préréglage. Ainsi, ces objets de manipulation ne restent pas cantonnés dans leur rôle de saisie des paramètres, mais offrent aussi un retour d'information visuel précis, indispensable à la composition, et potentiellement éclairant pour l'interprétation.

17. « [...] la manipulation directe permet de déplacer un objet pour juger, au fur et à mesure de la manipulation, de son bon placement. Alors que le mode conversationnel procède du langage, code qu'il faut apprendre pour communiquer, la manipulation directe exploite nos capacités sensori-motrices élémentaires : montrer, prendre, déplacer, lâcher. » [Beaudoin-Lafon, 1999, p. 124]

De plus, afin de visualiser les différents signaux audionumériques pendant le fonctionnement du logiciel, plusieurs objets `number~` (*signal monitor*) ont été disséminés dans les deux modules *Loop* et *Fleur* :

- deux objets ont été placés dans le module *Loop* pour surveiller d'une part la sortie audio (même les très faibles variations y sont visibles) et d'autre part la valeur courante de la vitesse de lecture (montrant la variation continue provoquée par l'oscillateur),
- et trois objets ont été placés dans le module *Fleur* pour visualiser les deux mêmes paramètres plus un troisième, le facteur d'amplitude courant appliqué par l'enveloppe dynamique.

Enfin, les boîtes de quatre sous-patchs cruciaux sont laissées visibles, afin de pouvoir voir leur contenu par un double clic sur leur boîte. Ainsi, les trois sous-patchs moteurs des trois modules sont laissés visibles et cliquables : `[fib-DelayEngine]`, `[loop-LoopPlayer]` et `[fleur-FleurPlayer]`. De même, le sous-patch `[fleur-SwitchMapping]` a été laissé apparent, pour pouvoir vérifier le schéma de correspondance entre les quatre premiers préréglages et les quatre pédales idoines, et entre la commande d'arrêt et sa pédale. Les trois sous-patchs moteurs, une fois ouverts, montrent eux-mêmes plusieurs objets de visualisation pour pouvoir surveiller le déroulement des processus, ou encore pour aider à comprendre les algorithmes principaux.

7.3.7 Le module « Delay » : un écho dissimulé

Calqué sur le modèle du logiciel FeedItBack développé précédemment, le module *Delay* recèle une nouveauté majeure imaginée par Santiago : la gestion d'enveloppes automatiques de fondu sur le signal entrant. Cette idée semble lui venir de sa pratique de guitariste lorsqu'il utilisait sa pédale de délai. Il avait en effet développé un jeu subtil qui masquait l'attaque du son afin d'éviter la reconnaissance évidente de l'écho d'une note de guitare, car l'écho reste trop perceptible lorsqu'il est appliqué à des sons à l'enveloppe de type percussive. Ainsi, en associant l'écho à cette suppression de l'attaque par fondu entrant, il arrivait à obtenir un effet sonore continu, sans attaque, avec une oscillation d'amplitude parfois à peine perceptible (dûe à la régularité de l'écho), et de toute façon sans commune mesure avec l'écho d'un son attaqué. Nous voulions alors essayer d'adapter cet effet à la caisse claire...

En effet, l'évitement de la répétitivité constitue une préoccupation compositionnelle importante pour Santiago, ce qu'il m'a confié dès ses premières expérimentations avec FeedItBack[18] : « Je n'aime pas tellement comme un rythme est perceptible avec les boucles... À quelle condition la perception de la périodicité est-elle évitable ? » ; je lui proposais deux réponses partielles : un délai long (au-delà des 2 secondes maximales de sa pédale d'origine), et la variation dynamique des différents paramètres.

18. Une telle préoccupation semble parfaitement légitime, car comme le note J.B. Barrière, « en insistant trop sur la permanence, on produit des tautologies et, dès lors, une forme de conservatisme s'impose à travers un discours épuisant les mêmes formules. » [Barrière, 1992, p. 78] Il poursuit d'ailleurs en notant que la réciproque est vraie, conformément à la théorie de l'information : « Le trop probable engendre l'ennui, mais l'improbable engendre tout aussi sûrement l'endormissement, sinon l'agacement. Une information connue n'en est déjà plus une ; une information incompréhensible n'en sera jamais. ».

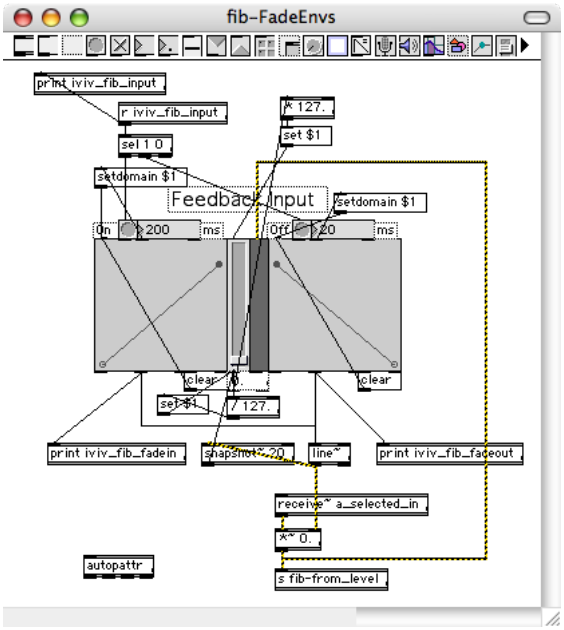


FIG. 7.8 – *Enveloppes de fondu déclenchables*

Nous avons finalement opté pour une modélisation de la solution empirique des fondus de Santiago, présentée figure 7.8, mais de façon *automatisée* : c'est-à-dire sans utiliser les pédales d'expression, afin d'éviter à la fois les difficultés d'écriture et surtout celles d'apprentissage et d'exécution par l'instrumentiste, sachant le nombre limité de répétitions avec le pédalier. Le principe devint donc le suivant : un premier appui sur la pédale étiquetée 1 active le bouton 1 *Delay on/off* qui déclenche l'enveloppe de fondu ouvrant (en envoyant le message `1` à l'objet `r iviv_fib_input`), tandis qu'un deuxième appui sur la pédale 1 désactive le bouton 1 *Delay on/off* qui déclenche l'enveloppe de fondu fermant (en envoyant le message `0` à l'objet `r iviv_fib_input`). Après plusieurs essais, nous avons retenu des enveloppes linéaires avec des durées de 200 ms pour l'enveloppe ouvrante contre 50 ms pour l'enveloppe fermante, car ces valeurs reproduisaient le mieux l'effet désiré à partir d'un son de caisse claire jouée aux balais frottés sur la peau, comme au début de la partition *In Vivo / In Vitro*, dès la première mesure (cf. figure 7.9).

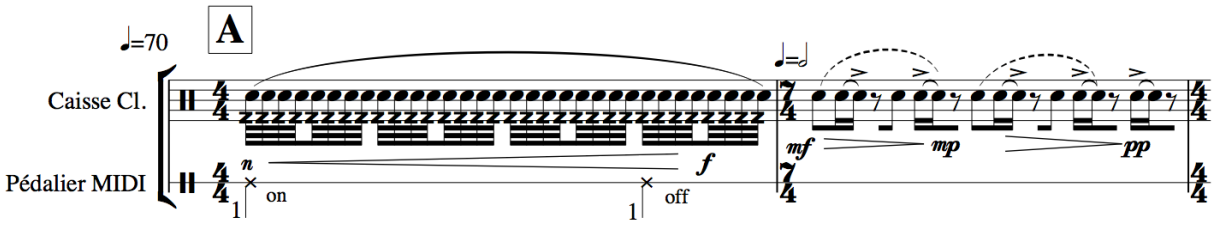
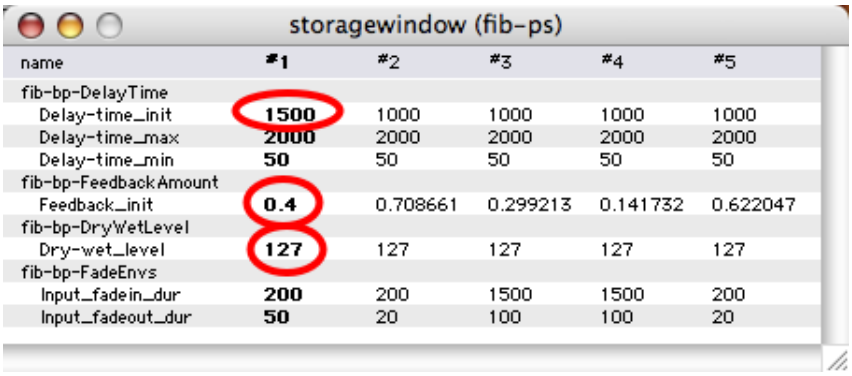


FIG. 7.9 – *In Vivo / In Vitro – Première mesure avec délai*

La première mesure joue donc sur cet effet particulier, en un geste d'introduction énergique, progressif et croissant. Ce geste lance la deuxième mesure, qui expose quant à elle un motif rythmique en forme d'écho « à la main », sans électronique, en répétant le motif cinq fois, en decrescendo et avec des variations d'accentuation. Sur la partition, l'écriture de la deuxième portée dédiée au pédalier MIDI indique clairement l'usage de la pédale 1 en activation sur le premier temps, soit en même temps que le début du jeu sur



name	#1	#2	#3	#4	#5
fib-bp-DelayTime					
Delay-time_init	1500	1000	1000	1000	1000
Delay-time_max	2000	2000	2000	2000	2000
Delay-time_min	50	50	50	50	50
fib-bp-FeedbackAmount					
Feedback_init	0.4	0.708661	0.299213	0.141732	0.622047
fib-bp-DryWetLevel					
Dry-wet_level	127	127	127	127	127
fib-bp-FadeEnvs					
Input_fadein_dur	200	200	1500	1500	200
Input_fadeout_dur	50	20	100	100	20

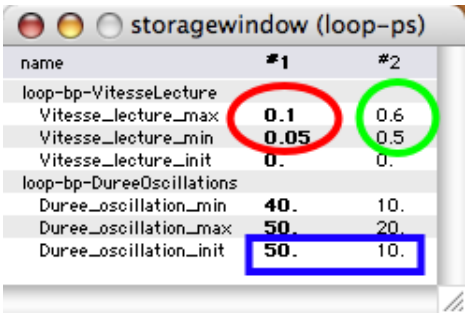
FIG. 7.10 – *In Vivo / In Vitro – Préréglages du délai*

la caisse claire. De cette façon, d'une part l'attaque éventuelle du 1er temps sera masquée par la rampe ouvrante de 200 ms, et d'autre part le son retardé commencera en tuilage, en apparaissant seulement un peu avant le 3e temps de la 1re mesure. En effet, le son entrant dans la ligne à retard sera rediffusé 1,5 seconde plus tard selon l'écriture des préréglages reproduite en figure 7.10 : pour le préréglage #1[19], *Delay-time_init* = 1 500 ms, et *Dry-wet_level* = 127, valeur qui signifie que seul le son retardé sera diffusé.

La désactivation du délai est écrite sur le quatrième temps, c'est-à-dire de façon à ce que l'écho se poursuive pendant les 2e et 3e mesures en décroissant, assurant à la fois un sommet d'intensité avant la décroissance et la possibilité pour l'instrumentiste de ne plus avoir à penser aux pieds pendant le jeu rythmique de la deuxième mesure. Enfin, le dosage de la réinjection, à la fois présent et modéré (*Feedback_init* = 0.4), épaissit la texture retardée sans entretenir l'écho trop longtemps, de façon à laisser la place à ce qui suit, comme les motifs rythmiques de la mesure 2.

7.3.8 Le module « Loop » : pour un fond texturé

L'objectif du module *Loop* découlait de l'idée musicale de disposer d'un *fond sonore* ou d'un *environnement sonore* pour certains moments de la pièce, d'étoffer ainsi le son global en regard de l'aspect ascétique de l'instrumentarium acoustique limité à une caisse claire *solo*, et de faire apparaître différents plans sonores.



name	#1	#2
loop-bp-VitesseLecture		
Vitesse_lecture_max	0.1	0.6
Vitesse_lecture_min	0.05	0.5
Vitesse_lecture_init	0.	0.
loop-bp-DureeOscillations		
Duree_oscillation_min	40.	10.
Duree_oscillation_max	50.	20.
Duree_oscillation_init	50.	10.

FIG. 7.11 – *In Vivo / In Vitro – Préréglages du bouclage*

19. Les autres préréglages ne sont que traces des différents essais pour la composition, seul le préréglage #1 est utilisé par la pédale 1.

Pour obtenir cet effet d'environnement sonore, Santiago a écrit deux préréglages qui ralentissent la vitesse de lecture d'une plage sonore enregistrée en direct : autour d'un facteur 15 pour le ralentissement du premier préréglage (une vitesse de lecture comprise entre 0.1 et 0.05) et autour seulement d'un facteur 2 pour le second (une vitesse entre 0.6 et 0.5), comme le montre la figure 7.11 page précédente. Ces ralentissements de la vitesse de lecture déforment de façon drastique le résultat sonore, produisant souvent des textures avec du souffle et des effets spatiaux, dans le registre grave.

Par ailleurs, Santiago m'a demandé d'éviter que cette vitesse de lecture reste fixe, toujours dans ce souci de garder l'information musicale « vivante », ce qui m'a conduit à lui proposer un oscillateur à basse fréquence (cf. figure 7.12) pour faire varier la vitesse de lecture, entre deux bornes préréglables. Il a ainsi défini la fréquence du Lfo à 50 secondes pour le premier préréglage et à 10 secondes pour le second (cf. *Duree_oscillation_init* figure 7.11 page précédente), renforçant ainsi la différenciation des traitements associés à la pédale 6 *« On Loop1 »* et à la pédale 9 *« On Loop2 »*.

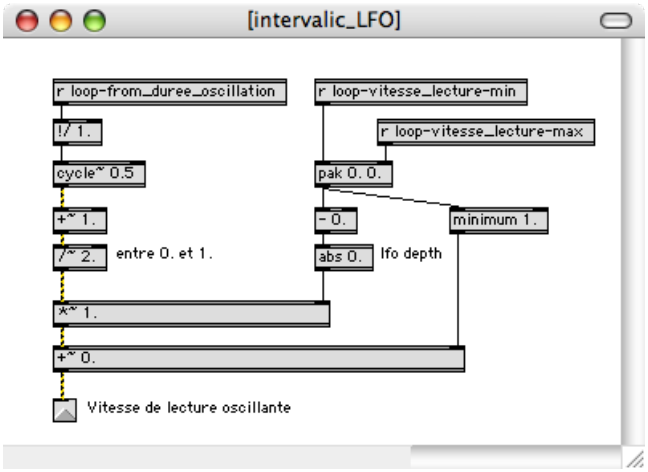


Fig. 7.12 – Lfo *sur la vitesse de lecture*

Avec le son de la caisse claire, les préréglages délivrent un son plus lent et plus grave pour le premier traitement (étirement temporel oscillant entre 10 et 20), avec des artefacts de spatialisation du type réverbération importants, et un son plus « actif » pour le second traitement, avec un comportement qui s'apparenterait à un contrepoint « à la blanche » sur un tom grave, coloré par le changement oscillatoire de la hauteur, parfaitement audible.

Techniquement, le module *Loop* est construit à partir de l'objet `groove~` (placé dans son mode *loop*) plutôt qu'à partir d'un autre objet de lecture comme `play~`, précisément pour ses commodités de bouclage et pour celles de spécification de la vitesse de lecture[20]. Lu par cet objet `groove~`, l'objet `buffer~` nommé *iviv_buf* centralise l'auto-enregistrement pour l'ensemble du logiciel. Ce tampon général ne se trouve pas dans le module *Loop* mais dans le sous-patch `[iviv-RecordBuf]`, et se trouve d'ailleurs partagé par un autre objet `groove~`, dans le module *Fleur*, ainsi que par un objet `waveform~`

20. Le tutoriel Msp *« 14. Playback with loops »* plébiscite clairement cet objet : *« The groove~ object is the most versatile object for playing sound from a buffer~. You can specify the buffer~ to read, the starting point, the playback speed (either forward or backward), and starting and ending points for a repeating loop within the sample. » [Dobrian et al., 2006, p. 130]*

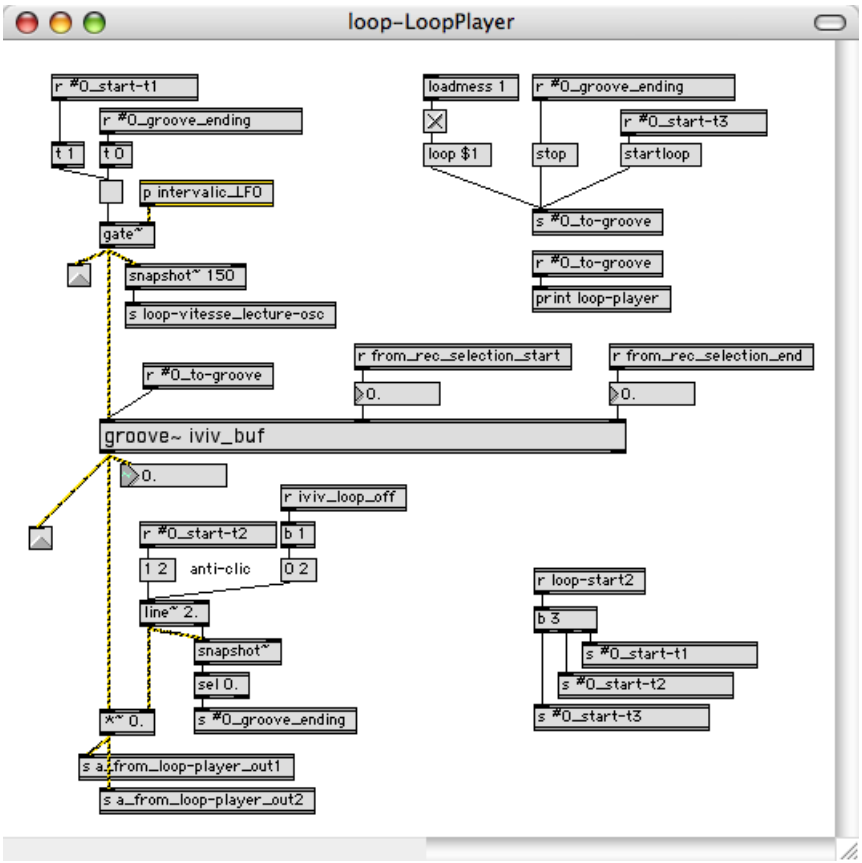


FIG. 7.13 – *Arcanes du bouclage d'Iviv*

dans le bpatcher `[iviv-RecordWaveform]` destiné à visualiser la forme d'onde du son enregistré.

La figure 7.13 montre que l'algorithme de bouclage ne présente pas de difficulté intrinsèque, puisqu'il suffit de transmettre le message `loop 1` à l'objet `groove~`, mais que néanmoins un certain ordre doit être respecté dans le positionnement des différentes variables :

- la variable *loop-start2*, déclenchée après la variable *loop-start1*, déclenche elle-même trois variables de démarrage dans un ordre précis :
  1. d'abord *#0_start-t1* pour positionner la vitesse de lecture,
  2. puis *#0_start-t2* pour déclencher la rampe anti-clic,
  3. et enfin *#0_start-t3* pour autoriser la lecture ;
- l'arrêt effectif de la lecture doit avoir lieu après la fin de la rampe d'anti-clic ; l'astuce employée ici consiste à sélectionner uniquement les valeurs égales à zéro lors des prélèvements automatiques de fin de rampe.

7.3.9 Le module « Fleur » : des saillances orientées

Au-delà du fond texturé obtenu par le module de bouclage ralenti, Santiago souhaitait entendre pour cette pièce des éléments plus saillants de la part de la machine, différemment de ce que propose le module de délai, simplement en contrepoint de la caisse claire. En

baptisant ces éléments « Fleur » sur ses esquisses, et en les écrivant parfois simultanément aux sons *loop*, Santiago suggérait un module dédié et indépendant. Dans la partition, ces éléments constituent en fait des événements, écrits en dialogue ou en contrepoint avec la partie de caisse claire.

Santiago imaginait ces « fleurs » comme donnant l'impression à l'écoute soit de s'ouvrir soit de se refermer – d'où leur nom. Ainsi, ces fleurs s'ouvrent et se referment selon quatre modalités différentes, spécifiées numériquement dans les préréglages du module (cf. figure 7.14) :

1. *« On Fleur »* déclenche une enveloppe exponentielle[21] ascendante de 3 secondes, passant du grave à l'aigu (de 0.2 à 2. pour la vitesse de lecture), et poursuit la lecture en boucle tant qu'un événement « fermant » n'est pas déclenché (comme la pédale 4 *« Off Fleur »* ou bien la pédale 5 *« Rev Fleur »*[22]) ;
2. *« Rev Fleur »* déclenche une enveloppe linéaire descendante de 3 secondes, passant de l'aigu au grave (de 6. à 1. pour la vitesse de lecture), et ferme le niveau d'amplitude (l'enveloppe termine à 0.) ;
3. *« On Zoom »* déclenche une enveloppe relativement brève (1 seconde) qui croît pendant presque toute sa durée et décroît assez brutalement à la fin (un parcours de 1. à 6. puis 1. pour la vitesse de lecture), dans un geste sonore franc d'apparition-disparition ;
4. *« Final Fleur »* déclenche une longue enveloppe décroissante de 16 secondes, passant de l'aigu au grave (de 4. à 0.2 pour la vitesse de lecture) en un geste dramatique ou en tout cas assez conclusif.

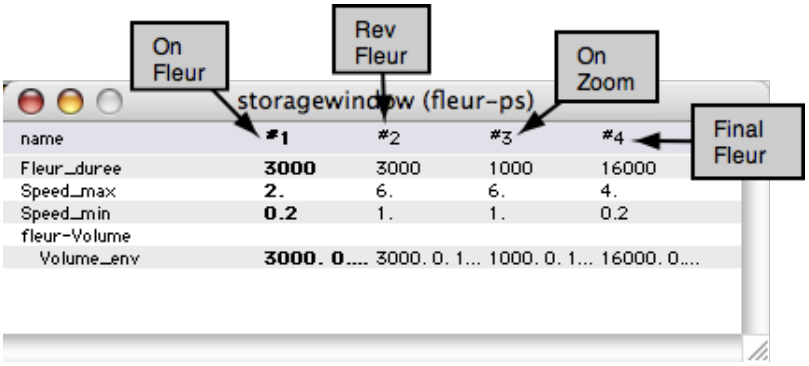


Fig. 7.14 – *In Vivo / In Vitro – Préréglages du module « Fleur »*

Techniquement, l'algorithme principal ressemble beaucoup à celui du module *Loop* présenté précédemment figure 7.13 page ci-contre, fondé sur la relecture du tampon sonore auto-enregistré. L'une des différences réside dans le fait qu'une courbe *unique* définit à la fois les évolutions du paramètre d'amplitude et de celui de la vitesse de lecture, au lieu de deux courbes, ce qui corrèle profondément ces deux paramètres. Cette économie nous a paru tout à fait acceptable, compte tenu de la possibilité de paramétrer les bornes

21. Il s'agit plus exactement d'une courbe affine par morceaux d'inspiration exponentielle, dessinée à partir de l'objet `function`.

22. Pour *« reverse »*.

minimum et maximum de la vitesse de lecture indépendamment pour chaque préréglage. Par exemple pour *« On Fleur »*, à l'écoute, la conjugaison d'un profil dynamique en crescendo accéléré et une augmentation de la vitesse de lecture sur le même profil nous a convaincu, par rapport à la sensation d'ouverture et d'accroissement d'activité sonore recherchée. D'ailleurs, c'est toujours grâce à l'écoute que la spécification numérique des paramètres a pu se faire précisément, comme on peut le voir sur la figure 7.14 page précédente, dans une boucle action-perception favorisée par l'architecture temps réel du logiciel.

7.3.10 Conclusion : deux improvisations impromptues

Le logiciel Iviv a passé l'épreuve du concert[23] avec un fait remarquable : en plus de l'exécution de la pièce *In Vivo / In Vitro*, Santiago et Clarissa se sont proposés pour faire chacun une improvisation avec Iviv, alors que cela n'était pas prévu. Il faut souligner qu'ils ne disposaient pas du pédalier chez eux pour s'exercer avant le concert, et qu'il n'y a eu aucune répétition pour ces improvisations...

Le fait que ces deux improvisations aient eu lieu dans ces conditions et qu'elles se soient déroulées sans encombre montre d'une part la stabilité de ce logiciel, puisqu'il a résisté à un usage d'improvisation pour lequel il n'avait pas été prévu (toute les répétitions s'étaient faites sur la pièce), et d'autre part l'efficacité de l'ergonomie d'Iviv. Cette efficacité de l'ergonomie d'Iviv provient essentiellement du respect de trois contraintes simples proposées par Santiago :

- « un bouton = une fonction » (*i. e.* un déclenchement ou bien une commutation de type marche/arrêt),
- l'utilisation d'une seule banque, pour ne pas avoir à naviguer,
- l'absence d'utilisation des pédales progressives (la pédale A n'est rattachée à aucun paramètre, tout au plus la pédale B peut servir à fermer le volume général, si besoin).

Par ailleurs, la qualité sonore de ces deux improvisations tend à montrer que la conception modulaire variée – avec des idées de *plans* différents – est pertinente musicalement. En effet, dans l'improvisation de Clarissa au vibraphone comme dans celle de Santiago à la guitare, les boucles ralenties et oscillantes produisent des arrière-plans faciles à produire pour l'instrumentiste tout en restant le plus souvent dignes d'intérêt, alors que les fleurs permettent de produire des événements sonores qui ressortent clairement.

23. Le 2 février 2007, cf. section 4.3.2 page 93.

Chapitre 8

a2m, Loterie, Emzed et Nappe / Mauricio Meza

Résumé du chapitre : Ce chapitre montre d'abord en quoi la pièce *Woes-war-sollichwer-den* de Mauricio Meza est un projet ambitieux, en présentant la multiplicité des ressorts techniques et esthétiques invoqués. Ensuite, en tenant compte de l'inachèvement du projet, ce chapitre analyse sélectivement certains aspects des quatre logiciels programmés au cours de cette collaboration : a2m, Loterie, Emzed et Nappe.

8.1 Woes-war-sollichwer-den

8.1.1 Un projet ambitieux

Avec Mauricio, compositeur, nous nous sommes rencontrés à l'université Paris 8, après ma présentation sur le logiciel Mimi contrôlé par le pédalier multiple et alimenté par le capteur intra-bec pour la clarinette. Il a accepté de participer à une collaboration qui intégrerait ces trois éléments, en vue d'un concert avec une pièce qu'il écrirait.

En première approche, la complexité de ce projet se laisse deviner par l'aspect quantitatif : à l'importance de l'annexe musicale qui lui est consacrée par rapport aux autres partitions (45 pages, cf. annexe E.2 page 297), et au nombre de programmes développés (4 logiciels, soit plus que pour les autres collaborations). En réalité, il faut ajouter à cela l'intégration tout azimut d'esthétiques diverses : entre œuvre écrite et œuvre ouverte, détermination et indétermination, temps réel et temps différé, aléatoirité informatique et improvisation humaine, interaction logicielle et participation du public. Toutefois, de cette intégration touffue peuvent ressortir prioritairement deux objectifs : le projet artistique sous-jacent vise une œuvre à la fois *immersive* et *participative*.

L'aspect *immersif* doit résulter de la conjonction de plusieurs facteurs : la présence de l'instrumentiste sur scène, l'improvisation, l'amplification sonore, la progression globalement crescendo de la pièce (le climax est écrit à la section E juste avant la coda, cf. conduite page 302), le processus d'« entropie » croissante (de plus en plus de processus se superposent au fil de la pièce), et la participation du public.

L'aspect *participatif* reste plus ponctuel : seule la section C centrale, qui doit durer 4 à 5 minutes, repose sur un jeu oral de type « question/réponse ». Les réponses sont enregistrées par l'ordinateur et rejouées de façon déformée par l'instrumentiste à travers le pédalier pendant le jeu même, puis elles sont réutilisées par les logiciels jusqu'à la fin de la pièce. Cependant, malgré son usage interactif limité à une seule section, cet aspect participatif compte pour Mauricio au point d'inférer le nom de la pièce – *Woes-war-sollichwer-den* (Mauricio s'en explique dans la notice, annexe E.2 page 297).

8.1.2 Une méta-partition complexe

L'ensemble des documents écrits nécessaires à l'interprétation de la pièce réunit 45 pages, réparties sur 4 documents différents selon le tableau 8.1.

Notice	4 pages	Notice explicative sur l'interprétation et la construction de la pièce.
Conduite	10 pages	Synopsis individuels des 5 sections, des 4 transitions et de la coda.
Partition	19 pages	Les 9 « modules » pour la clarinette, joués dans les sections A, B, et les transitions t_{AB}, t_{BC}, t_{CD}, t_{DE}.
Panneaux	12 pages	Réservoirs de mots pour la section C.

Tab. 8.1 – *Récapitulatif des documents de Woes-war-sollichwer-den*

Parmi ces documents, la notice et la conduite ne sont réservés qu'à la préparation de l'interprétation, mais la partition et les panneaux restent pour le concert (soit 31 pages).

8.1.3 Quatre logiciels

La notice rédigée par Mauricio, très complète et jointe en annexe, dispense ce chapitre d'une analyse plus approfondie de cette pièce, d'autant qu'elle n'a pas encore pu être exécutée en concert faute de temps (cf. section 8.4 page 178). Ce chapitre se concentrera donc essentiellement sur les quatre logiciels et leur rôle dans le projet : a2m, Loterie, Emzed, et Nappe, dont le tableau 8.2 donne un premier aperçu.

composition	a2m		Génération de matériau précompositionnel
pré-concert	Loterie		Tirage au sort juste avant A
A	Emzed	5'00	Interprétation, clarinette enregistrée par le logiciel
t_{AB}	Emzed	3'00	Improvisation, clarinette et pédalier, Emzed obligé
B	Emzed	5'30	Interprétation, clarinette et pédalier
t_{BC}	Emzed	3'00	Improvisation, clarinette et pédalier, Emzed libre
C	E.+Nappe	4'30	Participation orale, enregistrée par le logiciel
t_{CD}	E.+Nappe	3'00	Improvisation, clarinette et pédalier
D	E.+Nappe	5'00	Improvisation, clarinette et pédalier
t_{DE}	E.+Nappe	2'00	Improvisation, clarinette et pédalier
E	Nappe	3'00	Tacet, autonomie du logiciel vers un paroxysme
Coda	Nappe	10"	Arrêt automatique sur détection d'une série de gestes agressifs improvisés à la clarinette

TAB. 8.2 – *Place musicale des 4 logiciels a2m, Loterie, Emzed et Nappe au cours de la pièce*

Les minutages indiqués dans ce tableau étant nécessairement approximatifs, on peut conclure que la pièce dure environ 35 minutes, avec une marge de plus ou moins 5 minutes. Par ailleurs, cette durée conséquente peut justifier pour une large part le degré de complexité mis en œuvre dans ce projet, l'échafaudage d'une architecture structurée qui se renouvelle dans « la grande forme », ainsi que son caractère inachevé.

Enfin, il faut distinguer les deux logiciels *performatifs*[1] que sont Emzed et Nappe des deux autres logiciels, a2m relevant de la CAO, et Loterie du petit utilitaire de tirage au sort.

8.2 a2m : un traducteur précompositionnel

Le logiciel a2m tient une place particulière dans ma thèse, parce qu'il figure comme le logiciel le plus proche du temps différé relativement aux autres logiciels développés au cours de ce doctorat (cf. figure 3.6 page 79). Il s'agit fondamentalement de *traduire* une dimension acoustique sur une dimension solfégique.

1. Cf. section 5.4 page 109.

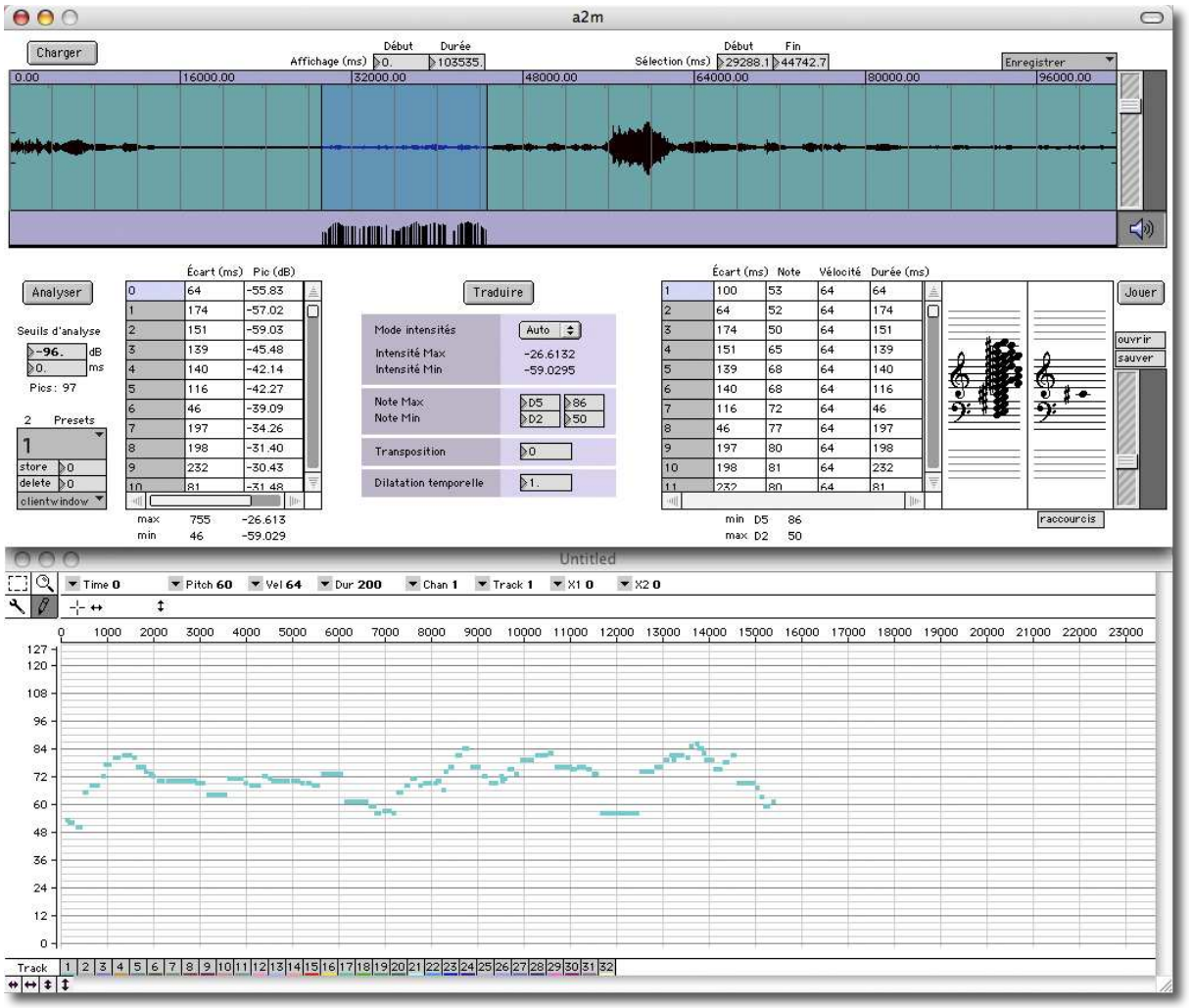


FIG. 8.1 – *Capture de l'interface visuelle de a2m*

Mauricio a utilisé a2m pour la partition de clarinette de *Woes-war-sollichwer-den*, et plus tard dans *Cinq miniatures micro-tonales* pour clavier MIDI, puis dans *Tamborita de lejos* pour saxophone soprano.

8.2.1 Un artisanat du détournement

En effet, a2m répond au souhait de Mauricio de produire du matériau précompositionnel – ici des séquences monophoniques MIDI – à partir de caractéristiques morphologiques d'un fichier audio (cf. annexe F page 327). Déjà, la fonction principale de a2m – la traduction – transparaît dans la structure de son nom, inspiré de certaines commandes unix de traduction[2] où le « 2 » central signifie *« to »*, c'est-à-dire « vers » : le nom « a2m » désigne ici une traduction *« audio to material »*, c'est-à-dire une traduction « audio vers matériel » (précompositionnel). Voici d'ailleurs un extrait du cahier des charges :

2. On peut citer par exemple `a2ps` (pour *« ascii to postscript »*), `csv2latex`, `latex2html`, ou `dos2unix`.

> *Vers un moteur de traduction* — Nous souhaitons développer un programme de génération de matériau précompositionnel à travers un algorithme de mise en parallèle morphologique. Il s'agit donc d'analyser des « saillances » morphologiques dans un fichier son et de les repérer dans le temps afin d'y faire correspondre des couples « date relative » / « hauteur MIDI », dans un format compatible avec l'objet `detonate` (durée en millisecondes ; MIDI *pitch*).

Comme caractéristique acoustique à traduire, nous avons retenu la détection des pics d'amplitude, car les pics d'amplitude représentent des *événements* acoustiques auxquels peuvent correspondre de façon assez intuitive les *événements* musicaux que sont les notes. Techniquement, l'amplitude de chaque pic est mesurée par l'objet `analyzer~` de Tristan Jehan[3], puis traduite en hauteur MIDI (de 0 à 127), en reprenant les durées enregistrées entre les pics détectés.

Mauricio a utilisé a2m pour écrire la partition de clarinette de *Woes-war-sollichwer-den* (cf. annexe E.2 page 308) : en particulier, il a analysé et traduit des extraits du *Dialogue de l'ombre double*[4]. La référence à cette pièce peut se lire au moins à deux niveaux : celui de la relation intime entre la clarinette et son double électroacoustique, ici en temps réel (avec Emzed), ou celui du concept de dialogue, extrapolé jusqu'à l'aspect participatif et verbal évoqué précédemment (cf. section 8.1.1), avec toutefois pour cet aspect une influence plus directe de l'usage de la théâtralité dans les œuvres de Vinko Globokar.

Quoi qu'il en soit, Mauricio opère un rapprochement entre *Woes-war-sollichwer-den* et le *Dialogue de l'ombre double* par un détournement d'extraits métamorphosés avec a2m... De fait, il ne s'agit ni d'une simple transposition, ni d'une homothétie quelconque (sur les hauteurs ou sur les durées par exemple), mais bien d'un changement de structure, si important que les séquences d'origine deviennent immanquablement méconnaissables.

8.2.2 Analyse des pics d'amplitude

Après le chargement du fichier audio, un clic sur l'objet `button` annoté « Analyser » déclenche la lecture de la sélection audio (ajustable dans l'objet `waveform~`) ainsi que l'analyse de ce signal en temps réel (technique) à travers l'objet `analyzer~`[5].

La notion de pic d'amplitude prend ici une définition rudimentaire mais pertinente par rapport à notre projet : une valeur d'amplitude renvoyée par `analyzer~` est un pic d'amplitude si elle constitue un sommet local d'ordre 2 parmi le flux des valeurs d'amplitude, c'est-à-dire si elle est supérieure à ses voisins immédiats et qu'eux-mêmes sont

3. Ci-reproduits les droits de cet objet, tels qu'ils apparaissent dans *Max/MSP* :
`Analyzer~ object version 1.3.1 by Tristan Jehan`
`copyright © 2001 Massachusetts Institute of Technology`
`Pitch tracker based on Miller Puckette's fiddle`
`copyright © 1997-1999 Music Department UCSD.`

4. Une des pièces mythiques de Pierre Boulez, pour clarinette et bande, réalisée à l'IRCAM en 1984 et 1985 avec Andrew Gerso, et créée par Alain Damiens à la clarinette.

5. L'objet `bonk~` de Miller Puckette propose cette fonction de détection des pics d'amplitude, mais il n'était pas disponible sur notre plate-forme (macintel) au moment du développement, donc nous avons travaillé avec l'objet `analyzer~`.

supérieurs à leurs autres voisins. Une seule inégalité stricte dans l'expression suffit à éviter les « plateaux » (signal constant). Ainsi, le test suivant applique la définition précédente au flux des valeurs d'amplitude :

```
if (($f1 <= $f2) && ($f2 < $f3) && ($f3 >= $f4) && ($f4 >= $f5))
 then $f3
```

Cette définition simple détecte effectivement les pics qui nous intéressent, parce qu'elle ne suit pas le signal audionumérique à l'échelle de l'échantillon, ce qui donnerait probablement beaucoup trop de sommets locaux, mais qu'elle suit le flux des données produites par `analyzer~` à une période bien supérieure : 11.6 ms, correspondant aux 512 échantillons du pas d'analyse (*hop size*). Notre fenêtre d'analyse couvre finalement 2 560 échantillons au total, soit 58 ms à 44.1 kHz, une durée qui convient aux sons de clarinette analysés.

Par ailleurs, l'utilisation de l'objet `analyzer~` possède un autre intérêt, car les trois autres types de données qu'il produit sont déjà disponibles pour de futurs développements, en élargissant les possibilités de traduction à partir de la hauteur du signal (*pitch*), de la brillance (*brightness*), ou encore de l'inharmonicité (*noisiness*).

Enfin, deux seuils permettent de réduire le nombre de résultats donnés par la phase d'analyse des pics d'amplitude : un premier seuil sur leur niveau minimum (en dB) et un second sur l'écart temporel minimum entre deux pics consécutifs (en ms).

8.2.3 Paramétrage de la traduction

On a vu que l'analyse des pics d'amplitude constitue le cœur du moteur de traduction – soit la partie automatique du travail. Cependant, le résultat de cette analyse nécessite la plupart du temps un travail supplémentaire pour devenir exploitable – soit la partie humaine, incarnée par les possibilités de paramétrage proposées par a2m.

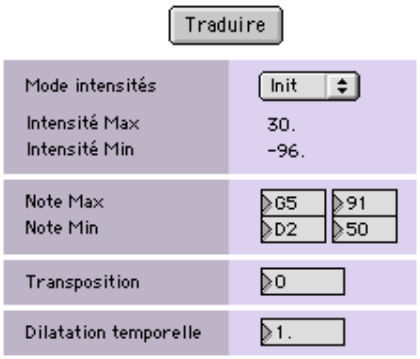


FIG. 8.2 – *Interface de traduction de a2m*

La fonction de traduction de la liste des pics d'amplitude vers la liste des hauteurs MIDI est assurée par l'objet `zmap`, qui fait correspondre de façon linéaire un intervalle fini en entrée avec un intervalle fini en sortie :

Soient : i l'intensité d'un pic d'amplitude définie pour des nombres « flottants » sur un intervalle $D_i = [-96., 30.]$ au plus large, n la hauteur d'une note Midi définie pour des

nombres entiers sur un intervalle $D_n = [0, 127]$ au plus large, et t une fonction de traduction qui associe à tout i de D_i une unique note n appartenant à D_n telle que $n = t(i)$. Les quatre bornes i_{min}, i_{max}, n_{min} et n_{max} permettent de régler la mise en correspondance de l'intensité des pics d'amplitude avec la hauteur des notes MIDI, c'est-à-dire de régler la fonction de traduction elle-même, telle que :

$$t : \left| \begin{array}{ccc} [i_{min}, i_{max}] & \longrightarrow & [n_{min}, n_{max}] \\ i & \longmapsto & n \end{array} \right.$$

Pour l'intervalle d'entrée $[i_{min}, i_{max}]$, a2m propose trois modes de spécification, enregistrables dans le fichier de préréglages :

- Auto (0), qui récupère automatiquement les valeurs d'intensité minimum et maximum parmi l'ensemble des pics retenus par la phase d'analyse ; ce mode minimise automatiquement l'intervalle d'entrée pour la traduction ;
- Init (1), qui initialise les deux bornes à leur valeur limite, soit -96 dB et +30 dB ;
- Perso (2), qui permet de personnaliser la valeur de chaque borne, à l'intérieur du domaine $[-96., 30.]$ toutefois.

Par ailleurs, si une valeur d'entrée sort de l'intervalle D_i, alors l'objet `zmap` ramène n à l'intérieur de D_n (à n_{min} ou n_{max}).

Pour l'intervalle de sortie $[n_{min}, n_{max}]$, a2m propose deux représentations : les hauteurs sont spécifiables à la fois sous forme de notation anglo-saxonne (plus intuitive, cf. figure 8.2 page ci-contre), et sous forme de numéro MIDI (celle qui est effectivement enregistrée dans le fichier de préréglages).

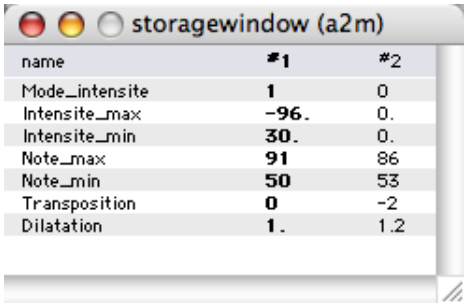


FIG. 8.3 – *Paramètres de a2m*

Au total, a2m comporte sept paramètres enregistrables, présentés sur la figure 8.3), ainsi que plusieurs commodités pour la traduction :

- trois modes de spécification pour l'intervalle de l'intensité des pics en entrée ;
- deux bornes minimum et maximum pour l'intensité des pics en entrée ;
- deux bornes minimum et maximum pour l'ambitus des notes en sortie ;
- une valeur de transposition, en demi-tons ;
- un facteur de dilatation temporelle ;
- un panneau de gestion des préréglages ;
- une visualisation synthétique de toutes les notes produites (`nslider`) ;
- la possibilité de visualiser et d'écouter la séquence MIDI produite, avec l'objet `detonate` (cf. partie inférieure de la figure 8.1 page 172).

8.2.4 Présentations du résultat Midi

Pour que la partie humaine soit efficace avec le logiciel, il faut permettre des corrections « au jugé » sur le résultat, si possible en le présentant sous plusieurs formes ; il s'agit d'exploiter en quelque sorte le « savoir » des sensations et des émotions dans le cycle perception/action. À la suite de ce principe, a2m peut présenter à l'utilisateur deux formes visuelles et une forme sonore du résultat Midi :

- l'agrégat synthétique des différentes notes sur quatre portées (objet `nslider`) qui donne immédiatement une vision d'ensemble de la répartition des hauteurs verticalement ;
- la fenêtre en rouleau de piano de l'objet `detonate`, qui montre la ligne mélodique déployée dans le temps horizontalement ;
- le bouton « Jouer », qui permet d'écouter la séquence Midi[6], avec une onde carrée pour s'approcher grossièrement du timbre de la clarinette (objet `rect~`).

* * *

Le logiciel a2m a permis à Mauricio de générer du matériau précompositionnel – des séquences mélodiques Midi – à partir d'extraits audionumériques du *Dialogue de l'ombre double* de Pierre Boulez, en traduisant une dimension acoustique vers une dimension solfégique. Cette traduction paramétrable métamorphose ainsi littéralement ces extraits détournés, qui deviennent méconnaissables.

8.3 Loterie : une ouverture pour l'œuvre ouverte

8.3.1 Une ouverture garantie

Le patch Loterie intervient dans la pièce *Woes-war-sollichwer-den* comme une garantie du renouvellement du parcours entre les modules. À nouveau, Pierre Boulez avait déjà exploré une écriture fragmentaire confiant le choix du parcours à l'interprète (p. ex. dans les *Domaines*) ; or, dans les faits, les interprètes choisissent souvent un parcours définitif, et un parcours fini rapidement par s'imposer et faire école pour plusieurs générations d'interprètes, supprimant les autres parcours possibles, et par là même l'intérêt de ce type d'œuvres ouvertes, qui se referment de fait.

8.3.2 Une improvisation encadrée

Ici, le tirage au sort doit être lancé avant chaque interprétation, et assure ainsi que l'ouverture de la pièce est bien préservée. Mauricio est allé encore un peu plus loin, dans

6. Incidemment, on peut remarquer que cette lecture a lieu en temps réel technique (en accord avec le taux d'échantillonnage), comme la phase d'analyse, mais que la traduction se fait « plus vite que le son » pour reprendre l'expression de Miller Puckette, c'est-à-dire *faster than sound*, comme le système « FTS » de l'Ircam.

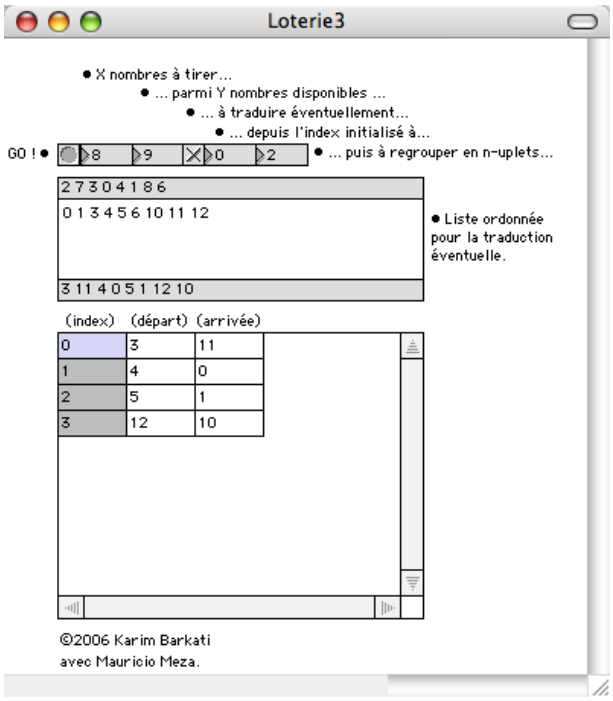


FIG. 8.4 – *Capture de l'interface visuelle de Loterie*

le sens où ce tirage aléatoire détermine non pas un parcours mais uniquement l'introduction et la conclusion des quatre transitions, le reste devant être improvisé (cf. notice page 297). De cette façon, les deux modules tirés au hasard orientent nécessairement l'improvisation, dont l'alpha et l'oméga viennent d'être déterminés. On peut considérer que cette détermination partielle et tirée au sort donne une légitimité particulière au recours à l'improvisation au sein d'une œuvre écrite, renouvelant le concept d'œuvre ouverte...

8.4 De Mimi à Emzed : discussion acoustique

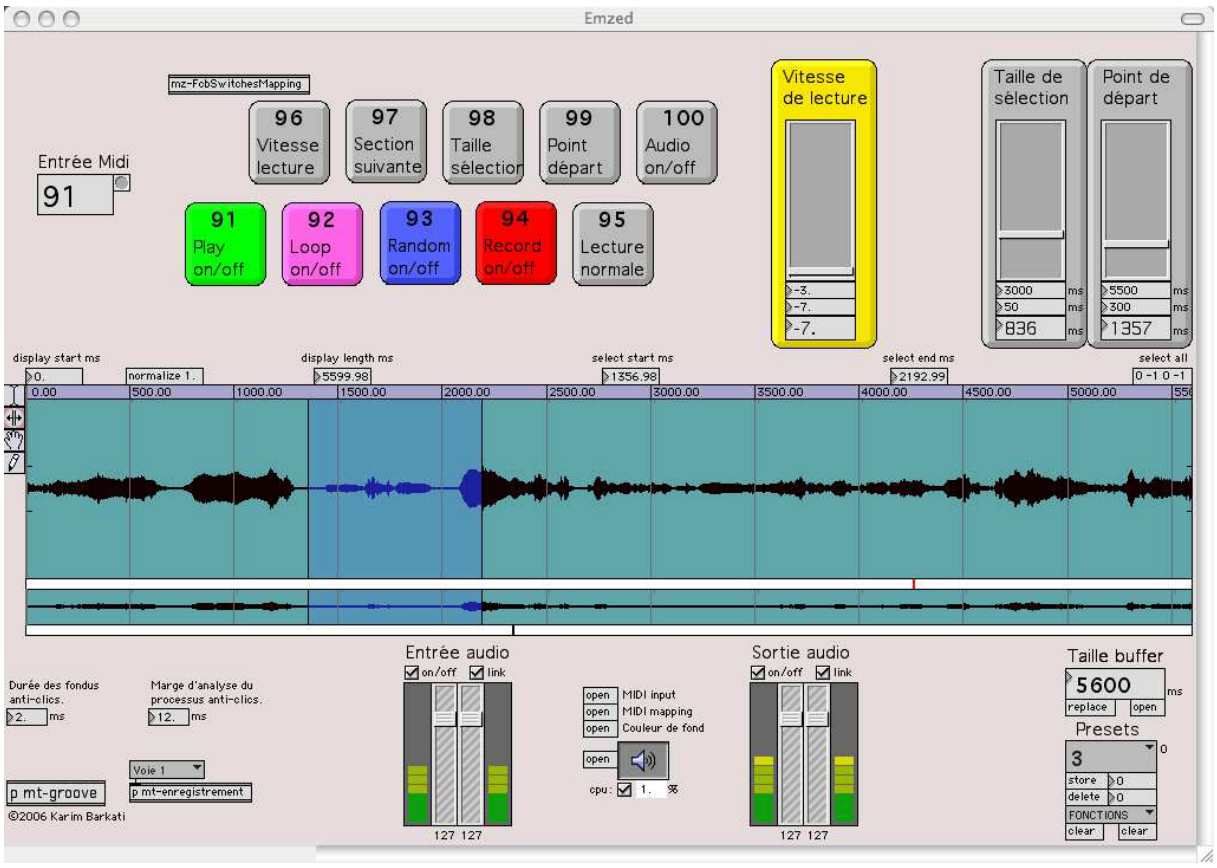


FIG. 8.5 – *Capture de l'interface visuelle de Emzed*

Le projet de départ devait directement intégrer le logiciel Mimi (cf. section 6.2 page 120), mais les premiers tests avec la clarinette sonnaient fort mal : en effet, des clics survenaient systématiquement dans la lecture. D'une part, ce problème nous a d'abord beaucoup surpris, puisque nous n'avions jamais constaté ces effets avec l'alto, et d'autre part la tentative de résolution de ce problème – baptisée Emzed – s'est révélée terriblement chronophage, compromettant malheureusement les chances du projet d'être prêt à temps pour l'échéance du concert.

Cette section examine ainsi les raisons susceptibles d'expliquer une telle différence dans le comportement d'un même logiciel selon qu'il se trouve confronté à l'alto ou à la clarinette.

8.4.1 Un premier examen algorithmique

Pourquoi *le même logiciel* sonne bien lorsqu'il est appliqué à l'alto et sonne mal lorsqu'il est appliqué à la clarinette ? De quelle partie du logiciel viennent ces clics ? Ces questions, formulées de cette manière, nous ont d'abord aiguillés sur une piste biaisée : un algorithme de passage par zéro (*zero-crossing*).

Sur le principe, il s'agit de corriger la position précise des deux bornes de la sélection de lecture dans le tampon, qu'elle ait été tirée au hasard ou directement spécifiée par l'utilisateur. Techniquement, ce sous-patch[7] de correction des sélections par passage à zéro (cf. figure 8.6 page 181) parcourt le tampon échantillon par échantillon, à la recherche du passage par zéro le plus proche de la date proposée, à l'intérieur de la sélection, à l'aide des objets `uzi`, `counter` et `peek~` ; pour ce faire, il utilise conjointement trois informations – la date proposée, la direction de recherche et le nom du tampon. À l'échelle de l'échantillon, un passage par zéro peut être repéré simplement, de deux façons différentes : soit l'échantillon vaut exactement zéro, soit il change de signe par rapport à l'échantillon précédent. Mais ces simples corrections n'ont pas suffi.

Comme des clics subsistaient, d'autres restrictions ont été mises en place dans l'espoir de les éradiquer, dont :

- trois zéros consécutifs au lieu d'un seul, pour les échantillons nuls exactement,
- des transitions symétriques (*i. e.* croissantes ou bien décroissantes autour de zéro),
- le blocage de la prise en compte des changements de valeur durant la lecture d'une sélection,
- un dispositif d'impression pour le débogage.

Nous avons d'ailleurs pu vérifier le placement exact des bornes de la sélection sur des points de passage par zéro en agrandissant la représentation de la forme d'onde (avec l'objet `waveform~`), mais à l'audition, avec des sons de clarinette entretenus, des clics continuaient d'apparaître...

8.4.2 Un second examen acoustique

Ces échecs nous ont contraint à reformuler nos questions autrement : pourquoi *le son de la clarinette* génère-t-il des clics dans ce logiciel, là où celui de l'alto n'en génère pas ? Qu'est-ce qui, dans le son de la clarinette, produit ces clics ? Cette reformulation invite naturellement à un autre type de réponse.

Rappelons que Emzed (et Mimi) se basent essentiellement sur de l'auto-échantillonnage, monophonique, avec deux fonctions principales : l'enregistrement et la relecture, dans un même tampon. L'unicité de ce tampon, associée au fait que la vitesse de lecture est variable, implique que le pointeur de lecture peut parfois croiser le pointeur d'enregistrement, donc que deux échantillons lus consécutivement peuvent appartenir à deux enregistrements différents, c'est-à-dire potentiellement générer des clics. Mais ceci n'explique pas pourquoi ces clics étaient inaudibles avec le son de l'alto... De même, pourquoi le forçage du passage par zéro n'a-t-il pas résolu les clics de la clarinette ? Il faut en déduire que quelque chose *dans le son*, une propriété acoustique, doit nécessairement intervenir dans l'explication.

La lecture d'Emzed procède en quelque sorte par « collage numérique » dynamique, par juxtaposition des sélections au cours du temps : en effet, lorsque le pointeur de lecture atteint la fin de la sélection, il consulte la valeur du point de départ pour s'y rendre immédiatement, et consulte la taille de sélection pour lire cette nouvelle sélection[8]. Or

7. Le patch `[kb-nearest_zc]` est utilisé dans le bpatcher `[mt-Pedal]`, pour corriger les valeurs des deux curseurs du point de départ et de la taille de sélection.

d'une part, le point de départ n'a pas de rapport avec le point de fin, puisque l'enregistrement est pris sur le vif et que la sélection se fait indépendamment de l'enregistrement, et d'autre part ce point de départ se trouve susceptible de changer, soit par tirage aléatoire si la fonction est active, soit par contrôle de l'utilisateur via la pédale A.

Nous pensons que c'est cette absence de rapport entre le point de fin et le nouveau le point de départ qui pose des problèmes lors de l'enchaînement de deux sélections : la juxtaposition génère un clic à l'audition pour le son de la clarinette, même si l'on force le passage par zéro à l'échelle de l'échantillon. Ce doit donc être à une échelle supérieure, de l'ordre de la milliseconde, que le signal subit un saut lors de la transition, un saut perceptible avec un son de clarinette mais pas avec un son d'alto...

En observant la forme d'onde de sons entretenus produits avec ces deux instruments, à différentes échelles d'agrandissement, on peut remarquer un aspect plus « massif » pour le son de la clarinette par rapport à un aspect plus « volubile » pour le son de l'alto. Nous faisons l'hypothèse que ces différences découlent de la différence de vibrato en amplitude et en fréquence, plus important et plus rapide pour l'alto (avec nos instrumentistes), et de la différence de stabilité fréquentielle et spectrale au cours du temps, plus forte pour la clarinette. La stabilité du son de la clarinette résulte probablement elle-même du mode de production sonore relativement *fixe* au regard de la combinaison bec / anche / pression d'air, à comparer avec le mode de production sonore relativement *labile* de l'alto au regard de la combinaison corde / colophane / vitesse et pression d'archet. Autrement dit : au cours de la lecture, si l'on considère *a posteriori* une fenêtre temporelle autour du point de transition, on peut supposer que la discontinuité acoustique que constitue ce point aura potentiellement un impact important sur un signal assez stable et un impact moindre sur un signal plus variable.

Ainsi, les différences observées dans la morphologie acoustique nous suggèrent de conjecturer une « résistance acoustique » au processus de la lecture par juxtaposition, ou encore une « raideur acoustique », cette « raideur » étant incarnée par la persistance des clics pour certains types de signaux seulement, et de penser cette raideur comme le revers de la stabilité acoustique de ces signaux.

* * *

Emzed diffère relativement peu de Mimi et représente une tentative d'éradication algorithmique des clics audibles avec le son de la clarinette mais pas avec le son de l'alto. En fin de compte, il s'est avéré que ces clics proviennent d'une « raideur acoustique » de la clarinette, qui ne supporte pas la lecture par juxtaposition.

8. Pour tenter de réduire encore la quantité de clics, tout changement de ces deux valeurs a d'ailleurs été rendu inopérant pendant la durée de la lecture de la sélection.

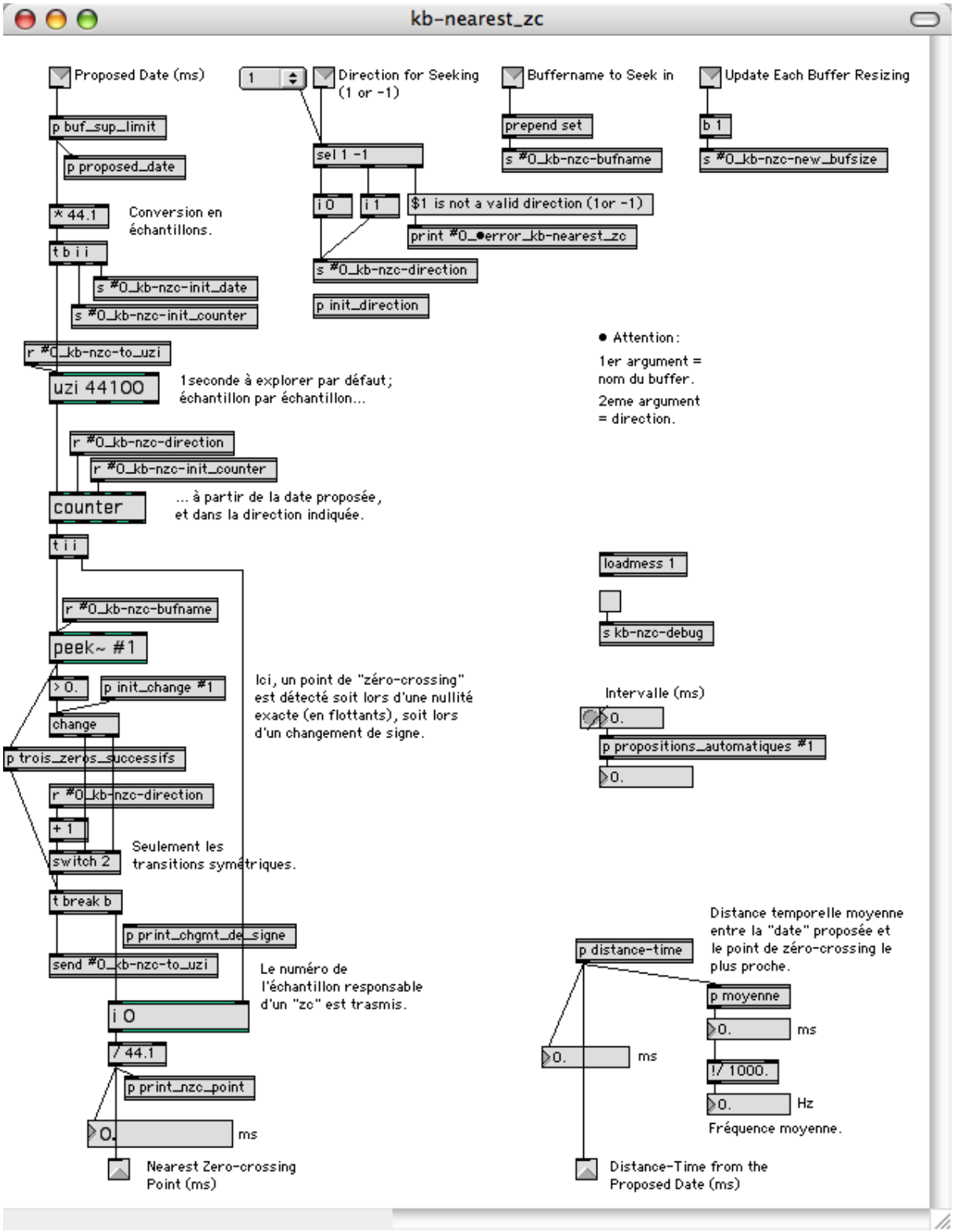


FIG. 8.6 – *Algorithme de passage par zéro de Emzed*

8.5 Nappe : résumé d'une complexité logicielle

Le logiciel Nappe est à l'image de la pièce *Woes-war-sollichwer-den* : complexe. En effet, Nappe comporte de nombreux traitements sonores, certains étant automatiques ou aléatoires, et d'autres contrôlables depuis le pédalier. Compte tenu du fait que Nappe n'est pas tout à fait achevé, cette section se limite à présenter sa complexité.

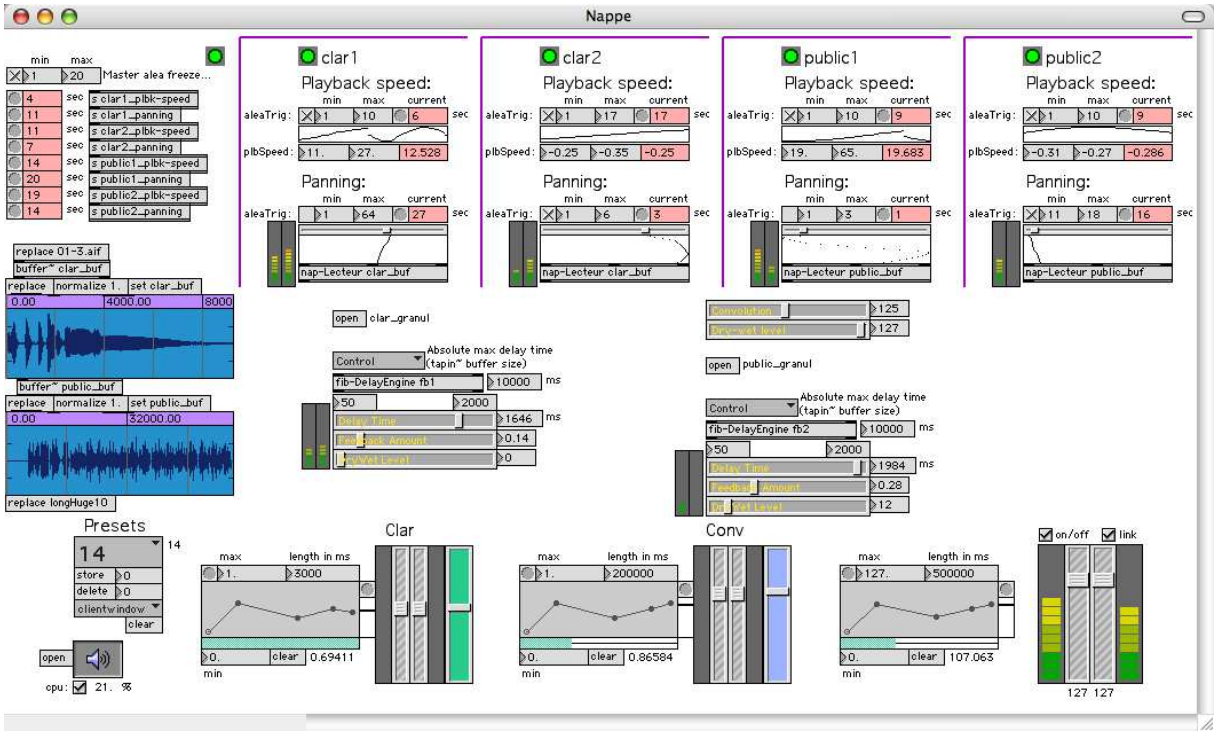


FIG. 8.7 – *Capture de l'interface visuelle de Nappe*

8.5.1 De nombreux traitements

On peut ainsi dénombrer :

- un module global pour le paramétrage de l'aléatoire ;
- deux tampons audionumériques (un pour la clarinette et un pour le public) ;
- quatre lecteurs : un à l'endroit et un à l'envers pour les deux tampons ;
- quatre OBFI[9] semi-aléatoires bornés pour varier la vitesse de lecture ;
- quatre OBFI semi-aléatoires bornés pour varier la panoramisation ;
- un moteur de convolution source-filtre[10] débrayable (la source étant un mixage des deux lectures variées du tampon du public, et le filtre étant un mixage des deux lectures variées du tampon de la clarinette) ;
- deux granulateurs[11] débrayables (un sur le mixage des deux lectures variées du tampon de la clarinette, l'autre sur le résultat de la convolution) ;

9. Oscillateurs basse fréquence *intermittents* : un paramètre aléatoire active ou désactive l'oscillateur à chaque nouveau cycle, lui-même tiré au sort entre deux bornes.

10. Adapté depuis le patch d'exemple `[convolution-wkshop2003.pat]` de Les et Xoaz.

– deux modules de délai[12] débrayables (en sortie des deux granulateurs) ;
– trois longues enveloppes d'amplitude déclenchables par le pédalier (à travers les préréglages) : une pour la ligne de la clarinette, une pour la ligne de la convolution, et une pour la sortie générale ;
– un module de gestion des préréglages (`[kb-Storage-mng]`).

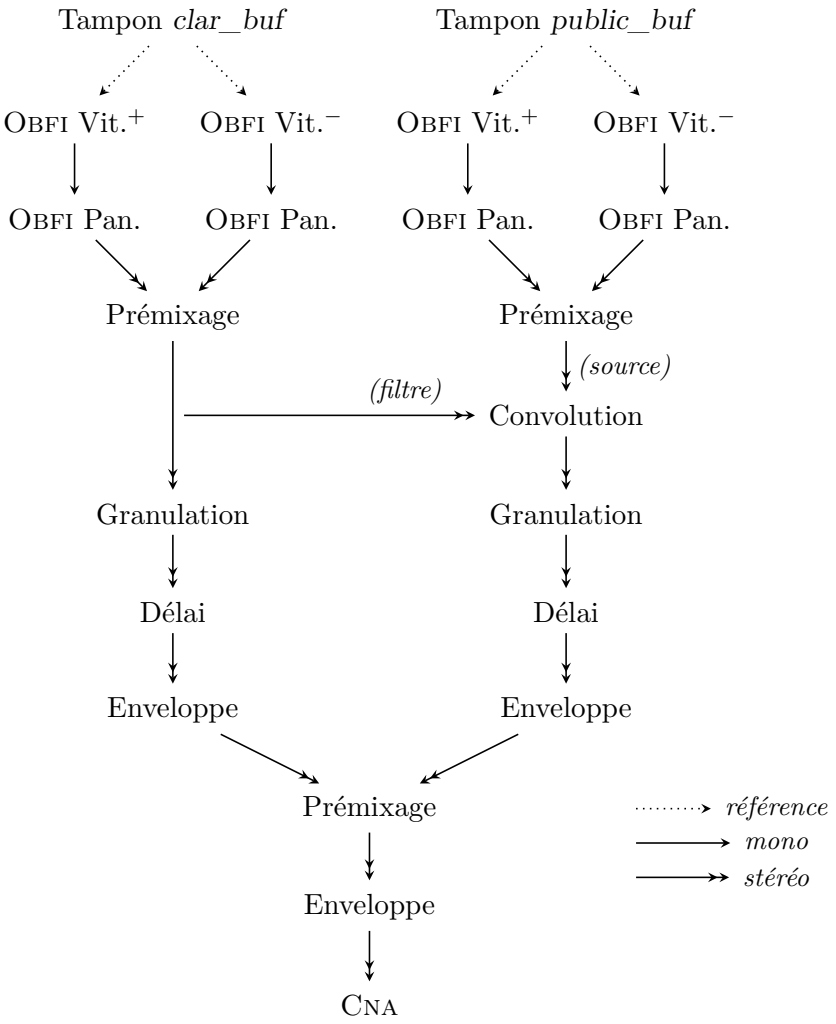


Fig. 8.8 – *Diagramme fonctionnel de Nappe*

Par ailleurs, un autre facteur de complexité intervient dans Nappe : l'interfaçage avec Emzed qui doit fonctionner en parallèle. Car d'une part Emzed doit fournir les enregistrements audio pour les deux tampons *clar_buf* et *public_buf*, et d'autre part Emzed sert de passerelle pour le contrôle au pédalier pour les deux logiciels, de façon à éviter les conflits potentiels avec deux interfaces redondantes en parallèle.

11. Utilisant l'objet `munger~` de Dan Trueman (du *Computer Music Center* de l'université Columbia, et l'interface modifiée par Manuel Poletti.

12. Reprenant le moteur de FeedItBack, cf. section 7.2 page 150.

8.5.2 Quatre-vingt-dix-neuf paramètres

La figure 8.8 page précédente montre l'organisation des différents modules de Nappe, tandis que la figure 8.5.2 page 186 témoigne de sa relative complexité de préréglage, au travers des 99 paramètres... En particulier, le réglage des deux granulateurs demande 44 paramètres (22×2), et celui des différents processus aléatoires demande 26 paramètres[13]. Cette importante quantité de paramètres complique évidemment la composition, et contribue à expliquer l'inachèvement de la pièce.

Code 8.1 – *Fichier XML des 99 paramètres de Nappe*

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>

<pattrstorage name = "nap-ps">
  <slot number = "1">
    <patcher name = "nap-PannedPlayer">
      <patcher name = "nap-RandLFO-bp">
        <pattr name = "clar1_speed_random_dur_max" value = "10" />
        <pattr name = "clar1_speed_random_dur_min" value = "1" />
      </patcher>
      <patcher name = "nap-CycleMapping-bp">
        <pattr name = "clar1_speed_max" value = "9.67" />
        <pattr name = "clar1_speed_min" value = "0.38" />
      </patcher>
      <patcher name = "nap-RandLFO-bp[1]">
        <pattr name = "clar1_pan_random_dur_max" value = "83" />
        <pattr name = "clar1_pan_random_dur_min" value = "1" />
      </patcher>
    </patcher>
    <patcher name = "nap-PannedPlayer[1]">
      <patcher name = "nap-RandLFO-bp">
        <pattr name = "clar2_speed_random_dur_max" value = "24" />
        <pattr name = "clar2_speed_random_dur_min" value = "1" />
      </patcher>
      <patcher name = "nap-CycleMapping-bp">
        <pattr name = "clar2_speed_max" value = "-0.56" />
        <pattr name = "clar2_speed_min" value = "-0.26" />
      </patcher>
      <patcher name = "nap-RandLFO-bp[1]">
        <pattr name = "clar2_pan_random_dur_max" value = "76" />
        <pattr name = "clar2_pan_random_dur_min" value = "1" />
      </patcher>
    </patcher>
    <patcher name = "nap-PannedPlayer[2]">
      <patcher name = "nap-RandLFO-bp">
        <pattr name = "public1_speed_random_dur_max" value = "21" />
        <pattr name = "public1_speed_random_dur_min" value = "1" />
      </patcher>
      <patcher name = "nap-CycleMapping-bp">
        <pattr name = "public1_speed_max" value = "0.22" />
        <pattr name = "public1_speed_min" value = "0.1" />
      </patcher>
      <patcher name = "nap-RandLFO-bp[1]">
        <pattr name = "public1_pan_random_dur_max" value = "44" />
        <pattr name = "public1_pan_random_dur_min" value = "1" />
      </patcher>
    </patcher>
    <patcher name = "nap-PannedPlayer[3]">
      <patcher name = "nap-RandLFO-bp">
        <pattr name = "public2_speed_random_dur_max" value = "25" />
        <pattr name = "public2_speed_random_dur_min" value = "1" />
      </patcher>
      <patcher name = "nap-CycleMapping-bp">
        <pattr name = "public2_speed_max" value = "-1.38" />
        <pattr name = "public2_speed_min" value = "-0.88" />
      </patcher>
```

13. 4 modules de lecture × 6 paramètres locaux + 2 paramètres généraux.

```
56        <patcher name = "nap-RandLFO-bp[1]">
57          <pattr name = "public2_pan_random_dur_max" value = "78" />
58          <pattr name = "public2_pan_random_dur_min" value = "1" />
59        </patcher>
60      </patcher>
61      <patcher name = "nap-FreezeAlea-bp">
62        <pattr name = "dur_max" value = "20" />
63        <pattr name = "dur_min" value = "1" />
64      </patcher>
65      <patcher name = "nap-Convolution-bp">
66        <pattr name = "conv_drywet" value = "127" />
67        <pattr name = "conv_index" value = "127" />
68      </patcher>
69      <patcher name = "nap-Granul">
70        <pattr name = "mung1_drywet" value = "127" />
71        <patcher name = "Mung4.dsp~">
72          <patcher name = "Munger-ctrl">
73            <pattr name = "scale" value = "0" />
74            <pattr name = "delaylength" value = "200" />
75            <pattr name = "direction" value = "0" />
76            <pattr name = "ingain" value = "127" />
77            <pattr name = "maxvoices" value = "10" />
78            <pattr name = "minsize" value = "10" />
79            <pattr name = "outgain" value = "127" />
80            <pattr name = "pitch" value = "0" />
81            <pattr name = "pitch. variation " value = "0." />
82            <pattr name = "position" value = "-1" />
83            <pattr name = "power" value = "1" />
84            <pattr name = "ramptime" value = "10" />
85            <pattr name = "rand-gain" value = "0" />
86            <pattr name = "rate. variation " value = "0" />
87            <pattr name = "record" value = "1" />
88            <pattr name = "separation" value = "10" />
89            <pattr name = "size" value = "100" />
90            <pattr name = "size. variation " value = "0" />
91            <pattr name = "stereo-spread" value = "127" />
92            <pattr name = "tune" value = "0" />
93            <pattr name = "voices" value = "10" />
94          </patcher>
95        </patcher>
96      </patcher>
97      <patcher name = "nap-Granul[1]">
98        <pattr name = "mung2_drywet" value = "127" />
99        <patcher name = "Mung4.dsp~">
100         <patcher name = "Munger-ctrl">
101           <pattr name = "scale" value = "0" />
102           <pattr name = "delaylength" value = "200" />
103           <pattr name = "direction" value = "0" />
104           <pattr name = "ingain" value = "127" />
105           <pattr name = "maxvoices" value = "10" />
106           <pattr name = "minsize" value = "10" />
107           <pattr name = "outgain" value = "127" />
108           <pattr name = "pitch" value = "0" />
109           <pattr name = "pitch. variation " value = "0." />
110           <pattr name = "position" value = "-1" />
111           <pattr name = "power" value = "1" />
112           <pattr name = "ramptime" value = "10" />
113           <pattr name = "rand-gain" value = "0" />
114           <pattr name = "rate. variation " value = "0" />
115           <pattr name = "record" value = "1" />
116           <pattr name = "separation" value = "10" />
117           <pattr name = "size" value = "100" />
118           <pattr name = "size. variation " value = "0" />
119           <pattr name = "stereo-spread" value = "127" />
120           <pattr name = "tune" value = "0" />
121           <pattr name = "voices" value = "10" />
122         </patcher>
123       </patcher>
124     </patcher>
125     <patcher name = "nap-Feedback">
126       <pattr name = "fb1_absolute_max_buffering" value = "10000" />
127       <patcher name = "fib-bp-FeedbackAmount">
128         <pattr name = "fb1_feedback_amount" value = "0.76378" />
129       </patcher>
```

```
      <patcher name = "fib-bp-DryWetLevel">
        <pattr name = "fb1_dry-wet_level" value = "69" />
      </patcher>
      <patcher name = "fib-bp-DelayTime">
        <pattr name = "fb1_delay-time_init" value = "295" />
        <pattr name = "fb1_delay-time_max" value = "2000" />
        <pattr name = "fb1_delay-time_min" value = "50" />
      </patcher>
    </patcher>
    <patcher name = "nap-Feedback[1]">
      <pattr name = "fb2_absolute_max_buffering" value = "10000" />
      <patcher name = "fib-bp-FeedbackAmount">
        <pattr name = "fb2_feedback_amount" value = "0.448819" />
      </patcher>
      <patcher name = "fib-bp-DryWetLevel">
        <pattr name = "fb2_dry-wet_level" value = "60" />
      </patcher>
      <patcher name = "fib-bp-DelayTime">
        <pattr name = "fb2_delay-time_init" value = "1600" />
        <pattr name = "fb2_delay-time_max" value = "2000" />
        <pattr name = "fb2_delay-time_min" value = "50" />
      </patcher>
    </patcher>
    <patcher name = "nap-Env-bp">
      <pattr name = "master_env" value = "10000. 0. 1. 0. 0. 0 2125.984131 1. 0 6141.732422 0.52 0 8582.679688
          0.826667 0 10000. 0. 0" />
      <pattr name = "master_env_length" value = "10000" />
      <pattr name = "master_env_max" value = "1." />
      <pattr name = "master_env_min" value = "0." />
      <patcher name = "CB-EnvEdit">
        <patcher name = "CB-EnvFunctionEdit">
          <pattr name = "master_edit_ps" value = "-1" />
        </patcher>
      </patcher>
    </patcher>
    <patcher name = "nap-Env-bp[1]">
      <pattr name = "conv_env" value = "10000. 0. 127. 0. 0. 0 2125.984131 127. 0 6141.732422 66.04 0 8582.680664
          104.986664 0 10000. 0. 0" />
      <pattr name = "conv_env_length" value = "10000" />
      <pattr name = "conv_env_max" value = "127." />
      <pattr name = "conv_env_min" value = "0." />
      <patcher name = "CB-EnvEdit">
        <patcher name = "CB-EnvFunctionEdit">
          <pattr name = "conv_edit_ps" value = "-1" />
        </patcher>
      </patcher>
    </patcher>
    <patcher name = "nap-Env-bp[2]">
      <pattr name = "clar_env" value = "10000. 0. 127. 0. 0. 0 2125.984131 127. 0 6141.732422 66.04 0 8582.680664
          104.986664 0 10000. 0. 0" />
      <pattr name = "clar_env_length" value = "10000" />
      <pattr name = "clar_env_max" value = "127." />
      <pattr name = "clar_env_min" value = "0." />
      <patcher name = "CB-EnvEdit">
        <patcher name = "CB-EnvFunctionEdit">
          <pattr name = "clar_edit_ps" value = "-1" />
        </patcher>
      </patcher>
    </patcher>
  </slot>
</pattrstorage>
```

* * *

Nappe reste aujourd'hui un logiciel inachevé mais il possède un fort potentiel musical à l'instar du projet tout entier, dont l'inachèvement traduit *in fine* l'ambition musicale et la complexité.

Chapitre 9

Plugiscope, Ifso et Ultraviolette / Iván Solano

Résumé du chapitre : La collaboration avec Iván Solano a suscité trois logiciels interactifs : Plugiscope, Ifso et Ultraviolette. Ces programmes en temps réel utilisent en effet tous les trois le pédalier MIDI ; Plugiscope comme interface générique pour jouer de façon interactive avec des plugiciels sur le son entrant, Ifso comme interface de déclenchement ergonomique pour soixante-dix fichiers audionumériques, et Ultraviolette comme interface de contrôle du flux de cellules rythmiques. Ce chapitre présente successivement ces trois logiciels. L'annexe B page 253 rapporte par ailleurs un entretien conséquent avec Iván Solano.

9.1 Clarinettiste et compositeur

La collaboration avec Iván s'est déroulée en deux étapes : d'abord en tant que clarinettiste, puis en tant que compositeur.

9.1.1 Clarinettiste

La plupart des logiciels développés au cours de ce doctorat ont été initiés et suivis par des collaborations avec des compositeurs, en vue d'une pièce musicale (cf. section 4.3.2 page 93). Ainsi, Iván a d'abord participé au projet du concert en tant que clarinettiste, en interprétant *Chop Chich 2* de Pedro Castillo-Lara à la clarinette basse (cf. annexe E.3 page 322), et en travaillant sur *Woes-war-sollichwer-den* de Mauricio Meza (cf. chapitre 8 page 169).

En effet, outre sa sensibilité et sa virtuosité instrumentale, Iván représente un partenaire idéal pour une telle collaboration. D'une part, c'est un instrumentiste parfaitement rompu aux techniques contemporaines de jeu et d'écriture, jusqu'à l'électronique en temps réel, car il bénéficie d'une grande expérience (notamment avec l'atelier de composition de l'université, où il intervient et interprète les pièces des étudiants depuis plusieurs années). D'autre part, il est lui-même un compositeur tout à fait impliqué dans la musique contemporaine, d'ailleurs avec une activité de plus en plus importante depuis la fin de ses études auprès d'Ivan Fedele.

De plus, nous avons un point commun qui a joué un rôle catalyseur : la clarinette. En particulier, il a pu tester le système du capteur intra-bec (cf. annexe D.2 page 291) avec le pédalier et le logiciel Mimi, et présenter l'ensemble du dispositif lors d'un séminaire à l'université Paris 8. Ce système intra-bec vise à limiter fortement les risques de larsen avec la clarinette[1], ce que nous avons pu vérifier en répétition, sur des esquisses de la pièce *Woes-war-sollichwer-den* de Mauricio Meza.

9.1.2 Compositeur

Cependant, après le concert à Gennevilliers, nous avons prolongé notre collaboration dans le sens de la composition cette fois-ci, pour donner lieu à trois nouveaux logiciels : Plugiscope, Ifso et Ultraviolette.

D'abord, la rencontre avec Iván a rapidement stimulé la réalisation d'un premier programme – Plugiscope –, qui permet d'interagir avec les paramètres de plugiciels à l'aide d'un pédalier pour transformer le son de la clarinette (provenant du capteur placé dans le bec) en temps réel, en vue d'une pièce pour choeur de chambre et électronique intitulée *Kyōgen*. Plus tard, Iván m'a proposé de développer un deuxième programme – Ifso – qui

1. La clarinette reste un instrument difficile à sonoriser car son rayonnement acoustique dépend en effet des doigtés : le son ne sort par le pavillon que lorsque tous les trous sont bouchés, et sort sinon principalement par le premier trou ouvert (en partant du bec). Associée à la longueur de la clarinette, cette particularité oblige souvent à utiliser trois microphones pour la prise de son : un au niveau du pavillon, un au-dessus de la main droite, et un au-dessus de la main gauche, multipliant les risques de larsen en situation fortement amplifiée.

permet de déclencher des séquences audionumériques à l'aide du pédalier pour pouvoir jouer de la clarinette basse simultanément, pour des improvisations et pour une pièce pour clarinette basse et électronique intitulée *Ki*. Puis nous avons récupéré parmi mes esquisses un troisième programme – Ultraviolette –, en vue d'une pièce pour percussion et électronique intitulée *Corps Noir*.

9.2 Plugiscope : discussion et programmation

La forme actuelle de ce logiciel relativement simple dans son idée provient de deux souhaits successifs : initialement, il s'agissait modestement de pouvoir faire la démonstration du pédalier comme instrument interactif aux instrumentistes susceptibles de participer à des collaborations, dont Iván en particulier. Puis, le temps passant, ce premier objectif de démonstration s'est mu en un second souhait d'exploration plus systématique et fonctionnelle de ce que les plugiciels (ou *plug-ins*) pouvaient apporter musicalement dans ce contexte fortement interactif et instrumental.

Avant de présenter davantage ce logiciel, deux sections préliminaires s'imposent : une première section expose les raisons d'un choix terminologique inhabituel, puis une deuxième section formule une critique nécessaire à la contextualisation du développement de Plugiscope.

9.2.1 Parenthèse terminologique préalable

Pour remplacer le terme *plug-in*, anglicisme complètement installé dans le domaine de l'informatique musicale française, de nombreuses alternatives ont été proposées par différentes institutions chargées des questions terminologiques pour la langue française :

- *module externe* ;
- *module enfichable* ;
- *module d'extension* (adopté par la Commission générale de terminologie et de néologie en 1999[2]) ;
- *extension* (sa forme abrégée) ;
- *greffon* (plus original) ;
- ou encore *plugiciel* (terme proposé par l'Office québécois de la langue française[3] en 1996).

Cependant, pour l'instant, aucune de ces propositions n'a réussi à s'imposer en France dans le domaine de l'informatique musicale, où tout semble se passer la plupart du temps comme si la langue anglaise pouvait se porter garante du sérieux d'un logiciel, pour les

2. L'ensemble des recommandations terminologiques officielles du gouvernement est consultable en ligne via la base de données Criter (Corpus du Réseau Interministériel de Terminologie), à l'adresse http://franceterme.culture.fr.

3. Les recommandations de l'Oqlf sont elles aussi consultables en ligne, à l'adresse http://www.granddictionnaire.com, à partir d'un fonds plus complet que celui de la Dglf : depuis et vers le français, l'anglais et le latin, mais aussi dans une forme plus détaillée pour les définitions jointes, et dans des délais plus prompts, suivant leur objectif d'agir avant que le terme anglais ne devienne indélogeable.

sociétés commerciales comme pour les instituts de recherche (que penser du nom des logiciels-phares français « *OpenMusic* » de l'Ircam ou de « *GRM-Tools* » du Groupe de Recherches Musicales ?).

D'abord, la concision du terme anglais *plug-in* (qui perd même souvent son trait d'union à l'écrit avec *plugin*), constitué de seulement deux syllabes, élimine pratiquement les trois premières propositions (*module externe*, *module enfichable*, *module d'extension*) longues de quatre à cinq syllabes (dont la recommandation française *module d'extension*). Ensuite, le terme *extension* reste trop flou ; *greffon*, bien que sympathique, désigne depuis la biologie quelque chose qui doit pousser, ce qui n'est pas le cas ici puisqu'il s'agit de code binaire compilé donc figé. Enfin, *plugiciel* (prononcer à la française, avec « u » comme dans « plus » et « gi » comme dans « lo*gi*ciel ») fait bien sens informatiquement : en tant que module lo*giciel* qui apporte au programme principal une ou plusieurs fonctionnalités en *plus* ; malgré cela, *plugiciel* ne recueille guère d'audience qu'au Canada, concurrencé en France par la recommandation officielle (tardive) de *module d'extension*. D'un point de vue technique, la recommandation française s'avère aussi juste que *plugiciel* mais formulée en plus long, plus flou et surtout en moins évident quant à la possibilité d'établir le parallèle avec le terme anglais, déjà complètement implanté.

C'est pourquoi nous choisissons ici d'employer le néologisme *plugiciel* recommandé par l'Oqlf : pour sa proximité syntaxique avec *plug-in*, ce qui lui garantit d'être compris par le plus grand nombre sans trop d'ambiguïté, ainsi que pour sa pertinence sémantique comme élément *logiciel*. En effet, le suffixe *-giciel* est aujourd'hui assez répandu (cf. *progiciel*, et de plus en plus fréquemment *partagiciel*, *gratuiciel* ou encore *contributiel*), et le préfixe *plu-* indique bien qu'il apporte quelque chose de *plus* (du latin *plus*), et donc aussi qui ne saurait fonctionner seul puisqu'il vient *en plus*[4]. Notons que *logiciel* est lui-même une traduction francophone, adoptée en France en 1981, du mot *software*, traduction forgée à partir de *logique*, provenant du grec « science du raisonnement », et du suffixe « *-iel* » par opposition au *matériel*.

Pour refermer cette parenthèse terminologique, indiquons que le suffixe « *-scope* », utilisé le plus souvent pour former des noms d'appareils de visualisation, provient du grec ancien « observer » – ce qui correspond à notre dessein initial – et permet de prononcer « Plugiscope » de façon correcte aussi bien à la française qu'à l'anglaise.

9.2.2 Constat d'une dérive

Les précisions terminologiques précédentes éclairent légèrement le sujet mais laissent entière la question suivante : pourquoi s'intéresser aux plugiciels ?

En effet, il paraît légitime de douter de l'intérêt de ces modules logiciels aux fonctionnalités restreintes[5] et qui prolifèrent, à grand renfort de publicité, se vendent souvent

4. D'ailleurs, une nouvelle catégorie d'extensions logicielles émerge sous le nom « *add-on* » (littéralement « ajout au-dessus », justement avec l'idée de quelque chose « en plus »), sans doute pour exprimer une nuance par rapport à *plug-in* (littéralement « brancher dedans »), mais pour l'instant cette nuance reste très variable selon le contexte (système d'exploitation, navigateur, jeu vidéo, etc.).

5. Pour une description technologique, voir la section 2.3.2 page 57.

à un prix relativement élevé, et finalement assujettissent nombre d'usages de l'informatique musicale à des effets de mode, entraînant à la fois obscurantisme (les algorithmes utilisés restent opaques), fétichisme (désir et attachement à ces objets de mode mythifiés), et dépersonnalisation (beaucoup de gens les utilisent avec des résultats stéréotypés généralisés)...

Une modularité réductionniste

Il faut peut-être remonter au concept structurel et opératoire de la *modularité*, qui a saisi l'informatique depuis ses tout débuts avec Max Mathews sans jamais cesser de l'imprégner, ce que rappelle Jean-Claude Risset en rétablissant d'ailleurs un point historique vis-à-vis des synthétiseurs analogiques :

> Contrairement à ce que pensent beaucoup, la conception de Mathews ne copie pas celle des synthétiseurs : elle a au contraire inspiré des dispositifs de Moog et Buchla, réalisés analogiquement en tirant parti de la commande par tension... mais seulement à partir de 1964, alors que Music III a été écrit en 1959. En fait, la conception modulaire de Mathews a marqué la plupart des programmes de synthèse – comme Music 360, Music 11, CMusic, Csound – et des synthétiseurs analogiques ou numériques – comme Arp, DX7, 4A, 4B, 4C, 4X, et Syter – aussi bien que des langages de simulation de circuits électroniques, et plus tard un langage de création d'interactions temps réel comme Max. [Risset, 1999a, p. 25]

Ainsi, les plugiciels audionumériques pourraient représenter les plus jeunes héritiers de ce principe de la conception modulaire au sein de l'histoire de l'informatique musicale, s'ils n'étaient dénués de capacités de mise en réseau élaborées au profit du carcan de la superposition linéaire dans une logique pauvre d'accumulation qui sert manifestement des intérêts marchands (l'atomisation fonctionnelle permet une multiplication artificielle des produits commercialisables).

Structurellement, la modularité des plugiciels se trouve réduite à sa plus simple expression : une atomisation descendante, privée des procédures ascendantes potentiellement complexes dans la mise en relation propre à la conception modulaire telle que la retrace Risset, aussi bien dans les développements numériques, les synthétiseurs numériques et les langages de programmation, que pour les synthétiseurs analogiques où le contrôle par voltage permettait de réaliser des cascades complexes d'oscillateurs.

Une analogie usurpée

Loin d'une préoccupation de conception modulaire élaborée, la première norme de plugiciels développée en 1996 par la société Steinberg et baptisée « VST » pour *Virtual Studio Technology* porte en son nom des indices révélateurs de ses ambitions de l'époque : virtualiser le studio analogique (dans l'environnement de la marque par exemple), afin que tout un chacun puisse acquérir l'objet analogique virtuel dont il a rêvé, pour un coût inférieur. Or si la réalité du virtuel en tant que tel est indéniable, il plane toujours comme un mensonge ou une omission à propos de l'objet virtualisé : il est et restera virtuel, irrémédiablement. Autrement dit, un plugiciel imitant par exemple un synthétiseur analogique Minimoog n'est pas, n'a jamais été et ne sera jamais un vrai Minimoog... Cela

n'est pas grave en soi, puisque le plugiciel permet d'obtenir des résultats sonores même s'ils sont différents, néanmoins la distinction devrait être claire au lieu d'être camouflée, voire niée[6].

Comme l'a démontré Claude Cadoz, et de manière tout à fait générale, le numérique constitue une rupture de fait avec l'analogique, par l'intermédiaire du principe de *relais*, systématisé et exclusif, qui découple le continuum énergétique qui règne dans le monde analogique entre les causes et les effets[7]. Sans doute l'exploitation du paradigme de la mimique analogique s'épuisera-t-elle d'elle-même prochainement, au moins en informatique musicale lorsque tous les objets du studio analogique auront été virtualisés, pour laisser davantage de place aux véritables potentialités du numérique, plus proches de son essence informatique.

9.2.3 Vers un objectif raisonné

Le préambule critique précédent permet d'abord de délimiter négativement le champ de recherche, en pointant les écueils à éviter lorsqu'on travaille avec des plugiciels, puis de retenir un objectif raisonné – double en ce qui concerne Plugiscope : l'observation des paramètres et l'interaction musicale. En effet, positivement, cette application vise d'une part à observer, inspecter, étudier les plugiciels en montrant clairement leurs paramètres internes, levant ainsi une partie du voile, et vise d'autre part à interagir en temps réel avec l'ensemble de ces paramètres dévoilés, notamment avec un pédalier Midi ou tout autre contrôleur Midi.

9.2.4 Trois listes fouineuses

Sur trois listes verticales bien visibles, la fenêtre principale de Plugiscope (cf. figure 9.1 page 195) détaille et permet de sélectionner individuellement (de gauche à droite) :

1. les plugiciels d'un répertoire,
2. les programmes du plugiciel sélectionné (ensembles de préréglages de tous les paramètres),
3. leurs paramètres du plugiciel sélectionné.

6. La confusion est parfois entretenue par l'auteur de l'objet analogique d'origine lui-même, comme dans ce cas précis où Robert Moog déclare en personne que l'exemplaire virtuel est identique à l'original analogique (c'est nous qui soulignons) : « *Arturia has done it again. The* minimoog V*'s sound quality captures the magic of the original classic Minimoog. The graphic interface looks classy and responds smoothly, making it fun and easy to use. We at Moog Music are happy to lend our name to this fine product.* » http://www.arturia.com/fr/minimoog/minimoogv.php.

7. « La fonction relais est indissociable des notions d'amplification, de signal et d'information. Ce que l'on désigne aujourd'hui globalement par le terme *numérique* suppose également le relais. Le relais implique par essence un principe de coupure (pendant technologique de la coupure épistémique des linguistes). [...] Entre le clavier et l'écran, cela ne se passe qu'à travers le relais de millions de transistors et par le fait d'électrons dont l'agitation fébrile au milieu de labyrinthes de silicium ne dépend d'aucune manière de l'énergie avec laquelle on frappe le clavier. Or, cette situation ne se présente de manière aussi absolue, et avec cette échelle, dans aucune des situations technologiques antérieures. » [Cadoz, 1999, p. 170]

Bien sûr, Plugiscope ne révèle pas les algorithmes employés par un plugiciel, mais ses trois listes clarifient tout de même son fonctionnement. En effet, elles présentent les programmes et les paramètres internes dans leur intégralité et à un même niveau hiérarchique (c'est-à-dire par leur nom et sans artifice graphique), en complément de la fenêtre graphique d'origine du plugiciel. De cette façon, elles montrent explicitement les correspondances entre les paramètres internes du plugiciel et les mécanismes de visualisation externe de la fenêtre d'origine (visualisation plus ou moins fantaisiste, outrée, simpliste, austère selon l'auteur du plugiciel).

Chaque liste peut afficher verticalement 20 éléments dans une fenêtre à l'aide des objets `jit.cellblock`, ce qui suffit pour montrer la totalité des programmes et des paramètres de la plupart des plugiciels, et recourt à un mécanisme d'ascenseur lorsque la quantité d'éléments dépasse ce seuil.

La liste des plugiciels

La première liste, la plus à gauche dans la fenêtre principale, recense tous les fichiers susceptibles d'être des plugiciels au format VST : un algorithme filtre tous les fichiers d'un répertoire spécifié par l'utilisateur, récursivement jusqu'à une profondeur spécifiée par l'utilisateur, et ne renvoie que les fichiers portant l'extension « .vst » lorsqu'un système Macintosh est détecté, ou bien uniquement les fichiers portant l'extension « .dll » lorsqu'un système Windows est détecté (attention, la plupart des fichiers « .dll » ne sont pas des plugiciels VST).

Cet algorithme (cf. figure 9.2 page 196) exploite les commodités offertes par l'objet `ubumenu` pour la gestion des répertoires : une fois sa variable interne autopopulate positionnée à 1, le message prefix suivi du chemin d'un répertoire valide (stipulé par l'utilisateur dans l'objet `textedit` en haut de la fenêtre principale) prépare le remplissage automatique de l'objet `ubumenu`. Le remplissage sera effectivement déclenché lors de la réception du message populate[8], ou bien du message depth suivi d'un nombre entier, pour préciser la profondeur de récursion à partir du répertoire d'origine. À chaque fois qu'un remplissage automatique est déclenché, le patch `[kb-filtre_de_suffixe]` filtre les fichiers selon leur extension (.vst ou .dll) pour remplir à son tour l'objet `coll` rattaché enfin à l'objet `jit.cellblock` qui nous intéresse, par un message refer.

Loin d'être une coquetterie, ce détour par l'objet `coll` pour remplir l'objet `jit.cellblock` répond à un problème important. En effet, les plugiciels conçus à partir de *Max/MSP* (avec son système *Pluggo*) sont incompatibles avec tout objet considéré de type fenêtre par le système d'exploitation : par exemple, les objets `jit.cellblock` et `ubumenu` font brutalement quitter l'application lorsqu'ils communiquent directement avec. Or ces plugiciels « *Pluggo* » sont justement souvent intéressants, pour leur originalité par rapport aux plugiciels commerciaux. Ainsi, le développement de Plugiscope a subi plusieurs contorsions pour pouvoir tolérer ces plugiciels *Pluggo*, au prix aussi d'une certaine discipline de la part de l'utilisateur : si des plugiciels *Pluggo* sont chargés, il ne faut pas cliquer sur les objets-fenêtres devenus instables, mais utiliser les procédures alternatives mises en place,

8. On peut traduire l'emploi informatique de *to populate* par « garnir », « remplir » ou « charger » (ce dernier correspondant davantage à *to load*).

c'est-à-dire un périphérique MIDI (pédalier, clavier, etc.) ou encore les raccourcis-clavier du pédalier virtuel. Une fenêtre d'avertissement décrit ces procédures à l'utilisateur (cf. figure 9.3 page 196), et aucun de ces problèmes ne survient tant qu'on ne sélectionne pas de plugiciel *Pluggo*.

Concernant les autres limitations, pour l'instant, seuls les plugiciels au format VST sont accessibles dans l'environnement *Max/MSP*, à l'aide de l'objet `vst~`, et les plugiciels VSTi ne sont pas vraiment pris en compte ici, étant donné que nous avons privilégié l'utilisation d'un instrument acoustique en entrée et non pas celle d'un instrument MIDI ; en revanche, si les notes MIDI n'entrent pas dans les plugiciels en tant que telles, elles peuvent toutefois servir à la sélection des éléments dans les listes, grâce aux options de configuration de Plugiscope.

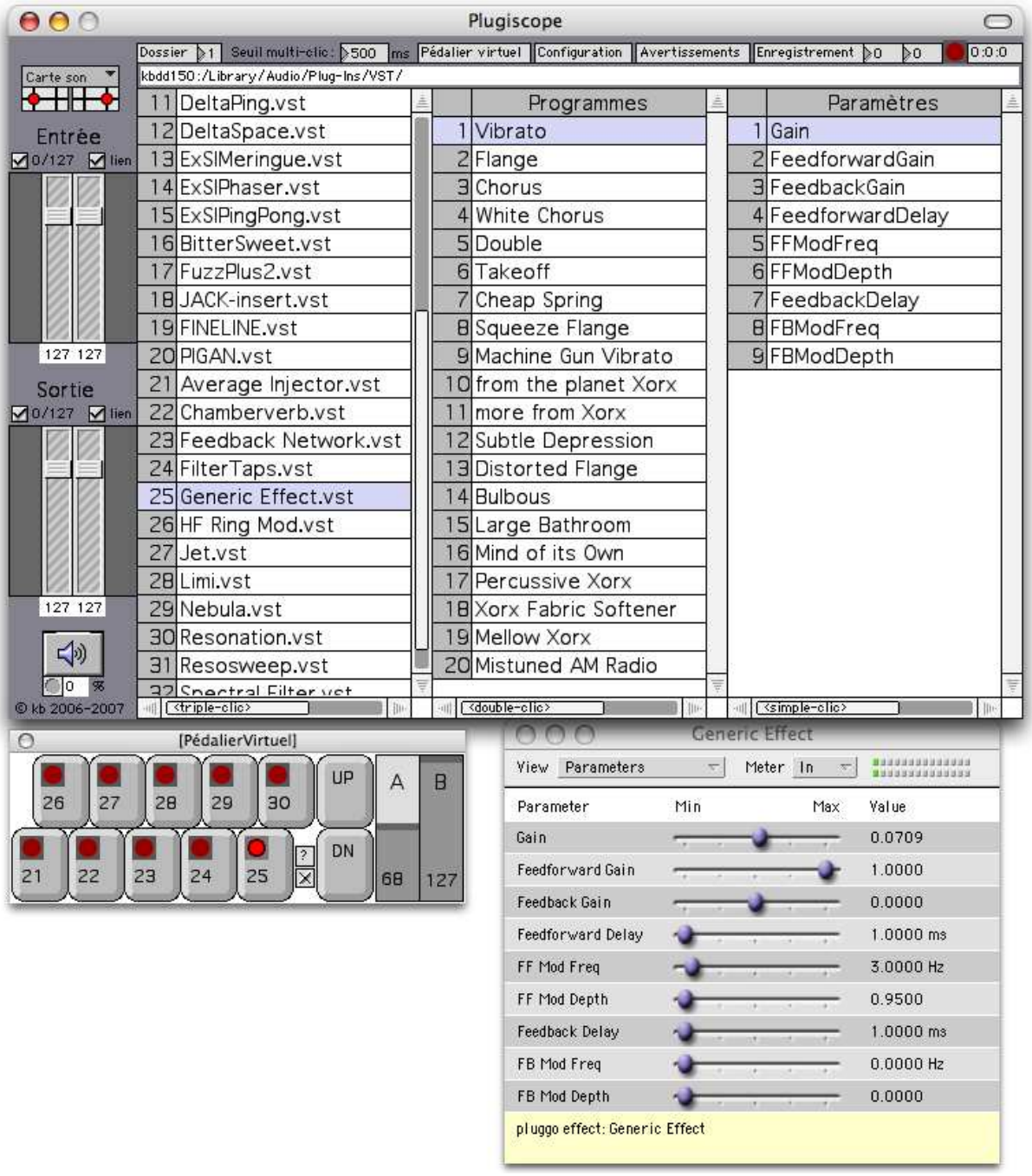


FIG. 9.1 – *Capture de l'interface visuelle de Plugiscope*

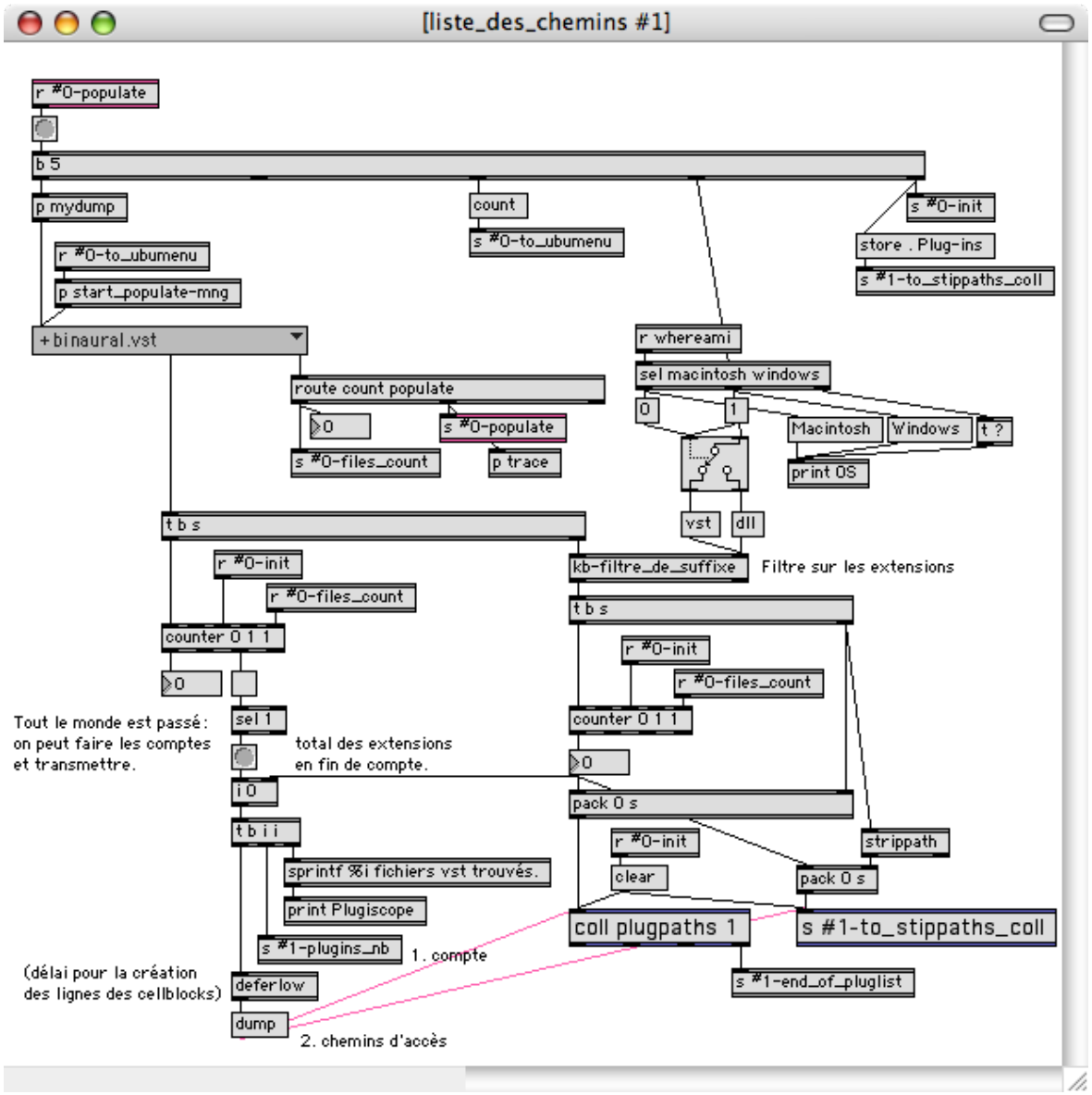


FIG. 9.2 – *Patch de constitution de la liste des plugiciels*

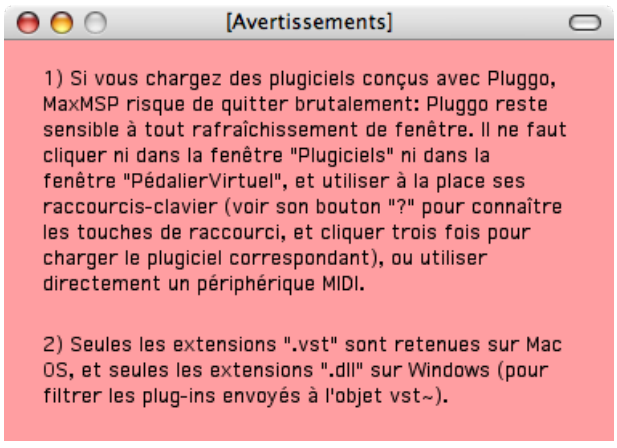


FIG. 9.3 – *Fenêtre d'avertissement*

La liste des programmes

Musicalement, cette liste est peut-être la plus utile : d'une part sur le plan de la visualisation, parce que dans la fenêtre d'origine des plugiciels les programmes sont généralement gérés par un menu qui ne peut en montrer qu'un seul à la fois quand le Plugiscope peut en montrer une vingtaine, et d'autre part sur le plan de l'interaction, puisqu'elle permet de changer l'ensemble des paramètres en une seule commande[9], et non-linéairement.

Techniquement, dès qu'un plugiciel a été sélectionné dans la liste de gauche, l'objet `vst~` est consulté pour recueillir les informations dont Plugiscope a besoin pour mettre à jour les deux autres listes, à l'aide des messages adaptés (cf. figure 9.4) :

- get -3 pour obtenir le nombre de programmes enregistrés,
- get -4 pour obtenir le nombre de paramètres,
- pgmnames pour obtenir les noms de chaque programme,
- params pour obtenir les noms de chacun des paramètres.

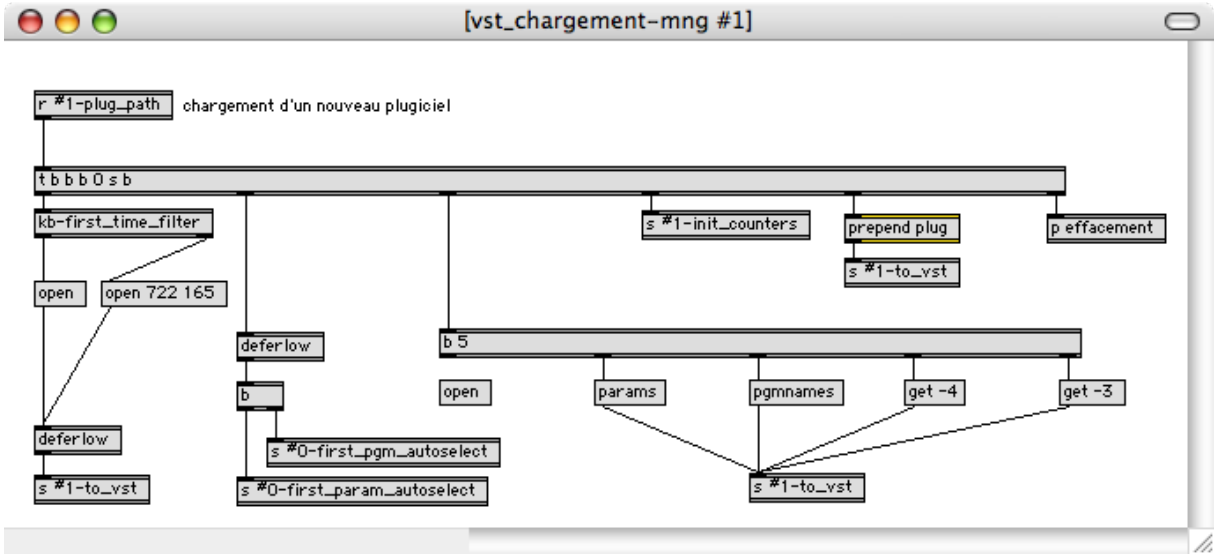


FIG. 9.4 – *Patch de chargement des plugiciels*

Une fois les deux listes remplies, le premier programme et le premier paramètre sont sélectionnés automatiquement, puis la fenêtre d'origine du plugiciel est ouverte (par défaut à l'extérieur de la fenêtre de Plugiscope et sous celle du pédalier virtuel).

La liste des paramètres

C'est la liste la plus à droite et surtout la plus accessible, puisqu'il suffit d'un simple-clic de pédalier pour changer de paramètre, alors qu'il faut un double-clic pour changer de programme et un triple-clic pour changer de plugiciel.

9. Nous avons conservé le terme « programme » pour désigner un enregistrement des valeurs précises pour l'ensemble des paramètres, afin de ne pas entraver le parallèle avec l'intitulé « *program* » qui figure sur de nombreux plugiciels, bien que « préréglage » soit sans doute plus correct.

Techniquement, sélectionner un paramètre revient ici à mettre en correspondance ce paramètre et la « pédale A », qui dépend de la configuration Midi enregistrée (à moins d'utiliser le pédalier virtuel). À la différence des programmes, la sélection d'un paramètre permet un contrôle graduel et discret de sa valeur, typiquement de 0 à 127, ce qui fait son intérêt musical ; d'ailleurs, la première version de démonstration de Plugiscope ne contenait que cette liste.

Une triple « listoscopie »…

En résumé, Plugiscope permet en quelque sorte de « voir » à l'intérieur d'un plugiciel ; non pas l'algorithme lui-même, qui reste opaque de toute façon, mais la liste des paramètres ainsi que la liste des préréglages (les « programmes ») en regard de la fenêtre d'origine du plugiciel, en plus de montrer la liste des plugiciels disponibles à partir d'un répertoire.

9.2.5 Détournement interactiviste

L'usage premier des plugiciels, on l'a vu dans la section critique, relève du traitement sonore à la manière analogique, c'est-à-dire coincé dans une piste audio virtuelle d'une table de mixage virtuelle[10]. Ainsi, les différents formats de plugiciels ont d'abord spécifié des systèmes de modification des paramètres des plugiciels en conformité avec l'objectif (à valeur de paradigme) de la mimique analogique, puis, plus proche de la nature numérique des plugiciels, les stations de travail audionumériques ont développé des mécanismes d'automation de ces paramètres.

Plugiscope propose quant à lui une double visualisation qui s'inscrit dans la perspective de l'interaction : tandis que les trois listes de Plugiscope révèlent clairement tout ce qui peut être sélectionné par l'utilisateur, la fenêtre originale du plugiciel ouvert montre le plus souvent une représentation intuitive des modifications des paramètres ; cette double visualisation favorise l'apprentissage du système ainsi que l'interaction elle-même.

Ainsi, Plugiscope détourne les plugiciels vers un usage *performatif* (cf. section 5.4 page 109) :

- en situation pratique de temps réel, l'inconvénient de la pauvreté fonctionnelle devient l'avantage de la simplicité, gage de jouabilité ;
- les modalités d'interactions ont été diversifiées : d'abord le son entrant peut provenir aussi bien de la carte audio que d'un fichier audio, puis le contrôle peut se faire à

10. Daniel Teruggi s'indignait déjà en 1999 : « Sur ce point [la représentation des outils], j'aurais un sérieux reproche à adresser à l'informatique en général ; quand il s'agit d'accès de ce type, on les calque sur des modèles analogiques, comme bouton, tirette, qui sont associés à des gestes qu'on effectue dans la vie courante. Je pense que l'environnement informatique permet justement de créer d'autres types d'outils de contrôle, d'une virtualité qui ne corresponde pas à notre expérience. Ce serait, je trouve, le domaine le plus intéressant d'évolution vers le futur ; arrêter de faire des tables de mixage qui n'en sont pas. On a produit virtuellement des outils avec lesquels on ne retrouve pas la souplesse de contrôle qu'apportaient leurs modèles analogiques. C'est une direction qui me semble complètement erronée, propre au monde des constructeurs commerciaux d'outils. » [Teruggi, 1999b, p. 226].

la souris, au clavier ou par différents contrôleurs Midi (grâce au sous-patch de configuration `[mfc-MidiinConfig]` en figure 9.5) ;
– Plugiscope respecte une hiérarchie pratique pour l'accessibilité aux différents éléments : un simple clic suffit pour activer un paramètre quand il faut un double clic pour activer un programme et un triple clic pour changer de plugiciel.

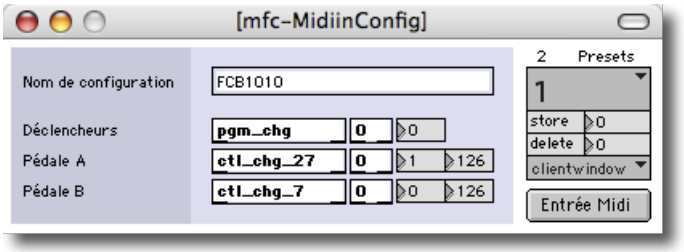


Fig. 9.5 – *Interface de configuration Midi*

* * *

Essentiellement, l'intérêt de Plugiscope réside dans la combinaison de son système de visualisation multiple avec ses capacités interactives, combinaison qui ouvre de fait la voie à l'exploration interactive ; autrement dit, Plugiscope permet à l'utilisateur – instrumentiste ou compositeur – d'entrer en dialogue avec la fonction particulière d'un plugiciel, dans son aspect limité, et via ses paramètres, de manière dynamique.

9.3 Ifso : panorama succinct

Le projet d'Ifso étant largement relaté dans l'entretien avec Iván, cette section présente simplement ses caractéristiques principales. La pièce correspondante, en cours d'écriture, s'appelle *Ki* (« respiration » est l'une de ses significations en japonais).

9.3.1 Présentation générale

Pour résumer, Ifso est un lecteur polyphonique interactif pour pédalier Midi, permettant de contrôler 7 traitements sur 70 fichiers audio : vitesse, panoramique, volume, transposition, granulation, bouclage, inversion. Il permet aussi de contrôler 4 de ces traitements sur l'entrée audio : panoramique, volume, transposition, et granulation.

L'interface visuelle d'Ifso s'appuie sur une métaphore du pédalier (cf. section 4.3.5 page 98) : un bloc de dix boutons à gauche et un bloc de deux curseurs verticaux à droite ; entre ces deux blocs s'insère un bloc consacré aux paramètres.

Ensuite, lorsque la lecture d'un fichier audio est déclenchée, alors une petite fenêtre rouge dénommée « lecteur » apparaît sous la fenêtre principale, avec une ombre de lecture qui progresse sur la forme d'onde, et le bouton concerné devient vert. Pour compléter le code couleur, les sélections sont signalées en jaune, pour les fichiers comme pour les paramètres.

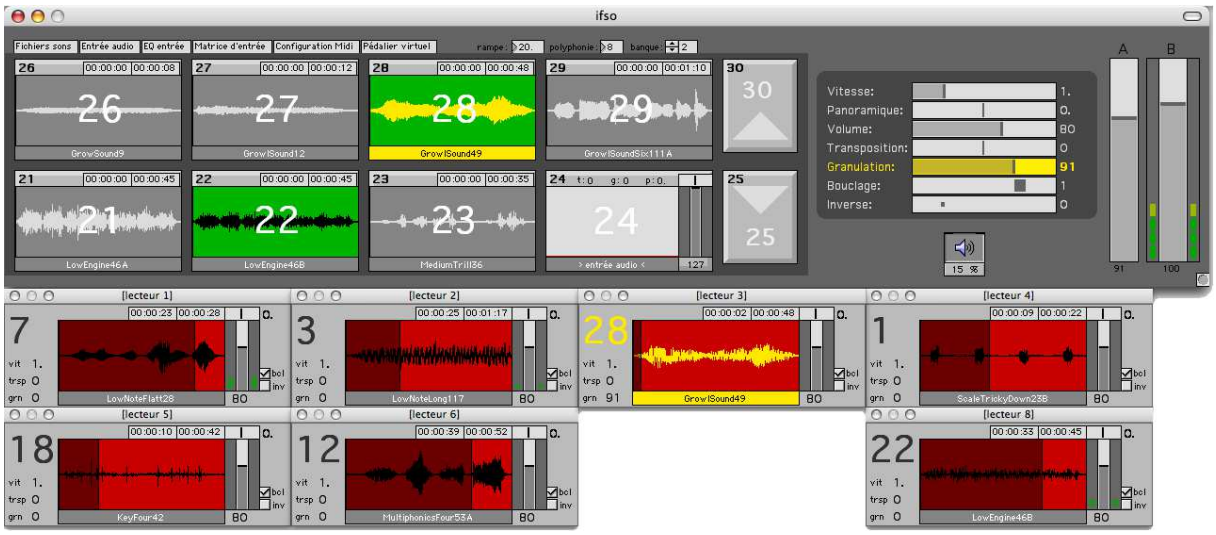


FIG. 9.6 – *Capture de l'interface visuelle d'Ifso*

9.3.2 Gestion des fichiers audio

À la différence de Rose amère ou d'Iviv, Ifso utilise le changement de banque (les pédales UP et DOWN), afin de mettre à disposition le plus de fichiers possibles, soit 70 fichiers pour 10 banques. En effet, il reste 7 fichiers déclenchables par banque lorsqu'on ôte le bouton n° 4 pour l'entrée audio et les boutons n° 5 et n° 10 pour la navigation entre les paramètres aux 10 boutons disponibles pour une banque.

Une interface de gestion des fichiers audio permet d'organiser et d'enregistrer la correspondance entre les 70 boutons de façon ergonomique (cf. figure 9.7 page suivante) : il suffit de donner le chemin du répertoire qui contient les fichiers pour que les 70 menus se remplissent automatiquement avec ces fichiers, puis simplement de choisir un fichier dans chaque menu pour l'affecter au bouton idoine. Pour faciliter cette opération, les lignes sont numérotées et regroupées selon la répartition des boutons réservés aux fichiers : 1, 2, 3 puis 4, 5, 6, 7 pour la 1re banque ; 11, 12, 13 puis 14, 15, 16, 17 pour la 2e ; etc. Lors du démarrage d'Ifso, le préréglage n° 1 est automatiquement chargé.

Une fois que les fichiers audio sont chargés, la procédure d'utilisation des fichiers comporte trois étapes : sélection – lecture – arrêt. À cause du mécanisme de sélection, il faut distinguer 4 états possibles[11] et 2 types de clics (cf. figure 9.8 page 202) :

- le clic interne « c_{int} », correspondant à un appui sur le bouton, qui permet d'abord de sélectionner, puis alternativement d'activer et de désactiver la lecture ;
- le clic externe « c_{ext} », correspondant à un appui sur tout autre bouton que le bouton courant, qui n'interrompt pas nécessairement la lecture.

11.

	lecture inactive	lecture active
désélection	00	10
sélection	01	11

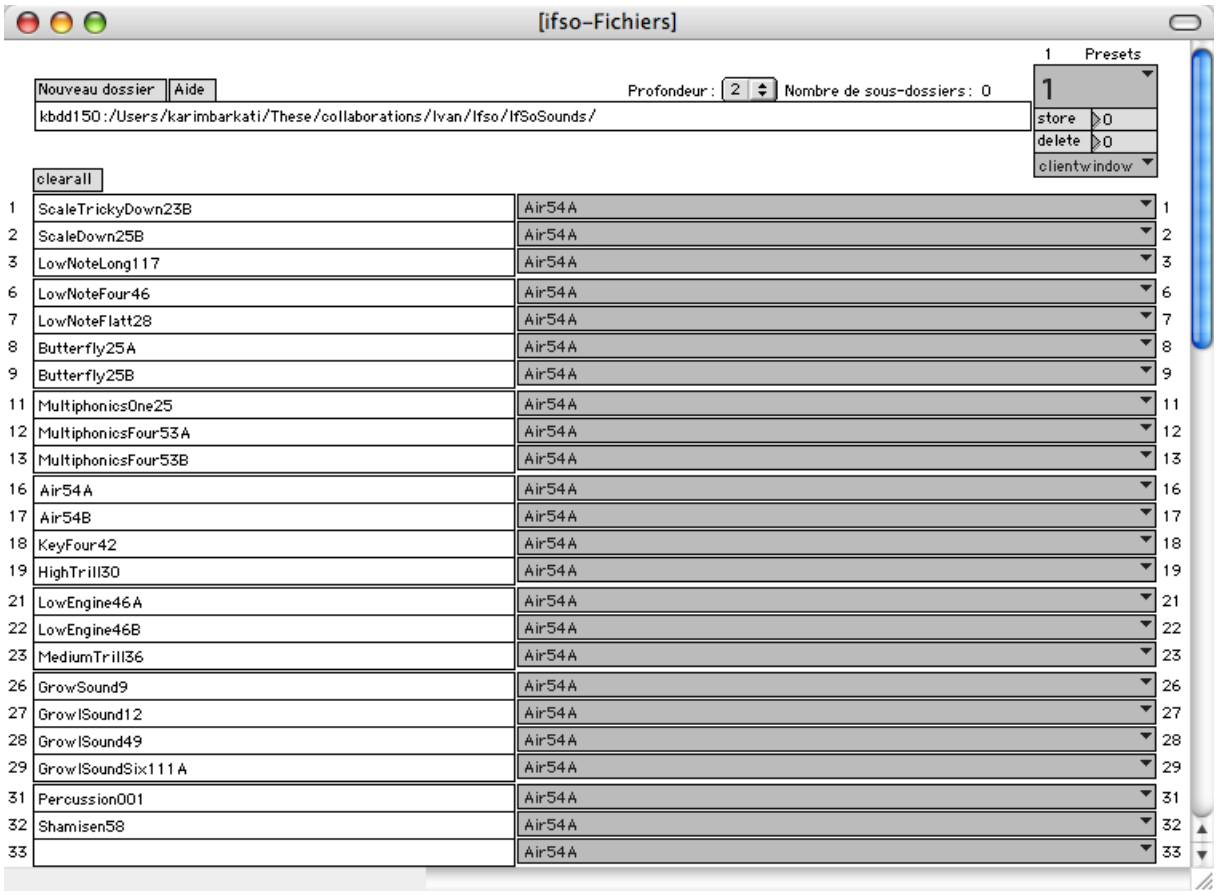


FIG. 9.7 – *Interface visuelle de gestion des fichiers audio d'Ifso*

9.3.3 Contrôle des paramètres

Dans Ifso, la A-permutation (cf. section 6.2.2 page 123) a été remplacée par un mécanisme en deux étapes : d'abord la sélection du module à traiter (un lecteur ou l'entrée audio), puis le choix parmi les traitements (7 pour les lecteurs et 4 pour l'entrée audio, cf. tableau 9.1).

La sélection du module se fait directement sur le bouton correspondant, suivant le diagramme représenté figure 9.8 page suivante. La dissociation entre la sélection et la lecture permet de régler les paramètres d'un module avant de lancer sa lecture, ou encore de revenir sur ses réglages en venant d'un autre module sans interrompre sa lecture.

	min	*max*	*type*	*init*
vitesse	0.1	4.	curseur	1.
panoramique	-1.	1.	curseur	0.
volume	*0*	*127*	*curseur*	*80*
transposition	-12	12	curseur	0
granulation	0	127	curseur	0
bouclage	0	1	interrupteur	1
inverse	0	1	interrupteur	0

TAB. 9.1 – *Les 7 paramètres d'Ifso*

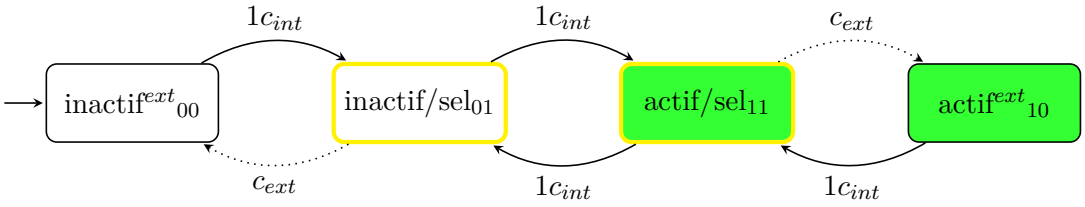


FIG. 9.8 – *Diagramme performatif des fichiers audio dans Ifso*

Le choix du paramètre à régler se fait cette fois-ci à travers deux boutons au lieu d'un seul : soit par incrémentation (boutons 10, 20, 30, etc., *i. e.* en haut à droite) soit par décrémentation (boutons 5, 15, 25, etc., *i. e.* en bas à droite) dans la liste circulaire, afin de minimiser le nombre de clics nécessaires pour atteindre un traitement dans la liste (au maximum, une liste en A-permutation dans Rose amère contient 3 éléments). C'est pour cette raison que l'élément sélectionné par défaut – le volume – n'est pas le 1er de la liste mais le 3^{e}, de façon à proposer visuellement de remonter sur la panoramique ou la vitesse, autant que de descendre dans la liste.

Par ailleurs, un système d'accrochage a été développé pour la pédale A ; ce système évite les sauts de valeur trop important lorsqu'on change de paramètre ou de module, grâce au sous-patch `[kb-accrocheur127]` (cf. figure 9.9), qui bloque la transmission des valeurs tant qu'elles ne s'approchent pas suffisamment de la valeur courante, c'est-à-dire à moins de 10 unités sur 128.

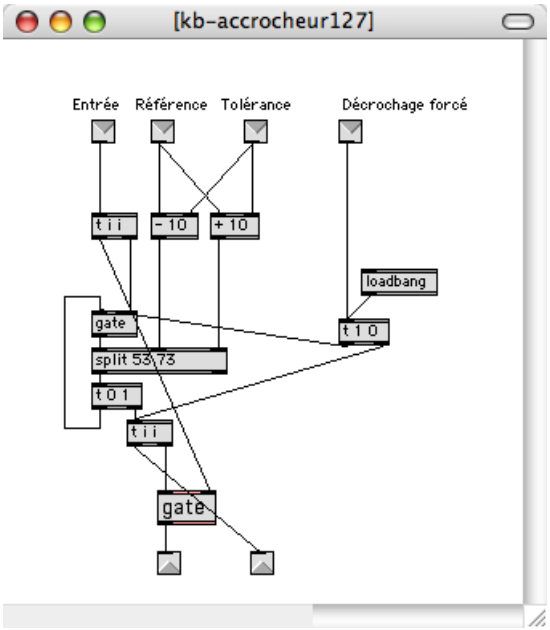


FIG. 9.9 – *Sous-patch de gestion de l'accrochage*

9.3.4 Une interface optimisée pour le volume

Enfin, pour l'ajustement du paramètre de volume, Ifso n'utilise pas l'objet standard `gain~` de Max/MSP qui ne rend pas une course linéaire, car la première moitié du curseur

de `gain~` est presque inutile (cf. figure 9.10, où les tables tracent les valeurs mesurées par les objets `meter~`).

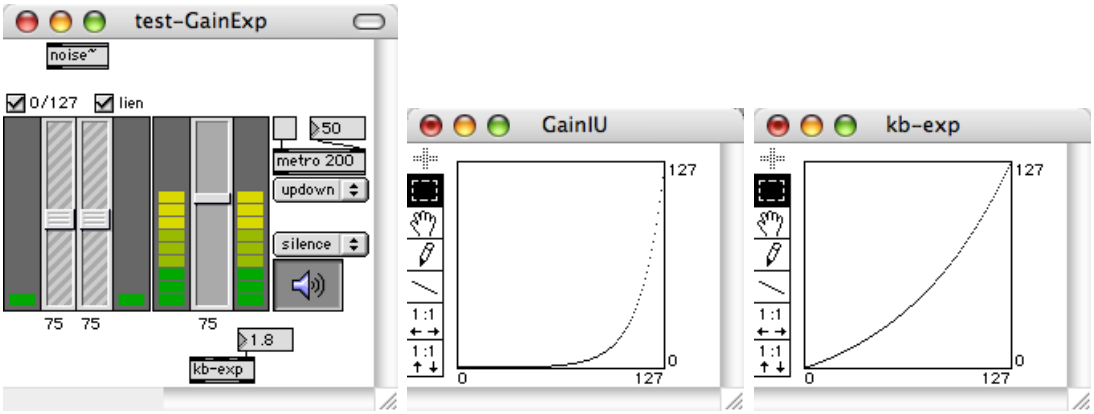


FIG. 9.10 – *Comparaison des curseurs de volume*

À la place, Ifso, Rose amère et Ultraviolette utilisent `[kb-exp]` (cf. figure 9.11), un sous-patch personnel qui permet de regagner une modification de l'amplitude perceptible tout au long de la course, de 0 à 127, avec un coefficient $k = 1.8$ pour l'exponentielle.

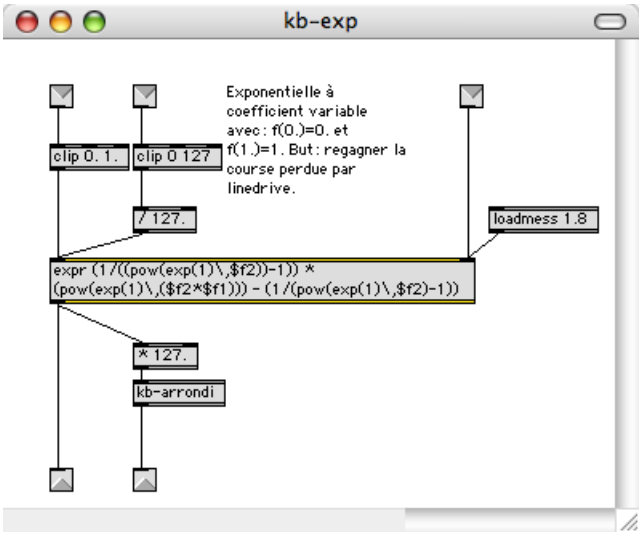


FIG. 9.11 – *Sous-patch de gestion de l'exponentielle du volume*

La fonction de transfert employée provient d'une hypothèse sur la forme exponentielle d'une telle fonction, soit $f(x) = a.e^{kx} + b$, et de deux conditions, à savoir $f(0) = 0$ et $f(1) = 1$, ce qui donne l'équation suivante :

$$f(x) = \frac{e^{kx} - 1}{e^k - 1}, \text{ avec } k \neq 0. \tag{9.1}$$

Ou encore, en termes informatiques (pour Max/MSP), l'expression suivante :

```
expr (pow(exp(1)\,($f2*$f1))-1)/(pow(exp(1)\,$f2)-1)
```

Cette fonction restitue une sensibilité aux pédales d'expression qui gèrent le volume, y compris dans la première partie de leur course, ce qui s'est révélé musicalement important en situation de concert.

9.4 Ultraviolette : panorama succinct

Le projet avec Ultraviolette n'apparaît pas dans l'entretien parce qu'il lui est postérieur. Il s'agit de la partie logicielle de *Corps Noir*, une pièce d'Iván en cours d'écriture, pour percussion et électronique en temps réel.

9.4.1 Présentation générale

Ultraviolette peut être décrit comme un générateur de flux rythmique interactif. Le flux rythmique est généré à partir d'une collection de cellules rythmiques prédéfinies et d'un dossier de fichiers audio pré-enregistrés, et l'interaction se fait sur l'interpolation interactive du tempo et de la vitesse de lecture.

Visuellement, l'interface d'Ultraviolette réinterprète la métaphore du pédalier d'une façon circulaire : les dix boutons s'organisent en deux demi-cercles superposés (de 1 à 5 pour le demi-cercle inférieur, de 6 à 10 pour le demi-cercle supérieur), le bouton n° 1 étant légèrement excentré pour marquer sa fonction d'initialisation. Les deux curseurs verticaux qui représentent les pédales A et B restent à droite des boutons, comme sur le pédalier.

9.4.2 Cellules rythmiques et hasard

L'objectif musical premier d'Ultraviolette consistait à renouer avec le rythme, sans tomber dans des séquences trop prévisibles ni trop imprévisibles d'une part, et d'autre part en allant jusqu'à la frontière entre rythme et texture sonore.

Ainsi, pour chercher un point d'équilibre entre prévisibilité et imprévisibilité, j'ai combiné l'utilisation d'une collection de cellules rythmiques avec un tirage au sort par cheminement aléatoire (*random walk*) dans la collection : à chaque nouvelle impulsion, une nouvelle cellule rythmique est choisie à une distance de la précédente de 1 pas au plus[12].

Ce cheminement aléatoire pas-à-pas garantit une forme de cohérence dans le résultat rythmique global, à la condition qu'une cohérence ait été introduite dans la collection ordonnée des cellules rythmiques, ce qui est le cas pour l'exemple donné en figure 9.13 page 206.

Techniquement, le principe consiste ensuite à tirer au hasard un fichier audio dans le dossier indiqué (cf. figure 9.14 page 206), cette fois-ci de façon plus désordonnée, simplement avec l'objet `random`. Pour l'instant, le nombre d'éléments d'une cellule détermine

12. Ce tirage par cheminement aléatoire est effectué par l'objet `drunk 12 2`, si la taille de la collection vaut 12, et le pas maximum d'avancement à été fixé à 2 (*i. e.* les pas de 0 et 1 sont permis).

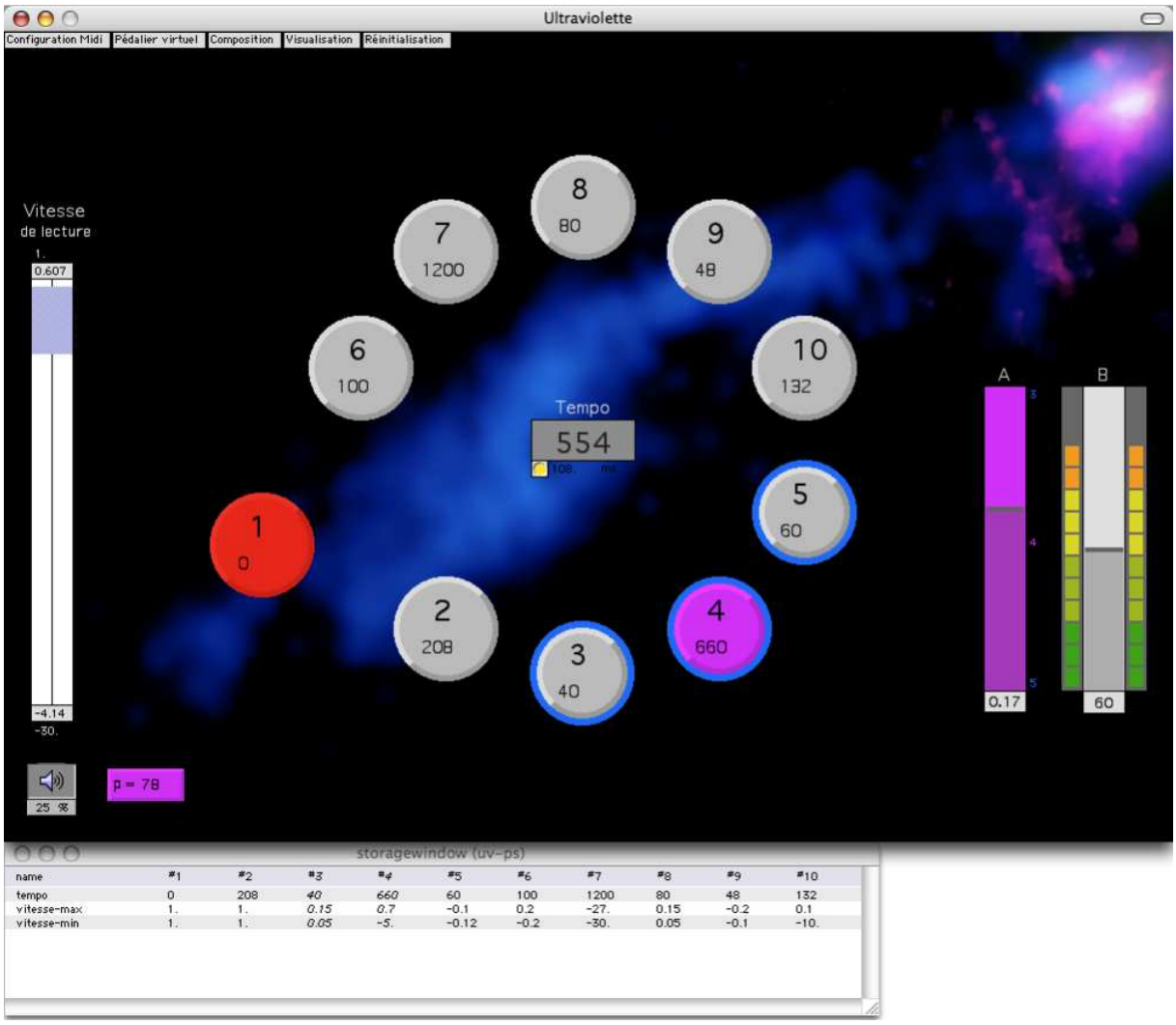


FIG. 9.12 – *Capture de l'interface visuelle d'Ultraviolette*

la subdivision de la pulsation, puis la valeur 1 (une « impulsion[13] ») déclenche le tirage au sort et la lecture d'un fichier audio tandis que la valeur 0 n'en fait rien. Bien que ce système fonctionne, je souhaite utiliser à terme 128 valeurs afin de pouvoir travailler sur la dynamique, au lieu des seuls 0 et 1.

Au maximum, Ultraviolette peut ainsi lire jusqu'à 128 fichiers audio (grâce à l'objet `poly~`)... Cette valeur relativement importante permet deux résultats sonores en pratique : premièrement d'approcher des textures granulaires lorsque le tempo est élevé (p. ex. 1200 dans la 7e configuration en figure 9.15 page suivante), et secondement de garder longtemps en mémoire les longs sons inversés qui peuvent alors se faire entendre seulement des dizaines de secondes plus tard (p. ex. la 5e configuration, avec une vitesse de lecture comprise entre -0.12 et -0.10).

13. À ne pas confondre avec la pulsation.

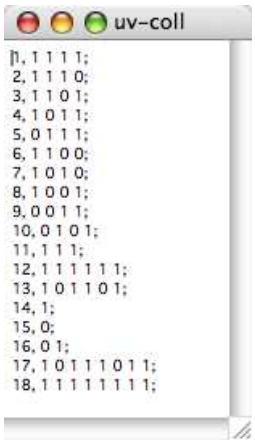


FIG. 9.13 – *Une collection de cellules rythmiques pour Ultraviolette*

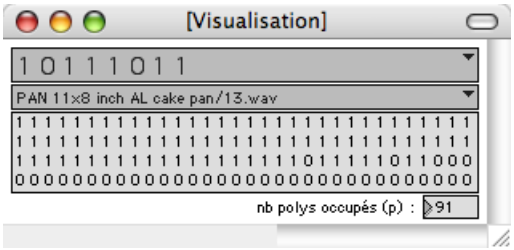
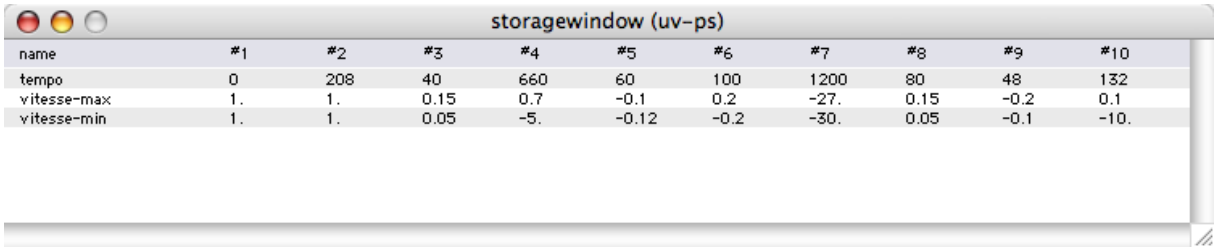


FIG. 9.14 – *Visualisation de la charge d'Ultraviolette*

9.4.3 Dix configurations

Les 10 premières configurations correspondent aux 10 premiers préréglages du fichier *uv-ps.xml* et sont affectées aux 10 boutons d'Ultraviolette ; on notera sur la figure 9.15 que le tempo de la 1re configuration est à 0, de façon à pouvoir arrêter le déclenchement de nouveaux fichiers depuis le pédalier.



name	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
tempo	0	208	40	660	60	100	1200	80	48	132
vitesse-max	1.	1.	0.15	0.7	-0.1	0.2	-27.	0.15	-0.2	0.1
vitesse-min	1.	1.	0.05	-5.	-0.12	-0.2	-30.	0.05	-0.1	-10.

FIG. 9.15 – *Les 10 configurations d'Ultraviolette*

Chaque configuration comporte trois paramètres : *tempo*, *vitesse-max* et *vitesse-min*. Le paramètre *tempo* détermine simplement la vitesse de la pulsation (pour l'objet `metro`), mais les paramètres *vitesse-min* et *vitesse-max* bornent un intervalle pour un troisième tirage au sort : à chaque impulsion, la vitesse de lecture du fichier tiré est elle-même tirée au hasard dans l'intervalle [*vitesse-min*, *vitesse-max*], afin de pouvoir renouveler la sonorité des fichiers.

Afin de spécifier commodément ces trois paramètres pour les 10 configurations, la fenêtre présentée figure 9.14 rassemble trois boîtes d'édition numérique et le patch de

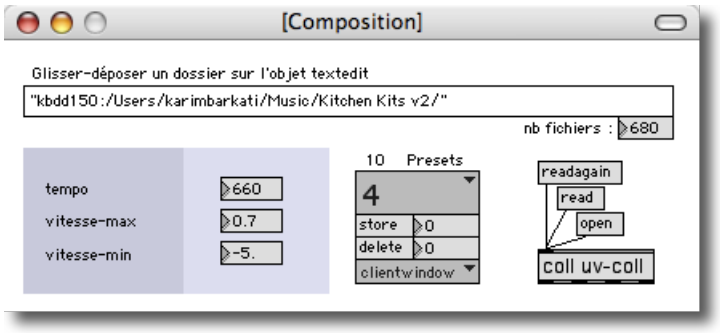


FIG. 9.16 – *Interface de composition des configurations d'Ultraviolette*

manipulation des préréglage `[kb-Storage-mng]`. Cette fenêtre permet en outre d'éditer la collection des cellules rythmiques et d'indiquer le répertoire des fichiers audio[14].

9.4.4 Une interpolation par voisinage

Pour Ultraviolette, le fonctionnement de la pédale A se distingue de celui des autres logiciels par un comportement plus complexe. En effet, sa course se divise en 5 zones selon le patch `[uv-interpolateur]` (cf. figure 9.17 page suivante) et le tableau 9.2 :

	intervalle	*taille*	*variable*	*effet*
5	[122; 127]	5	*uv-suivant*	rotation anti-trigonométrique
4	[68; 121]	54	*uv-au-dessus*	interpolation avec *uv-suivant*
3	[59; 67]	9	*uv-calife*	retour à la configuration courante
2	[5; 58]	54	*uv-en-dessous*	interpolation avec *uv-precedent*
1	[0; 4]	5	*uv-precedent*	rotation trigonométrique

TAB. 9.2 – *Les 5 zones de la pédale A pour Ultraviolette*

Parmi ces 5 zones, 3 d'entre elles sont consacrées au mécanisme de rotation (zones 1, 3 et 5), tandis que les 2 zones intermédiaires (1 et 4) sont dédiées à l'interpolation ; d'ailleurs ces dernières zones occupent à elles deux la plus grande part du curseur (108 valeurs sur 128, soit 84%), de façon à préserver une progressivité dans les interpolations exécutées avec la pédale d'expression.

Avec les 3 courtes zones de rotation, la pédale A permet de parcourir le cercle des boutons dans un sens ou dans l'autre, avec deux particularités : d'une part le bouton n° 1 (marche/arrêt) est exclu de ce parcours, et d'autre part la combinaison du parcours circulaire avec la métaphore du pédalier provoque deux autres discontinuités dans le parcours, soient les transitions 5–10 et 6–2. Au total, le parcours emprunte l'ordre suivant, dans les deux sens : … 8, 9, 10, 5, 4, 3, 2, 6, 7… ce qui correspond à l'ordre des voisins sur le cercle des boutons, en partant du bouton le plus haut.

Le principe de la rotation est simple : si le curseur atteint son extrémité supérieure (la zone 5), alors le voisin suivant dans le sens anti-trigonométrique (*i. e.* dans le sens des

14. Ce répertoire peut éventuellement contenir un grand nombre de fichiers, il en contient ici 680 (cf. figure 9.16).

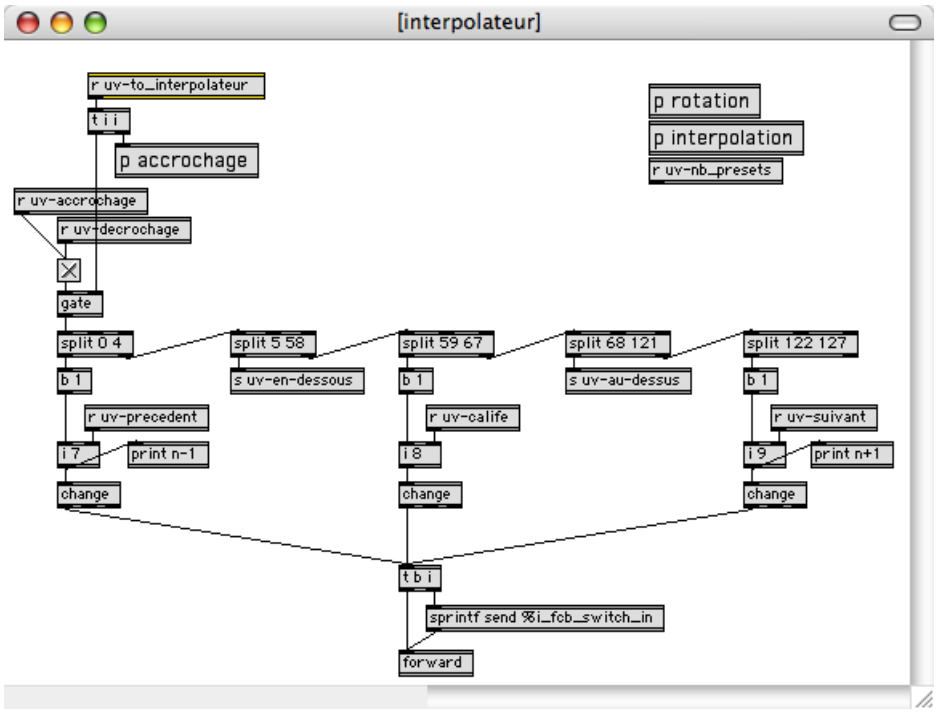


FIG. 9.17 – *Patch d'interpolation d'Ultraviolette*

aiguilles d'une montre) devient la nouvelle configuration courante et le curseur décroche ; réciproquement, s'il atteint son extrémité inférieure (la zone 1), alors le voisin précédent devient la nouvelle configuration courante et le curseur décroche. Pour raccrocher le curseur, il suffit ensuite de retrouver le centre du curseur (la zone 3). Par ailleurs, les boutons restent aussi en accès direct, ce qui permet de se déplacer n'importe où sur le cercle sans passer par le curseur.

Pour finir, ci-suit le code source du script qui dispose les 10 boutons en cercle sur la fenêtre principale du patch Ultraviolette, en javascript. En particulier, la fonction `boutons()`, à partir de la ligne 63, calcule les coordonnées des boutons, selon qu'ils appartiennent au demi-cercle inférieur ou supérieur.

Code 9.1 – *Positionnement circulaire des 10 boutons d'Ultraviolette, en javascript*

```
1  // uv- interface_lisse . js
2  //
3  // génère automatiquement 10 boutons
4  // disposés en cercle autour de l'objet
5  // nommé "tempo"
6  //
7  // kb 20080606
8  //
9
10 // inlets et outlets
11 inlets = 1;
12 outlets =1;
13 autowatch = 1;
14
15 // variables globales
16 var numboutons = 0;
17 var cx = 0;
18 var cy = 0;
19 var rBouton = 90 / 2;
20 var rCercle = 200;
21 var nbBoutons = 10;
22
```

```
23 // Maxobj variables
24 var lesboutons = new Array(10);
25 var lesPanel = new Array(10);
26 var lesNbbox = new Array(10);
27 var lesUbutton = new Array(10);
28 var lesComment = new Array(10);
29 var lesSendDown = new Array(10);
30 var lesSendUp = new Array(10);
31
32
33 // méthodes
34
35 gettempo.local = 1;
36 function gettempo()
37 {
38   tempo = this.patcher.getnamed("tempo");
39   tx1 = tempo.rect[0];
40   ty1 = tempo.rect[1];
41   tx2 = tempo.rect[2];
42   ty2 = tempo.rect[3];
43   cx = tx1 + ((tx2 - tx1) / 2);
44   cy = ty1 + ((ty2 - ty1) / 2);
45   // post("cx = " + cx + ", cy = " + cy + ", pi = " + Math.PI + "\n");
46 }
47
48 function efface_tout()
49 {
50   var i = 0;
51   for(i=1;i<11;i++)
52   {
53     if (b = this.patcher.getnamed(i+"_uv_panel")) this.patcher.remove(b);
54     if (b = this.patcher.getnamed(i+"_uv_ubutton")) this.patcher.remove(b);
55     if (b = this.patcher.getnamed(i+"_uv_comment")) this.patcher.remove(b);
56     if (b = this.patcher.getnamed(i+"_uv_nbbox")) this.patcher.remove(b);
57     if (b = this.patcher.getnamed(i+"_uv_send_up")) this.patcher.remove(b);
58     if (b = this.patcher.getnamed(i+"_uv_send_down")) this.patcher.remove(b);
59   }
60 }
61
62 // boutons -- génère les 10 boutons en cercle dans le patch
63 function boutons()
64 {
65   var k = 0;
66   var x = 0;
67   var y = 0;
68   var d = 0;
69
70   gettempo();
71
72   for(k=0;k<5;k++) // dans le sens trigonométrique
73   {
74     a = Math.PI * (nbBoutons + 1 + (2 * k)) / nbBoutons;
75     r = rCercle;
76     if (k==0) r += rBouton * 2; // bouton 1 décalé du cercle
77
78     x = parseInt(r * Math.cos(a) + cx - rBouton);
79     y = parseInt(r * -1 * Math.sin(a) + cy - rBouton);
80     //post("k, x, y = " + k + ", " + x + ", " + y + "\n");
81
82     constructionBouton(k, x, y)
83   }
84
85   for(k=5,d=9;k<10;k++,d-=2) // dans le sens inverse
86   {
87     a = Math.PI * d / nbBoutons;
88     x = parseInt(rCercle * Math.cos(a) + cx - rBouton);
89     y = parseInt(rCercle * -1 * Math.sin(a) + cy - rBouton);
90     //post("k, x, y = " + k + ", " + x + ", " + y + "\n");
91
92     constructionBouton(k, x, y)
93   }
94   numboutons = 10;
95 }
96
```

```
97  // appels successifs des constructeurs, dans l'ordre des 'bringtofront'.
98  function constructionBouton(k, x, y)
99  {
100   creationPanel(k, x, y);
101   creationComment(k, x, y);
102   creationNbbox(k, x, y);
103   creationSendDown(k, x, y);
104   creationSendUp(k, x, y);
105   creationUbutton(k, x, y);
106   creationConnexions(k, x, y);
107 }
108
109 function creationPanel(k, x, y)
110 {
111   var i = k + 1;
112   var p = this.patcher;
113
114   lesPanel[k] = p.newdefault(x, y, "panel");
115   lesPanel[k].rect = [x, y, x+90, y+90];
116   lesPanel[k].varname = i+"_uv_panel";
117   lesPanel[k].message("rounded", 90);
118   lesPanel[k].message("border", 0);
119   lesPanel[k].message("shadow", 5);
120   p.bringtofront(lesPanel[k]);
121 }
122
123 function creationUbutton(k, x, y)
124 {
125   var i = k + 1;
126   var p = this.patcher;
127
128   lesUbutton[k] = p.newdefault(x, y, "ubutton");
129   lesUbutton[k].rect = [x, y, x+90, y+90];
130   lesUbutton[k].varname = i+"_uv_ubutton";
131   p.bringtofront(lesUbutton[k]);
132 }
133
134 function creationSendDown(k, x, y)
135 {
136   var i = k + 1;
137   var p = this.patcher;
138
139   lesSendDown[k] = p.newdefault(x+26, y+100, "send", i+"_uv_ubutton_down");
140   lesSendDown[k].varname = i+"_uv_send_down";
141   lesSendDown[k].hidden = true;
142   p.bringtofront(lesSendDown[k]);
143 }
144
145 function creationSendUp(k, x, y)
146 {
147   var i = k + 1;
148   var p = this.patcher;
149
150   lesSendUp[k] = p.newdefault(x, y+118, "send", i+"_uv_ubutton_up");
151   lesSendUp[k].varname = i+"_uv_send_up";
152   lesSendUp[k].hidden = true;
153   p.bringtofront(lesSendUp[k]);
154 }
155
156 function creationConnexions(k, x, y)
157 {
158   var i = k + 1;
159   var p = this.patcher;
160
161   p.hiddenconnect(lesUbutton[k], 0, lesSendUp[k], 0);
162   p.hiddenconnect(lesUbutton[k], 1, lesSendDown[k], 0);
163 }
164
165 function creationComment(k, x, y)
166 {
167   var i = k + 1;
168   var p = this.patcher;
169
170   lesComment[k] = p.newobject("number", x+30, y+10, 51, 24, 0, 0, 8352, 3, 0, 0, 0, 221, 221, 221, 222, 222, 222, 0, 0,
```

```
        0);
171   lesComment[k].varname = i+"_uv_comment";
172   //lesComment[k].message("flags", 32 + 128 + 8192);
173   p. bringtofront (lesComment[k]);
174 }
175
176 function creationNbbox(k, x, y)
177 {
178   var i = k + 1;
179   var p = this.patcher;
180
181   lesNbbox[k] = p.newobject("number", x+20, y+50, 51, 14, 0, 0, 8352, 3, 0, 0, 0, 221, 221, 221, 222, 222, 222, 0, 0,
        0);
182   //lesNbbox[k]. rect = [x, y, x+90, y+90];
183   lesNbbox[k]. varname = i+"_uv_nbbox";
184   //lesNbbox[k]. message("flags", 32 + 128 + 8192);
185   p. bringtofront (lesNbbox[k]);
186 }
```

* * *

La collaboration avec Iván en tant que compositeur n'ayant débuté que récemment, les trois pièces prévues – *Kyōgen* pour Plugiscope, *Ki* pour Ifso et *Corps Noir* pour Ultraviolette – ne sont pas encore terminées. Néanmoins, les trois logiciels nous semblent suffisamment prometteurs, c'est-à-dire réfléchis et aboutis, pour pouvoir espérer les entendre bientôt en concert.

Chapitre 10

Réalisateur en informatique musicale ?

Résumé du chapitre : Ce bref chapitre final constitue un « pas de côté » dans la thèse, un regard sur mon travail de programmation en collaboration avec les compositeurs et les musiciens. Ce faisant, il tentera de dégager quelques points de réflexion autour de la collaboration en tant que telle dans le domaine de l'informatique musicale, à partir des expériences exposées dans les chapitres précédents.

10.1 Qu'est-ce qu'un RIM ?

L'annexe (section B.3 page 275, qui rapporte un entretien approfondi avec Iván Solano, aborde déjà la question du réalisateur en informatique musicale par rapport à la collaboration avec le compositeur, à travers notre propre collaboration.

Par ailleurs, l'annexe G page 331 reproduit un texte tout à fait éclairant quant au métier de RIM et aux problématiques qui s'y rapportent, extrait de la brochure des *Journées professionnelles sur le métier de réalisateur en informatique musicale* qui se sont tenues à l'IRCAM les 23 et 24 juin 2007. Ce texte présente d'abord le RIM comme « un interlocuteur privilégié » du compositeur lors du « travail dans un studio », puis complète la liste de ses fonctions (pour les années 90 et à l'IRCAM) par : « le réalisateur devait aussi assumer de façon plus affirmée l'encadrement des compositeurs, la gestion les projets de production et, enfin, assurer, en collaboration avec l'ingénieur du son et le compositeur, l'exécution de l'œuvre ». Ce texte indique enfin que le RIM n'est pas une exclusivité de l'IRCAM, mais qu'il n'existe toujours pas une « identité professionnelle partagée », ni de « reconnaissance publique du métier ».

Aussi, afin de préciser davantage le profil du RIM, voici l'extrait d'un entretien donné par Franck Madlener, nouveau directeur de l'IRCAM depuis 2006, au journal *ResMusica* [Madlener, 2006][1] :

> *ResMusica : L'activité de l'*IRCAM *s'inscrit-elle maintenant autour de l'assistant musical ?*
>
> *Franck Madlener* : Plus précisément, l'assistant est le médiateur par excellence, doté de la culture du singulier artistique et de la généralité scientifique. À partir d'un projet en train de se concevoir, il est le premier à pouvoir identifier des éléments appelés à devenir des outils communs.
>
> *RM : L'assistant musical, est-il un musicien ou un scientifique ?*
>
> *FM* : Il n'y a pas de règle. En fait, si l'IRCAM a inventé une fonction dans la création, c'est bien ce personnage. Il peut être un musicien, qui a la fibre scientifique, ou un chercheur, qui cherche à établir des contacts beaucoup plus directs avec la création. Il peut venir des mondes informatique ou scientifique qui travaillent sur les formalismes, la logique. Outre la diversité des profils, l'assistant musical doit être mobile. Passant par le renouvellement des compositeurs, la nouvelle politique de l'IRCAM sous-tend un renouvellement de l'élaboration des œuvres. Dans d'autres cultures, par exemple en Allemagne, le compositeur et l'assistant musical entretiennent des échanges extrêmement étroits, presque d'égal à égal, relations dont nous pourrions nous inspirer. Autre particularité, ici, compositeurs et assistants musicaux forment de vieux couples, Andrew Gerzso / Pierre Boulez, Eric Daubresse / Emanuel Nunes, Serge Lemouton / Philippe Manoury, ce qui a produit le meilleur, œuvres, logiciels, façons de travailler. Néanmoins, un assistant musical a aussi besoin de prendre l'air, de se retrouver de temps à autre face à des expériences différentes, de créer une courroie de transmission avec d'autres studios. L'IRCAM doit être capable de prendre en charge une partie de l'histoire de ce qui se fait dans d'autres

1. Cet article de 2006 emploie encore l'expression « assistant musical » qui est aujourd'hui remplacée par « réalisateur en informatique musicale », sans doute à cause de la connotation péjorative de l'« assistanat ». Cette question de la dénomination de ce métier est toujours ouverte, d'après les débats qui se sont tenus aux *Journées professionnelles sur le métier de réalisateur en informatique musicale*, les 23 et 24 juin 2007 à l'IRCAM.

> studios, comme celui de la Wdr à Cologne ou le *Heinrich Strobel* à la Swr de Freiburg. Ces échanges nous permettraient aussi de tirer profit de l'expérience des autres centres de recherche.

On peut retenir de cet extrait deux points qui semblent importants : le rôle de médiateur d'une part, et l'existence de « vieux couples » d'autre part. Ce dernier point doit révéler quelque chose à mon sens, car une telle longévité ne peut pas être neutre. Je pense en effet que le tandem est la plus petite formation viable lorsqu'on considère des projets d'une certaine ampleur, en particulier quand on mêle musique instrumentale et informatique, parce que les compétences nécessaires dans les deux domaines demandent de nombreuses années de formation et de pratique. Autrement dit, il me semble qu'un projet mixte qui souffrirait d'une faiblesse trop importante dans l'un des deux domaines risque fort d'être immédiatement rejeté.

Cela n'empêche pas le compositeur d'avoir des compétences en informatique, ni le réalisateur d'avoir des connaissances en musique : au contraire, pour pouvoir se comprendre, un minimum reste nécessaire. D'ailleurs, Eric Daubresse rappelle :

> L'assistant doit au contraire tout faire pour maintenir sa part de créativité car il ne saurait être considéré comme un simple technicien. Ce que les compositeurs reconnaissent volontiers mais ce dont le public n'a en général pas conscience car il cesse parfois d'applaudir lorsque l'assistant musical vient saluer en se demandant qui il est !

Et Emmanuel Nunes d'ajouter :

> Quand je parle musique à l'assistant, il est fondamental pour moi de ne rien avoir à expliquer. Avec Eric, on s'entend au premier mot car il connaît ma pensée de l'intérieur.

Ces deux citations sont d'ailleurs extraites d'un article dont le titre, en forme d'aphorisme, pourrait résumer la fonction du Rim : *L'assistant musical, trait d'union libre entre recherche et création* [Gervasoni, 2000]. Ainsi, sauf pour quelques rares personnalités qui jouissent à la fois d'un don pour l'orchestration et d'une science de la programmation, il me semble que la collaboration reste une stratégie efficace face aux défis posés par la création de musique électroacoustique, et à plus forte raison par la musique interactive.

Cependant, la croyance en l'émergence d'un véritable métier doit être tempérée : il reste très difficile de se prononcer sur l'avenir du Rim en tant que profession, tant cela concerne une population confidentielle et tant l'informatique et ses pratiques progressent rapidement. D'aucuns pensent même que ce métier est déjà obsolète devant l'ergonomie croissante et le coût décroissant des systèmes informatiques personnels, convaincus que les compositeurs apprendront dorénavant la programmation comme ils apprennent le solfège. D'autres soutiennent que ce métier ne peut qu'émerger devant la complexité croissante de la recherche scientifique et de la création musicale. Pour ma part, je trouverais déraisonnable de me prononcer sur une question finalement spéculative.

10.2 Une brève réflexion sur mon rôle

À défaut de pouvoir répondre à la question de l'avènement du métier de Rim, je peux essayer d'aborder la question plus restreinte de ma place dans les différentes collaborations auxquelles j'ai participées au cours de mon doctorat. Bien entendu, cette tentative possède une limite non négligeable, étant donné qu'il est impossible de développer un point de vue objectif depuis l'intérieur d'une relation, cependant mon point de vue subjectif pourrait présenter un intérêt, à condition qu'il soit lu comme tel.

10.2.1 Rien que la programmation

D'abord, ma double formation informatique et musicale (cf. section 4.2.1 page 89) me rapproche du profil du Rim, qui doit impérativement posséder la double compétence. Pierre Gervasoni note à ce sujet qu'« il n'est pas d'assistant musical qui ne soit compositeur, même si les parcours divergent » [Gervasoni, 2000].

Ensuite, dans chaque collaboration, j'ai clairement investi la programmation – et uniquement la programmation – autant que possible : de fait, j'avais décidé dès le début d'écarter à la fois la composition et l'interprétation de mon travail au sein de ces collaborations, et je m'y suis tenu jusqu'au bout, suffisamment occupé d'ailleurs par la programmation des logiciels et par l'écriture de la thèse... Ainsi, j'ai effectivement réalisé la partie informatique, comme *médiateur* entre le projet musical du compositeur ou de l'instrumentiste et l'ordinateur, ce qui rapproche effectivement mon travail de celui d'un réalisateur en informatique musicale.

Est-ce à dire que j'ai abandonné l'idée de composer ? Non : je crois, pour moi comme pour d'autres programmeurs en informatique musicale, qu'il s'agit plus d'une étape de maturation que d'un « piège ». C'est aussi ce que note Vlad Spear[2] à propos de son travail de programmation :

> *Spending all your time creating music-making tools in Max/Msp and none of your time actually making music is a well-known trap amongst the patcher obsessed, but I see it as part of one long compositional process.*

10.2.2 Un « sur-mesure » effectivement chronophage

J'ai effectivement consacré un temps important à la programmation des logiciels, un temps globalement équivalent à celui que j'ai consacré à l'écriture du mémoire de thèse. Le respect de l'idée du « sur-mesure » (cf. section 4.3.1 page 91), appliqué à chaque collaboration, explique en partie cette durée conséquente.

En effet, lorsque je programme pour moi, j'adopte naturellement une certaine souplesse avec mes intentions de départ, en les accommodant aux tendances de l'environnement de

2. Dans un entretien qu'il donne à Marsha Vdovin, posté le 24 septembre 2007 sur le site de *Cycling'74*, http://www.cycling74.com/?op=displaystory;sid=2007/9/24/12623/3682.

programmation, pourvu qu'une autre forme de satisfaction arrive à émerger. Or cette souplesse par rapport aux buts d'origine s'évanouit presqu'intégralement lorsque je travaille pour quelqu'un d'autre, et l'inverse se produit alors : j'essaie d'accorder l'environnement de programmation aux intentions stipulées dans le cahier des charges, quitte parfois à devoir opérer des contorsions malcommodes[3].

Bien sûr, durant ce doctorat, les cahiers des charges ont été élaborés en concertation avec moi alors que j'avais déjà une connaissance des limites et des possibilités de *Max/MSP*, et c'est d'ailleurs pourquoi la plupart des logiciels que j'ai développés ici sont orientés vers le temps réel, car ce dernier reste le champ d'application nettement privilégié de cet environnement. Néanmoins, il me semble que j'ai plutôt eu tendance à essayer de relever ce qui m'apparaissait comme difficile pour moi avec *Max/MSP*, par curiosité et surtout par empathie avec le projet esthétique de mon partenaire, une fois que j'avais compris sa logique interne.

10.2.3 Un fort potentiel d'innovation

D'une manière générale, je retiens de ces expériences que la collaboration entre un programmeur et un spécialiste d'un autre domaine crée une dynamique à fort potentiel d'innovation.

Il y a évidemment quelques conditions pour que ce potentiel puisse se réaliser, comme le temps disponible, les outils, une bonne entente générale, le degré de compréhension mutuelle quant au projet, mais au-delà de ces considérations, si le temps dévolu au projet est suffisant, un facteur d'innovation assez paradoxal peut intervenir : la *naïveté* face à l'informatique de celui qui ne progamme pas, une naïveté qui doit cependant être contrebalancée par une compétence solide dans l'autre domaine pour cette même personne.

Ce facteur qui m'était d'abord apparu comme un handicap, typiquement à cause des demandes que je trouvais parfois farfelues en connaissant les limites de l'informatique, s'est progressivement révélé fécond, d'une part en remettant fréquemment en question mes croyances ou mes préjugés sur ce qu'il était possible de faire ou pas avec l'informatique, me procurant ainsi une forme de *désinhibition* face à la machine, et d'autre part en imposant un cycle de validation particulièrement exigeant et large. Par exemple, ce cycle garantit finalement une certaine qualité à des niveaux auxquels un informaticien ne prette pas forcément une attention aussi aiguë lorsqu'il travaille seul, comme l'interface graphique, l'ergonomie dans tous ces détails, et l'expérience utilisateur en général. Or, comme Hugues Vinet le précise : « les interfaces graphiques constituent, du point de vue du concepteur informaticien, le support de médiation entre l'utilisateur et les fonctions de l'application informatiques. » [Vinet, 1999, p. 100] Cet aspect de la « convivialité » d'un logiciel, ce que les anglo-saxons ont consacré par l'expression *user-friendly*, quitte alors son statut trop souvent facultatif pour devenir proprement crucial, c'est-à-dire à la croisée entre les chances de survie du logiciel et l'expérience de utilisateur avec ce logiciel. Les

3. Il faut d'ailleurs saluer le travail de programmation remarquable parce qu'à contre-courant de Carlos Caïres avec son logiciel de micromontage *Irin*, au titre de la spécialisation de *Max/MSP* vers le temps réel, l'activité de micromontage relevant davantage d'une pratique en temps différé, ce qui entraîne la multiplication de contorsions de programmation virtuoses « contre-nature » pour *Max/MSP*.

interfaces visuelles d'Iviv, de Rose amère, d'Ifso et d'Ultraviolette témoignent des souhaits graphiques très personnels de leur commanditaire respectif. Je crois qu'en art plus que dans d'autres domaines ce paramètre visuel se révèle déterminant, parce qu'il pèse immédiatement en favorisant ou en défavorisant des processus cognitifs comme l'inspiration en des termes qualitatifs, et non pas simplement sur la productivité en termes quantitatifs.

Travailler avec un initié interroge sans doute moins les limites de la programmation que travailler avec un néophyte, mais, dans les deux cas, la dissociation des tâches entre celui qui programme et celui qui dirige le projet artistique présente de toutes façons plusieurs avantages majeurs :

- la complémentarité des points de vue, ou plus exactement la complémentarité des « niveaux » de vue ;
- le décalage chronologique des suivis, frais pour l'un et éprouvé pour l'autre, pour la programmation comme pour la composition ;
- le recul du compositeur qui contourne la perception nécessairement biaisée du programmeur, due à l'inévitable rétrécissement du jugement global lorsqu'on baigne un certain temps dans la réalisation de choses locales ;

Je ne prétends pas qu'il est impossible de réaliser seul un projet de musique avec informatique, mais simplement que, dans la pratique, la dissociation des points de vue « microscopique » et « macroscopique » par rapport au logiciel facilite la réalisation du projet et améliore sa qualité ; d'un côté celui qui programme, plongé dans le code, est relativement épargné par les doutes d'ordre esthétique, et de l'autre côté celui qui ne programme pas, épargné par les problèmes de la programmation, ne perd pas sa hauteur de vue sur les directions artistiques, d'autant plus qu'elles sont nécessairement en mouvement dans ce type de projet.

10.2.4 La boucle collaborative énaction / résolution

De fait, comme de nombreux informaticiens[4], je conçois l'activité de programmation comme étant essentiellement une activité de résolution de problèmes, mais je considère plusieurs étapes dans mes collaborations dont :

- la détection et le filtrage des attentes informatiquement légitimes ou illégitimes ;
- la traduction des attentes légitimes dans un formalisme informatique (incluant la modélisation calculable et la transcription syntaxiquement correcte) ;
- la chasse aux inévitables bogues ;
- le détournement éventuel d'outils ;
- la vérification de l'adéquation entre le programme et le cahier des charges ;
- la maintenance du système informatique ;
- en outre, la compréhension de l'esthétique du partenaire constitue encore un autre type de problème à résoudre et non des moindres, à la fois indispensable pour pouvoir collaborer et impossible à réaliser totalement...

De ce point de vue, c'est-à-dire de l'informatique comme activité de résolution de problèmes, le collaborateur idéal de l'informaticien devient celui qui peut *soulever* des problèmes, c'est-à-dire qui possède la compétence du « soulevage[5] » de problèmes (*« problem-raising »*), qui intervient nécessairement avant celle de la résolution de problèmes (*« problem-solving »*). C'est dans ce couplage-là que la « partie de ping-pong » évoquée à la section 5.2 (page 106) acquière le plus grand potentiel morphophorique.

* * *

4. Par exemple, Marc Zanoni précise dans sa thèse – *La récursivité en programmation, l'abord psychologique* – que la programmation est une « activité de résolution de problème impliquant le traitement d'un problème "du monde" au moyen d'outils et de concepts calculables par un dispositif informatique. » [Zanoni, 1998]. D'une façon plus approfondie, Harald Wertz synthétisait quant à lui une première définition de la notion de problème rapportée à la programmation, dans son article *Quelques remarques sur le cadre psychologique de la programmation* [Wertz, 1983, p. 3-4] :

> Puisqu'une des hypothèses de départ de tout notre travail est que la programmation est une activité de *résolution de problèmes* nous aimerions, afin de préciser notre sujet, donner une courte définition de la notion de *problème*.
>
> Nous dirons que nous sommes confrontés à un *problème*, si nous avons un état initial α que nous voulons transformer en un état final β, et que nous ne savons pas immédiatement quelle série d'actions peut réaliser cette transformation.
>
> $$\begin{array}{rcl} \alpha & \longrightarrow & \beta \\ \text{état initial} & \longrightarrow & \text{état final} \end{array}$$
>
> Autrement dit : un *problème* est la recherche d'une suite d'*opérateurs* qui transforment un état initial α donné dans un état final β désiré. Chacun des *opérateurs* est soit un *opérateur primitif*, soit une *combinaison* d'opérateurs primitifs. Les objets définissant l'état initial et l'état final peuvent être – en général – de n'importe quel type (des symboles mathématiques, des figures du jeu d'échec, les briques pour construire une maison, etc.).

5. La traduction française n'est pas aussi évidente pour désigner le concept *« problem-raising »* (nommé parfois *« problem-arising »*), que pour *« problem-solving »* par « résolution de problèmes ». Nous proposons « soulevage de problèmes » ou bien « énaction de problèmes » (pour insister davantage sur le caractère émergent).

Loin d'incarner l'unique façon de faire, ce mode de travail en tandem a porté ses fruits pour la plupart des projets entrepris ici, et si tous n'ont pas débouché sur une structure fonctionnelle qui se rapprocherait de celle des couples « mythiques » de l'informatique musicale, il me semble que tous ont au moins bénéficié d'une véritable synergie, à la suite de cet équilibre dissocié entre *celui-qui-programme* et *celui-qui-compose* ou *celui-qui-va-improviser*, et que certaines de nos collaborations ont même bénéficié d'une stimulation féconde et souvent réciproque, dans une sorte de spirale dynamique auto-entretenue, à travers les échanges.

10.3 Éloge des impossibles traductions

L'aspect de division du travail, le *co-labeur*, ne constitue qu'un avantage quantitatif dont l'intérêt s'efface, à mon sens, devant l'apport qualitatif de la *co-élaboration.* De mon point de vue, une bonne partie de l'intérêt intellectuel des collaborations réside dans l'impossible traduction du souhait de mon partenaire à moi-même, puis à l'environnement de programmation dans son « pauvre » langage informatique...

Dans un premier temps, je dois retraduire à mon collaborateur ce que j'ai compris de son souhait pour vérifier que j'ai compris l'idée centrale de l'instant, ce qu'il doit lui-même retraduire avant de me répondre. À ce stade, rien de cette idée n'est encore programmé.

Au cours de ce processus, écrire et dessiner sur papier est un lieu d'échange privilégié pour des raisons pratiques évidentes : le support est stable dans le temps immédiat, ce qui n'est pas le cas de nos mémoires, qui doutent, sans compter que le graphisme possède intrinsèquement une suggestivité et une éloquence souvent supérieures aux tentatives verbales équivalentes. Malgré cette stabilité salutaire du papier, nous sommes conscients que nous ne nous comprendrons jamais exactement, et ce hiatus forme justement une source importante de créativité : car dans la tentative d'explication ou d'explicitation à l'autre d'une idée, jaillissent souvent d'autres idées, et bientôt des réseaux entiers d'associations d'idées.

Ces flots nous submergent individuellement fréquemment, ce qui ralentit encore la traduction par réduction de notre disponibilité intellectuelle, occupés que sommes alors à mémoriser et à faire le tri dans ces émergences intempestives. La rédaction – d'un article, d'un courrier électronique, d'un passage de la thèse – intervient devant ces chaos provisoires comme une mise en forme et une fixation de nos pensées pour enfin devenir des pensées à part entière (non fugitives) et parfois neuves à notre échelle.

Il s'agit donc d'un cycle hautement créatif, au-delà d'une simple émulation, la discipline imposée par les traductions multiples et nécessaires générant de la pensée « nouvelle » au gré des formulations, des formules, des reformulations, des associations d'idées, des prises de conscience, dans l'expérience des différences et des divergences. L'autre, nous réfléchissant déformée notre réflexion, nous fait immanquablement réfléchir ; et réciproquement.

Dans ce mode de création à deux (le travail), plus qu'une simple *création-à-deux* (l'œuvre), le langage et l'expression occupent évidemment une place importante. En effet, le langage supportant la pensée, toute tentative de reformulation, et encore davantage face à un constat d'incompréhension, entraîne une réflexion sur le choix des mots, et parfois

une seconde réflexion à propos des origines possibles de l'incompréhension. Ainsi, tenter de préciser sa pensée à l'autre, ça n'est pas dire la même chose d'une façon différente, mais plutôt se mettre à penser différemment de notre pensée première, ce qui revient souvent à préciser sa pensée à soi-même. En définitive, l'impossible traduction se révèle être, par suite et pour chacun, une puissante stratégie individuelle de création, que le travail solitaire ne possède pas.

Conclusion

« Entre temps réel et temps différé » : voilà sans doute la condensation la plus brève de la thèse développée ici. Effectivement, toute la première partie théorique s'est attachée à montrer qu'un dépassement de la dichotomie traditionnelle du temps réel et du temps différé est possible en informatique musicale – précisément dans le champ sémantique *musical* de ces expressions – en formalisant un espace *entre* temps réel et temps différé – un axe « latentiel ». La seconde partie illustre quant à elle l'existence de cet espace entre temps réel et temps différé, avec nos logiciels qui sont à la fois *performables* et *composables*.

Bilan

Un axe « latentiel »

Nous avons commencé par une approche historique à plusieurs fins : premièrement pour montrer l'ontologie informatique des notions de temps réel et de temps différé en général (en regard de l'ancrage analogique des notions antérieures de direct et de différé), deuxièmement pour dater l'apparition de ces notions au début des années 60, troisièmement pour rappeler que la dichotomie historique s'est produite en informatique musicale dans le champ *technique* (avec la synthèse sonore), et quatrièmement pour évoquer le glissement sémantique de cette dichotomie vers le champ *musical* dans les discours sur la musique contemporaine (dès la fin des années 70).

Puis nous avons développé une critique du cloisonnement de ces deux catégories. En effet, l'étude du seuil censé séparer le temps réel du temps différé – la latence – révèle deux difficultés : une frontière floue, lorsqu'elle repose sur des critères perceptifs, et des définitions multiples, éventuellement incompatibles entre elles. Ce flou nous a amené à distinguer trois champs lexicaux concernant le temps réel et le temps différé : le champ *technique*, qui se rapporte au taux d'échantillonnage, le champ *pratique*, qui se rapporte à la boucle perception/action de l'interaction, et le champ *musical*, qui se rapporte à la fonction musicale en tant que choix esthétique. Une analyse de quelques contradictions apparentes (telles que la préparation souvent conséquente nécessaire aux œuvres en temps réel ou que le fort degré d'interactivité des logiciels actuellement conçus pour le temps différé), permet de valider la polysémie dégagée dans cette partie critique.

Enfin, nous avons proposé une représentation susceptible d'incarner cette posture non dichotomique : un axe « latentiel ». Il s'agit d'un axe tendu entre le temps réel et le temps différé considérés comme deux pôles idéaux (c'est-à-dire, schématiquement, entre une latence nulle et une latence infinie). Ensuite, à défaut de pouvoir munir cet axe d'une

norme, il a été muni d'une relation d'ordre qui fonctionne terme à terme, engendrant un ordonnancement *relatif*. Afin d'illustrer cette représentation axiale, un logiciel d'aide au classement relatif a été développé et appliqué dans trois analyses axiales : sur nos logiciels musicaux, sur leurs fonctionnalités, et sur un annuaire généraliste de logiciels musicaux.

Des projets musicaux intégrant temps réel et temps différé

À l'image des logiciels musicaux, qui intègrent à la fois des aspects temps réel et temps différé tout en se distinguant les uns des autres sur l'axe latentiel, nos projets musicaux intègrent aussi ces deux aspects tout en se distinguant les uns des autres par rapport à ces mêmes aspects. Autrement dit, toutes les musiques qui ont été créées au cours de ce doctorat intègrent à la fois du temps réel et du temps différé.

En effet, d'une part, chacun des projets développés ici implique le jeu d'un instrumentiste avec son propre instrument acoustique et confie l'interprétation de la partie informatique à ce même instrumentiste, via le pédalier multiple. Cette interprétation numérique signe la dimension *performative* de ces projets musicaux. D'autre part, ces mêmes projets font tous appel à des logiciels dont les préréglages restent à composer, avec une certaine finesse d'ailleurs, ce qui signe leur dimension *compositionnelle* (au niveau numérique, s'ajoutant à la partition traditionnelle pour ceux qui en utilisent). Ainsi, chaque projet musical se situe de fait quelque part entre temps réel et temps différé, dissocié des extrémités, à une place relative à l'ensemble des projets considérés.

La CBox, développée avec Mario Lorenzo, reste le logiciel qui possède le plus fort degré de composabilité numérique, tandis que Mimi et Rose amère, les deux logiciels développés avec l'altiste Stéphanie Réthoré s'inscrivent historiquement uniquement dans la perspective de l'improvisation (aujourd'hui, une pièce écrite pour alto et Rose amère n'est cependant pas exclue). Incidemment, le logiciel Iviv développé avec Santiago Quintáns a montré la plasticité de ces logiciels face à la musique écrite et à la musique improvisée, lors d'un même concert qui a fait entendre non seulement sa pièce *In Vivo / In Vitro* pour caisse claire augmentée, mais aussi une improvisation à la guitare et une seconde improvisation au vibraphone, avec le même logiciel Iviv…

Le projet développé avec Mauricio Meza, *Woes-war-sollichwer-den*, utilise largement différents ressorts du temps réel et du temps différé, avec les quatre logiciels spécifiques : a2m pour générer du matériau précompositionnel, Loterie pour tirer au sort parmi la partition modulaire, Emzed pour l'instrumentiste, et Nappe pour générer des textures sonores en temps réel. Quant aux projets développés avec Iván Solano, ils utilisent tous trois le dispositif podophonique interactif : Plugiscope pour contrôler un plugiciel, Ifso pour déclencher et contrôler la lecture de fichiers audio, et Ultraviolette pour contrôler le tempo du générateur de flux rythmique.

En dernier lieu, une réflexion sur le réalisateur en informatique musicale rappelle qu'en amont de la réalisation de ces logiciels, l'essentiel de nos efforts a résidé dans la mise en relation entre le programmeur et le musicien, relation à laquelle nous croyons dans le domaine de la création musicale[6].

6. Comme le soutient Daniel Teruggi : « L'originalité dans une démarche en quête de nouveaux outils aptes

Perspectives

Perspectives théoriques

L'élaboration théorique principale entreprise dans cette thèse – à savoir le passage d'une dichotomie ensembliste à une liste ordonnée – pourrait ouvrir la voie à un prochain travail de catégorisation beaucoup plus fine que les deux catégories de départ, à partir de l'axe temps réel / temps différé. En effet, la représentation ordonnée établie dans la première partie – l'axe *latentiel* – offre un support susceptible d'inciter au regroupement d'éléments voisins sur l'axe, sachant que le regroupement est une opération profondément naturelle :

> Grouper les « choses » (dans le sens le plus général) constitue l'élément le plus profond, le plus indispensable, de notre perception et de notre conception du réel. Bien qu'il soit évident que jamais deux choses ne seront exactement identiques, c'est parce qu'on ordonne le monde en groupe d'éléments ayant en commun une propriété importante (groupes qui se recoupent d'une façon complexe et se superposent) qu'on donne une structure à ce qui ne serait autrement qu'un chaos, une fantasmagorie. [Watzlawick *et al.*, 1975, p. 28]

Le premier pas vers une organisation *latentielle* a été franchi avec notre idée de coupler une relation d'ordre terme à terme avec deux pôles idéaux, permettant de sortir de la logique d'exclusion du temps réel et du temps différé pour entrer dans une logique d'inclusion. Mais un deuxième pas, précisément celui de choisir et de justifier des regroupements plus fins, reste à faire dans une prochaine recherche qui s'appuierait sur le classement axial.

À notre connaissance, il existe d'ailleurs au moins un tel type de travail de regroupement dans un autre domaine : l'analyse méthodique des distances inter-individuelles par Edward T. Hall dans son essai pionnier *La dimension cachée* (1966). L'auteur aboutit à quatre catégories de distances – intime, personnelle, sociale et publique – et propose de forger un nouveau concept : la « proxémie[7] ». Ainsi, on pourrait à notre tour tenter un nouveau concept, transposé dans le champ temporel plutôt que spatial : la « chrono-proxémie », ou bien la « latentialité »… Ainsi, ce néologisme pourrait achever le cadre d'une future typologie temporelle, en désignant clairement la nature temporelle de la dimension à explorer, en particulier dans le champ musical.

à la création musicale, réside dans la capacité à créer des rapports entre les concepteurs et les utilisateurs. Les rapports sont organisés autour d'une modélisation des problématiques des compositeurs et d'une évolution progressive des outils. Les modifications d'un outil donné sont immédiatement utilisables et valident l'évolution tout en se nourrissant de l'expérimentation des développements successifs. » [Teruggi, 1999a, p. 186]

7. Edward T. Hall s'explique ainsi : « Le terme de "proxémie" est un néologisme que j'ai créé pour désigner l'ensemble des observations et théories concernant l'usage que l'homme fait de l'espace en tant que produit culturel spécifique. » [Hall, 1971, p. 13]

Perspectives pratiques

Tout d'abord, les logiciels développés ici portent en eux un potentiel pour de nouvelles réalisations musicales : par exemple, Mauricio Meza a continué d'utiliser a2m pour d'autres pièces, et Stéphanie Réthoré a improvisé dans plusieurs concerts avec Rose amère. Ces logiciels sont d'ailleurs, pour la plupart, mis à disposition sur internet[8].

Ensuite, les trois pièces d'Iván Solano sont encore en cours d'écriture et devraient bientôt porter leurs fruits, d'autant que les logiciels correspondants – Plugiscope, Ifso, et Ultraviolette – sont aboutis ; seul Ultraviolette subira probablement quelques modifications.

En outre, du point de vue personnel, j'imagine et j'espère que je pourrai continuer à réaliser de nombreux projets en informatique musicale en collaboration avec des musiciens, tant l'expérience de cette thèse m'a montré que ce couplage peut se révéler créatif, musicalement intéressant, et humainement enrichissant.

Enfin, mais il ne s'agit là que d'une spéculation, le décloisonnement impliqué par la représentation axiale intégrative pourrait bien influencer les mentalités et les pratiques, en suggérant que la création musicale navigue toujours *entre* le temps réel et le temps différé…

8. http://karim.barkati.online.fr/Universite/Doctorat/Logiciels/.

Table des figures

Liste des tableaux

Extraits de code

Bibliographie

[Adelstein *et al.*, 2003] Bernard D. Adelstein, Durand R. Begault, Mark R. Anderson, et Elizabeth M. Wenzel. Sensitivity to haptic-audio asynchrony. Dans *ICMI '03 : Proceedings of the 5th international conference on multimodal interfaces*, pages 73–76, New York, NY, USA, 2003. ACM.

[Adorno, 1963] Theodor W. Adorno. Vers une musique informelle. Dans *Quasi una fantasia*, pages 289–340. Gallimard, Paris, 1963. Traduction Jean-Louis Leleu, 1982.

[Agamben, 2006] Giorgio Agamben. *Qu'est-ce qu'un dispositif ?* Rivages poche / Petite Bibliothèque, 2006. Traduit de l'italien par Martin Rueff en 2007. Titre original : *Che cos'è un dispositivo ?*

[ARCEP, 2006] ARCEP. Comparaisons 1999 / 2005 : Le marché de l'Internet. 2006, http://www.arcep.fr/index.php?id=7704. Autorité de Régulation des Communications Électroniques et des Postes.

[Barbanti *et al.*, 2004] Roberto Barbanti, Enrique Lynch, Carmen Pardo, et Makis Solomos. *Musiques, arts, technologies – pour une approche critique.* L'Harmattan, Paris, 2004.

[Barkati, 1999] Karim Barkati. Programmation d'un échantillonneur logiciel. Mémoire de Maîrise, Université Paris 8, 1999. Directeur de recherche : M. Horacio Vaggione.

[Barkati, 2000a] Karim Barkati. Programmation de bibliothèques C compatibles C++ pour la gestion du son sous linux. Mémoire de Dea, Université Paris 8, 2000. Directeur de recherche : M. Daniel Goossens.

[Barkati, 2000b] Karim Barkati. Programmation d'outils logiciels pour les compositeurs. Mémoire de Dea, Université Paris 8, 2000. Directeur de recherche : M. Horacio Vaggione.

[Barkati, 2001] Karim Barkati. K-Box : programmation d'une extension logicielle pour instument percussif. Dans *Actes des* Jim, Bourges, 2001.

[Barkati et Lorenzo, 2005] Karim Barkati et Mario Lorenzo. Entre temps réel et temps différé. Dans *Actes des* Jim, 2005.

[Barkati et Saint-James, 2003] Karim Barkati et Emmanuel Saint-James. Svg & Xslt : une portée notable. Dans *Actes des* Jim, 2003.

[Barrière, 1992] Jean-Baptiste Barrière. Le sensible et le connaissable. Dans *Les cahiers de l'*Ircam *: recherche et musique*, volume 1, pages 77–86. Ircam – Centre Georges-Pompidou, 1992.

[Battier, 1999a] Marc Battier. L'approche gestuelle dans l'histoire de la lutherie électronique – étude de cas : le theremin. Dans *Les nouveaux gestes de la musique*, pages 139–150. Parenthèses, Marseille, 1999.

[Battier, 1999b] Marc Battier. Les polarités de la lutherie électronique – instrument, machine, représentation. Dans *Méthodes nouvelles, musiques nouvelles – Musicologie et création*, pages 307–318. Presses universitaires de Strasbourg, 1999.

[Beaudoin-Lafon, 1999] Michel Beaudoin-Lafon. Moins d'interface pour plus d'interaction. Dans *Interfaces homme-machine et création musicale*, Chapitre 6, pages 123–141. Hermes, Paris, 1999.

[Biesenbender, 1992] Volker Biesenbender. *Plaidoyer pour l'improvisation dans l'apprentissage instrumental*. Van de Velde, 1992. Traduction de Catherine Barret, 2001.

[Bonnet et Demeure, 1999] Christian Bonnet et Isabelle Demeure. *Introduction aux systèmes temps réel*. Hermes, Paris, 1999.

[Bossis, 2005] Bruno Bossis. Introduction à l'histoire et à l'esthétique des musiques électroacoustiques. 2005, http://portal.unesco.org/culture/fr/ev.php-URL_ID=26167&URL_DO=DO_TOPIC&URL_SECTION=201.html.

[Cadoz, 1999] Claude Cadoz. Continuum énergétique du geste au son — simulation multisensorielle interactive d'objet physiques. Dans *Interfaces homme-machine et création musicale*, Chapitre 8, pages 165–181. Hermes, Paris, 1999.

[Couchot, 1985] Edmond Couchot. À la recherche du « temps réel ». *Traverses*, 35 :41–45, 1985.

[Courtot, 1993] Francis Courtot. Entre le décomposé et l'incomposable. Dans *La composition assistée par ordinateur*, volume 3, pages 61–88. Ircam – Centre Georges-Pompidou, 2e trimestre 1993.

[Cristofol, 2003] Jean Cristofol. Temps réel, direct, différé. sep 2003, http://temporalites.free.fr/?browse=Temps%20r%C3%A9el,%20direct,%20diff%C3%A9r%C3%A9.

[Déchelle, 1999] François Déchelle. jMax : un environnement de programmation pour l'interactivité et le temps réel. Dans *Interfaces homme-machine et création musicale*, Chapitre 4, pages 85–94. Hermes, Paris, 1999.

[Desain *et al.*, 1993] Peter Desain, Henkjan Honing, Robert Rowe, Brad Garton, Roger Dannenberg, Dean Jabobs, Cort Lippe, Zack Settel, Stephen Travis Pope, Miller Puckette, et George Lewis. Putting max in perspective. *CMJ*, 17(2) :3–11, 1993.

[Di Scipio, 2008] Agostino Di Scipio. Émergence du Son, Son d'Émergence – Essai d'épistémologie expérimentale par un compositeur. Dans *Intellectica – Musique et Cognition*, volume 48–49, pages 221–249. Association pour la Recherche Cognitive, 2008.

[Dobrian *et al.*, 2006] Chris Dobrian, David Zicarelli, Andrew Pask, Darwin Grosse, Gregory Taylor, Joshua Kit Clayton, jhno, Richard Dudas, R. Luke DuBois, et Ben Nevile. *MSP 4.6 Tutorials and Topics*. Cycling'74, jun 2006.

[Durieux, 1992] Frédéric Durieux. Réseaux et création. Dans *Composition et environnements informatiques*, volume 1, pages 87–103. Ircam – Centre Georges-Pompidou, automne 1992.

[Dusapin, 2007] Pascal Dusapin. *Composer – Musique, paradoxe, flux*. Collège de France / Fayard, 2007.

[Fober *et al.*, 2004] Dominique Fober, Stéphane Letz, et Yann Orlarey. MidiShare : une architecture logicielle pour la musique. Dans *Informatique musicale : du signal au signe musical*, Chapitre 5. Hermes, mai 2004.

[Gervasoni, 2000] Pierre Gervasoni. L'assistant musical, trait d'union libre entre recherche et création. *Le Monde*, hors-série Agora, juin 2000.

[Gonord, 2001] Alban Gonord. *Le temps*. GF Flammarion, Paris, 2001.

[Hall, 1971] Edward T. Hall. *La dimension cachée*. Seuil, 1971. Edition originale 1966.

[Josèphe, 2008] Pascal Josèphe. *La société immédiate*. Calmann-Lévy, Paris, 2008.

[Kelly, 2005] Kevin Kelly. We are the web. aug 2005, http://www.wired.com/wired/archive/13.08/tech_pr.html.

[Lago et Kon, 2004] Nelson Posse Lago et Fabio Kon. The quest for low latency. Dans *Proceedings of the International Computer Music Conference*, pages 33–36, 2004.

[Laliberté, 1999] Martin Laliberté. Archétypes et paradoxes des nouveaux instruments. Dans *Les nouveaux gestes de la musique*, pages 121–138. Parenthèses, Marseille, 1999.

[L'Espresso, 2007] L'Espresso. Progression de la radio sur le net. *Newsletter*, mar 2007.

[Lestienne, 2007] Rémy Lestienne. *Les fils du temps – causalité, entropie, devenir*. Cnrs Éditions, Paris, mar 2007.

[Levitin *et al.*, 1999] D.J. Levitin, K. MacLean, M. Mathews, L. Chu, et E. Jensen. The perception of cross-modal simultaneity. *Proceedings of International Journal of Computing Anticipatory Systems*, 1999.

[Lévy, 1992] Pierre Lévy. *De la programmation considérée comme un des beaux-arts*. La Découverte, Paris, 1992.

[Madlener, 2006] Frank Madlener. Frank Madlener – Vers un nouvel Ircam. *ResMusica*, jan 2006. Rédacteur : Bruno Serrou.

[Manoury, 1992] Philippe Manoury. Entretien. Dans *Composition et environnements informatiques*, volume 1, pages 35–47. Ircam – Centre Georges-Pompidou, automne 1992. Propos recueillis par Danielle Cohen-Levinas.

[Manoury, 2007] Philippe Manoury. Considérations (toujours actuelles) sur l'état de la musique en temps réel. *L'étincelle*, 3, nov 2007.

[Marescaux *et al.*, 2002] Jacques Marescaux, Joel Leroy, Francesco Rubino, Michelle Smith, Michel Vix, Michele Simone, et Didier Mutter. Transcontinental robot-assisted remote telesurgery : feasibility and potential applications. *Annals of surgery*, 235(4) :487–492, Apr 2002.

[McAdams, 1994] Stephen E. McAdams. Audition : physiologie, perception et cognition. Dans M. Richelle, J. Requin, et M. Robert, éditeurs, *Traité de psychologie expérimentale*, volume 1, pages 283–344. Presses Universitaires de France, Paris, 1994.

[McCartney, 1996] James McCartney. SuperCollider : a new real time synthesis language. Dans *Proc. ICMC 1996*, 1996.

[McCartney, 2003] James McCartney. A few quick notes on opportunities and pitfalls of the application of computers in art and music. Dans Hatje Cantz, éditeur, *CODE – The Language of Our Time*, Linz, sep 2003. Ars Electronica.

[Morin, 2005] Edgar Morin. *Introduction à la pensée complexe*. Seuil, 2005. 1re édition chez Esf en 1990.

[Murail, 1992] Tristan Murail. Entretien. Dans *Composition et environnements informatiques*, volume 1, pages 19–34. Ircam – Centre Georges-Pompidou, automne 1992.

[Orlarey, 2006] Yann Orlarey. Introduction. Dans *Le feedback dans la création musicale*, Lyon, mar 2006. Grame, Rencontres musicales pluridisciplinaires.

[Orlarey *et al.*, 1999] Yann Orlarey, Dominique Fober, et Stéphane Letz. Lambacalcul et composition musicale. Dans *Interfaces homme-machine et création musicale*, Chapitre 3, pages 67–83. Hermes, Paris, 1999.

[Piéron, 1913] Henri Piéron. Recherches sur les lois de variation des temps de latence sensorielle en fonction des intensités excitatrices. *L'année psychologique – Revue de psychologie cognitive*, 20, 1913.

[Puckette, 2002] Miller Puckette. Max at seventeen. *CMJ*, 26(4) :31–43, 2002.

[Puckette, 2004] Miller Puckette. A divide between 'compositional' and 'performative' aspects of Pd. *First International Pd Convention*, 2004.

[Puckette, 2006] Miller Puckette. Computing while composing. Dans Carlos Agon, Gérard Assayag, et Jean Bresson, éditeurs, *The OM Composer's Book*, volume 1. Delatour France / Ircam, 2006.

[Puckette, 1996] Miller S. Puckette. Pure Data : another intergrated computer music environment. Dans *Proc. ICMC 1996*, pages 37–41, 1996.

[Risset, 1998] Jean-Claude Risset. Rapport art, science, technologie. Rapport Technique, Ministère de l'Éducation Nationale, de la Recherche et de la Technologie, 1998.

[Risset, 1999a] Jean-Claude Risset. Évolution des outils de création sonore. Dans *Interfaces homme-machine et création musicale*, Chapitre 1, pages 17–36. Hermes, Paris, 1999.

[Risset, 1999b] Jean-Claude Risset. Temps en musique numérique. Dans *Le temps en musique électroacoustique*, volume Actes V, pages 141–145, Bourges, 1999. Académie Bourges, Mnemosyne.

[Roads, 1992] Curtis Roads. Des instruments pour un son organisé. Dans *Composition et environnements informatiques*, pages 107–125. Ircam – Centre Georges-Pompidou, Paris, automne 1992. Traduit de l'anglais par Serge Grünberg.

[Roads, 1998] Curtis Roads. *L'audionumérique*. Dunod, 1998. Version française : Jean de Reydellet.

[Roads, 2007] Curtis Roads. L'art de l'articulation : la musique électroacoustique d'Horacio Vaggione. Dans Makis Solomos, éditeur, *Espaces composables – essais sur la musique et la pensée musicale d'Horacio Vaggione*, pages 69–88. L'Harmattan, Paris, 2007.

[Rondeleux, 1997] Luc Rondeleux. Quarante années de représentations numériques au service de la création. *Alliage – Statut esthétique de l'art technologique*, 33-34 :80–91, 1997.

[Rondeleux, 1999] Luc Rondeleux. Une histoire de l'informatique musicale entre macroforme et microcomposition. Dans *Actes des* Jim, pages 1–9, Issy-Les-Moulineaux, 1999.

[Schaeffer, 1966] Pierre Schaeffer. *Traité des objets musicaux*. Seuil, Paris, 1966.

[Segal, 2003] Jérôme Segal. *Le Zéro et le Un : Histoire de la notion scientifique d'information au 20^e siècle*. Syllepse, 2003.

[Seleborg, 2004] Carl Seleborg. Interaction temps-réel/temps différé – Élaboration d'un modèle formel de Max et implémentation d'une bibliothèque Osc pour OpenMusic. Dea Atiam, Université Aix-Marseille II, Paris, jun 2004.

[Solomos, 2004] Makis Solomos. Une musicologie et son temps. Dans Roberto Barbanti, Enrique Lynch, Carmen Pardo, et Makis Solomos, éditeurs, *Musiques, arts, technologies – pour une approche critique*, pages 281–290. L'Harmattan, Paris, 2004.

[Solomos *et al.*, 2007] Makis Solomos, Jean-Claude Risset, Curtis Roads, Anne Sedes, Horacio Vaggione, Osvaldo Budón, Paul Méfano, Martin Laliberté, Antonia Soulez, Philip Mead, Elsa Justel, Pascale Crition, et Agostino Di Scipio. *Espaces composables - essais sur la musique et la pensée musicale d'Horacio Vaggione*. L'Harmattan, Paris, 2007.

[Spence et Squire, 2003] Charles Spence et Sarah Squire. Multisensory integration : maintaining the perception of synchrony. *Current Biology*, 13(13) :R519–21, jul 2003.

[Szendy, 1998] Peter Szendy. Musique, temps réel. *Résonance*, 14, oct 1998.

[Tanenbaum, 2005] Andrew Tanenbaum. *Architecture de l'ordinateur*. Pearson Education, 5e édition, 2005.

[Teruggi, 1999a] Daniel Teruggi. L'interactivité dans les processus de création sonore et musicale. Dans *Interfaces homme-machine et création musicale*, Chapitre 9, pages 185–193. Hermes, 1999.

[Teruggi, 1999b] Daniel Teruggi. Table ronde. Dans *Interfaces homme-machine et création musicale*, Chapitre 12, pages 215–236. Hermes, 1999. Animation et transcription : François Delalande.

[Teruggi, 2005] Daniel Teruggi. Le temps du temps réel. Dans *Archives GRM*. INA, 2005.

[Truchet *et al.*, 2001] C. Truchet, C. Agon, G. Assayag, et P. Codognet. Cao et contraintes. Dans *Actes des* Jim, volume 8, Bourges, 2001.

[Vaggione, 1998a] Horacio Vaggione. L'espace composable : sur quelques catégories opératoires dans la musique électroacoustique. Dans J.M. Chouvel et M. Solomos, éditeurs, *L'espace : musique-philosophie*, pages 153–166. L'Harmattan, Paris, 1998.

[Vaggione, 1998b] Horacio Vaggione. Son, temps, objet, syntaxe. Dans *Musique, rationalité, langage*, pages 169–201. L'Harmattan, Paris, 1998.

[Varela, 1984] Francisco J. Varela. The creative circle : sketches on the natural history of circularity. Dans Paul Watzlawick, éditeur, *The invented reality*. Norton, New York, 1984.

[Varela, 1987] Francisco J. Varela. *Autonomie et connaissance – Essai sur le vivant*. Seuil, Paris, 1987. Trad. P. Bourgine et P. Dumouchel 1989.

[Varela, 1989] Francisco J. Varela. *Invitation aux sciences cognitives*. Seuil, Paris, 1989.

[Veitl, 2004] Anne Veitl. Le compositeur à l'ordinateur (1955-1985) : des moyens de rationnalisation aux outils de *réelisation*. Dans Roberto Barbanti, Enrique Lynch, Carmen Pardo, et Makis Solomos, éditeurs, *Musiques, arts, technologies – pour une approche critique*. L'Harmattan, Paris, 2004.

[Vercoe et Ellis, 1990] B. Vercoe et D. Ellis. Real-time csound : Software synthesis with sensing and control. Dans *Proc. ICMC 1990*, pages 209–211, Glasgow, 1990.

[Vinet, 1999] Hugues VINET. Concepts d'interfaces graphiques pour la production musicale et sonore. Dans *Interfaces homme-machine et création musicale*, Chapitre 5, pages 97–121. Hermes, 1999.

[Wang, 2008] Ge WANG. *The ChucK Audio Programming Language : An Strongly-timed and On-the-fly Environ/mentality*. PhD thesis, Princeton University, 2008.

[Wang et Cook, 2003] Ge WANG et Perry R. COOK. ChucK : A concurrent, on-the-fly, audio programming language. Dans *Proc. ICMC 2003*, 2003.

[Wang et Cook, 2007] Ge WANG et Perry R. COOK. *The ChucK Manual 1.2.1.2*. Princeton University, 2007.

[Wang *et al.*, 2007] Ge WANG, Rebecca FIEBRINK, et Perry R. COOK. Combining analysis and synthesis in the ChucK programming language. Dans *Proc. ICMC 2007*, Copenhagen, 2007.

[Watzlawick *et al.*, 1975] Paul WATZLAWICK, John WEAKLAND, et Richard FISCH. *Changements*. Seuil, 1975. Traduction de Pierre Furlan.

[Wertz, 1983] Harald WERTZ. Quelques remarques sur le cadre psychologique de la programmation. *Rapport interne*, oct 1983.

[Wessel, 1979] David L. WESSEL. Timbre space as a musical control structure. *Computer Music Journal*, 3(2) :45–52, 1979.

[Winer, 2003] Dave WINER. RSS 2.0 specification. 2003, http://blogs.law.harvard.edu/tech/rss.

[Zanoni, 1998] Marc ZANONI. *La récursivité en programmation, l'abord psychologique*. Thèse de doctorat, Université Paris 8, 1998.

[Zicarelli *et al.*, 2006a] David ZICARELLI, Gregory TAYLOR, Joshua Kit CLAYTON, JHNO, Richard DUDAS, R. Luke DUBOIS, et Andrew PASK. *MSP 4.6 Reference Manual*. Cycling'74, 4.7 édition, aug 2006.

[Zicarelli *et al.*, 2006b] David ZICARELLI, Gregory TAYLOR, Joshua Kit CLAYTON, JHNO, Richard DUDAS, et Ben NEVILE. *Max 4.6 Reference Manual*. Cycling'74, 4.7 édition, aug 2006.

Annexes

Annexe A

Article des JIM 2005

Cet article a été rédigé en collaboration avec Mario Lorenzo, au cours du projet CBox, et publié dans les actes des Journées d'informatique musicale 2005.

CBOX : ENTRE TEMPS REEL ET TEMPS DIFFERE

Karim Barkati et Mario Lorenzo

CICM - Université Paris VIII
MSH Paris Nord
karim.barkati@free.fr
mariolorenzo@fr.st

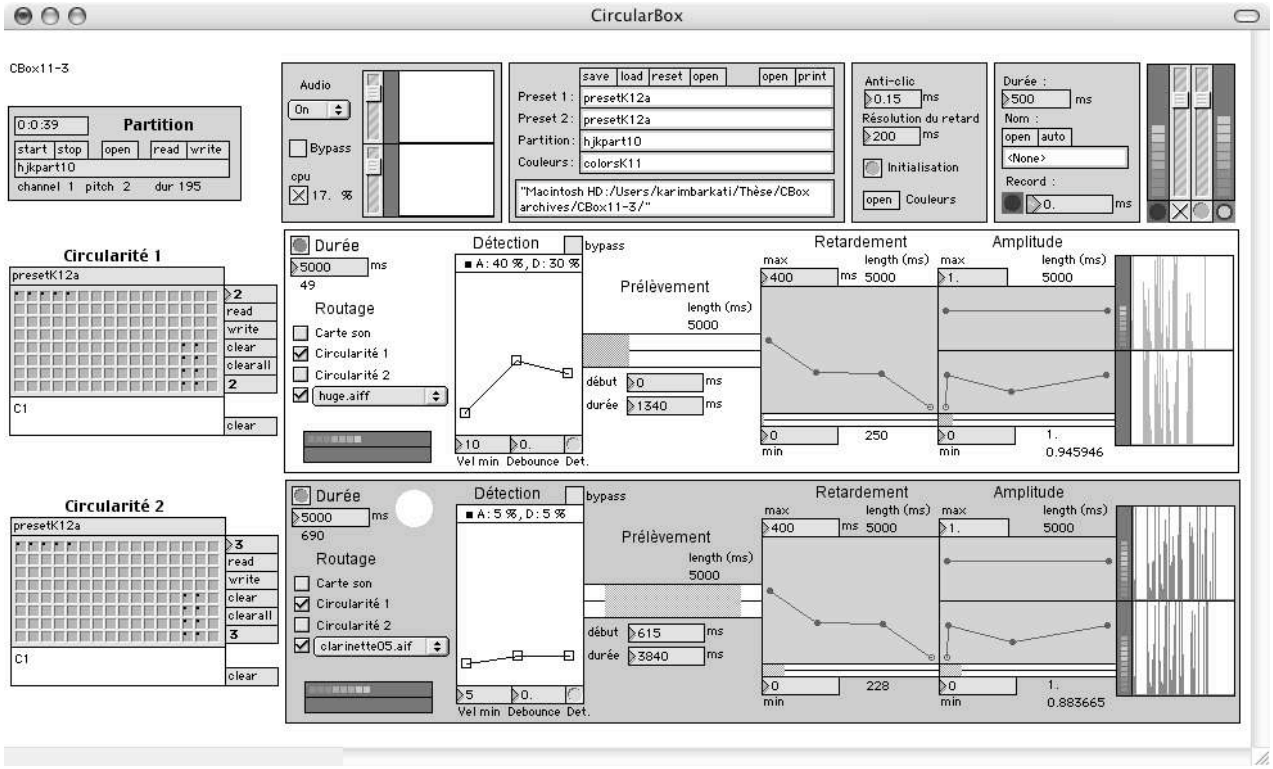


Figure 1. Interface graphique de la CircularBox11-3 jouant une partition programmée.

RÉSUMÉ

Nous souhaitons présenter ici une partie de nos recherches universitaires dans leur état actuel de « work in progress », non pour démontrer un résultat définitif, mais pour ce qu'elles mettent en relief des questions qui nous intéressent : autour du temps réel et du temps différé, de la circularité, de la réactivité logicielle, et du rapport entre micro temps et temps réel.

Ainsi, nous étudierons d'abord la mixité logicielle entre temps réel et temps différé à travers l'exemple de la CBox, un programme que nous avons développé dans l'environnement de programmation Max/MSP. Ensuite, nous présenterons la circularité d'un point de vue conceptuel, puis d'un point de vue plus pratique, telle qu'elle est implémentée dans ce logiciel, pour distinguer deux types structurels de bouclages dynamiques et réactifs. Enfin, nous reposerons la question du rapport entre le micro temps et l'informatique temps réel en apportant des éléments de réponse guidés par un souci musical.

1. INTRODUCTION

L'idée de *circularité* proposée par Francisco Varela [19] pour caractériser les systèmes autonomes a été le point de départ d'une réflexion sur quelques problématiques contemporaines et du développement de notre logiciel, dont la version actuelle se nomme CBox11-3 (pour CircularBox).

Bien que nous nous soyons inspirés du paradigme connexionniste et en particulier de l'étude des systèmes auto-organisés de Varela et Maturana [8], il est important de préciser que notre approche ne vise pas à établir des modèles[1], ni à s'opposer aux applications

[1] Nous pensons par exemple aux premières expériences de créativité artificielle (Hiller, Barbaud, Xenakis), plus récemment aux automates cellulaires de Miranda [9], et à l'implémentation d'agents autonomes en Max/MSP de Malt [7]

d'orientation plus déterministes. En effet, en tant que musiciens, nos travaux relèvent essentiellement de la recherche en composition assistée par ordinateur (CAO), à la croisée de l'algorithmique, de l'interaction et de l'interprétation : au sein des techniques de synthèse et transformation du son, il s'agit de trouver des alternatives aux approches déterministes, en incorporant, dans une certaine mesure, des idées cognitives susceptibles d'apporter de nouvelles réponses à nos problématiques musicales.

CBox est un logiciel programmé dans l'environnement Max/MSP, à la fois temps réel (TR) et temps différé (TD). Essentiellement, il s'agit d'un circuit composé de deux *processus circulaires*, ayant tous les deux exactement la même architecture et pouvant être connectés entre eux ou simplement à eux-mêmes (feedback), ou dans d'autres combinaisons possibles selon les entrées utilisables (Carte son, C1, C2, Fichiers son).

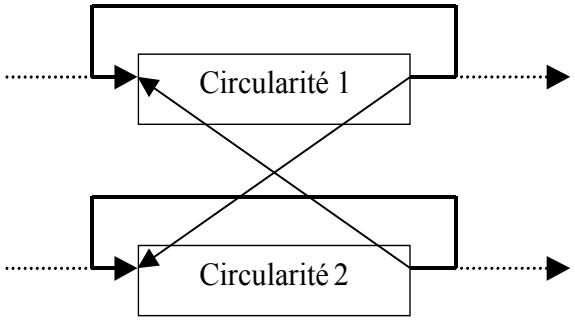


Figure 2 : schéma des branchements circulaires.

Du point de vue de la lutherie informatique, le logiciel CBox pourrait être considéré comme un instrument à la fois composable et interactif. Il fonctionne en temps réel et reçoit le signal sonore pour le transformer dynamiquement (cf. Fig.3) par des opérations de :

- routage (avec l'objet *gate~*)
- détection d'attaques (avec l'objet *bonk~*)
- prélèvement (avec *line~* et *delay~*)
- retardement (avec *tapin~* et *tapout~*)
- réinjection. (avec *send~* feedback)

Ces opérations sont doublées en deux modules identiques, les circularités C1 et C2. Les paramètres sont enregistrés puis rappelés depuis l'objet graphique *preset*. Finalement, toutes les configurations stockées dans les deux *presets* peuvent être déclenchées par une partition MIDI à deux voix (grâce à l'objet *detonate*), ou en temps réel par un instrumentiste à l'aide du clavier alphanumérique (objet *key*).

2. MIXITE TEMPS REEL ET TEMPS DIFFERE

« Temps réel » et « temps différé » sont deux expressions souvent opposées dans le champs de l'informatique musicale pour distinguer des applications selon le contexte des pratiques musicales, sonores et compositionnelles. Le dictionnaire des arts médiatiques[1] donne la définition suivante du temps réel :

« Modalité temporelle des systèmes de traitement de l'information dans lesquels il n'y a pas de délai entre la sortie d'informations et l'entrée de données, ou, si l'on veut, dans lesquels l'output suit immédiatement l'input. Le temps réel est caractéristique du mode interactif. (...) Certains traitements ou synthèses sonores informatiques exigent trop de calculs pour être effectués en temps réel. On obtient donc le résultat avec un délai qui peut parfois être très long : c'est le « temps différé ». Mais la puissance actuelle des ordinateurs est telle que de plus en plus de systèmes de synthèse réagissent en « temps réel », ce qui permet au compositeur d'avoir un contrôle perceptif immédiat et continu de son travail. »

De notre point de vue de musiciens, la pratique habituelle amène à considérer globalement que le « temps réel » désigne les situations de concert ou de sonorisation où l'informatique intervient directement et activement, avec une instantanéité apparente, tandis que le « temps différé » désigne davantage les situations de studio associées au temps de l'acte compositionnel et de la réflexion. Nous étudierons à travers le cas de la CBox une approche mixte, où TR et TD coexistent de façon nécessaire.

2.1. Temps différé

« (...) les propositions du temps réel manquent de variété, et elle incitent à des réactions réflexes et donc irréfléchies, donnant souvent lieu à des stéréotypes vite usés ou périmés. » [13]

Historiquement, les applications informatiques étant limitées par la faible puissance de calcul des premiers ordinateurs, les premiers logiciels ne pouvaient calculer du signal sonore qu'en temps différé, c'est-à-dire qu'il fallait d'abord saisir les données, puis lancer l'opération de calcul du son, et enfin attendre jusqu'à des heures entières avant de pouvoir l'écouter, et éventuellement recommencer en modifiant quelques paramètres si le résultat n'était pas satisfaisant… Citons Music V et Csound parmi les logiciels historiquement marquants.

En dépit de cet inconvénient de l'attente, aujourd'hui largement réduite, le « temps différé » s'avère incontournable à l'heure de composer. En effet, le retour sur des choix compositionnels, les connexions multiples entre différentes échelles temporelles, l'inclusion de singularités, le travail détaillé au niveau du micro temps, sont parmi les « calculs » dont le temps réel ne fourni que des « résultats » partiels. Car la composition musicale, telle que nous l'envisageons, est intimement liée aux aspects cognitifs, où l'interaction a

[1] Dictionnaire des arts médiatiques © 1996, Groupe de recherche en arts médiatiques - UQAM

un rôle central. Dans ce sens, la possibilité d'inclure l'improvisation à la composition grâce au temps réel, certes peut apporter des réponses mais l'encadrement doit être précis. Le risque de tomber dans des gestes stéréotypés, non réfléchis, est grand. Nous reviendrons plus tard sur cette position.

Il s'ajoute une autre contrainte, cette fois-ci liée à la problématique du marché : le phénomène d'obsolescence propre à l'informatique, qui risque de rendre les œuvres « périssables » pour reprendre l'expression de Jean-Claude Risset[1].

L'un des enjeux de ce projet réside dans la possibilité de composer entièrement une pièce avec la CBox, c'est-à-dire de disposer du maximum de fonctions, de commodités d'écriture et de mémorisation des paramètres, d'espaces de structurations multi-échelles, pour en faire un outil de composition à part entière, complètement envisageable comme un outil « temps différé ». Plusieurs mécanismes ont été implémentés dans ce but :

- la partition MIDI, avec l'objet *detonate*
- l'interface graphique de saisie des paramètres, avec des enveloppes, des espaces pour les nombres et pour du texte (commentaires)
- la mémorisation et la sauvegarde des paramètres, avec les objets *preset*
- la mémorisation et la sauvegarde des différents fichiers utilisés, dans un unique fichier de démarrage
- l'enregistrement du résultat sonore sur disque dur

2.2. Temps réel

Un deuxième enjeu important de ce logiciel réside dans la possibilité de l'utiliser en situation de concert, qu'il s'agisse d'improvisation ou bien d'interprétation d'une pièce écrite avec une partition « mixte » pour le programme et le ou les interprètes. On peut imaginer d'autres applications, comme pour des installations sonores, puisque la CBox est capable de réagir à l'environnement sonore en détectant des attaques dans le signal d'entrée (cf. ci-dessous 3.1 « Ecoute » et réactivité logicielles).

Le temps réel possède des intérêts non négligeables au regard de nos préoccupations musicales. Depuis que la puissance de calcul des ordinateurs le permet, beaucoup d'applications temps réel sont apparues - traitements, effets, plug-ins, interfaces interactives - ainsi que plusieurs environnements de programmation dédiés au temps réel comme Max/MSP, SuperCollider, Pure Data, jmax. Ces programmes sont susceptibles d'interagir pendant une performance musicale de façons très diverses : en captant les mouvements ou déplacement d'un danseur, en projetant des séquences vidéo en fonction d'évènements sonores, en suivant une interprétation musicale par rapport à la partition pour déclencher précisément des fichiers sons ou des traitements sonores, ou de toute autre manière calculable en temps réel par un système informatique, c'est-à-dire sans délai trop perceptible. Ainsi la plupart des programmes temps réel donnent accès à une interactivité séduisante et parfois spectaculaire, ils dotent d'une dimension presque « vivante » les performances interactives, et peuvent ouvrir un dialogue entre l'activité humaine et l'activité de la machine, tant qu'ils préservent une cohérence causale perceptible. L'informatique peut alors accompagner, prolonger, voire décupler le geste humain mis en scène, ou bien, sur un autre plan, ouvrir un dialogue homme-machine[2].

En ce qui concerne notre démarche par rapport au temps réel, c'est cette possibilité de dialogue qui nous intéresse, soit dans le travail de studio, soit pendant la performance ou le concert. Avec la CBox, ce dialogue a lieu selon des modalités particulières dont une partie est à préciser lors de la composition de la partition du logiciel, et dont les autres sont inhérentes aux spécificités de l' « écoute informatique » telle qu'elle a été programmée dans ce logiciel, ici à partir de l'objet *bonk~*.

Comme nous le verrons par la suite, cette « écoute » du logiciel introduit une part d'imprévisibilité dont les conséquences posent des questions différentes, selon qu'il s'agit de composition ou d'interprétation.

2.3. La mixité comme région fractionnaire

La frontière entre temps réel (TR) et temps différé (TD) se fait de moins en moins nette, soit que les logiciels considérés comme TD se rapprochent du TR grâce au progrès technologique, soit que les logiciels TR incorporent de plus en plus d'éléments TD. Plus les logiciels incorporent une mixité entre TR et TD, plus cette frontière devient ténue, et plus nous pouvons concevoir des régions fractionnaires entre ces deux pôles.

La CBox autorise une exploration particulière de cette nouvelle région, car nous pouvons envisager ce logiciel de deux manières différentes et complémentaires en considérant qu'il permet de faire : soit de "l'improvisation pré-composée", soit de la "composition réactive". Quel que soit l'angle avec lequel on aborde la CBox, on ne peut pas ne pas passer par les deux temps. En effet, une pratique improvisatrice, considérée comme TR, ne pourra pas ici faire l'économie d'une phase de composition des paramètres du logiciel, phase considérée comme TD. Réciproquement, une pratique compositionnelle ne pourra pas faire l'économie d'une phase d'improvisation, de tâtonnement interactif et dynamique avec ce logiciel. Il ne s'agit pas d'une conséquence fortuite ou d'un constat postérieur, mais

[1] *"Le véritable problème est celui de l'évolution très rapide des machines commerciales assez puissantes pour traiter le son en temps réel. En effet, comme le remarque Risset, « l'obsolescence technologique » risque ici tout particulièrement de « rendre l'oeuvre périssable », puisque celle-ci se refuse à être fixée sur un support (comme dans le cas des musiques sur bande) et ne fait que coder des protocoles d'interaction avec un ordinateur... soumis aux lois du marché."* [16]

[2] Pour une approche du paradigme interactif, cf. Wegner [21].

bien d'une volonté double dès le début de la conception ; il se trouve par ailleurs que les tendances de chacun des deux concepteurs par rapport à la priorité TR ou TD s'opposent, de sorte qu'à chaque étape, lorsque que la programmation du logiciel avançait dans une direction, l'étape suivante compensait dans l'autre direction.

Cette « double compétence » du logiciel, outre de fournir un cas d'étude exemplaire, présente quelques avantages et quelques inconvénients. D'abord, ne pas choisir l'un des deux pôles comme cible principale alourdit le programme, puisqu'il faut implémenter les couches spécifiques à la fois du TR et du TD. Ensuite, passé ce problème technique, l'improvisation intuitive autorisée par les aptitudes TR de la CBox stimule la composition : en jouant avec le programme interactif on peut se familiariser rapidement. Enfin, la nécessité de "pré-composer" l'improvisation à diverses échelles temporelles simultanément permet potentiellement d'échapper au risque du manque de réflexion, tout en garantissant pour la performance un dialogue homme-machine.

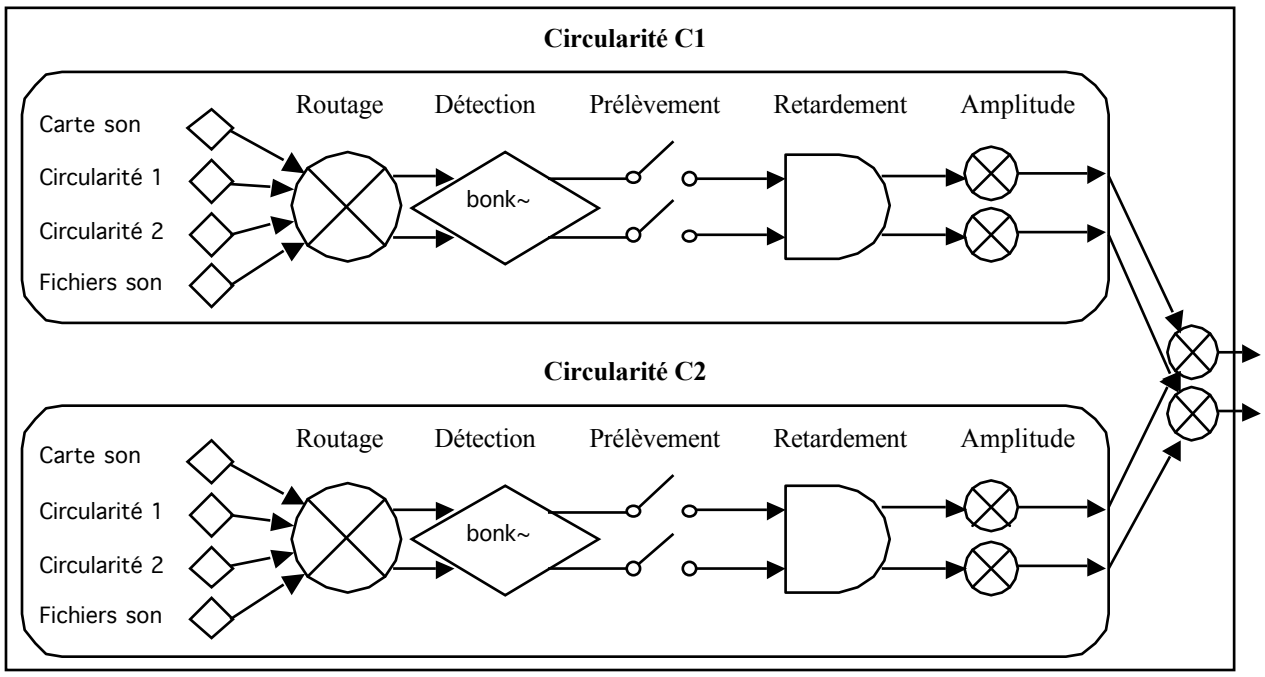


Figure 3 : schéma structurel de la chaîne de traitement du signal.

3. CIRCULARITE

Le principe fondateur de la CBox repose sur l'idée de *circularité* à la fois comme simple mécanisme de réinjection du signal de sortie à l'entrée de courtes chaînes opératoires et comme possibilité de connexion réciproque de ces deux processus. Nous distinguons ainsi des circularités de « type 1 », qui bouclent seulement sur elles-mêmes, et des circularités de « type 2 », qui connectent deux processus entre eux de façon à ce qu'ils se définissent mutuellement[1].

3.1. « Ecoute » et réactivité logicielles

Ces processus circulaires incluent chacun une « écoute » informatique, ce qui nous permet d'introduire l'idée de *réactivité logicielle.* Pour l'instant, cette « écoute » reste assez élémentaire : elle repose essentiellement sur la détection d'attaques opérée par l'objet bonk~[2]. Bonk, programmé par Miller Puckette [11], détecte les attaques définies par un changement de forme dans l'enveloppe spectrale. Ainsi, la CBox reçoit un signal audionumérique et le convertit en temps réel en une suite de valeurs résultant de l'analyse de ce signal. L'analyse de bonk~ est effectuée sur une fenêtre de 256 points, soit 6 millisecondes pour une fréquence d'échantillonnage de 44100 Hertz, et répétée tous les 128 échantillons, soit toutes les 3 millisecondes.

[1] Nous nous sommes inspirés des écris de Francisco Varela [20] mais les idées de feedback et de circularité apparaissent dès l'origine des sciences cognitives. Dans la première cybernétique déjà, Wiener introduit en 1946 le concept de boucle de rétroaction (feedback), inhérent à toute régulation d'un système, sur la base de l'écart observé entre son action effective (output) et le résultat projeté (goal) [3]. Quant à l'idée de circularité en tant qu' « auto-organisation», on peut en trouver des traces chez Paul Weiss. Ainsi, d'après Dupuy :
« …Weiss énonce un principe de causalité circulaire entre niveaux d'intégration emboîtés. (…) Dans un « système », les lois de la physique laissent aux éléments individuels de nombreux degrés de liberté. Cette indétermination à la base va être réduite par les contraintes exercées par le tout, lesquelles résultent elles-mêmes de la composition des activités élémentaires. Le tout et les éléments se déterminent mutuellement. » [3]

[2] Nous avions déjà utilisé cet objet pour du déclenchement de sons dans une autre application, K-Box [1].

Cette sensibilité d'écoute nous permet théoriquement de travailler dans le domaine du micro temps, essentiel à la composition des sons[1]. Les limites de l'usage pratique de l'objet bonk nous a cependant conduits à nous reposer la question de l'écoute informatique, en particulier le souhait d'accéder à d'autres axes « perceptifs » que la seule détection d'attaques. Nous avons donc envisagé la substitution de bonk par des analyses plus complètes grâce à l'exploitation des données spectrales, par exemple avec l'objet **analyzer** de Tristan Gehan[2] à partir duquel nous avons développé un programme Visualyzer pour expérimenter les possibilités de ces analyses. Cet objet révèle plusieurs paramètres grâce à l'analyse en TR des données spectrales et énergétiques : l'attaque, l'amplitude instantanée, la hauteur (pitch), la brillance (brightness), le degré d'harmonicité (noisiness). Plusieurs inconvénients sont apparus : d'abord, la fenêtre d'analyse d'analyzer demande 23ms contre 6ms pour celle de bonk, ensuite, *« …une représentation temps-fréquence n'est qu'un point de départ, mais nullement une photographie de notre image mentale du son évolutif. »* [13]. Nous avons donc songé à élaborer quelque chose de plus proche de notre propre écoute, un parcours d'écoute composé, c'est à dire une bibliothèque de formes dynamiques subjectives à partir de parcours paramétriques dans un temps donné. Cette bibliothèque d'abord personnelle et subjective pourrait devenir plus générale en augmentant de façon critique sa taille et la variété des formes enregistrées. Nous avons renoncé devant la lourdeur de la tâche et l'ouverture de la question de « l'écoute informatisée », qui pourrait peut-être d'ailleurs trouver quelques réponses avec le connexionnisme[3].

Bien que bonk~ ne permette qu'une écoute limitée à la détection d'attaques dans le signal, la CBox, grâce à une interface de détection, exploite les possibilités de paramétrage de cet objet: *high threshold*, *low threshold*, *minvel*, et *debounce*. Ainsi, une enveloppe graphique à deux pentes définit la forme de la croissance du signal, qui doit dépasser le seuil supérieur (*high threshold*) puis passer sous le seuil inférieur (*low threshold*) pour déterminer une attaque. Par ailleurs, deux boîtes numériques définissent deux filtres: *minvel* pour filtrer les attaques d'amplitude inférieure à ce seuil, et *debounce* pour imposer un temps minimum entre deux attaques.

Il faudrait aussi insister sur le fait que cette réactivité est accompagnée par des choix compositionnels (les prélèvements) et leur mise en relation (routage) ce qui le distingue d'une approche purement aléatoire. C'est de cette façon que nous établissons notre dialogue avec l'outil, entre temps réel et temps différé, entre déterminisme et indéterminisme. Une voix moyenne.

3.2. Degré d'imprévisibilité

Cette voix moyenne, entre déterminisme et indéterminisme, résulte en grande partie du mécanisme d'écoute logicielle et apporte une imprévisibilité encadrée, un côté peut-être plus « vivant » à l'outil. Il s'agit de préciser les modalités de la réaction du logiciel, sans que l'on puisse être certain du résultat qui sera effectivement produit. L'imprévisibilité pourrait être intégrée musicalement à plusieurs niveaux, mais si possible seulement là où on le souhaite, car un dialogue sans évidence causale risque d'engendrer une frustration musicale rédhibitoire. Imprévisible ne signifie pas aléatoire. Nous distinguons au moins deux niveaux d'imprévisibilité selon qu'il s'agit du temps de la composition ou bien du temps de la performance.

Comme nous l'avons vu précédemment, la réactivité du logiciel permet d'obtenir des résultats imprévus. Ces imprévisibilités, loin d'être des « erreurs », sont du point de vue de la composition riches en conséquences, porteuses de formes. Elles permettent d'emprunter des chemins qui n'auraient pas pu être envisageables d'avance (par une formule quelconque) et ainsi de composer des singularités, qui seront ensuite réintroduites dans le processus circulaire.

« Aujourd'hui nous sommes dans une situation où le compositeur ne se limite plus à planifier un processus pour le regarder marcher tout seul, en attendant qu'il lui donne quelque chose : il interagit à tout moment avec lui, pour produire du formel. Même si c'est le compositeur lui-même qui a construit le système, il peut s'investir à tout moment dans une « performance » avec les données fournies par son système ; les sorties ne sont donc pas automatiques, mais du point de vue du système, imprévisibles. On peut dire donc qu'il produit ainsi des singularités. » [19]

Mais, si pour la composition ces imprévisibilités ont un rôle formel (porteuses de formes), il en est autrement lors d'une performance, en particulier s'il s'agit d'une interprétation. En effet, l'intérêt d'une réactivité logicielle au niveau micro temporel réside dans la possibilité de donner à l'interprète une sensibilité de jeu (propre à toute interprétation) sans pour autant perdre la cohérence compositionnelle. Nous souhaitons que les fluctuations micro temporelles de l'interprétation engendrent des résultats différents, sans que ces résultats n'aient rien à voir entre eux ; c'est-à-dire formaliser des abstractions qui définissent des classes de résultats en intégrant à la fois cohérence et imprévisibilité. Pour cela, d'une part, l'instrumentiste doit se familiariser avec l'œuvre (en tant que composition réactive) et, d'autre part, le travail de composition doit prendre en compte le degré d'imprévisibilité du geste humain.

3.3. Un facteur de complexité

Le mécanisme global de la CBox autorise le couplage des deux processus circulaires (C1 et C2) du logiciel, ce qui engendre rapidement une certaine complexité en établissant une circularité de « type 2 ». De fait, la

---

[1] Cela dit, en termes fréquentiels nous sommes évidemment limités. À cet égard, nous travaillons actuellement sur d'autres mécanismes compensatoires, dont un moteur de répétition micro temporelle séparé.

[2] http://web.media.mit.edu/~tristan/maxmsp.html

[3] *«… l'action cognitive ou perceptive, appréhendée comme ontogenèse, en tant qu'elle fait naître, suivant des processus temporels propres, des sensations à partir d'un environnement, ainsi que des associations sur des sensations, ne peut se décrire en termes de représentations…* [5]

composition musicale est porteuse de complexité lorsque les échelles temporelles qu'elle véhicule se co-déterminent par une même fonction, lorsqu'elles interagissent dans des processus circulaires de « type n », avec n supérieur à notre niveau 2 élémentaire. D'après Lévy-Leblond, la complexité, *« c'est la conjugaison à la fois d'une hétérogénéité structurelle et d'une réciprocité fonctionnelle. »*. [19]

Certes, le recours à l'aléatoire ou à l'improvisation partielle, pour palier à la difficulté de déterminer à la fois l'ensemble et les détails, peut être une source d'inattendus, mais ils restent souvent d'une complexité faible, car dépourvus des fonctions réciproques où les niveaux temporels restent séparés. L'improvisation, par son immédiateté, empêche la possibilité systématique de travailler sur plusieurs niveaux, comme dirait Dahlhaus, elle est « monolithique », elle se concentre presque toujours sur un seul élément. [2] Ces limitations nous ont conduits à implémenter une circularité de « type 2 », par le couplage de deux niveaux temporels (qu'on pourrait associer d'une manière schématique au local et au global) et cela nous permet déjà d'aborder empiriquement la complexité. Dans le sens post-biologique, il faudrait beaucoup de circularités pour s'approcher de la complexité des simulations de vie artificielle, et surtout beaucoup plus de puissance de calcul. Il faudrait aussi imaginer une autre façon de composer, car écrire pour 2000 processus circulaires au lieu de 2 implique nécessairement des changements radicaux.

Même avec notre puissance de calcul limitée, avec beaucoup de compromis, la CBox pourrait finir par devenir un logiciel normal plutôt qu'expérimental, mais pour l'instant elle nous a déjà permis de formuler des questions qui nous intéressent. En tout état de cause, le travail soigné des prélèvements à différentes échelles - des singularités - et de leur mise en relation grâce au couplage de deux circularités permet d'établir cette fonction commune qui caractérise la complexité.

4. QUELLE NECESSITE MICRO TEMPORELLE ?

Le niveau micro temporel est devenu aujourd'hui un domaine incontournable dans la composition musicale assistée par ordinateur. Les célèbres travaux d'analyse par synthèse de Risset [12] ont montré l'importance de l'articulation au domaine du micro temps et, de fait, le micro temps est désormais, non sans difficultés, un champs supplémentaire à composer.

4.1. Entre déterminisme et aléatoire

Depuis les premières recherches du physicien Denis Gabor dès 1946 sur les « quantas sonores », de nombreuses recherches et applications ont été réalisées autour de ce qu'on appelle aujourd'hui le micro son [15], dont la synthèse granulaire [14], [17], l'une des techniques les plus développées[1]. Mais pouvoir descendre jusqu'aux échelles les plus petites n'est pas toujours synonyme de pouvoir composer le micro temps. Si paradoxal que cela puisse paraître, la synthèse granulaire ne garantit pas une véritable incorporation du micro temps à l'intérieur d'une composition.

En effet, en raison de la quantité massive de données nécessaires pour piloter cette technique, la synthèse granulaire fait appel à des unités d'organisation de haut niveau [15]. Ianis Xenakis, un des premiers à concevoir une utilisation musicale des grains sonores et de leur distribution dans le temps, a utilisé les théories mathématiques des probabilités [22]. Etant donné la qualité de ses œuvres, il est certain que ses recherches ont abouti à des résultats très innovants et lui ont permis de trouver une alternative à l'approche sérielle. Cependant, si les probabilités apportent des résultats inattendus, source de nouveaux matériaux sonores, elles envisagent finalement souvent du macro timbre ou des macro sons continus ou discontinus [10], ce qui devient paradoxal pour une approche micro temporelle au départ. Par ailleurs, comme écrit Roads :

« Un caractère purement aléatoire est un idéal, et fait référence à une absence complète de biais dans le choix des valeurs. Des algorithmes logiciels pour l'aléatoire ont un caractère pseudo-aléatoire (déterministe). Les valeurs qu'ils produisent tendent à être uniformes et adirectionnelles, ne tendant que vers la moyenne. » [15]

4.2. Inter-relation des échelles temporelles

« L'idée d'une musique qui procède totalement d'un seul principe, où le tout et les détails sont déterminés par la série, cette idée s'est avérée utopique. » [2]

Ainsi, avec la synthèse granulaire, on obtient des flux ou des nuages de grains qui, certes, sont porteurs de « couleurs », mais dont le niveau de complexité reste souvent faible. C'est la raison pour laquelle lorsque le compositeur fait appel à ces techniques, il doit toujours par la suite « adapter » le nouveau matériau, par des actions manuelles[2], parfois très nombreuses, au tissu complexe de l'œuvre. En tout état de cause, nous soutenons que des techniques qui sont uniquement organisées par des unités temporelles d'ordre supérieur, et qui par conséquent négligent l'inter-relation entre les différents niveaux temporels, manquent de pertinence musicale[3]. Notre démarche se révèle à nouveau intermédiaire, entre la génération et la composition,

[1] Pour une autre approche, voir la transformée en ondelettes cf. Kronland-Marinet [6]

[2] Même Xenakis, qui a utilisé avec rigueur les lois des probabilités, à l'heure de composer n'hésite pas à laisser la place à des actions directes, plus intuitives, au profit de l'œuvre. [4]

[3] L'ordinateur se montre aujourd'hui particulièrement performant pour générer du matériau, avec toutefois quelques dérives possibles. Comme le signale Vaggione : *«… de notre point de vue concernant la composition musicale, on ne saurait échapper au niveau du matériau, bien entendu ; mais on est on droit d'excercer une pensée musicale concernant ce niveau – car le « matériau » peut être aussi illusoire qu'aliénant, du moment qu'on ne le pense pas musicalement, c'est-à-dire qu'on ne le compose pas. »* [19]

puisqu'on peut interagir à tout moment avec ce qu'on est en train de produire.

Si les travaux fondateurs de Risset sur la brillance sont d'une grande importance pour l'aspect purement micro temporel, ils le sont encore davantage pour l'inter-relation des échelles qui sont tissées à l'intérieur même du micro temps. Le micro temps, est essentiel pour la composition de sons, mais, outre le rapport à la perception, il n'implique pas de nouveaux problèmes (si problèmes il y a) compositionnels. C'est seulement un niveau de plus à composer, à intégrer.

4.3. Micro temps et temps réel

« tout système numérique ne peut atteindre en temps réel qu'une certaine limite de complexité… L'oreille adaptée aux finesses acoustiques, est particulièrement exigeante. Et il restera toujours le principe d'une limite. » [13]

Actuellement, le seuil de prédictibilité du logiciel avoisine les 40 à 60 ms dans de bonnes conditions selon la machine, ce qui représente justement une zone critique entre micro et macro temps. Différentes causes concourent à cet état de fait, à plusieurs niveaux des systèmes TR. A moins d'opérer sur des heuristiques, l'immense majorité des logiciels TR calculent le signal par rapport au passé et pâtissent donc fatalement d'un temps de latence intrinsèque à l'opération à effectuer, proportionnellement à la taille de sa fenêtre temporelle et sa vitesse de glissement. En outre, les interfaces de contrôle et leurs pilotes logiciels sont eux-mêmes cadencés à des vitesses qui interdisent un accès micro temporel précis. Enfin, la machine, la carte son, et le système d'exploitation possèdent leur propre temps de latence qui peut se cumuler aux autres.

Paradoxalement, le temps réel lui-même finit par devenir porteur de macro temps d'un point de vue informatique, une fois que le micro temps dont il pouvait être porteur a été filtré par les différents temps de latence rencontrés lors de la traversée de la chaîne informatique.

Nous étudions aujourd'hui la possibilité de lever l'impasse du micro temps dans la CBox grâce à une interface de prélèvement à plusieurs lignes temporelles qui permettraient de mieux opérer et de formaliser un travail micro temporel. Nous pourrions ainsi paramétrer une forme précise pour plusieurs prélèvements micro temporels en séquence, dans une échelle temporelle supérieure qui resterait confortable pour la machine et le logiciel. D'autre part, nous envisageons un moteur fréquentiel local de répétitions micro temporelles qui pourrait s'avérer complémentaire.

5. CONCLUSION

La frontière entre temps réel et temps différé se fait de moins en moins nette, maintenant que les logiciels dédiés au studio produisent le résultat de leurs calculs dans un temps proche de l'immédiateté, et réciproquement que les logiciels dédiés aux performances demandent parfois davantage de travail de composition en amont. La CBox autorise une exploration particulière *entre* ces deux pôles qui sont réellement indispensables ici ; on ne peut pas ne pas expérimenter son interactivité, ni s'abstenir de faire des choix compositionnels. Nous avons donc défini la notion de région fractionnaire pour qualifier les applications simultanément TR et TD, lorsque les deux temps sont obligatoirement expérimentés quelle que soit l'orientation préférentielle de la pratique effective.

La pratique et la réflexion autour de ce logiciel nous a aussi permis de clarifier le concept de circularité, en distinguant deux types, correspondants au nombre de processus circulaires concernés par le bouclage dynamique, du simple feedback à des inter-relations de niveau *n*.

Ensuite, la CBox demande de reformuler la question de l'articulation musicale du micro temps et du temps réel, en apportant des éléments de réponses techniques et structurels, grâce à des mécanismes compensatoires des temps d'analyse et de latence comme l'écriture explicite de prélèvements en séquence dans le signal et bientôt un moteur fréquentiel local de répétitions micro temporelles.

Enfin, pour ne pas finir, puisque nous préparons un concert pour clarinette et CBox, il y aura encore de nombreux allers et retours entre la programmation de ce curieux logiciel et nos expérimentations musicales, compositionnelles et scéniques, comme une conclusion circulaire…

6. REFERENCES

[1] Barkati, K. « K-Box : Programmation d'une extension logicielle pour instrument percussif. » *Textes de communications JIM 2001*, pp.27-37, Bourges, 2001.

[2] Dahlhaus, C. *Essais sur la Nouvelle Musique*, Editions Contrechamps, Paris, 2004.

[3] Dupuy, J.P. *Aux origines des sciences cognitives*, Editions La Découverte, Paris, 1994.

[4] Gibson, B. « Théorie et pratique dans la musique de Iannis Xenakis : à propos du montage », *Thèse de doctorat*, Ecole des Hautes Etudes en Sciences Sociales, Paris, 2003.

[5] Kiss, J. *Composition musicale et sciences cognitives*, Editions L'Harmattan, Paris, 2004.

[6] Kronland-Martinet, R. « The use of the wavelet transform for the analysis, synthesis and processing of speech and music sounds », *Computer Music Journal 12 n°4*, pp.11-20, MIT Press, Cambridge, Massachusetts, 1988.

[7] Malt, M. « Les mathématiques et la composition assistée par ordinateur, concepts outils et modèles », *Thèse de doctorat de Musique et musicologie du XXème siècle*, Ecole des hautes Etudes en Sciences Sociales, directeur de thèse Marc Battier, Paris, 2000.

[8] Maturana, H. et Varela, F. *L'arbre de la connaissance*, Addison-Wesley France, Paris, 1994.

[9] Miranda, E. R. « Music composition using cellular automata », *Languages of Design Vol.2*, pp. 105-117, USA, 1994.

[10] Pape, G. et Bokesoy, S. « Présentation de Stochos », *Actes des JIM 2003*, Montbéliard, 2003.
[11] Puckette, M. *Real-time audio analysis tools for Pd and MSP*, http://www.crca.ucsd.edu/~msp, 1998.
[12] Risset, J.C. et Wessel, D. « Exploration du timbre par analyse et synthèse », *Le timbre, métaphore pour la composition*, Editions Christian Bourgois, pp.102-131, Paris, 1991.
[13] Risset, J.C. « Temps et musique numérique », *Le temps en musique électroacoustique, Actes V*, pp.141-145, Editions Mnemosyne, Académie Bourges, 1999/2000.
[14] Roads, C. "Automated granular synthesis of sound". *Computer Music Journal,* **2** (2), pp. 61-62, MIT Press, Cambridge, Massachusetts, 1978.
[15] Roads, C. "Microsound", MIT Press, Cambridge, Massachusetts, 2001.
[16] Szendy, P. « Musique, temps réel ». *Résonance n°14*, Ircam - Centre Georges- Pompidou, Paris, octobre 1998.
[17] Truax, B. "Real-time granular synthesis with the DMX-1000." *Proceedings of the 1986 International Computer Music Conference*, pp.231-235, San Francisco, 1986.
[18] Vaggione, H. « Sons, temps, objet, syntaxe. Vers une approche multi-échelle dans la composition assistée par ordinateur. » pp.169-202, *Musique, rationalité, langage*, L'Harmattan, Paris, 1998.
[19] Vaggione, H. « De l'opératoire », *Formel/Informel*, L'Harmattan, pp.221-235, Paris, 2003.
[20] Varela, F. « Autonomie et connaissance. », *Essai sur le vivant*, Edition du Seuil, Paris, 1989.
[21] Wegner, P. "Why interaction is more powerful than algorithms", *Communications de l'ACM*, New York, 1997.
[22] Xenakis, I. « Musiques Formelles », *La Revue Musicale n°253-254*, Paris, 1963.

Annexe B

Entretien avec Iván Solano

Cet entretien a lieu le 4 avril 2008 chez Iván, après une première séance d'improvisation à la clarinette basse, avec Ifso et le pédalier.

B.1 Les projets Ifso et Plugiscope

B.1.1 Le projet avec Ifso

Karim Barkati : *J'aimerais bien que tu me parles de tes projets en lien avec notre collaboration, et en particulier avec les logiciels Ifso et Plugiscope.*

Iván Solano : Pour Ifso, il s'agissait de quelque chose d'adapté à des sons préenregistrés, prétravaillés, qui puisse s'insérer dans une partition, ou dans une improvisation. Donc des éléments qui sont préconçus, précomposés, mais aussi des éléments qui pourraient être préimprovisés et enregistrés en tant que matériau sonore, dans le but de construire une forme par la suite.

— *Et maintenant ?*

— Maintenant, il y a en tout cas l'idée d'une pièce, d'une durée d'une trentaine de minutes.

— *C'est beaucoup pour une pièce solo !*

— D'après l'improvisation qu'on a faite aujourd'hui, 10 minutes sont passées hyper vite ! Sachant qu'on a utilisé une polyphonie de seulement 5 lecteurs (plus l'instrument traité) ! Donc je crois qu'avec une plus grande densité ou une restructuration de la forme, on peut atteindre tranquillement les 45 minutes... Finalement, 30 minutes, je crois que c'est une durée acceptable. Le projet en soi, je crois que je le conçois de plus en plus en relation avec la résidence à Kyoto. Donc je vais essayer de focaliser, ou de diriger, toute l'énergie, ou toute l'envie, de Ifso vers une pièce pour clarinette basse et électronique en temps réel, même si l'électronique en temps réel, dans ses 80% est du temps réel « différé » parce que c'est préenregistré.

Je crois qu'avec un nombre de paramètres sur les sons préenregistrés au total assez restreint (on a : volume, panoramisation, vitesse, transposition, granulation, inverse, et un paramètre qui n'en est pas vraiment un – le bouclage), je crois qu'on est capable de faire beaucoup de choses. J'ai déjà la sensation que ça marche très bien. Plus, de l'autre côté, ce qu'il se passe à la clarinette basse, la source captée par les microphones (j'espère qu'on pourra en utiliser trois pour la pièce : j'ai la sensation que c'est bien de prendre tout l'instrument dans sa longueur, depuis l'embouchure jusqu'à la note la plus grave, et pouvoir travailler avec ça).

Pour le projet, je vais essayer de le diriger vers une idée, c'est une résidence à Kyoto, à la villa Kujoyama.

— *C'est quand ?*

— Normalement, je dois envoyer des dossiers au début de l'année 2009, pour une résidence en juin 2009 ; je crois que cette résidence dure trois mois.

Mon idée, c'est justement, à la fin de cette résidence, d'avoir enregistré du son avec mon instrument, du son dans la nature, et dans des milieux pas si naturels que ça, en milieu urbain, ou des voix, je ne sais pas trop, mais d'essayer de faire une sorte de parallèle entre le zen et la pièce. Le seul parallèle que je voulais faire, sans prétention mais plutôt comme une « excuse », c'est de se dire : il y a des choses qui, quand on est en méditation, passent par notre esprit, et ce genre de choses nous fait réagir mentalement, même si le zen nous dit de laisser passer les pensées, de les laisser circuler. Aujourd'hui, pendant l'improvisation, j'ai essayé de penser à ça, et de me dire qu'il y a des « particules sonores » dans les sons préenregistrés que je peux connaître mais que je vais « ré-analyser » en temps réel, ça va me faire penser à ça, et ça va changer ma façon de réagir avec mon instrument ; et ça va créer aussi un rythme, une respiration, des idées musicales, des phrasés, des textures…

— *Tu veux dire que tu te laisses faire par les idées, les pensées qui t'arrivent à l'écoute de ce qui sort des haut-parleurs ?*

— Oui, mais en même temps que je décide de ce que j'aimerais bien entendre ; parce que j'ai un pédalier qui me permet de lancer des sons que je connais d'avance. Par exemple, si je me dis que j'aimerais bien avoir une harmonie à dix notes mais que malheureusement je n'ai qu'une clarinette basse pour jouer, alors sur Ifso je peux lancer trois sons multiphoniques qui sont préenregistrés et après travailler en plein milieu de tous ces sons, et j'ai alors quelque chose qui est une composition verticale. Au niveau rythmique, c'est pareil : si je peux lancer un rythme, un bruit de clé, ou une respiration, un bruit de souffle, ou une gamme qui se répète plusieurs fois à intervalle régulier, alors je peux créer une sensation de rythme, que je peux aussi développer dans une granulation sur un son continu… Je crois justement que ce sont les trois paramètres de l'écoute : l'écoute du passé, celle du présent et celle du futur. Je veux dire que quand on est en train de jouer quelque chose, forcément, dans l'instant, on réfléchit – on écoute – ce qu'on a joué, autrement dit on sait d'où on vient, on est en train de jouer et d'entendre ce qu'on joue à ce moment précis, et on est aussi en train de penser à ce qu'on va jouer après. On doit faire la même chose avec Ifso, simplement c'est un autre instrument, qu'on joue en parallèle à son instrument acoustique auquel on doit réfléchir en même temps.

— C'est très vrai dans l'improvisation, mais cette triple « attention temporelle » sera peut-être moins vraie avec une pièce écrite, non ?

— Oui. C'est pour ça que je t'avais dit que la pièce écrite, à l'intérieur d'elle-même, va sûrement permettre la possibilité d'insérer des choses qui me permettront d'improviser. C'est comme des bulles d'air, à l'intérieur d'une chose dont tu sais qu'elle a une forme préétablie, mais qui vont te permettre d'évoluer d'une certaine façon... Peut-être qu'on peut appeler ça « évoluer d'une façon différente » mais ça n'évolue pas de façon différente plutôt ça « *in*volue d'une façon différente », à la l'intérieur de la pièce une autre fois qu'on la rejoue.

— Ce serait toi l'interprète dans ce projet-là ?

— Oui. Parce que d'abord j'ai travaillé avec toi, pour Ifso. Ensuite, parce que je pars de la base de préenregistrer mes propres sons, tant à la clarinette que d'autres sons. Je pars de la base de composer, d'écrire ce que j'aimerais bien dans la pièce, d'autant que j'ai les éléments et la capacité de jouer de l'instrument. Et la capacité d'écrire la pièce, celle de travailler avec Ifso et le pédalier... donc je crois que pour l'instant j'aurais une difficulté à penser à quelqu'un d'autre. Ça ne veut pas dire que je ne veux pas que cette pièce soit une pièce pour être jouée, mais je sais qu'il y a maintenant peut-être très peu de gens qui ont envie : un – de faire une pièce avec électronique en temps réel, deux – peut-être qu'il y a beaucoup de gens qui ont envie mais beaucoup moins qui en ont les possibilités, pour des questions techniques, parce qu'il faut un pédalier, une carte son, un ordinateur, et si possible il faut que ce soit des éléments que tu connaisses, trois micros, des câbles, des pieds, un système de diffusion, au moins trois haut-parleurs, une table de mixage, Max/Msp ou son *runtime*. Cela demande que quelqu'un aille chercher un *runtime*, qu'il se dise « qu'est-ce que c'est Max/Msp ? », « pourquoi j'ai besoin d'un *runtime* ? », « qu'est-ce que c'est cette application ? », « de quoi il s'agit ? », « pourquoi je dois télécharger un *runtime* pour faire marcher ce truc ? », et après, aussi, je pense qu'il n'y a pas tellement de gens qui ont envie d'amener la clarinette et un ordinateur pour jouer une pièce. Et si en plus c'est amener sa clarinette, un ordinateur, trois haut-parleurs, trois micros, une carte son, et un pédalier, je crois que ça fait beaucoup. Et donc c'est pour ça que je ne me dis pas au premier abord « je donne », ou « je vais faire cette pièce pour », non, je pense à la pièce pour moi... aussi parce que je pense que c'est quelque chose qui est assez « intérieur » dans le projet.

— Je connais des compositeurs qui n'écrivent pas pour eux-mêmes, toi tu arrives à écrire pour toi.

— Je ne l'ai jamais fait jusqu'à maintenant ! Chose qui est très curieuse... Donc c'est la première pièce à laquelle je pense pour moi. Je crois que c'est quelque chose qui part de l'idée basique d'une sorte de pièce « intérieure ». Et que Ifso a la possibilité de pouvoir exprimer en même temps plusieurs idées, sur une partition, sur une pièce, et sur un instrument qui est le mien. C'est très bizarre, je ne vois pas ça comme écrire pour quelqu'un... C'est une sorte de : « je vais réfléchir, je réfléchis sur ça, et je l'expose », et cette exposition fait la pièce, même si elle est écrite. C'est très curieux parce que je ne pense pas de la même façon à cette pièce qu'à la pièce pour chœur d'enfants par exemple, ou à la pièce pour flûte et percussion...

B.1.2 Le projet avec Plugiscope

Karim Barkati : *Tu peux m'en dire plus sur Plugiscope ?*

Iván Solano : L'idée du Plugiscope, pour lequel on n'a pas beaucoup travaillé ensemble pour l'instant, part de l'envie d'avoir un *multi*pédalier à la maison. Pourquoi, moi qui suis un instrumentiste d'instruments acoustiques, je n'aurais pas le droit d'avoir une pédale de guitare électrique sous mon pied ? Je pars de cette idée que je vais essayer « d'électroniser » mon instrument, mais le faire avec un peu de sens. Donc si je me dis qu'il y a déjà des gens qui ont construit des *plugins* qui peuvent transformer le son, alors pourquoi pas avoir un lecteur de ces *plugins* qui peut s'adapter avec une entrée microphone pour faire une sortie avec une transformation du son ? Pourquoi pas ?

Il n'y a pas une pièce à la base du projet, il n'y a pas une idée musicale précise, l'idée part simplement d'une certaine jalousie ! C'est un peu le mec qui voit une guitare électrique et qui se dit : « la vache ! ça serait super de pouvoir faire ça, dans une impro, aux Instants Chavirés[1] avec ses copains ! ». Peut-être que ça provient d'une expérience que j'avais faite justement aux Instants Chavirés avec un groupe qui s'appelle Mou Lipps. J'avais fait deux CD avec Hervé Boghossian, et sur son premier CD qui s'appelait RVB la première partie comprenait trente minutes de larsen... Larsen hyper-aïgu, que je voulais travailler avec la clarinette basse, et avec le deuxième clarinettiste on essayait de jouer avec ce larsen de la guitare. On incorporait une texture, un son acoustique à un son électrique ; je ne dis pas « électronique » parce que je crois que c'est électrique la sensation quand tu approches une guitare électrique avec sa cellule près du haut-parleur de l'ampli, pour moi ça sonne électrique. Ça pétait les oreilles, carrément.

Après on a fait un autre disque, où j'ai fait une autre pièce avec Mathieu Saladin : des sons de clarinette basse qu'on a enregistrés, puis qu'on a bidouillés chacun de notre côté, sur ordinateur, puis qu'on a mis ensemble pour construire la pièce ; pièce en temps différé. Mais, quand on a fait le premier CD, on est parti aux Instants Chavirés pour jouer avec plusieurs personnes qui avaient une idée très minimaliste de l'improvisation, une sorte de « pureté » du son et de statisme dans le développement... et donc, dans les concerts qui ont suivi le CD, parfois je partais dans des choses complètement différentes et complètement barrées ! Parce que j'avais vraiment envie, quand ils étaient dans un roulement hyper lent, je commençais à jouer des sons ou des gammes hyper fort dans les aigus ou dans les graves, un peu free-jazz, style Jimmy Giuffre. C'est là que je me suis dit que j'aimerais bien que mon instrument acoustique soit capable d'être encore plus « performant » : donc si j'ai envie que mon son soit distordu, transformé, harmonisé, transposé, ou complètement changé, alors soit j'aurais dû m'acheter dix pédaliers différents, des pédaliers de guitare, un octavieur pour jouer plus grave, un préenregistreur pour déclencher des plages, etc., et donc j'ai pensé que s'il y a la possibilité d'un patch[2] qui lit des *plugins* qui sont déjà faits, et qui font tout ce travail, je n'aurais plus besoin que d'un ordinateur et d'un seul pédalier qui puisse le contrôler, et ma clarinette, parce que les *plugins*, je les ai déjà !

1. Les *Instants Chavirés* sont une salle de concert, située à Montreuil en région parisienne, dont la programmation est tournée vers l'improvisation et l'expérimentation.

2. Dans cet entretien, un « patch » désigne un programme musical réalisé avec l'environnement de programmation Max/Msp.

— *Ça fait longtemps que tu travailles avec des* plugins *?*

— Oui. Je travaille beaucoup sur ProTools avec toute sorte de *plugins*, mais comme les Vst ne sont pas compatibles avec ProTools, voilà ce que je fais en général : j'enregistre mes sons, je les travaille en Vst avec Peak, puis je les remets dans ProTools. Je travaille avec des traitements audio depuis une quinzaine d'années. J'aime beaucoup la matière sonore, j'aime bien travailler avec elle, et en fait le début de la composition pour moi vient de là ! J'ai commencé par faire de la musique pour le théâtre : à seize ans à peu près, j'ai eu une occasion ; mon seul accès c'était mon instrument ou tout ce que je pouvais enregistrer sur ordinateur… Tout au début, je jouais des mélodies avec la clarinette en si bémol, et après j'ai travaillé avec des sons, parfois très réalistes, des sons concrets, de voix, de rue, d'animaux, et donc j'essayais d'adapter ce que je faisais à la pièce de théâtre. Pour moi la musique était aussi un personnage, qui pouvait discuter avec les autres personnages, ou un personnage qui pouvait collaborer, dans une rythmique, ou encore aller contre l'action. Donc il y avait trois ou quatre paramètres différents dans ce personnage.

— *Maintenant que Plugiscope est fini d'être programmé, qu'est-ce que tu envisages de faire avec ?*

— J'ai une pièce à faire pour un chœur de chambre, le chœur de chambre de Strasbourg, avec 4 femmes et 2 hommes pour cette pièce[3]. J'ai l'idée de voir deux trios – 2 femmes / 1 homme –, avec un positionnement sur scène : un trio à gauche, l'autre à droite, et sûrement les femmes devant et les hommes derrière. C'est en parlant avec la chef de chœur que m'est venue l'idée d'utiliser l'électronique. *A priori*, ce n'était pas de l'électronique en temps réel, mais un CD, avec éventuellement des pistes séparées à lancer au fur et à mesure de la pièce. Donc des séquences qui ne bloqueraient pas la rythmique de la partie écrite et chantée, mais qui ajoutent aux possibilités des chanteurs avec des choses qu'ils ne peuvent pas faire.

Quand je compose quelque chose avec électronique, j'essaie toujours de me dire : « comment pourrais-je utiliser l'électronique en différé de façon à ce que la personne qui joue en profite ? ». Si c'est un chanteur, qu'il puisse avoir des références sonores pour prendre ses notes à un moment précis, sans devoir frapper un diapason et l'approcher de son oreille, ou lui donner une rythmique précise. Si c'est des enfants, comme la pièce pour chœur d'enfants, leur donner justement des notes de référence pour qu'ils puissent les chanter après. Et aussi faire des choses qu'on ne peut pas faire avec l'instrument ou la voix : par exemple des grands intervalles à grande vitesse avec une justesse exacte, ce qui n'est pas organique pour la voix, mais faisable avec un ordinateur ; je peux prendre des sons, et construire un chant très véloce avec des intervalles très grands et très exacts, un peu comme dans Farinelli. Pour ces raisons, je me suis dit qu'il y avait besoin de l'électronique, en temps différé, et après, en parlant de l'esthétique de la pièce avec la chef de chœur, j'ai pensé à une esthétique japonisante (ce qui n'est pas le cas d'Ifso, malgré les apparences). Qu'est-ce que ça veut dire une esthétique « japonisante » ? Peut-être que c'est un cliché… Je pars de l'idée du kabuki et des intervalles qui sont chantés dans le kabuki, justement avec des sauts très grands. Je te fais écouter « Le fil d'Ariane », une pièce électroacoustique de 7 minutes qui date de 2006, pour cinq comédiennes qui chantent

3. Cette pièce devrait s'appeler *Kyōgen – 22 tableaux sur l'amour et la vie*. On peut traduire l'expression japonaise *« Kyōgen »* par « parole sauvage ».

et qui lisent des textes dans cinq différentes langues, où j'avais préenregistré la chanteuse. Le début contient un chant avec des attaques aiguës qui glissent vers la note tenue, comme dans le kabuki. On entend trois voix mais il n'y a qu'une seule voix, les deux autres voix étant de simples transpositions, à des intervalles qui construisent des accords spéciaux, mais pas en temps réel : c'est préenregistré et prétravaillé ; le chanteur sait quand il doit partir pour sonner avec l'accord.

Mais pour la future pièce, l'idée est justement de produire cette même sensation harmonique, en temps réel. D'où mon idée d'utiliser Plugiscope, et deux microphones (un pour chaque trio) que j'utilise quand je veux des transformations qui soient des harmonisations de ce genre. Donc j'ajoute un transpositeur, ou un vocodeur, ou une sonorisation différente à la voix, au fur et à mesure que la pièce se déroule. Durée de la pièce : à peu près 10 minutes. C'est là que je me suis dit : tiens, je peux utiliser Plugiscope que tu avais développé quand je te l'avais demandé, et qui est fini. Il y a beaucoup de paramètres à l'intérieur, qui sont modifiables – chose qui est très bien –, c'est-à-dire que tous les paramètres d'un *plugin* Vst sont modifiables à travers le pédalier – chose qui n'est pas tout à fait courante dans un pédalier de guitare par exemple. Donc on a une multiplication infinie du *plugin* premier, et qui en plus peut être changé en temps réel. Je me suis dit qu'il faudrait l'appliquer. Maintenant, il faut que je prenne Plugiscope et que j'arrive à retrouver cette sonorité harmonisée, pour pouvoir l'obtenir à partir des quatre voix de femme qui vont être amplifiées. Peut-être qu'il y a aussi la possibilité de ne pas en avoir besoin puisque j'ai deux femmes / un homme / deux femmes / un homme, j'ai déjà trois voix, donc c'est possible de faire ça seulement avec l'écriture, mais j'ai la sensation qu'un petit côté de profondeur, d'espace, m'est toujours nécessaire. Quand je dis « espace », je veux dire que le son que tu écoutes là a une réverbération derrière qui est donnée par la transformation d'une des voix avec les deux autres intervalles calculés, et par la position de ces voix transformées un peu décalée dans le temps à cause du traitement, ce qui donne une sorte de réverbération « naturelle » au traitement. Lorsque l'attaque ne se produit pas exactement en même temps mais quelques millisecondes après, on commence à avoir une réverbération, jusqu'au moment où on entend la deuxième attaque comme attaque.

En dessous de cette limite on a une réverbération « naturelle », et ça crée un espace. Cet espace, on ne peut pas l'écrire pour des instrumentistes, ni une réverbération. On peut écrire une note qui tient après l'attaque d'une autre, mais alors on dépense l'énergie d'un chanteur pour ça ; par contre, si quelqu'un chante une note et que l'électronique ajoute une réverbération derrière, il peut continuer à chanter pendant que la réverbération de cette note est installée.

— Donc là, sur cette pièce, c'est le « besoin d'espace » qui te conduirait à utiliser l'électronique temps réel ?

— Oui. Et là, il y a la vraie différence entre Ifso associé à l'idée de composer pour moi (même si je ne le vois pas comme une composition pour moi, je ne sais pas pourquoi), et l'utilisation du Plugiscope, parce que dans cette pièce pour chœur je suis le modèle de composition que j'utilise tout le temps… pas de façon esthétique, ou de façon formelle, ou de façon structurelle, mais de la façon dont je pense. Quand je pense à une pièce, je pense à combien d'instruments j'ai (la formation), je pense à leur disposition dans l'espace (hyper-important pour moi), je pense à la question de leur déplacement (ou pas) dans l'espace en même temps qu'ils chantent ou qu'ils jouent, une sorte de mise en scène qui ne vient pas du rapport théâtral de la pièce mais du rapport spatial du son et de son

déplacement dans l'air, une « spatialisation active », en temps réel et humaine ! Il y a une autre pièce, pour quatuor de saxophones et douze voix, où je pars dans un délire : il y aura du mouvement pendant qu'ils chantent… Il y aura peut-être des accords qui commencent à se déplacer dans l'espace, des notes qui commencent à aller ailleurs…

J'ai besoin de penser à l'espace, j'ai besoin de créer mon propre espace, par rapport à chacune des pièces, mais un espace « réel » en fait. Une chose qui m'a beaucoup plu dans Ifso, aujourd'hui, c'est la sensation d'avoir une tridimensionnalité, d'avoir un objet tridimensionnel, un objet qui a une profondeur, une hauteur, et une latéralité. Ça crée un espace tridimensionnel, et j'ai besoin que cet espace fasse partie de la pièce en permanence. Pareil pour la pièce pour le chœur de chambre et Plugiscope, où cet espace va être créé par Plugiscope qui va permettre d'utiliser des *plugins* pour transformer la voix, mais pas tout le temps, parce qu'il y a aussi des moments où j'ai envie que la voix soit proche, sans réverbération. Alors on peut s'approcher dans cet espace, on peut s'éloigner, on peut magnifier, on peut verticaliser, on peut faire des accords, ou simplement on peut lancer un son qui rajoute quelque chose à ce qui est en train d'être chanté.

— *Est-ce que tu sais déjà quels plugiciels tu vas utiliser pour cette pièce ? Et combien ?*

— Je pense déjà à un vocodeur, un VST de Prosonic qui s'appelle Orange, avec ses avantages et ses inconvénients. Il permet d'avoir des sons « vocalisés » : dans l'un des paramètres on peut choisir une voyelle, ou du souffle avec une application d'air « chanté » dans la note MIDI, parce qu'il y a un clavier dans ce *plugin* qui envoie des notes, éventuellement transformées par du bruit rose ou du bruit blanc ou par un modulateur en anneau, ou justement par des sons de voyelles.

— *Donc ce sont ses « qualités vocales » que tu aimerais utiliser avec le chœur ?*

— Ses qualités vocales, et aussi la possibilité d'avoir des notes très précises dans l'harmonie : le clavier produit des notes exactes, qu'on peut faire écouter aux chanteurs, en même temps que leur voix est en train de se faire transformer par ce *plugin*. Cette harmonisation est un outil qui permet aux chanteurs de chanter plus facilement. Je crois que le *plugin* ou le patch doit être au service de la personne qui chante ou qui joue quelque chose. Et pour arriver à ça, d'abord il faut choisir le *plugin* qui pourrait être intéressant, et après il faut donner à l'interprète la possibilité de se dire : « tiens, là il y a une note qui justement est celle que je vais devoir chanter plus tard », donc placer des repères auditifs.

— *Tu penses à d'autres plugiciels ?*

— J'avais pensé à un transpositeur, sûrement, un spatialisateur, peut-être, quelque chose qui fait bouger les sons, et éventuellement… je verrai bien : il y a tellement de VST, je ne me rappelle plus toutes les possibilités qu'on a, alors je verrai bien quand la pièce sera finie de composer, même si je vais la composer au fur et à mesure que je teste les VST.

— *À qui sera confié le déclenchement au pédalier ?*

— Ça va être fait par moi, donc il y a quand même dans cette pièce un technicien, quelqu'un qui déclenche les effets, Plugiscope, les *plugins*, ou des séquences sonores, à un moment donné en rapport avec la partition écrite. Pour permettre le maximum de

liberté au chef de chœur et aux chanteurs, parce que, là on revient à la même question, je ne crois pas tout le temps au fait d'impliquer les interprètes dans certaines questions techniques... J'y crois de mon côté, mais là c'est un projet qui est précis : il est fait pour un enregistrement. Peut-être que la pièce sera mise en concert par les chanteurs, mais pour l'instant ça part d'une idée d'enregistrement. Et donc je pense que je peux être là pour appuyer sur les boutons, puisque c'est moi qui vais composer la pièce et que je connais Plugiscope.

— *Tu peux m'en dire plus sur l'enregistrement ?*

— L'enregistrement est une démarche de Catherine Bolzinger, qui est la chef du chœur de chambre de Strasbourg. *A priori*, le projet se base sur une sorte d'hommage à Ivan Fedele, dans le sens où elle veut enregistrer certaines pièces chorales des élèves de la classe d'Ivan Fedele, qui ont déjà passé leur prix de composition pour certains et pas encore pour d'autres. C'est un projet qui va partir du conservatoire de Strasbourg.

Peut-être que c'est moi qui vais poser le plus de problèmes avec ma pièce, justement parce qu'elle a de l'électronique. Je ne suis pas sûr d'enregistrer l'électronique en même temps que j'enregistrerai les voix transformées en temps réel, mais j'ajouterai peut-être après les sons préenregistrés, la partie temps différé. Quatre haut-parleurs : deux haut-parleurs derrière chaque groupe, en stéréophonie. L'idée, c'est d'avoir une sorte de toile à quatre pattes plus un centre pour chaque groupe : la voix d'homme au milieu, deux femmes devant et deux haut-parleurs derrière qui joueront à partir des voix des femmes, ce qui rend des voix de femmes devant et derrière celle des hommes. Des deux côtés pareil. Donc on a une sorte de structure qui peut tourner autour des voix masculines, on peut faire bouger de droite à gauche de chaque côté, avec une double stéréo gauche et une double stéréo droite.

— *Est-ce que tu as déjà un nom pour la pièce pour le chœur de chambre et pour celle pour clarinette basse et électronique ?*

— Non, pas encore, mais la pièce pour clarinette basse et Ifso pourrait trouver son titre d'après le livre de l'auteur japonais Matsuo Bashô : *The Narrow Road to Oku*. Au dix-septième siècle, cet homme très cultivé, fils d'un samouraï, entreprend un voyage et écrit, à chaque étape, des haïkus, ces poèmes très courts de trois lignes consécutives composés en 5 puis 7 puis à nouveau 5 syllabes ; au total toujours 17 syllabes. Je ne vais pas utiliser cette structure 5-7-5, mais l'idée du voyage, parce que c'est une pièce pour le Japon, qui n'est pas tout près, et aussi parce qu'il y a un autre écrivain qui m'intéresse beaucoup : Kavafis, et son poème *Le voyage à Ithaque* qui dit en substance que ce n'est pas la destination qui importe mais le voyage en lui-même. C'est le voyage qui nous apprend des choses, sur le chemin, donc le but n'est pas d'arriver quelque part. Cela, curieusement, a une relation avec l'idée du zen pour moi, avec l'idée de laisser passer les pensées, sans les attraper mais laisser continuer la pensée à se développer : parce qu'on n'arrive pas à un but précis, simplement il y a des choses qui circulent.

C'est un texte profond, et mon résumé est peut-être trop synthétique mais il donne une première idée : ce n'est pas un but d'atteindre Ithaque, parce que peut-être qu'Ithaque n'existe pas. Ce n'est pas non plus un lieu tout à fait fictif car le voyage en soi est sûrement Ithaque, « Ithaque t'a donné le beau voyage » ; donc ce n'est pas forcément un lieu, mais un chemin, rempli de plusieurs matins, de plusieurs aventures, de plusieurs découvertes, de plusieurs regards...

B.1.3 La place de l'interaction

Karim Barkati : *Ces deux projets, l'un avec Ifso, l'autre avec Plugiscope, sont des projets où l'interaction semble prendre une place importante : quelle place accordes-tu à l'interaction en général et quelle place accordes-tu à l'interaction dans ces pièces en particulier ?*

Iván Solano : Concernant l'interaction en général, je crois que ça fait longtemps que je pense la même chose sur l'électronique en temps réel : pour moi, l'électronique en temps réel doit être composée. Qu'est-ce que je veux dire par *composée* ? Il y a un compositeur qui a un langage, par des techniques il développe ce langage pour transmettre une idée de façon musicale, et cette idée qui est transmise au papier doit être interprétée par l'instrumentiste, par quelqu'un qui puisse l'exprimer. L'instrument de cet interprète peut être un ordinateur, sauf qu'en général il n'y a pas d'instrumentistes qui jouent de l'ordinateur, ni dans les concerts de musique classique ni dans les concerts de musique contemporaine, mais il y en a dans la musique « électro » et dans la musique improvisée électronique. Comme je ne suis pas dans un parcours de « musique électronique », j'essaie de composer justement, de me dire que chaque partie électronique doit être précomposée. Il s'agit de connaître les paramètres de l'instrument qu'on va utiliser ; peut-être que cet instrument dépend d'un programme Max/Msp ou autre. L'interaction doit être quelque chose qui permet à l'instrumentiste – qui est en train de jouer d'un instrument – de jouer d'un autre instrument en même temps. Cet *autre* instrument, dont il joue en même temps qu'il joue de son instrument d'origine, va transformer d'une façon ou d'une autre le résultat sonore.

— *Est-ce que l'interaction c'est quelque chose d'important pour toi dans la composition ? Sachant que notre projet avec Ifso est très interactif...*

— Jusqu'à maintenant je ne me posais pas la question. D'abord parce que les instruments sonores auxquels j'ai eu accès dans les concerts, les choses que j'écoutais me paraissaient toujours semblables... J'ai eu des moments, à certains concerts à l'Ircam, à Paris 8 ou ailleurs, où je me suis dit : « tiens, ça sonne du Max/Msp ». C'est bien en tant qu'instrument, mais ce n'est pas bien en tant que compositeur ou instrumentiste : je crois que le son doit être proche de l'idée musicale qu'on veut transmettre. Si je mets une transformation en temps réel, et que le son me fait penser à un *plugin*, ou à un programme, comme Max/Msp, alors je suis en train d'enlever beaucoup des caractéristiques de l'instrument sonore en lui-même. Donc, j'ai toujours eu la sensation que ça aplatissait.

— *Donc l'interaction, c'est plutôt quelque chose que tu écartais ?*

— Oui, beaucoup. Et aussi parce que j'avais une complète méconnaissance de Max/Msp, donc pour des questions techniques, je ne me sentais pas capable, et d'ailleurs toujours pas. Je ne suis pas en train de composer l'électronique de la façon dont je t'ai parlé, je n'ai pas écrit exactement ce que j'aimerais que le son fasse à ce moment-là, même si je suis en train de le faire avec la pièce pour clarinette basse... mais je ne me sentais pas capable de pouvoir construire mes patchs pour faire ce que j'aimerais bien faire et que ça sonne comme j'aimerais bien que ça sonne.

Alors que je peux prendre un violon et lui dire de jouer *pizz bartok* une note, *pizz* normal une autre note, et *col legno battuto* la troisième, je sais ce que va être le résultat, j'en ai une idée.

— *Qu'est-ce qui fait que ces projets intègrent de l'interaction tout à coup, alors que tu l'écartais jusque là ?*

— Il y a un rapport avec quelqu'un, toi, qui part de l'interaction entre l'instrumentiste et la machine, et ce logiciel devient pour moi un instrument. Quand on parle d'Ifso et qu'on commence à se creuser la tête à son sujet, je ne te pense pas comme un programmeur, mais j'ai la sensation que tu es un facteur d'instrument, un luthier... dans le sens où c'est comme si j'arrivais à me procurer un alto et que tout d'un coup j'aimerais bien que tu fasses une caisse de résonance plus grande, pour accorder l'instrument différemment, et aussi j'aimerais bien que la touche soit un peu plus large, pour que mes *pizz bartok* sonnent beaucoup plus forts... J'ai la sensation que je m'adresse à quelqu'un qui est en train de fabriquer, de construire un instrument pour moi, qui va me permettre de le jouer en même temps que je vais jouer de ma clarinette.

— *J'ai cerné deux obstacles à l'interaction pour toi jusqu'à maintenant : le premier, c'est que ça sonne « toujours pareil », et le deuxième, ce sont les difficultés techniques. Il me semble que travailler avec un « luthier » peut résoudre la dimension technique mais pas forcément le fait que ça sonne toujours pareil, non ?*

— Si, aussi, parce que justement quand on pense à un luthier, l'instrument devient un instrument particulier. On peut même choisir le vernis !

— *C'est vrai qu'on a aussi travaillé sur les couleurs !*

— Justement, c'est : quelle est la réaction de tes yeux quand tu as l'instrument devant toi ? Quelle est la réaction de tes oreilles quand tu constates qu'un son déclenché à 127 est trop fort ? On le met à 80 au démarrage et ça, toi, tu le reprogrammes en deux secondes... En fait, toi, en tant que luthier, tu connais l'instrument qu'on est en train de construire, et cet instrument n'a pas un rapport avec Max/Msp de façon directe, c'est comme si Max/Msp devenait du bois pour construire quelque chose, et là c'est génial ! Tu arrives à faire un instrument qui ne va pas sonner pareil... Si au lieu de prendre du bois, ou du métal, ou une combinaison des deux, tu prenais seulement une anche, un tuyau et des clés, alors tu pourrais peut-être construire une flûte, une clarinette, un saxophone, et ça va sonner plus ou moins comme une flûte, une clarinette, un saxophone. Mais si par contre tu prends la matière première : comme on a remarqué lors de l'improvisation, tout d'un coup on entend plusieurs instruments... Est-ce que tu as la sensation que ça sonne comme du Max/Msp ?

— *Hmm, non. Peut-être la granulation, éventuellement, mais je l'ai vraiment retravaillée de manière particulière...*

— Justement : qu'est-ce que tu as pensé quand tu as commencé à travailler sur la granulation ?

— *Il y avait des préréglages standards, je les ai écoutés longuement, il y a une vingtaine de paramètres, j'ai essayé de penser à ton projet musical, je savais que tu voulais une*

pédale progressive unidimensionnelle de 0 à 127 qui parte de « pas de granulation » à « beaucoup de granulation », et donc que ce soit un seul paramètre alors qu'il y en a 20 ! Comment réduire quelque chose de multidimensionnel ?

— Donc ça te fait prendre des décisions, ça te fait penser à une couleur…

— *Absolument. Et je me suis dit qu'il fallait que je te propose effectivement un axe, pour qu'après, à partir de cette proposition, tes oreilles réagissent, pour que tu puisses me dire « plus de ceci », « moins de cela », ou « c'est bon, on garde comme ça ». Techniquement, je suis parti sur une interpolation entre deux préréglages des 20 paramètres, et je les ai faits de manière à ce que la continuité entre ces deux préréglages sonne de façon cohérente quand on utilise la pédale, je les ai testés moi-même, et vraiment j'ai passé trois bonnes demi-journées à ajuster ces préréglages…*

— C'est ce qui fait la différence entre se dire « tiens, j'ai un tuyau, une anche et des clefs » et se dire « je vais prendre un bon morceau de bois, et voir ce que je peux en sortir, et de façon à ce que les gens ne voient pas que c'est un morceau de bois mais un instrument ». Et donc on arrive à la couleur. On arrive aussi à la texture, et on arrive aussi à quoi ? Quand on a des *presets*, mais aussi du son en temps réel et aussi des sons préenregistrés qui ont une matière première qui est proche de la même texture (ici la clarinette basse), alors on commence à pouvoir écarter les sons électriques… Et d'une certaine façon on n'est plus avec un objet qui rappelle Max/Msp ! J'ai la sensation que quand on réécoute l'enregistrement de l'improvisation de tout à l'heure, on ne sens pas que c'est du Max/Msp… et ça, c'est génial. La seule chose qui me manque (et c'est une question personnelle parce que je crois qu'on a la possibilité d'aller dans le son), c'est d'avoir une égalisation.

— *C'est pour bientôt !*

— C'est simplement quelque chose qui fait partie du son naturel, et de l'idée de construire un espace. On est en train de construire un espace, parce qu'on a des haut-parleurs qui vont délimiter la sortie du son et transformer la perception de la personne qui est en face. Et je n'aimerais pas que ces gens-là se disent « bof, encore une pièce électroacoustique, encore une pièce électrique, encore un truc avec du Max/Msp, encore un granulateur qui fait les mêmes sons ». Ça ne sonne pas comme étant de la musique. Ça sonne trop aléatoire, ça ne sonne pas contrôlable, pas dirigeable, pas phrasé, pas musical, pas texturé, pas profond, pas vertical… Il manque alors tous les paramètres qui sont dans la composition.

— *La deuxième partie de ma question, c'est l'interaction dans ces pièces-là, et en particulier avec Ifso, peut-être qu'elle est plus limitée dans Plugiscope…*

— Je crois que ce sont deux choses différentes… Je crois qu'au contraire l'interaction est plus limitée dans Ifso. Dans Ifso, on a cadré les paramètres, avec un maximum de 6 paramètres par piste lancée (vitesse, transposition, panoramique, volume, granulation, inversion – je ne compte pas le bouclage comme paramètre). On a donc 6 paramètres qu'on a fixés, vraiment, alors que dans Plugiscope, on n'a pas ça, mais tous les paramètres de n'importe quel *plugin* Vst, donc une quantité de paramètres virtuellement infinie ! Donc justement, dans la pièce avec Plugiscope je vais devoir choisir au préalable parmi cette

marée de *plugins* Vst que j'ai dans mon ordinateur, une cinquantaine, multipliée par tous les paramètres modifiables par Plugiscope en temps réel (ou pas).

— *D'accord. Ce que je voulais dire, c'est que dans le projet avec Ifso il y a une demi-heure où l'instrumentiste, toi, sera avec le pédalier et tu où vas interagir, changer les traitements en temps réel, pendant que tu joues. Alors que dans l'autre pièce, qui est plus courte, les musiciens qui sont vraiment sur scène ne vont pas interagir directement avec l'ordinateur, puisque c'est toi qui auras le pédalier, hors scène...*

— C'est vrai qu'il n'y a pas une interaction dans le même sens, tu as raison. J'avais pensé à comment interagit le son avec la transformation, les possibilités de traitement. Il y a l'interaction au sens d'interface – où l'instrumentiste joue avec le pédalier, son instrument, l'ordinateur, les entrées et les sorties, où il est en complète interaction avec la machine –, et dans l'autre cas non. Mais l'interaction – ou l'action – de la machine sur le son est beaucoup plus pointue dans Ifso que dans Plugiscope ; en tout cas, à ce moment de la composition de la pièce avec Plugiscope.

B.1.4 L'écriture du pédalier

Karim Barkati : *Quelle écriture envisages-tu pour ces pièces, et en particulier en ce qui concerne le pédalier ?*

Iván Solano : Très bonne question ! On va partir de la pièce pour voix, parce que, comme tu dis, elle est moins « interactive ».L'écriture, dans cette pièce, va être carrément du déclenchement de pédales qui transforme le son à un moment donné. Donc il y a une partition, avec des numéros, chaque numéro est un préréglage. Le lancement d'un numéro fait que Plugiscope va chercher un *plugin* Vst, le charge et il l'applique avec les paramètres qu'on a préétablis.

— *Donc que du déclenchement et rien de progressif ?*

— Non, rien de progressif, que du déclenchement, pas de pédale d'expression pour l'instant. On a une palette de couleurs qu'on applique sur un son chanté.

— *Tu comptes l'écrire sur le conducteur ?*

— Oui, sur le conducteur, et sûrement que tout le monde sera au courant de ce qui va se passer, car chacun aura le conducteur ; en général, pour les chœurs je fais comme ça. Il y aura non seulement les numéros des pédales, mais aussi une graphie de l'électroacoustique différée avec la forme d'onde, que la référence soit visuelle aussi pour le chanteur, plus des mots (comme « sons de cloches », « attaque », « sforzando », « note grave », etc.), et avec une écriture proportionnelle.

— *Et pour Ifso ?*

— Pour Ifso, l'écriture va partir des pistes préenregistrées : je vais enregistrer les sons, je vais construire une structure formelle de 30 minutes, qui va indiquer plus ou moins à quels endroits ces sons sont déclenchés. Comme je connais la transcription écrite de ces

sons préenregistrés, je vais utiliser ces graphismes pour composer la partie de l'instrument, qui ne sera peut-être pas au même endroit que la piste qui est déclenchée, mais à un autre côté – bruits de clés, notes, composition normale –, en partant des sons qui sont déjà préenregistrés.

Ceci peut m'amener au texte, au souffle, aux bruits de clés, aux *flatterzunge*, aux trilles, aux multiphoniques, tous les modes de jeu.

— *Et pour l'écriture de la pédale d'expression ?*

— Il faudrait que je vois si je peux adapter à chaque moment un graphisme pour chacun des paramètres. Donc mon idée c'était : si on doit lancer la piste 1 (en préréglage ou pas) pour changer une liste de paramètres (par exemple « p » pour panoramique suivi d'un numéro), alors la personne qui regarde la partition voit « P 1 », et donc elle doit se dire qu'en préréglage de la piste 1, il faut appuyer sur la pédale numéro 1 ; la piste devient alors sélectionnée (jaune à l'écran), l'instrumentiste va sur le réglage de panoramique, il accroche la pédale d'expression gauche et fait monter la panoramique jusqu'à 1, et après il relance la lecture du tampon 1. Voilà, c'est pour un seul paramètre qui est « P » ; on utiliserait « G » pour « granulation », « V » pour « vitesse », « T » pour « transposition », etc.

Cette liste de paramètres pourrait s'appliquer de la même manière en temps réel que pour le préréglage. Si on est sur la panoramique pour un son long déjà en lecture (par exemple la piste 6), et que j'aimerais bien que le son circule de droite à gauche, j'écris une ligne qui commence par 1 et qui fini par 0, donc une flèche, avec deux numéros. L'interprète peut voir alors sur la partition que la pédale, sur ce son, doit passer de 1 à 0 sur la panoramique (en utilisant la pédale gauche d'expression), à la vitesse qui correspond proportionnellement aux durées écrites, comme on indique un changement de timbre pour le passage entre le son aérien et le son plein. On écrit un cercle vide et un cercle rempli, ou un losange vide et un losange rempli, et on trace une ligne horizontale entre les deux tout en haut des notes (au-dessus de la portée), pour indiquer le passage progressif entre les deux.

B.1.5 Comparaison entre Ifso et Plugiscope

Karim Barkati : *Pour toi, quelles sont les différences et les points communs d'Ifso et du Plugiscope ? Sachant qu'on a déjà abordé cette question précédemment...*

Iván Solano : Ifso est capable de jouer des pistes, et utilise des paramètres prédéfinis, très exacts, pour transformer tant les pistes que l'entrée de l'instrument. Plugiscope transforme le son à travers un *plugin*, donc pas directement par lui-même, et utilise les paramètres de ce *plugin* pour transformer l'entrée du microphone. Voilà déjà une grande différence, je crois.

— *Et les points communs ?*

— Sûrement la possibilité de transformation avec la pédale d'expression en temps réel : si je me dis qu'à un moment donné sur Ifso j'aimerais bien une panoramisation, et que sur Plugiscope aussi j'ai trouvé un *plugin* qui fait de la panoramisation que je peux modifier en temps réel avec la pédale d'expression, ça va devenir très similaire.

B.2 L'informatique et la musique

B.2.1 Un intérêt prononcé pour la matière et l'espace sonores

Karim Barkati : *Pourquoi travailler avec l'informatique ?*

Iván Solano : La même réponse qu'avant : jusqu'à maintenant, je n'ai pas trouvé la possibilité de m'approcher d'un instrument qui me plaise en informatique... Et toi, en tant que « luthier informatique », tu as réussi, au fur et à mesure qu'on travaille sur le projet, à construire un instrument, qui me paraît en plus très près de moi. Je crois que c'est aussi la sensation que tu as eu tout à l'heure, lors de l'improvisation.

— *Oui, tout à l'heure, je t'ai trouvé familier avec, à l'aise.*

— On avait parlé de la forme de l'instrument, du vernis, des cordes, des clés, des anches, on avait parlé de tous les éléments qui sont dans le logiciel. J'ai pris des références avec le pied gauche, et ces références marchent, au fur et à mesure que j'ai découvert cet instrument qui a été construit avec toi. Donc « pourquoi travailler avec l'informatique ? » : parce que je peux travailler avec toi.

— *C'est une réponse historique ou chronologique ponctuelle, mais ma question serait plutôt quel sens musical, en tant que compositeur – ou alors en tant qu'improvisateur, instrumentiste, ou interprète –, quel sens trouves-tu à travailler avec l'informatique ? Cet après-midi tu as fait une improvisation « avec informatique », et j'imagine que ça veut dire quelque chose pour toi...*

— Ok. En tant qu'interprète, je crois que ça fait des années que je suis là : je n'ai pas peur de prendre mon instrument devant un ordinateur, me mettre à jouer une pièce qui est écrite, et déclencher des événements, électroacoustiques ou en temps réel, qui sont écrits sur une partition. Donc du côté instrumentiste, c'est l'envie de se dire qu'il y a une relation entre l'instrument et l'électronique.

— *Par exemple, pourquoi tu ne te contentes pas de jouer de la clarinette, ou de composer pour des formations strictement acoustiques ?*

— Justement, je pars de là : ça fait peut-être dix ans que je joue de la musique contemporaine, et je me suis retrouvé avec des gens qui font de la musique avec électronique en temps réel. Ce temps réel me donne des idées parfois, parfois je reste imperméable, parfois je reste complètement impressionné par quelque chose, dans des cas plus rares. Et donc en tant qu'instrumentiste, il y a déjà une partie qui est touchée. Mais jusqu'à maintenant, au niveau « historique » comme tu dis, il n'y avait pas, à part la musique électroacoustique ou musique concrète, qui étaient en fait mes débuts compositionnels, la possibilité de travailler avec la matière chez moi, la possibilité de construire une verticalité, une profondeur, une tridimensionnalité, un espace, et le modeler jusqu'au moment d'obtenir une pièce, qui soit jouable tout le temps de la même façon. Parce que, heureusement ou malheureusement les CD sont là pour ça : on peut enregistrer ce qu'on veut, on le fait, on le joue, c'était ma pièce, la matière modifiable.

— Donc pour le temps différé, tu avais déjà adopté l'informatique depuis longtemps, pour réaliser un travail sur la matière sonore...

— Exactement. Et la matière sonore fait partie de moi en tant que compositeur. Après ces débuts, je commence à écrire après des choses pour ensemble instrumental, je commence à étudier ce qu'on appelle la composition « pure », même si j'avais déjà étudié le contrepoint ou l'harmonie. Et au fur et à mesure, à chaque fois, quand je vais composer quelque chose, je dois partir d'une idée, d'une idée sonore... À un certain moment tous les paramètres se mettent ensemble. Et donc d'un côté il y a : moi en tant qu'instrumentiste dans un projet pour un concert pour lequel on a travaillé avec des compositeurs, de l'autre côté, pour des pièces où j'ai joué avec des patchs que toi, d'un autre côté encore, tu développais pour ces compositeurs (mais aussi pour moi en tant qu'interprète, parce que c'était moi qui allait mettre les pieds dessus, donc tu étais en train de voir le rapport entre moi comme interprète qui regardais ça et le compositeur qui avait besoin de certaines choses). Et à un moment donné tout devient ensemble ! Alors d'un côté je me dis que je peux travailler la matière, et en plus j'aimerais bien que ça soit joué. J'aimerais bien pouvoir avoir une polyphonie, pour décider quand je lance certaines choses. Donc le temps devient temps réel même s'il est différé, parce que l'interaction entre deux pistes ne se fait pas en même temps à chaque fois.

— Si je comprends bien, tu veux dire qu'en tant que compositeur tu avais déjà un souci de la matière sonore travaillée par informatique, complètement différente de ce qu'on peut avoir avec de l'acoustique « strict », et qu'ensuite, avec ton expérience d'interprète, tu as eu l'occasion de travailler plusieurs pièces en temps réel, et qu'il y a un moment où s'est faite la jonction des deux, entre ton expérience de composition en temps différé avec l'informatique et tes expériences en tant qu'instrumentiste avec pédalier ou avec d'autres dispositifs pour contrôler en temps réel la matière sonore.

— Et aussi : en tant que compositeur, avec des pièces acoustiques, et en tant qu'interprète de pièces acoustiques, et toute la relation que cela a avec l'espace, musical et sonore. Je crois que quand on va a un concert, et qu'on s'assoit, la personne ou le groupe de personnes qui est en face, plus le compositeur qui a composé les pièces qu'on va écouter, ont besoin – pour moi c'est une règle – de composer un espace. On arrive de sa journée, on a eu des embouteillages sur le périphérique, la tête qui explose, mais on a un concert à écouter à vingt heures trente, dans une église, en plein milieu du marais... alors je crois que le compositeur a besoin, a presque un devoir, de se dire : « il faut que je crée un espace pour cette personne, et que je l'invite, par des façons propres à son langage ou à sa façon de faire, à rentrer dans mon espace, et à écouter ce qui se passe ». Je ne fais pas de jugement esthétique par rapport à ça.

— Et là, l'informatique ? Elle permet de façonner un espace ?

— L'informatique a la possibilité de faire une chose qu'on ne peut pas faire avec un son d'instrument : c'est de déplacer les sons physiquement dans l'espace. L'émission du son est faite à un endroit précis, qui dans ce cas-là pourra être loin de l'instrument en lui-même, ou très près.

Et aussi, une autre chose qui répond à ta question, c'est que jusqu'à maintenant je n'ai pas composé de pièce pour moi. Donc je crois que tout ça commence à venir un peu ensemble. D'un côté, j'ai fait mon chemin dans la matière sonore, je me mets à

l'intérieur, j'essaie de composer de façon « symphonique » avec des choses qui ne sont pas des instruments mais avec des sons, j'ai besoin d'un espace, d'une verticalité, d'un rythme, d'une horizontalité, d'établir beaucoup de choses... De l'autre côté, je suis instrumentiste, et quand je joue une pièce j'essaie justement de me mettre dedans, pour sortir toute la musicalité ou toute la musique qui *a priori* est inscrite dans ce texte. Encore d'un autre côté, il y a une démarche compositionnelle acoustique, « purement » acoustique, qui, au fur et à mesure qu'elle évolue, se mélange avec cette matière sonore prétravaillée. Et pourquoi pas arriver à un troisième niveau, dans lequel moi, en tant qu'interprète, je connais mon instrument, toi, tu construis un instrument pour moi en tant que luthier, qui a certaines qualités et certaines possibilités, et qui est d'une certaine façon *la matière sonore transformée en temps réel*, le son de l'instrument transformé en temps réel, le son composé écrit et le son composé à travers la matière sonore...

Donc je crois que c'est simplement une convergence d'éléments qui font que oui : je suis en train de penser à une pièce pour moi et un pédalier. Et aujourd'hui déjà, on est très content d'un premier résultat d'improvisation, qui pourrait marcher déjà comme ça dans n'importe quel concert.

— *C'est aussi ma sensation : j'ai l'impression que tu pourrais jouer tout de suite, juste avec ta clarinette basse, deux micros, un ordinateur et le pédalier !*

— Justement, c'est ça qui devient déjà un concept musical. Ça veut dire qu'on n'a pas un truc dont on ne sait pas comment il fonctionne.

— *C'est complètement opérationnel, d'une part, et d'autre part on est parfaitement familiarisé avec l'outil.*

— Et on n'a même pas eu toutes les semaines un rendez-vous, ou tous les deux jours, non. On a construit quelque chose en regardant ensemble, et tu as ajouté tous les éléments techniques dont tu disposais, parfois pour réfléchir de la façon dont toi tu pensais, et parfois où moi je pensais, et je te disais...

B.2.2 Un apport particulier : la virtuosité artificielle

Karim Barkati : *En fait, on a déjà répondu à la deuxième question...« Toutes tes pièces ne sont pas forcément avec électronique : qu'est-ce que l'informatique change dans ta façon de composer, et surtout qu'est-ce que l'informatique change dans ta musique ? »*

Iván Solano : Je peux répondre encore une chose, peut-être parce que de façon plus synthétique. La partie électronique, ici « concrète » ou « électroacoustique », d'une pièce, change en fonction du fait qu'elle peut être un atout qui permet à l'interprète d'avoir des références, de l'aider du côté pratique à jouer sa pièce, ou bien de faire des choses qu'il n'est pas capable de faire, de façon technique, parce que son instrument ne le permet pas.

— *Mais là c'est du point de vue de l'instrumentiste ces deux choses dont tu parles, mais du point de vue musical ?*

— Pareil ! Si on a un instrumentiste qui n'est pas capable de faire quelque chose que j'aimerais bien faire au niveau musical, parce que son instrument ne sonne pas comme

une voiture, ou que son instrument n'est pas capable de faire un saut de cinq octaves et demi à *noire* = 240 et répété pendant sept mesures de 4/4, alors pourquoi pas utiliser dans la composition quelque chose qui applique et qui donne ce son déjà préenregistré et déjà travaillé, et qui est un son qui est là.

— *Donc ce que tu es en train de dire, c'est que ce que l'informatique change à ta musique, c'est finalement...*

— ... de pouvoir ajouter des éléments que je n'ai pas, pour composer.

— *Donc c'est plus de possibilités ?*

— Oui. C'est comme j'avais besoin d'une couleur *vert-fluo* sans en avoir dans ma palette d'instruments, alors je sais qu'il y a toujours la possibilité d'amener un CD et des haut-parleurs, et mettre le *vert-fluo* ; parce que je sais le construire.

B.2.3 La place des nouvelles technologies

Karim Barkati : *Dans tes différentes pratiques musicales, quelle est la place que les nouvelles technologies occupent et celle qu'elles n'occupent pas ? Tu as déjà répondu en partie, mais quelles sont les frontières à ne pas dépasser que tu t'imposes par rapport à l'ordinateur ?*

Iván Solano : Voilà une frontière pour moi : lorsque l'électronique, temps réel ou temps différé, est un objet *anecdotique*. Un objet anecdotique est pour moi quelque chose qu'on met là parce que ça fait joli, ou parce qu'on m'a dit de faire une pièce avec électronique ; que ma pièce elle n'a pas un rapport *compositionnel* avec l'électronique, mais qu'elle se contente de l'ajouter, comme une chose qui vient se coller, une carapace qui vient se coller au corps de la musique qui a été composée. Donc tout ce qui devient anecdotique, tout ce qui n'est pas vraiment composé, ça sort de mes envies. C'est peut-être pour ça que je n'ai pas encore accédé à l'électronique en temps réel, parce que je crois qu'il faut être très conscient de tous les paramètres, alors il faut les connaître tous, que dès que tu touches à quelque chose, tu sais qu'il y a des choses qui peuvent arriver qui ne sont pas mesurables : ce n'est pas possible pour moi ; je ne crois pas à l'aléatoirité en composition, ni en musique électroacoustique, ni en musique en temps réel. Si je veux quelque chose, il faut que ce soit comme ça.

— *Donc l'aléatoire, pour toi, c'est aussi une frontière ?*

— Ce n'est pas une frontière : ça n'existe pas. Pour moi, l'aléatoire en musique n'existe pas, parce que je ne crois pas qu'il y ait quoi que ce soit d'aléatoire quand on joue une note, en temps réel, différé, ou avec un instrument acoustique. Quand on fait ça, on établit la base d'un langage, on établit une relation acoustique et spatiale, avec l'oreille de la personne qui écoute, et avec la note suivante ; il n'y a plus d'aléatoirité là. Là, la deuxième note doit avoir, même si elle sort de rien, une relation, et surtout c'est cette relation entre les deux notes est la chose qui est importante. C'est une phrase de Kurtág : « ce qui est important, c'est la relation qu'il y a entre les deux notes, le lien ». Et s'il n'y a pas de lien, il n'y a pas de forme ; s'il n'y a pas de forme, il n'y a pas de structure.

— *Est-ce qu'il y a des styles informatiques que tu exclus d'emblée ? Une pièce entièrement sur support ? La musique mixte ? Le temps réel ?*

— Je ne peux rien exclure *a priori*, surtout qu'avec toi, on arrive à une situation où beaucoup de choses peuvent se faire. Sauf une pièce entièrement pour instrument et temps réel que je n'ai pas fait encore, parce qu'il y a du temps différé dans Ifso, donc sauf une pièce « complètement » temps réel, auquel je n'ai pas encore touché parce que je ne m'en sentais pas capable seul.

— *Donc ça n'était que pour des raisons techniques, et pas des raisons esthétiques ?*

— Oui. La question esthétique ne fait pas partie de mes rapports avec l'électronique dans ce sens. Elle en fait partie tout au début quand je te dis que je n'ai pas envie de... c'est comme dans la musique concrète, ou bien quand on écoute une musique et qu'on reconnaît un *plugin* en particulier, un *freeze* du Grm, un vocodeur, et qu'on reconnaît exactement ce que c'est : là on sort de l'écoute musicale, on écoute l'outil et non pas la musique.

B.2.4 Du temps réel en musique

Karim Barkati : *Quelles sont tes positions sur le temps réel et le temps différé en général, ta relation musicale au temps réel et au temps différé, et comment est-ce que tu les penses dans ces projets ? Tu as déjà répondu sur ta relation musicale, sur tes projets aussi, implicitement, mais qu'en est-il sur tes positions « en général », peut-être comme auditeur, ou bien comme compositeur qui regarde ses pairs ?*

Iván Solano : Bon. Le temps réel, comme instrumentiste, comme compositeur, comme auditeur : « ras-le-bol » que tout soit pareil. Quand aujourd'hui tu as dit : « c'est génial d'avoir la sensation qu'il y a d'autres instrumentistes avec toi », là, je me suis dit qu'on est sur la bonne voie. Là, je sens que ce n'est pas « du Max/Msp ».

— *Donc le temps réel qui fait entendre l'outil, et pas la musique ou la composition...*

— Et surtout, qu'est-ce qui sort de tout ça ? Le temps réel qui n'est pas composé ! Parce que qu'est-ce que ça veut dire ? Quand ça sonne l'outil, pour moi ça veut dire qu'on n'a pas composé. Ça veut dire qu'on a pas eu assez de temps, de courage, de technique, ou de capacité pour se mettre à l'intérieur de cet outil et se dire : bon, je vais sortir un truc qui est là-dedans (comme tu as fait avec le granulateur d'Ifso), je vais mélanger deux granulateurs différents, 20 paramètres par granulateur, je vais faire un croisement entre les deux, et je vais voir ce que ça donne ; la personne qui ne pas fait ce travail fait preuve d'une pauvreté de moyens. C'est comme si on composait une pièce pour piano et qu'on jouait la note *do* tout le temps.

— *Ça a été fait : la première variation de Ligeti est sur une seule note, mais c'est différent à chaque fois, et c'est composé...*

— Oui, et le rythme a une importance. L'endroit où tu places telle note par rapport à sa résonance, et comment elle va faire vibrer les harmoniques, avec l'utilisation des

pédales, pour qu'on commence à entendre des choses qui ne sont pas *a priori* sur la note *do*. Voir, de Ligeti aussi, la pièce pour clavecin : il y a un moment où on commence à entendre des harmoniques qui ne sont pas là ; il y a des notes qui sortent – 15e harmonique, 17e harmonique – et qui flottent au-dessus d'un instrument qui *a priori* sonne « klang ! », et tout d'un coup tu commences à entendre « hiii », « huu », les harmoniques. C'est à peu près ça : si tu composes, tu dois composer pour tes outils, tes instruments.

— *Et le temps réel avec l'improvisation ?*

— Ah ça peut être très rigolo ! Mais comme instrumentiste, tu dois connaître tes outils là encore, tu dois connaître tes instruments. Donc justement, on revient à la même chose : pourquoi ça me plaît de faire ce projet Ifso ? Parce que, comme instrumentiste, j'ai mon instrument que je connais – clarinette basse –, et j'ai mon autre instrument, que je commence à connaître pas mal, qu'est Ifso, et je peux les utiliser tous les deux. Comme compositeur, j'ai ma partition que je peux écrire, pour l'instrument et pour Ifso et pour les parties préécrites, donc ça me ramène à la composition « pure », et comme compositeur électroacoustique, je peux travailler la matière avant de la lancer, et la placer plus ou moins là où je veux avec le pédalier ; donc voilà, on a tout ce qu'il faut.

— *Et est-ce que tu trouves en général que le temps réel improvisé s'en sort mieux à tes yeux, ou plutôt à tes oreilles, que le temps réel composé ?*

— Sûrement, parce que la personne qui est en train d'interpréter est en train d'écouter ce qu'elle fait. Elle n'est pas obligée d'aller dans une direction qui, si elle n'est pas précomposée, a un défaut : celui de manquer de direction, de phrasé. Par contre, un improvisateur est en général très attentif à ses phrasés. Et s'il a un instrument sur lequel on applique une électroacoustique en temps réel, il a en général les capacités d'écoute qui lui permettent d'évoluer.

B.2.5 Du temps différé en musique

Karim Barkati : *Et pour le temps différé ? En général, comment tu trouves ça ?*

Iván Solano : Je crois qu'il y a des écoles. Tout ça part, pour moi, de la même chose : tu fais une analyse, un jugement de ce que tu écoutes, et il y a un moment où tu passes à l'action, et là il faut que tu te dises que tu es cohérent avec ce que tu fais. Et parfois, dans la musique en temps différé, électroacoustique, dite parfois « concrète », il y a dans une grande majorité un manque de vision symphonique. Je crois que j'ai écouté très peu de pièces qui m'ont donné une sensation que j'appelle « symphonique », ça veut dire qu'on réfléchit à tous les paramètres : contrepoint, verticalité, horizontalité, profondeur, utilisation des fréquences, spatialisation, dynamique, réverbération, position de l'attaque. Tout cela forme un ensemble de paramètres qu'on peut contrôler, qu'on doit contrôler.

— *Donc quand tu dis « symphonique », il ne s'agit pas de la quantité de sons utilisés ?*

— Non. Je crois qu'avec un seul son on peut être « symphonique », dans le sens où il renferme en lui-même beaucoup de paramètres. Une note de piano a un spectre sonore très particulier, qui peut être transformé de son début jusqu'à sa fin, et on peut agir sur

ses harmoniques, sur son attaque, sur sa réverbération, sur sa longueur, sur sa vitesse, sa dynamique, son placement dans l'espace – plus près, plus loin, plus à gauche, plus à droite, plus haut ou plus bas... on peut agir sur beaucoup, beaucoup de choses. Tout cela n'est souvent pas pris en compte. Bien sûr, il y a Pierre Henry, Horacio Vaggione, Pierre Schaeffer, Bernard Parmegiani, George Graham (qui arrive dans sa musique dans un lieu très proche de la musique électroacoustique)... Tu connais Horacio Vaggione : pour moi, sa façon de travailler est une façon *symphonique*.

— *Il a une approche clairement multidimensionnelle.*

— Pour moi, c'est fondamental. Ça prend plus de temps, mais c'est fondamental.

— *Est-ce qu'on pourrait résumer en disant que les pièces en temps différé ne sont en général pas tout à fait abouties ? Qu'elles n'ont souvent pas creusé suffisamment de paramètres, sauf dans le cas de quelques compositeurs ?*

— Là, justement, il y a un problème : c'est le « problème » de la musique concrète. Je veux bien croire qu'à un moment donné, à un moment historique, on a eu besoin de dire que n'importe quel son peut devenir un objet musical, mais aujourd'hui on est dans une époque où même un tableau hyper-réaliste fait à la façon de Rubens ne serait pas tout à fait cohérent maintenant, ni avec son entourage, ni avec son époque. Je ne veux pas dire qu'on ne peut pas composer de façon prédéterminée, mais je crois que si aujourd'hui on fait du Mozart, j'ai peur que Mozart ait déjà fait mieux que nous ; parce qu'on va analyser un système, et essayer de le reproduire, et ça je doute que ça soit dans la cohérence de la découverte d'un langage personnel. Après on a du Berio, l'utilisation de passages musicaux qui n'étaient pas à lui, mais, comme objets, qui devenaient partie de sa musique à lui.

— *C'est complètement recomposé, ce qui n'a rien à voir avec de l'imitation.*

— Maderna, pareil ; c'est recomposé.

— *C'est souvent du collage camouflé, non reconnaissable... et lorsque c'est reconnaissable, c'est décontextualisé et recontextualisé.*

— Voilà. Donc qu'est-ce qui arrive pour moi dans la musique différée ? Il arrive que quelqu'un dise qu'une voix de femme qui dit un texte est déjà de la musique. C'est contestable... en étant conscient aussi qu'aujourd'hui les livres sonores existent. Donc si je veux écouter un texte, je le fais réciter par quelqu'un, et je le sort en tant que *livre audio*. Si je veux utiliser ce texte parce qu'il a des paramètres musicaux, alors il va falloir que je décontextualise et que je recontextualise dans ma musique ce texte.

— *Surtout que le texte a le pouvoir d'accaparer l'attention par la signification, et d'éloigner ainsi de la musique.*

— *A priori* oui, mais c'est comme la musique pour voix : en général, quel est le grand handicap de la musique vocale ? C'est que le texte est traité en tant que texte. Pourquoi l'opéra n'évolue pas ? Pourquoi ne deviendrait-il pas du théâtre musical ? Parce qu'à l'époque de l'opéra, le texte était le fondement de la production musicale ; donc le texte était représenté en tant que ligne mélodique surtout, harmonisé par un orchestre, mélodie que quelqu'un pouvait plus ou moins siffler quand il arrivait à la maison, ou

qui donnait une sensation représentative de ce texte en lui-même. Quand on passe à la musique vocale, si on essaie d'extraire d'un texte une idée, comme je t'ai dit par rapport à Kavafis ou par rapport à Bashô, si on veut essayer d'extraire, je crois qu'il va falloir aller au-delà des paroles, au-delà des mots.

C'est à peu près pareil pour la musique électroacoustique... Ça n'est pas parce qu'on met ensemble une chèvre, un téléphone, trois chanteurs, deux personnes qui soupirent, et un piano qui fait des *clusters*, qu'on est en train de faire de la musique ! Donc il faut aussi composer ; il faut aussi partir dans tous les paramètres que tout ça nous donne.

— *Donc pour toi il y a un peu trop de pièces qui ne sont pas assez composées ?*

— Il y en a beaucoup...

B.2.6 Limites de l'expressivité de l'ordinateur

Karim Barkati : *Pour toi, quelles seraient les possibilités et les limites de l'expressivité de l'ordinateur ?*

Iván Solano : Les limites, je n'en connais pas... parce qu'on ne connaît déjà pas les possibilités. On pourrait dire que l'ordinateur n'a pas de limites dans ce domaine...

— *C'est osé comme position ! Mais dans tes réponses précédentes j'entends aussi que finalement ça dépend aussi du compositeur... La limite de l'expressivité de l'ordinateur serait la limite de l'expressivité du compositeur ?*

— Quelle est la limite d'une clarinette ? La hauteur n'est pas un paramètre expressif, ni le registre...

— *Bon, je suis clarinettiste, mais j'aurais tendance à dire que la clarinette est un instrument très expressif, qui a beaucoup de possibilités ; j'aurais plus de facilité à répondre à cette question sur les limites de l'expressivité de la flûte à bec par exemple !*

— Pourquoi ?

— *J'ai le sentiment qu'elle a moins de possibilités sur la tessiture, sur la dynamique aussi, et je crois que c'est ça qui me gène le plus... Pour moi, c'est comme pour un bol tibétain : ça exprime très bien certaines choses, mais il y a des limites à son expressivité.*

(Iván sort une flûte à bec et se met à en jouer)

— *D'accord, jouée comme ça... Mais du Wagner à la flûte à bec... je ne sais pas...*

— Le problème quand tu parles de Wagner, c'est que tu penses à une complexité instrumentale, ce n'est pas de l'ordre de l'expressivité.

— *Pour exprimer une sorte de « dramatisme », de lourdeur, j'ai du mal à penser à une flûte à bec.*

— S'il s'agit d'une flûte contrebasse, oui : elle peut être très lourde, et très connotée, et très expressive.

— *Même une flûte contrebasse, j'ai du mal à l'entendre « lourde » ; pour moi elle va garder une forme de légèreté, dans l'absence de vibrato, dans ce son un peu sinusoïdal aussi… Peut-être qu'elle manque de* pathos*, ce genre de critère « romantisant » qui me paraît plus facile à obtenir à la clarinette.*

— Justement, en comparant les instruments, il y a très peu de répertoire pour la flûte à bec au XX^e^ siècle par rapport à la clarinette ; on pourrait en déduire qu'on a épuisé les possibilités expressives de cet instrument.

— *Si on a épuisé les possibilités expressives de cet instrument-là et pas d'un autre, c'est peut-être parce qu'il n'avait pas les mêmes, et donc qu'il a des limites.*

— Justement, pour moi un ordinateur n'a pas ces limites tant qu'il n'a pas été exploré et exploité complètement. Même la clarinette ! On pourrait dire : « 1791, Mozart compose son concerto pour clarinette, toutes les limites expressives de l'instrument sont déjà atteintes à ce moment-là ».

— *Ça me semble évident que non…*

— Évident pour qui ?

— *Tous les modes de jeu développés au* XX^e^ *siècle ont fait rebondir ses possibilités expressives.*

— Justement : de l'instrument que l'on connaît aujourd'hui – la clarinette –, à partir du début du XX^e^ siècle, ses possibilités commencent à se développer ; on entre dans la musique « contemporaine » (on va dire ça comme ça), on entre dans le « surréalisme » musical, on commence à explorer l'instrument d'une autre façon. Beaucoup d'instrumentistes ont d'ailleurs du mal, parce que cet instrument, selon leurs préjugés, ne doit pas faire ce que certains compositeurs demandent, voir Lachenmann. Un instrument en évolution, d'accord la clarinette est un instrument très jeune, plus jeune encore le saxophone, moins jeune le violon, et pourtant même pour le violon, au XX^e^ siècle, on trouve un développement de l'instrument : alors je ne crois pas qu'on soit près de trouver toutes les capacités internes d'un ordinateur ni toutes les potentialités de son utilisation au niveau musical ! Donc je crois que l'expressivité de l'ordinateur n'a pas de limites, et que ses possibilités dépendent toujours de nos capacités à comprendre et à musicaliser certains paramètres.

— *Une des limites est peut-être apparue avec le vocodeur qui était finalement trop lourd pour ta machine. La puissance de calcul est une première façon de parler de limite concernant l'ordinateur, mais ce n'est pas vraiment une limite « esthétique ».*

— Non. Ça n'est pas une limite, parce que ça dépend d'une machine particulière, la mienne, qui n'est pas la tienne, ni une autre machine qui pourrait être plus puissante. La machine n'est qu'un objet… c'est un peu comme si on jouait dans une salle sans aucune réverbération, une salle anéchoïque, et qu'on mettait un public à 20 mètres : on risque de ne pas entendre l'œuvre. Pour moi, la carcasse physique de l'instrument, c'est vrai qu'elle conditionne, mais ça n'est pas l'instrument en soi, ce n'est pas la limite.

B.3 La collaboration compositeur / RIM

B.3.1 Un potentiel enrichissant

Karim Barkati : *Pourquoi avoir entrepris une telle collaboration ?*

Iván Solano : Je ne sais pas, pourquoi pas ?

— *J'ai l'impression que tu es quelqu'un d'assez actif et d'assez pris aussi, j'ai le sentiment que tu ne t'engages pas à la légère, et que tu sélectionnes tes projets...*

— Justement. Il y a trois sortes de projets : les projets que tu fais pour l'argent, dans lesquels tu ne joues pas forcément ce que tu veux, parce que tu dois manger (les projets que j'appelle « gastronomiques »), dans lesquels tu t'embêtes à jouer un truc, mais tu le fais avec beaucoup de professionnalisme au moins, pour faire sonner une pièce que tu n'as pas forcément envie de jouer et qui ne rentre pas dans tes critères d'appréciation musicale.

Il y a des projets que tu as envie d'envisager, parce qu'ils sont tellement bien que pouvoir être là et le faire est déjà une sorte de récompense. Donc imagine-toi – je vais revenir au cliché du clarinettiste – jouer le concerto de Mozart avec le *BBC Symphony Orchestra*... Si quelqu'un me dit que je pourrai faire ça dans deux ans mais qu'on ne va pas me payer un sou, je le ferai quand même. Pas pour les choses que ça va m'apprendre (même s'il y en aura), mais pour ce qu'un tel projet va me procurer au niveau sentimental tellement de choses – parce que je suis clarinettiste, parce que l'histoire de l'instrument, parce que la composition, parce que Mozart, parce que le *BBC Symphony Orchestra*, et parce qu'il y a un concert – que je n'ai pas à me soucier de combien on va me payer.

Et après, il y a des projets que tu sens, qui ont vocation à te toucher en tant qu'individu, à te faire évoluer, parce qu'ils vont te mettre aux limites de tes possibilités et qui vont pousser ces limites. Ce sont des projets dans lesquels il y aura une grande partie d'humanité : tu vas rencontrer des gens avec lesquels tu vas quand même parcourir un bout de chemin de cet Ithaque qu'est la vie, tu vas apprendre ou faire apprendre, tu vas partager ta façon de penser, tu vas évoluer... Par exemple, travailler sur Ifso avec toi est un projet que je fais parce que je sens à l'intérieur de moi que ça amène quelque part.

Et enfin, tu as les projets décevants ! Ce sont les projets dont tu sens au bout de deux semaines que ça va nulle part, que personne n'a aucune idée de ce qu'ils sont en train de faire, et que toi tu n'as pas envie d'être là.

— *Donc ça fait quatre types de projets !*

— Voilà, j'avais dit trois...

B.3.2 Intégration de la collaboration dans la composition

Karim Barkati : *La composition est souvent un travail solitaire : comment le travail en collaboration s'intègre-t-il dans l'acte compositionnel pour toi ?*

Iván Solano : Tu peux répondre à ça peut-être ?

— *Moi ?*

— Oui ! Penses à moi en tant que quelqu'un qui va composer une pièce avec ton instrument…

— *Parfois, je me demande si ça n'est pas déstabilisant de travailler sur un instrument qui est en train d'être fabriqué, parce qu'on ne peut pas savoir exactement comment il va évoluer… et puis, finalement, tu n'as pas beaucoup joué avec Ifso, puisque je ne l'ai fini que récemment. Il y a une dimension de « jeunesse » qui pourrait être gênante…*

— Ifso est un instrument qui est déjà rempli de paramètres, des paramètres que nous avons décidés ensemble : certains que tu as décidé de ton côté, d'autres que j'ai décidé de mon côté, mais qui construisent un instrument à part entière, c'est vraiment comme une clarinette… Pourquoi ça pourrait être frustrant ?

— *L'autre chose, c'est que si tu écris ta pièce seul, alors tout ne dépend que de toi : est-ce que là il n'y a pas des choses qui ne dépendent pas de toi ?*

— Non, ça n'est pas vrai. Là, on revient aux questions précédentes, au rapport du compositeur à la musique en temps réel : il faut tout composer.

— *Donc tu as la sensation dans cette collaboration que tout dépend de toi ?*

— Non : j'ai la sensation dans cette collaboration que je vais pouvoir composer pour un instrument qui a été fait – je vais dire un mot un peu fort – « parfait », en rapport aux idées qu'on a, par rapport à ce projet. Je sais que j'ai un registre de 3 octaves et demi pour la clarinette basse, une polyphonie de 8 voies pour Ifso, avec des variables pour la vitesse, pour la dynamique, pour la panoramisation, pour la granulation, etc., comme les paramètres que je pourrais trouver sur une clarinette.

— *C'est vrai qu'Ifso est assez défini, « ferme », on en avait longuement parlé avant la programmation ; éventuellement en corrigeant des choses au passage…*

— Et justement c'est pour ça que je voulais que ce soit ferme, défini, et précis. Parce que si je ne peux pas me fier à ton instrument, alors je ne compose pas pour lui. Que fait un compositeur quand il va composer pour un instrument acoustique ? Avant tout : étudier l'instrument. Pas tout le monde mais certaines personnes vont jusqu'à louer l'instrument, Alberto Posadas fait ça ; s'il va composer une pièce pour violon, il loue un violon… Il a composé la pièce pour clarinette avec moi, et il avait une clarinette chez lui. Je crois que tu dois avoir les capacités pour connaître l'instrument pour lequel tu vas composer à travers les livres, tu l'étudies, tu apprends que l'harmonique *do* dièse sur l'alto c'est la 5^{e} position et que normalement un altiste va avoir peur de la jouer mais qu'il peut le faire, tu le notes, tu connais l'instrument, tu peux le mettre dans la partition, tu sais que cette note-là peut être faite. Si tu ne démarres pas de là, le pourcentage d'erreurs est quand même énorme.

Donc, *a priori*, ça dépend de toi, mais comme *luthier* de cet instrument, qui a un paramétrage très précis ; ça dépendra de moi en tant que compositeur. Heureusement, je

peux te demander : « au lieu de me faire une clarinette basse en *si* bémol qui descende jusqu'au *mi* bémol, est-ce que ça serait possible qu'elle descende jusqu'à l'*ut* ? » Et toi, tu peux te permettre de le faire.

— *Oui, il y aura bientôt un égaliseur !*

— Tu vois ce que je veux dire… on a entendu aujourd'hui que parfois certaines notes graves ne sont pas prises en compte par le transpositeur, peut-être parce qu'il est à sa limite, et je peux te demander de penser à ça, de me dire si on peut faire quelque chose pour ça. Ça ne dépend pas de moi en tant que compositeur, ça dépend de toi en tant que constructeur.

— *Sur la transposition, ça n'est probablement pas possible.*

— D'accord, donc tu me dis que cet instrument fait sauter la transposition d'une octave à partir d'une certaine note : je le sais. Quel souci ? Je connais alors les capacités, les possibilités de cet instrument, et ses limites.

B.3.3 Les compositeurs et la programmation

Karim Barkati : *D'après toi, est-ce que les compositeurs devraient ou devront maîtriser la programmation au point de réaliser eux-mêmes leurs propres logiciels ? Tu parlais de louer l'instrument tout à l'heure…*

Iván Solano : Oui, mais dans ce cas-là, l'instrument était déjà fabriqué. Je crois que faire un logiciel veut dire construire un instrument : or je ne suis pas luthier. Je peux monter et démonter ma clarinette, en tant qu'instrumentiste je connais ses capacités, mais un compositeur ne doit pas construire lui-même une clarinette. Par contre, je peux louer une clarinette pour voir comment elle réagit ; de même, je peux te demander de me prêter ton patch pour travailler dessus et voir les paramètres que je peux modifier. S'il y a des instrumentistes qui sont des luthiers, tant mieux, s'il y a des compositeurs qui sont des luthiers, encore mieux, mais normalement ça n'arrive pas ensemble : c'est une question de temps dans la vie.

— *Je crois qu'il y a plus souvent des luthiers qui sont instrumentistes…*

— Oui, mais pas compositeurs.

— *Donc pour toi la réponse est plutôt non ?*

— Plutôt non dans le concept de *compositeur*.

— *Ça veut dire que tu penses que l'interprète devrait connaître la programmation ?*

— Ça pourrait être intéressant à un moment donné, oui ; mais ça pourrait être intéressant à un moment donné aussi pour un compositeur, c'est une question de choix.

— *Il y a des gens qui affirment que le compositeur doit savoir programmer, avec l'argument que sinon l'électronique n'est pas composé.*

— Je crois que le compositeur doit connaître l'instrument, mais ça ne veut pas dire qu'il doit être capable de le créer lui-même, ça veut juste dire qu'il doit connaître tous les paramètres de cet instrument. Moi, en tant que compositeur, quand je me mets devant un patch Max/Msp, au bout d'une demi-heure... la première demi-heure est très agréable, mais franchement... je vais faire un granulateur, je vais prendre un patch qui est déjà préfabriqué en plus, parce que je ne vais pas construire mes outils, ou même si j'arrive à construire mes outils je commence avec un délai par exemple... un délai, je sais comment c'est : j'ai certaines notions de Max/Msp, c'est pour ça que je l'ai à la maison, donc je commence avec mon délai, et au bout d'une demi-heure je suis bloqué dans un truc par où je ne sais plus où aller ! Parce que je me dis « ah c'est joli », tigidigidi-tigidigidi-tigidigidi... une demi-heure de réinjection... Est-ce qu'on a besoin de faire nos propres crayons pour écrire sur un papier ? Est-ce qu'on a besoin de savoir presser notre papier pour écrire ? Là, on est quand même à la limite d'un endroit un peu...

— *Ce qui semble clair, c'est que ne pas savoir programmer ça n'empêche pas de composer.*

— Heureusement !

— *Rien n'empêche d'apprendre à programmer, sauf peut-être un facteur temps.*

— Un facteur temps, et aussi un facteur envie, et aussi un facteur... comment dire... c'est un peu l'autre côté. Je crois qu'un compositeur, quand Alberto Posadas loue une clarinette, il n'a pas forcément envie de savoir comment construire une clarinette.

— *Ni d'en jouer lui-même... il n'a pas forcément envie de devenir clarinettiste, alors que là on parle de devenir programmeur...*

— Oui, il n'a pas envie de devenir clarinettiste, ça c'est sûr ; mais il a envie de savoir quelles sont les possibilités, et il a envie de pouvoir « bidouiller » dans l'instrument, pour trouver des choses que quelqu'un d'autre n'a pas trouvé. Donc ça veut dire qu'une certaine connaissance de l'outil déjà fabriqué est nécessaire, mais jusqu'où on amène la connaissance de l'outil préfabriqué ? Certains diraient qu'il faut aller jusqu'à la programmation.

— *Oui, pour certains la programmation c'est un peu comme l'instrumentation, l'harmonie : juste une dimension de plus de la composition qui doit être intégrée.*

— Non. C'est justement pour ça, là je crois qu'on tombe dans les limites « illimitées » de l'ordinateur : vas-y, mets-toi dans Max/Msp ! Vas-y là-dedans, et essaie de te focaliser et de te centrer tellement pour être capable de sortir de là une idée musicale, zéro ! Une idée musicale, quelque chose de cohérent, avec une pièce, et sans le papier, sans le crayon, sans l'instrument : donc construis ton papier, ton crayon, ton instrument, et après construis ton langage sur cet instrument ! Construis ton langage harmonique, ton langage rythmique, ta spatialisation, ton spectre sonore, tes sons, ta texture, ta dynamique, tes espaces, ta réverbération...

— *Pour résumer, l'ordinateur intègre tous les paramètres musicaux ?*

— La seule chose que l'ordinateur n'intègre pas (en tant qu'instrument), c'est le son acoustique, qui n'est pas autre chose que des préparamétrages du son sinusoïdal. Donc on

va avoir de façon électronique un instrument acoustique ; qu'est qu'un instrument acoustique ? Un ajout de paramètres sur un son fondamental ; qu'est-ce qu'un son fondamental ? Une vibration ; qu'est-ce qu'une vibration ? Un rythme ; qu'est-ce qu'un rythme ? Une pulsation ; qu'est-ce qu'une pulsation ? La même chose que ce que produit un son électronique.

B.3.4 Réalisateur en informatique musicale

Karim Barkati : *Est-ce que tu penses que « réalisateur en informatique musicale », ou* Rim*, est un métier émergent ou pas ?*

Iván Solano : C'est quoi un Rim ?

— *« Rim » ou « réalisateur en informatique musicale » est le nouveau nom (l'ancien nom étant « assistant musical ») donné à l'*Ircam *et dans d'autres institutions.*

— Je veux bien que tu me définisses ça.

— *L'idée, c'est que certaines institutions proposent à un compositeur de travailler avec une personne qui connaît le matériel, les possibilités de diffusion, et les logiciels de l'institution, parce qu'en général ce sont des compositeurs qui viennent en résidence, pour une période limitée dans le temps. La nouvelle dénomination, « réalisateur en informatique musicale », est plus complète que « assistant musical », mais sans doute aussi plus juste. Par exemple, réaliser Ifso c'est de l'informatique musicale, et il a fallu effectivement que je le réalise : il y avait une sorte de cahier des charges qu'on avait établi ensemble, et c'est une réalisation quasi-palpable – on peut la télécharger, l'installer, la faire tourner, c'est quelque chose qui existe. Voilà, quelle est ton opinion sur l'avenir d'un tel « métier » ?*

— J'espère qu'un jour ça s'appellera « créateur en informatique musicale », même si « réalisateur » c'est déjà très bien, c'est proche... Je te vois comme un luthier.

Le compositeur, celui qui va démarrer le projet et écrire une pièce pour un instrumentiste, assisté par un réalisateur en informatique musicale, et qui demandera ou pas un instrument nouveau ou un instrument déjà construit, parce qu'au moment où il se décide à aller dans une institution pour faire une pièce, de faire de la composition « mixte » – pour instrument et électronique en temps réel – ce compositeur ne sait pas s'il aura besoin de construire un patch ou pas, il ne sait pas s'il devra utiliser des patchs qui sont déjà conçus et qu'il pourrait très bien déformer ou pas. Donc le réalisateur en informatique musicale va lui dire dans le meilleur des cas « voilà ce que tu veux, on a ça », dans le pire des cas il va faire la tête « tu veux ça et ça et ça ? », il va rentrer chez lui et passer deux semaines à composer son patch – je dis bien composer –, à construire son instrument, pour le donner au compositeur, qui ne connaîtra pas cet instrument de façon profonde, mais qui arrivera plus ou moins à faire quelque chose avec ce que le Rim aura fait...

— *J'assistais à une conférence à Lyon, et un compositeur, Gilles Grand, déclarait : « Je trouve que le plus difficile c'est de trouver un développeur qui me comprenne ; qu'il puisse y avoir une conversation ». Je crois que c'est là qu'il se passe quelque chose de particulier... Je reviens maintenant à ma question : est-ce que* Rim *est un métier émergent ou pas ?*

— Je reviens à l'histoire de la musique : toutes les pièces qui ont été composées pour des instruments ont été composées pour quelqu'un ou pour un instrument particulier.

— *Sauf les cas comme le* Clavier bien tempéré *de Bach, non ?*

— Oui, mais les œuvres de Mozart pour clarinette par exemple ont bénéficié de la collaboration avec son ami Stadler, qui possédait un cor de basset qui descendait plus grave que l'instrument habituel… L'instrumentiste passe par un luthier, mais il possède et il maîtrise son instrument, et toi tu possèdes ton instrument, l'ordinateur, et tu le maîtrises aussi. Tu construis quelque chose qui est dans les paramètres qu'on décide en rapport à ton instrument. Je peux te demander, parce que je vois de l'extérieur, de me sortir un instrument qui pourra me faire une octave plus grave… et on parle des couleurs de l'interface, du « vernis », on parle de la couleur du son, du son d'un granulateur : toi, tu creuses la question tout seul parce que j'ai dit le mot « granulateur », chapeau ! Je crois que tu as compris ce que je voulais dire, parce que quand tu écoutes le résultat tu es content avec, et moi aussi.

Est-ce que je crois qu'un métier est en expansion ? Je crois que ce métier est nécessaire, mais il passe par une relation toujours humaine, entre les gens. C'est comme composer pour un copain qui joue de la clarinette, ou composer pour un copain qui joue de l'accordéon, ou t'approcher du clavecin parce que tu as un ami qui joue du clavecin, même si tu n'as jamais pensé de ta vie que tu allais composer pour cet instrument-là. On va faire quelque chose avec l'électroacoustique ou l'électronique en temps réel ? Sûrement que je ne me serais jamais mis à le faire si on ne s'y était pas mis ensemble toi et moi, pour un projet pareil.

En revenant sur la question précédente, sur la croyance d'une soi-disant nécessité absolue pour les compositeurs de rentrer au milieu non pas d'un langage, mais de la première cellule linguistique d'un programme… jusqu'au développement d'un patch… et jusqu'à la possibilité que cela devienne quelque chose d'intéressant pour une pièce : je suis désolé, il y a des luthiers pour ça ! Il y a des gens qui savent le faire !

Donc, à propos de la question des Rim, j'espère que ce métier deviendra quelque chose qui progresse dans le temps.

Annexe C

Code source du programme d'aide au classement relatif

C.1 Page principale

L'interface d'aide au classement relatif se présente d'abord à l'utilisateur sous la forme d'une page HTML contenant aussi du javascript, nommée `classement.html`. Cette page comporte plusieurs fonctions programmées dans le langage javascript afin de la rendre dynamique, en particulier pour réaliser les calculs et les actualisations nécessaires au classement, en décrivant les actions des différents boutons. La partie javascript se trouve dans l'entête du document HTML, tandis que le corps du document décrit un formulaire HTML qui appelle le fichier `malignegeneree.php` en dernière instance, et s'occupe de mettre en page les différents éléments du formulaire par l'intermédiaire d'un tableau HTML. La fin du corps du document décrit une balise `<iframe>` nommée « hublot ».

Code C.1 – *Page HTML de l'interface d'aide au classement relatif*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Classement relatif</title>


<script type="text/javascript">

function insere(f,d){
  n1=f.liste1.selectedIndex
  n2=f.liste2.selectedIndex
  l1=f.liste1.length
  l2=f.liste2.length

  if (n1==-1) {alert ("Sélectionnez un logiciel en liste 1 !"); return}
  if (n2==-1 && l2>0) {alert ("Sélectionnez une place en liste 2 !"); return}
  e1 = f.liste1.options[n1].text
  if (l2==0){ // Traitement du premier élément placé en liste2.
    o=new Option(e1)
    f.liste2.options[0]=o
    f.liste2.selectedIndex=0
  }else{
    // D'abord décaler la partie inférieure de liste2 vers le bas.
    for (i=l2; i>n2+d; i--){
      t=f.liste2.options[i-1].text
      o=new Option(t)
```

```
30          f. liste2 . options [ i ]=o
31        }
32        // Ensuite insérer le nouvel élément à la bonne place (n2+d).
33        o=new Option(e1)
34        f. liste2 . options [n2+d]=o
35        f. liste2 . selectedIndex =n2+d
36      }
37      // Retirer l'élément de liste1 et redimensionner les deux listes .
38      f. liste1 . options [n1]=null
39      if (f. liste1 . length>1) f. liste1 . size = f. liste1 . length
40      if (f. liste2 . length>1) f. liste2 . size = f. liste2 . length
41
42      if (f. liste1 . length == 1){
43        f. liste1 . options [0]. selected = true;
44        montre('dessus') ;
45        montre('dessous') ;
46      }else{
47      montre(' retirer ') ;
48      cache('dessus') ;
49      cache('dessous') ;}
50
51       rafraichissement (f) ;
52    }
53
54    function ramene(f){
55      l1 = f. liste1 . length ;
56      l2 = f. liste2 . length ;
57      si1 = f. liste1 . selectedIndex ;
58      si2 = f. liste2 . selectedIndex ;
59
60      f. liste1 . options [l1] = new Option(f. liste2 . options [ si2 ]. text ); // Ajoute en fin de l1 .
61      f. liste1 . selectedIndex = l1 // Selectionne ce dernier element.
62      f. liste2 . options [ si2 ] = null
63
64      // Redimensionne l' affichage des listes .
65      if (l1 > 1) f. liste1 . size = l1+1;
66      if (l2 > 1) f. liste2 . size = l2;
67
68      if (l2 == 1){ // La liste2 va devenir vide .
69        cache(' retirer ') ; cache('dessus') ; cache('dessous') ;
70        f. liste1 . options [ si1+1]. selected = false ;
71      }else{
72        montre('dessus') ; montre('dessous') ;
73        if ( si2 == (l2-1))
74          f. liste2 . options [ si2 -1]. selected = true; // Reselection du precedent.
75        else
76          f. liste2 . options [ si2 ]. selected = true; // Reselection du suivant .
77      }
78       rafraichissement (f) ;
79    }
80
81    function charge(f){
82      entrees = new Array();
83      Expression = new RegExp("\r","g") // Retire ' carriage return ' pour Windows.
84      entrees = f. textareaL1 . value . replace ( Expression ,"") ;
85      entrees = entrees. split ("\n");
86
87      f. liste1 . length = 0;
88      f. liste2 . length = 0;
89
90      reg = /^\(.+\).*$/; // Détecte les chaines entre parenthèses .
91      par = /\(|\)/g; // Détecte les parenthèses .
92      flag = 0;
93      t = "";
94
95      for (i=0,j=0; i<entrees.length ; i++) {
96        t = entrees[ i ];
97        if (t != "") {
98          if (! reg . test (t)) {
99            f. liste1 . options [j++] = new Option(t);
100         }else{
101           // alert ("parentheses ! " + t);
102           if ( flag==0) {
103             f. label1 . value = t. replace (par ,"");
```

```
104             flag = 1;
105             f.label2.value = "";
106           } else   if(flag==1) {
107             f.label2.value = t.replace(par,"");
108             flag=2;
109           }
110         }
111       }
112     }
113     // Affichage entier et sans ascenceur individuel.
114     if (f.liste1.length>1) f.liste1.size = f.liste1.length;
115     rafraichissement(f);
116 }
117
118 function sel1(f){ // Selection d'un item dans liste1.
119     if(f.liste1.length == 1 || f.liste2.length == 0)
120       insere(f,0); // Placement immediat.
121     else {
122       montre('dessus'); montre('dessous'); }
123 }
124
125 function sel2(f){ // Selection d'un item dans liste2.
126     montre('retirer');
127     if (f.liste1.selectedIndex > -1) {
128       montre('dessus'); montre('dessous'); }
129 }
130
131 function imite(f){ // Miroir de liste2 dans le textarea de droite.
132     f.textareaL2.value = "";
133
134     for (i=0; i<f.liste2.length ; i++) {
135       t = f.liste2.options[i].text;
136       f.textareaL2.value += t + "\n";
137     }
138     f.submit(); // Redessine le graphique SVG.
139 }
140
141 function montre(btn) {document.fl[btn].disabled = false;}
142 function cache(btn) {document.fl[btn].disabled = true;}
143
144 function rafraichissement(f) {
145     imite(f);
146     document.getElementById("HL1").innerHTML = "Liste1 (" + f.liste1.length + ")";
147     document.getElementById("HL2").innerHTML = "Liste2 (" + f.liste2.length + ")";
148 }
149
150 </script>
151 </head>
152
153
154 <body>
155 <hr>
156 <h1 align="center">Interface d'aide au classement relatif </h1>
157 <hr>
158
159 <form action="svg/malignegeneree.php" name=fl method=post target="hublot">
160 <input type=hidden name=liste>
161
162 <table align=center border=0 cellpadding=10>
163 <tr>
164   <td></td>
165   <td><h2 id="HL1">Liste 1</h2></td>
166   <td><div align="center"></div></td>
167   <td><h2 id="HL2">Liste 2</h2></td>
168     <td> </td>
169     <td></td>
170 </tr>
171
172 <tr>
173 <td align=center>
174     <textarea name="textareaL1" rows="12">
175 &lt;Entrez vos donnees ici, separees par un retour a la ligne&gt;
176
177 (un premier element entre parentheses decrit le label1, un deuxieme le label2)
```

```
178   </textarea>
179   <p>
180   <input name="btn_charger" type="button" value="Charger vers L1" onclick="charge(this.form)">
181   </p>
182 </td>
183
184 <td bgcolor="#CCCCCC" >
185   <select name=liste1 size="2" onChange="sel1(this.form)">
186   </select>
187 </td>
188
189 <td align=center>
190   <p align="center">
191   <input class="b" type="button" name="dessus" value="Classer avant"
192   onclick="insere( this .form,0)" disabled  align="middle">
193   </p>
194   <p align="center">
195   <input class="b" type="button" name="dessous" value="Classer apr&egrave;s"
196   onclick="insere( this .form,1)" disabled  align="middle">
197     </p>
198 </td>
199
200 <td bgcolor="#CCCCCC" >
201   <select name=liste2 size="2" onChange="sel2(this.form)">
202   </select>
203 </td>
204
205 <td>
206   <input class="b" type="button" name="retirer" value="Retirer"
207   onclick="ramene(this.form)" disabled>
208 </td>
209
210 <td >
211   <input type="text" name="label1" id="label1" value="(label1)">
212   <br>
213   <textarea name="textareaL2" rows="12">
214 &lt; Sortie  du classement de la  liste  2&gt;
215   </textarea>
216   <br>
217   <input type="text" name="label2" id="label2" value="(label2)">
218   <p align="center">
219   <input type=submit value="G&eacute;n&eacute;rer le graphique SVG">
220   </p>
221 </td>
222 </tr>
223 </table>
224 </form>
225
226
227 <div align="center">
228 <iframe src="svg/vide.html" name="hublot" width="800" height="250" type="image/svg+xml"
229 align="middle" frameborder="1" scrolling="no"> <!-- marginheight="10" marginwidth="10" -->
230 <p>Votre navigateur ne peut malheureusement pas afficher de 'iframe'</p>
231 </iframe>
232 </div>
233
234 <hr>
235 <div align="right">
236   <pre>sr &amp; kb 2007 </pre>
237 </div>
238 <hr>
239
240 </body>
241 </html>
```

C.2 Précaution Svg

La balise `<iframe>` nommée « hublot » insère par défaut le fichier `vide.html` dans cette zone réservée de la page web. Tous les navigateurs n'intégrant pas automatiquement

le standard Svg [1] à ce jour, le fichier `vide.html` préconise à l'utilisateur d'installer dans son navigateur, si nécessaire, un plug-in [2] Svg, téléchargeable depuis l'adresse http://www.adobe.com/svg/viewer/install/main.html.

Code C.2 – *Cadre HTML d'avertissement pour SVG*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
</head>
<body>
<p align="center">
Si n&eacute;cessaire, vous pouvez t&eacute;l&eacute;charger le plug-in SVG (pour voir le graphique)
  &agrave; l'adresse suivante :</p>
<p align="center"><a href="http://www.adobe.com/svg/viewer/install/main.html" target="_blank">http://www.adobe.
    com/svg/viewer/install/main.html</a></p>
</body>
</html>
```

Remarquons que sur internet, l'image n'est pas encore arrivée à maturité, comparativement au texte par exemple ; la nécessité d'installer des plugiciels (Flash, Svg, etc.) et la redondance extrême des mises en page sur le web en témoignent. Ce retard peut sembler doublemen paradoxal dans nos sociétés où l'image a gagné une place parfois qualifiée d'hégémonique, ainsi que vis-à-vis de la maturité avérée du texte sur internet. Au demeurant, le web est toujours muet, exception faite de quelques rares sites qui sonorisent leurs pages avec plus ou moins de bonheur.

C.3 Graphique vectoriel de l'axe du classement

Le générateur Php du graphique Svg de l'axe

La balise `<iframe>` nommée « hublot » est actualisée à chaque modification du classement, avec un nouveau fichier graphique Svg, qui tient compte de ces modifications. Le fichier Svg est produit à partir d'un générateur placé sur le serveur, dans le langage Php [3].

Code C.3 – *Générateur PHP du fichier SVG*

```
<?php
  if (! isset ($_REQUEST["textareaL2"])) exit;

  $liste = explode(chr(13).chr(10), $_REQUEST["textareaL2"]);
  $tab = array();
  $label1 = $_REQUEST["label1"];
  $label2 = $_REQUEST["label2"];
```

1. Svg (*Scalable Vector Graphics*, en français « image vectorielle extensible ») est un langage de description Xml pour les images vectorielles. Notons que le rapprochement de *Scalable* et *Vector* constitue un pléonasme, puisque les formats graphiques vectoriels jouissent intrinsèquement de la propriété de redimensionnement (sans perte de qualité quelle que soit l'échelle), et réciproquement, puisqu'un format véritablement redimensionnable à loisir doit être décrit vectoriellement.

2. Les fichiers au format Svg peuvent être visualisés nativement avec certains navigateurs Web, comme Konqueror, Opera, et Mozilla Firefox, ou à l'aide d'un plug-in pour d'autres navigateurs.

3. Php, acronyme récursif de *Hypertext Preprocessor*, est principalement un langage de programmation web côté serveur pour générer du code qui pourra être interprété par un navigateur.

```
8   $flag = 0;
9
10  foreach( $liste as $ligne )
11    { if (trim( $ligne ) != "")
12      if ( $ligne [0] != '(' ) { // lignes normales
13        $tab[] = stripslashes ( $ligne ); }
14      else if ( $flag==0) { // labels
15        $label1 = str_replace(array("(",")"), "", $ligne );
16        $flag = 1; }
17      else $label2 = str_replace(array("(",")"), "", $ligne );
18    }
19
20  header("Content-type: image/svg+xml");
21  echo '<?xml version="1.0" encoding="iso-8859-1"?>'."\n";
22
23  $marge = 170;
24  $margetexte = 12;
25  $nbitems = count($tab);
26   $largeurnecessaire = ($marge * 2) + ($margetexte * 2 * $nbitems);
27  if (800 > $largeurnecessaire ) $largeur = 800; else $largeur = $largeurnecessaire ;
28  $hauteur = 240;
29   $lrect = $largeur - ($marge * 2);
30  $gauche = $marge;
31   $droite = $largeur - $marge;
32
33   $yligne = $hauteur - ($margetexte * 2);
34   $hrect = 8;
35   $strockerect = 2;
36   $strockeopacityrect = 1.0;
37   $dlptr = ($margetexte/2) - 1; // demi-largargeur du pointeur
38   if ($nbitems < 2) $pas = $lrect / 2;
39   else $pas = ( $lrect - ($margetexte * 2)) / ($nbitems - 1);
40 ?>
41
42 <svg contentScriptType="text/ecmascript" width="800" height="250"
43   xmlns:xlink="http://www.w3.org/1999/xlink" zoomAndPan="magnify"
44   contentStyleType="text/css"
45   preserveAspectRatio="xMidYMid meet"
46   xmlns="http://www.w3.org/2000/svg" version="1.0"
47   viewBox="0 0 <?=$largeur?> 250">
48
49 <defs>
50 <polygon id="pointeur" fill ="url(#MyGradient)" stroke="red" stroke-width="1" points=
51   "0, <?=($dlptr*1.5)?>
52   <?=($dlptr*2)?>, <?=($dlptr*1.5)?>
53   <?=($dlptr*4)?>, 0
54   <?=($dlptr*2)?>, <?=($dlptr*4)?>"
55 />
56
57 <linearGradient id="MyGradient">
58   <stop offset="5%" stop-color="#F60" />
59   <stop offset="95%" stop-color="#FF6" />
60 </linearGradient>
61 </defs>
62
63 <style type="text/css">
64   <![CDATA[
65     #mots { fill: blue; font-size:12pt; }
66     #repere { stroke: gray; }
67     #etiquettes { font-size:14pt; fill : gray; }
68   ]]>
69 </style>
70
71 <g id="repere" stroke-linecap="round">
72 <!-- Axe horizontal -->
73 <rect x="<?=$gauche?>" y="<?=$yligne?>" width="<?=$lrect?>"
74   stroke-width="<?=$strockerect?>" stroke-opacity="<?=$strockeopacityrect?>" fill="url(#MyGradient)"
75   rx="7.0" height="<?=$hrect?>"/>
76 <!-- Arcs de cercles extremes -->
77 <path d="M <?=$gauche-$margetexte-($hrect/2)?>,<?=$yligne-$margetexte?> a <?=$margetexte?>,<?=$margetexte
       ?> 0 0,1 0,<?=($margetexte*2)+$hrect?>"
78    fill ="none" stroke-width="<?=$strockerect?>" />
79 <path d="M <?=$droite+$margetexte+($hrect/2)?>,<?=$yligne-$margetexte?> a <?=$margetexte?>,<?=$margetexte
       ?> 0 1,0 0,<?=($margetexte*2)+$hrect?>"
```

```
80    fill ="none" stroke-width="<?=$strockerect?>" />
81  <!-- Disques extremes -->
82  <circle cx="<?=$gauche-$margetexte-($hrect/2)?>" cy="<?=$yligne+($hrect/2)?>" r="<?=($hrect/2)+2?>"
83    fill ="url(#MyGradient)" stroke-width="<?=$strockerect?>" stroke-opacity="<?=$strockeopacityrect?>"/>
84  <circle cx="<?=$droite+$margetexte+($hrect/2)?>" cy="<?=$yligne+($hrect/2)?>" r="<?=($hrect/2)+2?>"
85    fill ="url(#MyGradient)" stroke-width="<?=$strockerect?>" stroke-opacity="<?=$strockeopacityrect?>"/>
86  </g>
87
88
89  <g id="mots">
90  <?php
91  if ($nbitems < 2) $monx = $gauche + ($lrect / 2); else $monx = $gauche + $margetexte;
92  $mony = $yligne - ($dlptr*4) - $strockerect;
93  if ( isset ($tab) && is_array($tab) && count($tab)) {
94      foreach($tab as $elem) {
95      echo '<use x="'.($monx-($dlptr*2)).'" y="'.$mony.'" xlink:href="#pointeur" />'."\n";
96      echo '<text x="'.$monx.'" y="'.$mony.'" transform="rotate(-45, '.$monx.', '.$mony.')">'."\n";
97      echo $elem;
98      echo '</text>'."\n\n";
99      $monx = $monx + $pas;
100     }
101 }
102 ?>
103 </g>
104
105 <g id="etiquettes">
106 <text x="<?=$gauche-$margetexte-($hrect*2)?>" y="<?=$yligne+($hrect/2)+4?>" text-anchor="end"><?=$label1
        ?></text>
107 <text x="<?=$droite+$margetexte+($hrect*2)?>" y="<?=$yligne+($hrect/2)+4?>" text-anchor="start"><?=$label2
        ?></text>
108 </g>
109
110 <!-- Show outline of canvas using 'rect' element -->
111 <!-- <rect x="1" y="1" width="<?=$largeur-2?>" height="248" fill="none" stroke="lime" stroke-width="1" /> -->
112 </svg>
```

La première partie récupère les données textuelles des trois champs du formulaire Html « textareaL2 » (la liste triée), « label1 » et « label2 » (pour nommer les deux extrémités de l'axe), et calcule plusieurs variables en conséquence. La seconde partie prend en charge le code Svg proprement dit : avec d'abord l'entête Svg, puis les définitions graphiques de la forme des « pointeurs » et d'un gradient linéaire, ensuite une feuille de style Css [4] intégrée, et enfin les groupes Svg nommés « repere », « mots » et « etiquettes ».

L'interprète Php procède soit par substitution directe, lorsqu'il rencontre des balises délimitées par "`<?=`" et "`?>`", soit par impression explicite avec la fonction `echo` au sein d'un bloc Php (délimité par "`<?php`" et "`?>`"). Le cœur de l'algorithme se trouve ici dans la boucle `foreach` (ligne 94) qui distribue proportionnellement sur l'axe autant de mots qu'il y en a dans la liste triée, dans l'ordre, grâce aux balises Svg `<text>`.

Exemple de fichier Svg généré

Le résultat du calcul Php renvoyé par le serveur est un fichier complet au format Svg, affiché dans la balise `<iframe>` nommée « hublot » de la fenêtre principale par le navigateur assisté ou non d'un plug-in Svg.

Ce fichier est affiché par le navigateur, mais on peut aussi l'enregistrer et avoir accès au code source, par exemple pour le modifier soi-même, le convertir dans un autre format, ou

4. *Cascading Style Sheets* ou « feuilles de style en cascade » ; Css est utilisé pour décrire la présentation d'un document structuré écrit en Html ou en Xml

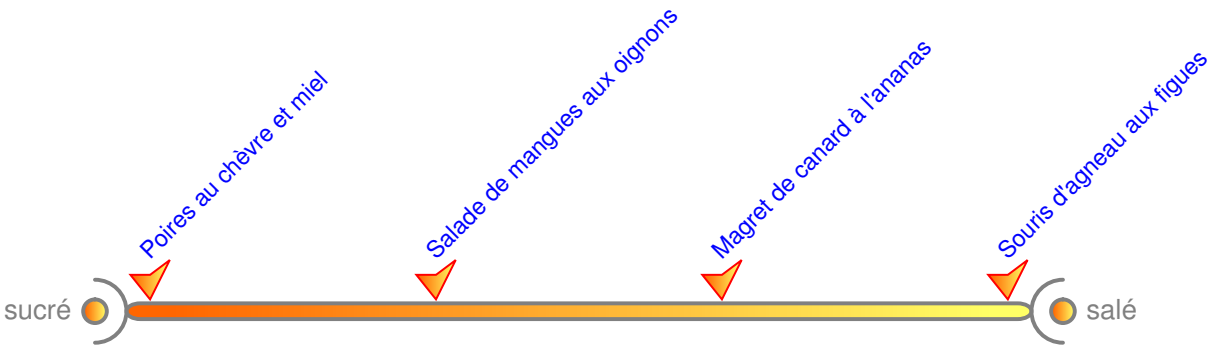


FIG. C.1 – *Visualisation d'un fichier* SVG *généré*

l'éditer dans un des nombreux logiciels qui reconnaissent ce format vectoriel. Sa syntaxe obéit au standard XML structuré par des balises, ce qui le rend relativement aisé à lire. En outre, le format vectoriel permet de zoomer à l'infini sans dégradation de la qualité de l'image.

Code C.4 – *Source du fichier SVG généré*

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<!-- Created by karim barkati kb2007 -->

<svg
  xmlns="http://www.w3.org/2000/svg" version="1.0"
   xmlns:xlink ="http://www.w3.org/1999/xlink"
  contentStyleType="text/css"
  contentScriptType="text/ecmascript"
  zoomAndPan="magnify"
  preserveAspectRatio="xMidYMid meet"
  width="800" height="250"
  viewBox="0 0 800 250">

<defs>
<polygon id="pointeur"  fill ="url(#MyGradient)" stroke="red" stroke-width="1" points=
  "0, 7.5
  10, 7.5
  20, 0
  10, 20"
/>

<linearGradient id="MyGradient">
  <stop offset ="5%" stop-color="#F60" />
  <stop offset ="95%" stop-color="#FF6" />
</linearGradient>
</defs>

<style type="text/css">

  <![CDATA[
    #mots { fill: blue; font-size:12pt; }
    #repere { stroke: gray; }
    #etiquettes { font-size:14pt; fill: gray; }
  ]]>
</style>

<g id="repere" stroke-linecap="round">
<!-- Axe horizontal -->
<rect x="170" y="216" width="460"
  stroke-width="2" stroke-opacity="1" fill="url(#MyGradient)"
  rx="7.0" height="8"/>
<!-- Arcs de cercles extremes -->
<path d="M 154,204 a 12,12 0 0,1 0,32"
   fill ="none" stroke-width="2" />
<path d="M 646,204 a 12,12 0 1,0 0,32"
```

```
    fill ="none" stroke-width="2" />
<!-- Disques extremes -->
<circle cx="154" cy="220" r="6"
    fill ="url(#MyGradient)" stroke-width="2" stroke-opacity="1"/>
<circle cx="646" cy="220" r="6"
    fill ="url(#MyGradient)" stroke-width="2" stroke-opacity="1"/>
</g>


<g id="mots">

<use x="172" y="194" xlink:href="#pointeur" />
<text x="182" y="194" transform="rotate(-45, 182, 194)">
Poires au chèvre et miel</text>

<use x="317.33333333333" y="194" xlink:href="#pointeur" />
<text x="327.33333333333" y="194" transform="rotate(-45, 327.33333333333, 194)">
Salade de mangues aux oignons</text>

<use x="462.66666666667" y="194" xlink:href="#pointeur" />
<text x="472.66666666667" y="194" transform="rotate(-45, 472.66666666667, 194)">
Magret de canard à l'ananas</text>

<use x="608" y="194" xlink:href="#pointeur" />
<text x="618" y="194" transform="rotate(-45, 618, 194)">

Souris d'agneau aux figues</text>

</g>

<g id="etiquettes">
<text x="142" y="224" text-anchor="end">sucré</text>
<text x="658" y="224" text-anchor="start">salé</text>
</g>

<!-- Show outline of canvas using 'rect' element -->
<!-- <rect x="1" y="1" width="798" height="248" fill="none" stroke="lime" stroke-width="1" /> -->
</svg>
```

Annexe D

Configuration matérielle utilisée pour le concert

D.1 Liste détaillée

À titre indicatif et par souci de reproductibilité du protocole expérimental, voici la liste complète du matériel (hors instruments de musique) employé pour les concerts :

- un pédalier MIDI Behringer FCB1010 ;
- un capteur intra-bec FWF[1] pour la clarinette, placé dans un bec Buffet Crampon percé latéralement ;
- un microphone dynamique super-cardioïde Sennheiser e608, col-de-cygne, utilisé sur l'alto ;
- un microphone Neumann prêté par Iván pour la prise de son des instruments ;
- un carte son externe et portable M-Audio FireWire 410, équipée de 2 entrées audio micro/ligne avec préamplificateurs et alimentation fantôme, de 8 sorties ligne, et d'un couple entrée/sortie MIDI ;
- un MacBook Pro Apple 15", processeur à 2 GHz, avec 1,5 Go de RAM et un disque interne de 150 Go ;
- une paire d'enceintes bi-amplifiées Gemini GX350 de 350 W chacune, bande passante de 40 Hz à 22 kHz ;
- une paire de microphones Studio Project C4 à condensateur de petit diaphragme, avec suspension anti-choc et deux capsules interchangeables omnidirectionnelles ou cardioïdes, utilisés pour l'enregistrement du concert ;
- un boîtier anti-larsen Behringer Shark DSP110, dont la largeur de bande des filtres reste étroite (jusqu'à 1/60 d'octave).

D.2 Capteur intra-bec

Le système FWF possède trois réglages : l'ouverture du capteur au niveau du bec, la sensibilité du signal en provenance du capteur au niveau du boitier-ceinture, et le filtre

1. Franck William FROMY, artisan installé à Paris.

de balance grave/aïgu dans le boîtier-ceinture. Ces trois réglages électro-mécaniques interdépendants agissent sur plusieurs paramètres perceptifs simultanément, essentiellement l'amplitude et la brillance, et demandent donc un certain temps d'apprentissage. En général, l'ouverture du capteur interne doit rester relativement petite, sans quoi, la membrane recevant beaucoup de pression, le son se retrouve complètement dénaturé en une sorte d'onde carrée saturée.

D.3 Vidéos

Le concert a été enregistré avec trois caméscopes numériques, enregistrement supervisé par Sophie Réthoré. J'ai monté quatre des pièces et improvisations, déposées sur une plate-forme de partage et de visionnage de clips vidéos en ligne, à l'adresse http://www.dailymotion.com/digitalsib.

Annexe E

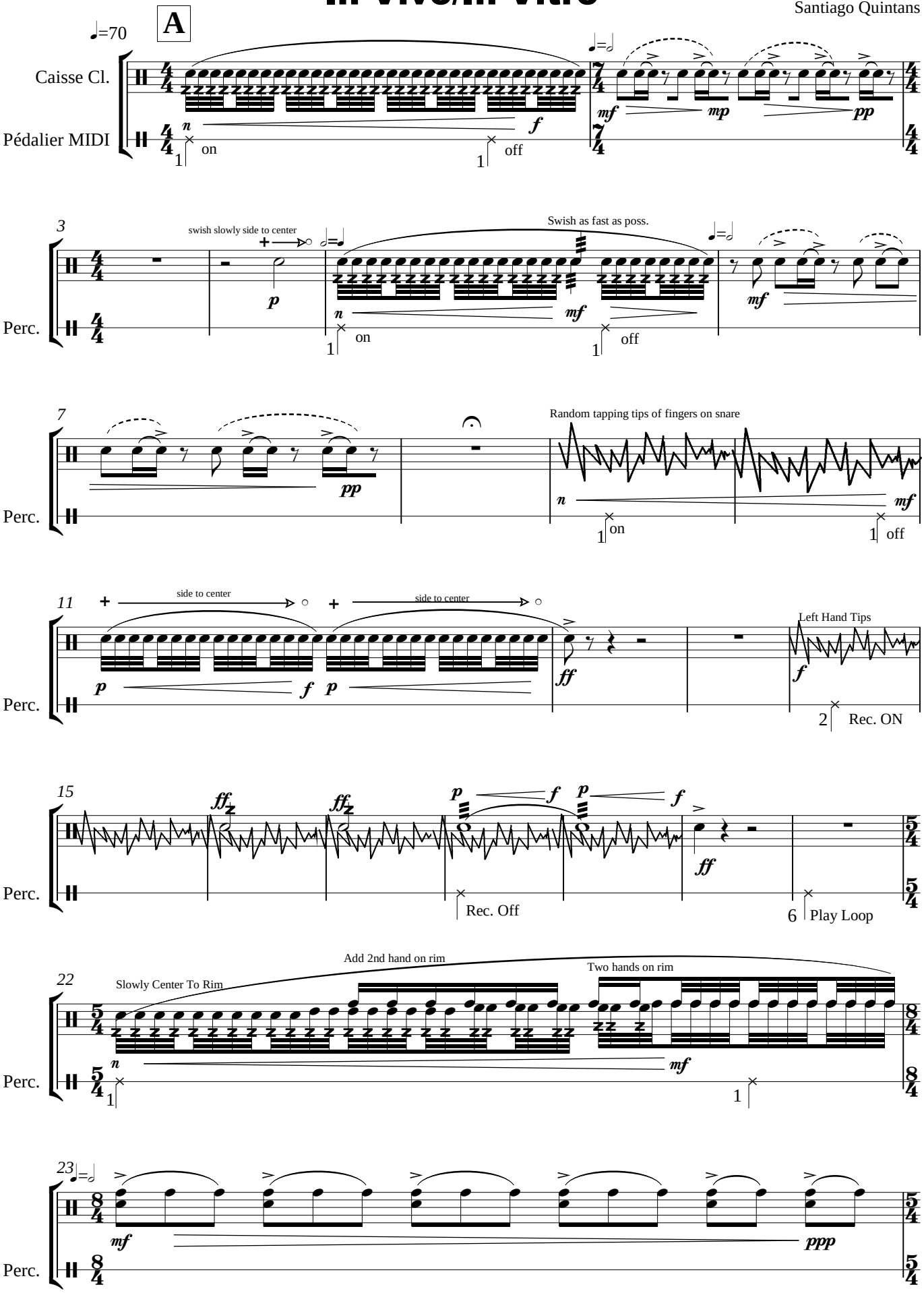
Partitions

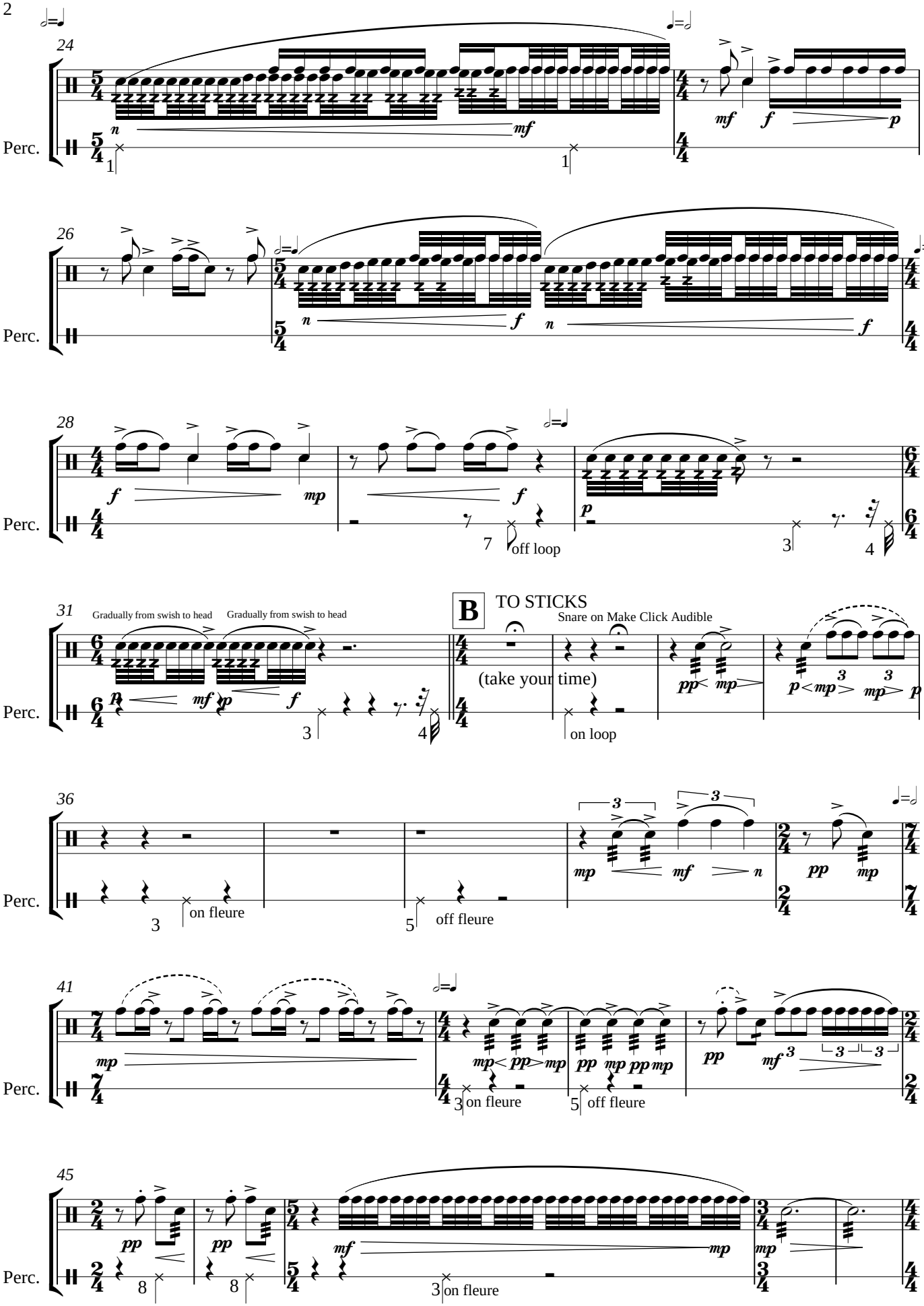
E.1 In Vivo / In Vitro de Santiago Quintáns

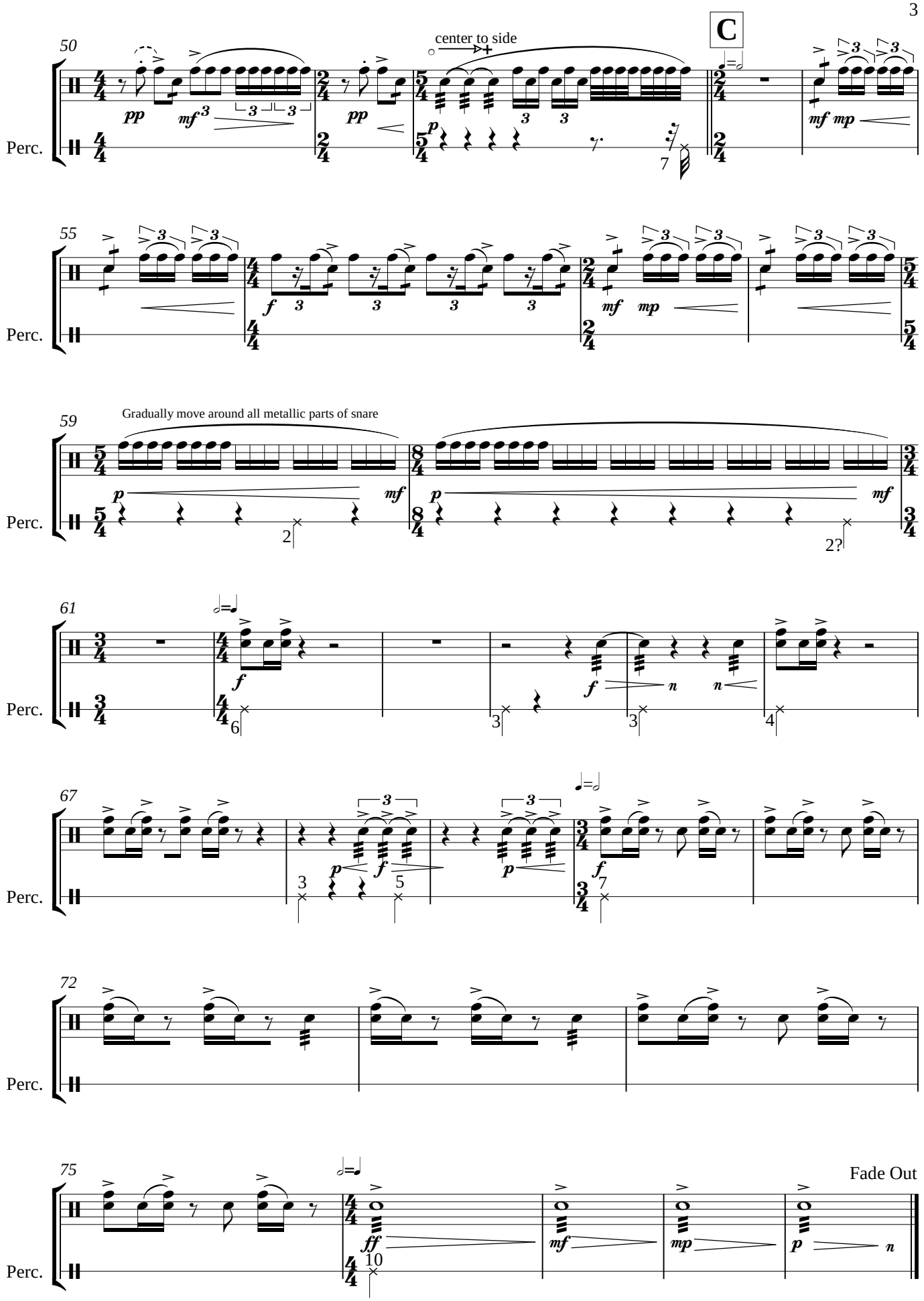
Ci-suit la partition complète de *In Vivo / In Vitro* pour caisse claire et pédalier MIDI, composée et copiée par Santiago Quintáns, dans sa forme originale : trois pages au format A4. Cette pièce a été créée en concert par Clarissa Borba le 2 février 2007 à Gennevilliers.

In Vivo/In Vitro

Santiago Quintans




24
Perc.
n
mf
1
1
mf
f
p
26
Perc.
n
f
n
f
28
Perc.
f
mp
f
p
7
off loop
3
4
31
Gradually from swish to head
Gradually from swish to head
B
TO STICKS
Snare on Make Click Audible
Perc.
n
mf
p
f
3
4
(take your time)
on loop
pp
mp
p
mp
3
3
mp
p
36
Perc.
3
on fleure
5
off fleure
3
3
mp
mf
n
pp
mp
41
Perc.
mp
3
on fleure
5
off fleure
mp
pp
mp
pp
mp
pp
mp
pp
mf
3
3
3
45
Perc.
pp
pp
8
8
mf
mp
3
on fleure
mp


50
center to side
C
Perc.
55
Perc.
59
Gradually move around all metallic parts of snare
Perc.
61
Perc.
67
Perc.
72
Perc.
75
Fade Out
Perc.

E.2 Woes-war-sollichwer-den de Mauricio Meza

Notice

Mauricio a rédigé une notice de quatre pages pour expliciter le projet, à l'interprète en particulier, tant au niveau de l'architecture relativement complexe, qu'au niveau des clés pour l'interprétation.

Woes-war-sollichswer-den

Pour Clarinette soprano en si b, Ordinateur, et Participation du public

Mauricio Meza
2006

Notice

Le titre de cette pièce provient de « Wo Es war soll Ich werden » qui veut dire en quelque sorte « Là ou était du Ça doit advenir du Moi[1] ». Il s'agit d'une célèbre formulation de Freud pour définir l'Inconscient. Dans le contexte de ce projet, ce titre a d'abord la fonction de rentrer en résonance avec la section centrale de la pièce, celle dans laquelle le public apporte sa voix.

Traitements :

Interactif : Basé sur des modes de lecture d'un buffer et contrôlés en temps réel par l'interprète, par le biais d'une interface de type pédale d'expression.

Tous les autres traitements sont contrôlés de façon automatique par l'ordinateur, notamment l'enregistrement de l'instrument et du public ainsi que la convolution, la granulation et le délai.

Dans toutes les transitions :

1) Toutes les improvisations ont un point de départ et une fin déterminés. Le choix des modules pour toutes les transitions sera proposé par un tirage au sort avant de jouer la section A ; l'interprète doit alors noter sur la partition de chaque transition les modules d'entrée et de sortie correspondants.

2) Uniquement dans les transitions, les modules doivent être joués rubato.

1 COLLECTIF, *L'homme cognitif*, WEIL-BARAS, Annick (dir.), Paris, Presses Universitaires de France, 1993, p. 89.

Section A

Dans cette section, il est question d'une situation "classique" de jeu et enregistrement.

L'interprète doit simplement jouer la série de modules 06 12 05 10 11 06 d'un seul tenant. Il doit aussi jouer cela en ignorant les indications de preset de traitement interactif, c'est-à-dire SANS TRAITEMENTS.

Il n'y a pas de traitement numérique du son, la seule tâche à accomplir par l'ordinateur est d'enregistrer l'interprétation. Cet enregistrement sera utilisé comme matériau dans les sections suivantes.

Transition A vers B

Après avoir joué le module d'introduction, improviser librement puis clore le discours avec le module de sortie correspondant. Utiliser le pédalier librement mais pendant toute la transition.

Section B

Même principe que pour la section A, mais cette fois-ci inclure les presets de traitement interactif.

Dans cette section, l'ordinateur introduit la première de ses interventions "autonomes".

Transition B vers C

Comme dans la transition précédente, à l'exception du pédalier qui peut être utilisé assez librement quant à sa présence pendant la transition.

Section C

Dans cette section, l'interprète a plus une fonction d'animateur que d'instrumentiste. Il s'agit de convoquer le public à entrer en résonance avec l'événement par le biais d'un jeu de type "question - réponse".

Le matériau pour le jeu c'est le langage naturel, le mot. Il s'agit de proposer, à tour de rôle, un mot choisi parmi les mots présents dans la liste qui accompagne cette partition, et de recueillir les associations d'idées proposées par le public par rapport au mot proposé.

L'interprète peut s'inspirer de cette situation théâtrale :

Interpète : Si je vous dis ...ballon... vous dites . . .

Quelqu'un du public : foot !
Quelqu'un du public 2 : (presque au même temps) volley !

Quelqu'un du public 3 : (un peu après) jonglage ?
Quelqu'un du public 4 : (tout de suite après) belle mère !
Quelques uns du public : (ensemble) Zizou !

Interprète : Merci.....si je vous dis ...anarchie... vous dites.....

(silence et perplexité)

Quelqu'un du public : John Cage !

Choix des mots :

Dans la liste, les mots sont divisées en deux groupes :

Groupe 1

Ce sont des mots qui ont une fonction ludique. Ils doivent servir à faire rentrer le public dans le jeu, quelques-uns d'entre eux doivent donc être proposés en début de jeu. Cependant ils pourront aussi avoir une utilité pour la modulation de la tension dramatique pendant le jeu.

Les mots de ce groupe pourront être remplacés par l'interprète si besoin. Également, l'ordre de leur apparition peut être déterminé par l'interprète.

Groupe 2

Ces mots ne sont pas remplaçables et leur présence dans le jeu est obligatoire. L'ordre de leur apparition peut être modulé au choix de l'interprète sauf quelques exceptions que j'expose tout de suite après.

Il s'agit donc de faire une transition du Groupe 1 vers le Groupe 2, ceci doit normalement donner comme résultat une certain tension dramatique.

Groupe 1---------->Groupe 1 + Groupe 2 ----------> Groupe 2

Cette partition est accompagné de 12 panneaux contenant chacun 4 mots. Cette organisation correspond au principe exposée ultérieurement. Dans tout agencement de ces panneaux ou d'autres, l'inclusion d'au moins une de ces séries est obligatoire :

Série 3, 4, 5.

Série 6, 7, 8.

Série 9, 10, 11.

La fin du jeu est toujours obligatoirement le panneau 12.

Les mots de ces panneaux peuvent être présentés au public dans n'importe quel ordre. La répétition de n'importe quel mot est interdite.

Les voix du public seront enregistrés par l'ordinateur pour constituer un

nouveau matériau sonore.

De plus, dans cette section l'interprète doit jouer au moins une fois avec le pédalier seul, de façon à faire entendre quelques transformations des voix enregistrées.

Transition C vers D

Cette transition fonctionne de la même manière que les transitions antérieures, la seule différence étant le changement de nature du matériau sonore, lequel inclut maintenant la convolution de l'instrument avec les voix enregistrées.

Section D

C'est une improvisation "libre". Toutes les matières sonores du corps de la pièce sont à disposition de l'interprète. Seule la courbe dynamique générale est imposée.

Depuis le début de la pièce, l'ordinateur mène un "processus de transformation caché". Dans cette section, il abouti à un résultat plastique particulier. Ce résultat sonore est autonome mais sensible au jeu de l'interprète.

Transition D vers E

Cette transition fonctionne comme les transitions antérieures, mais elle est plus courte en durée et comporte une courbe dynamique imposée.

Section E

Cette section est celle de la métaphore d'une autonomie de l'ordinateur. L'interprète peut se reposer.

Coda

Il s'agit d'arrêter le processus mené par la machine. L'interprète -improvisateur intervient au moment désiré. La durée de 10'' est globale. L'intervention de l'instrument peut se réduire à quelques gestes. Il est important de s'arrêter avant ou avec la machine, en aucun cas après. Rester immobile et laisser la résonance s'éteindre.

Conduite

Ci-après un document synoptique, support au développement pour le programmeur, et support synthétique au travail de la pièce pour l'interprète.

[A]

M.Meza
2006

4'30 à 5'

Cl.Bb.

Jouer cette série de Modules :

06 — 12 — 05 — 10 — 11 — 06

Ordinateur

Enregistrer l'interprétation instrumentale

[Transition A vers B]

M.Meza
2006

Environ 3'

Cl.Bb.

Jouer les Modules *sempre rubato*

Module d'entrée : Module de sortie

IMPROVISATION

Clarinette + Pédalier

Ordinateur

Interactivité exogène :

Traitement interactif (en temps réel) sur le son de la clarinette par le biais du pédalier.

Interactivité endogène :

Analyse et transformation de l'enregistrement de la section [A]. Mise en place de la nappe à déclencher en [B].

Woes-war-sollichwer-den

[B]

M.Meza
2006

5'30 environ

Cl.Bb.

Jouer la deuxième série de modules :

10 — 06 — 12 — 03 — 01 — 10

Clarinette + Pédalier

Ordinateur

Interactivité exogène :

Traitement interactif (en temps réel) sur le son de la clarinette par le biais du pédalier.

Interactivité endogène :

Fin de la transformation de l'enregistrement de la section [A] (processus caché).

Déclenchement de la nappe à 3'.

niente — ***mp***

Woes-war-sollichwer-den

[Transition B vers C]

M.Meza
2006

Environ 3'

Cl.Bb.

Jouer les Modules *sempre rubato*

Module d'entrée : Module de sortie :

IMPROVISATION

Clarinette + Pédalier

Ordinateur

Interactivité exogène :

Traitement interactif (en temps réel) sur le son de la clarinette par le biais du pédalier.

Interactivité endogène :

Maintien de la nappe en arrière plan. Enveloppe d'amplitude automatique. Mise en place des conditions pour l'interaction avec le public en [C].

[C]

M.Meza
2006

Interaction avec le public

Entre 4' et 5' environ

Laisser la clarinette et prendre le microphone.

Inviter de façon subtile le public à répondre aux suggestions des mots énoncés.*

Cl.Bb.

Contrôle de l'enregistrement des voix.
Improvisation avec le pédalier (contrôle de l'enveloppe d'amplitude et de l'index de convolution).
Metre en place un jeu d'alternance avec le public de façon à faire entendre ce que leur apport acoustique (voix) devienne une fois capté par le dispositif.

Interactivité exogène :
Traitement en temps réel au pédalier : Contrôle manuel de l'enveloppe d'amplitude et/ou de l'index de convolution voix + nappe et/ou du niveau du feed-back.

Ordinateur

Interactivité endogène :
Maintien de la nappe et du processus d'entropie appliqué sur celle-ci. Enregistrement des voix.
Feed-back automatisé (si celui-ci n'est pas contrôlé manuellement par l'interprète instrumentiste). Filtrage.

Public

Participation en groupe ou individuelle : réagir aux mots proposées par l'instrumentiste, faire des associations d'idées, s'exprimer librement, essayer de rester dans le jeu, un mot convoque d'autres mots.

* Liste et ordre des mots en annexe de cette partition.

[Transition C vers D]

M.Meza
2006

Environ 3'

Jouer les Modules *sempre rubato*

Module d'entrée : Module de sortie :

IMPROVISATION

Cl.Bb.

Clarinette + Pédalier

Contrôle sur l'index de convolution avec les voix enregistrées.

Interactivité exogène :
Traitement interactif (en temps réel) sur le son de la clarinette par le biais du pédalier.
Contrôle sur l'index de convolution avec les voix enregistrées.

Ordinateur

Interactivité endogène :
Maintien de la nappe en arrière plan. Enveloppe d'amplitude automatique. Maintien du processus d'entropie. Mise en place des conditions pour l'interaction avec l'instrumentiste sur la section [D].

[D]

Environ 5'

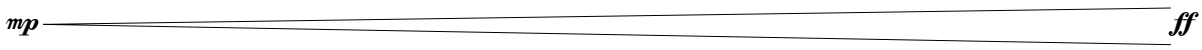
Cl.Bb.

IMPROVISATION

Dialogue avec l'ordinateur

Clarinette + Pédalier

Contrôle sur le traitement en temps réel de la clarinette.



Ordinateur

Interactivité exogène :

Traitement interactif (en temps réel) sur le son de la clarinette par le biais du pédalier.

Interactivité endogène :

Maintien du processus d'entropie. La nappe est autonome, elle a un comportement et est capable de réagir au jeu de la clarinette. Cette réaction se manifeste par l'amplitude, l'index de granulation et l'index de convolution.

[Transition D vers E]

Environ 2'

Cl.Bb.

Jouer les Modules *sempre rubato*

Module d'entrée :

Module de sortie :

IMPROVISATION

Clarinette + Pédalier

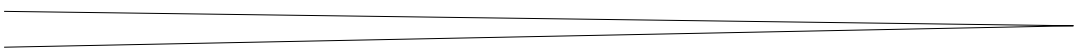
Ordinateur

Interactivité exogène :

Traitement interactif (en temps réel) sur le son de la clarinette par le biais du pédalier.

Interactivité endogène :

Maintien de la nappe en arrière plan. Enveloppe d'amplitude automatique. Maintien du processus d'entropie. Mise en place des conditions pour la section [E].



[E]

Environ 3'

Interactivité endogène :

Ordinateur

Mener le processus d'entropie à un niveau très élevé (maximale). Enveloppe automatique. Modulation automatique des processus. Incrément de feed-back. Enveloppe générale d'amplitude vers ***fff***. Paroxysme.

mf ——————————— ***fffff***

[CODA]

10"

Cl.Bb.

IMPROVISATION

Clarinette seule

Jouer contre l'ordinateur

Une série de gestes agressifs auront comme effet de l'arrêter.
Une fois la machine arrêté, arrêter de jouer et laisser s'éteindre la résonance par elle même.

——————————— *niente*

Interactivité exogène :

Captation de la clarinette.

Ordinateur

Interactivité endogène :

Arrêt soudain qui arrive comme conséquence du jeu agressif de la clarinette.
Résonance finale assez longue.

Modules

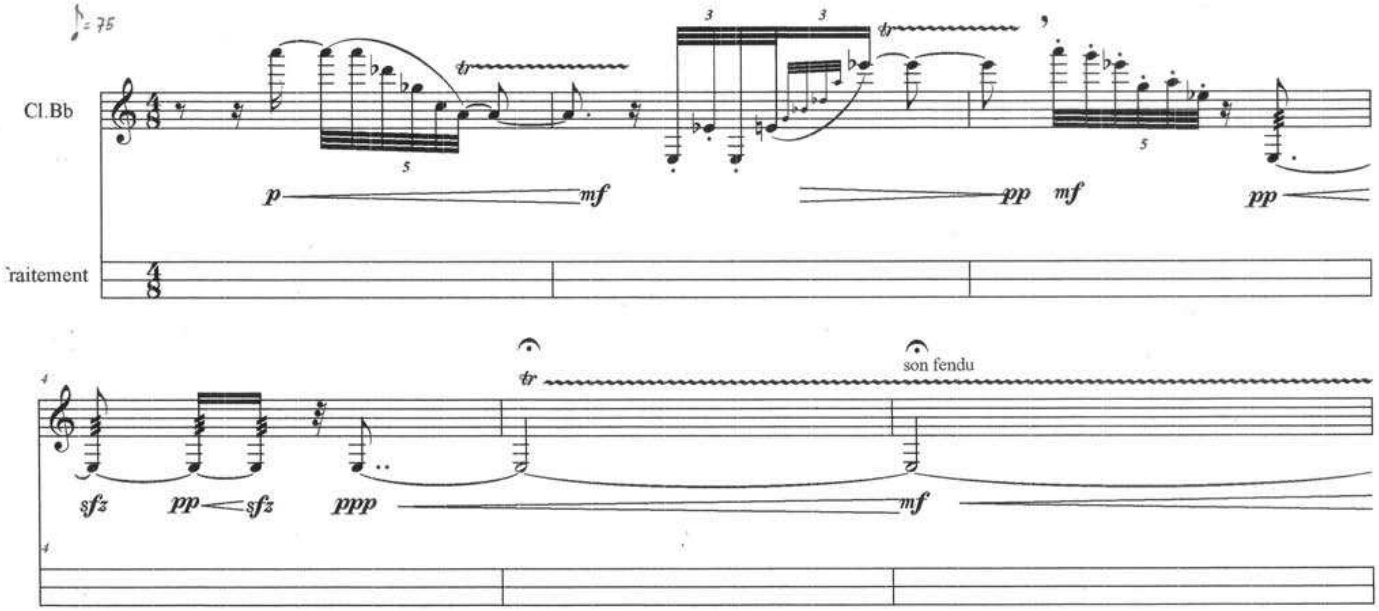
Les « modules » ouvrent ici l'usage de la musique écrite, en atomisant la partition et en proposant des supports à l'improvisation pour le couple « module de départ » / « module d'arrivée » tiré par le programme de loterie juste avant l'exécution de la pièce.

Il y a en tout 9 modules, numérotés de « 00 » à « 12 » :

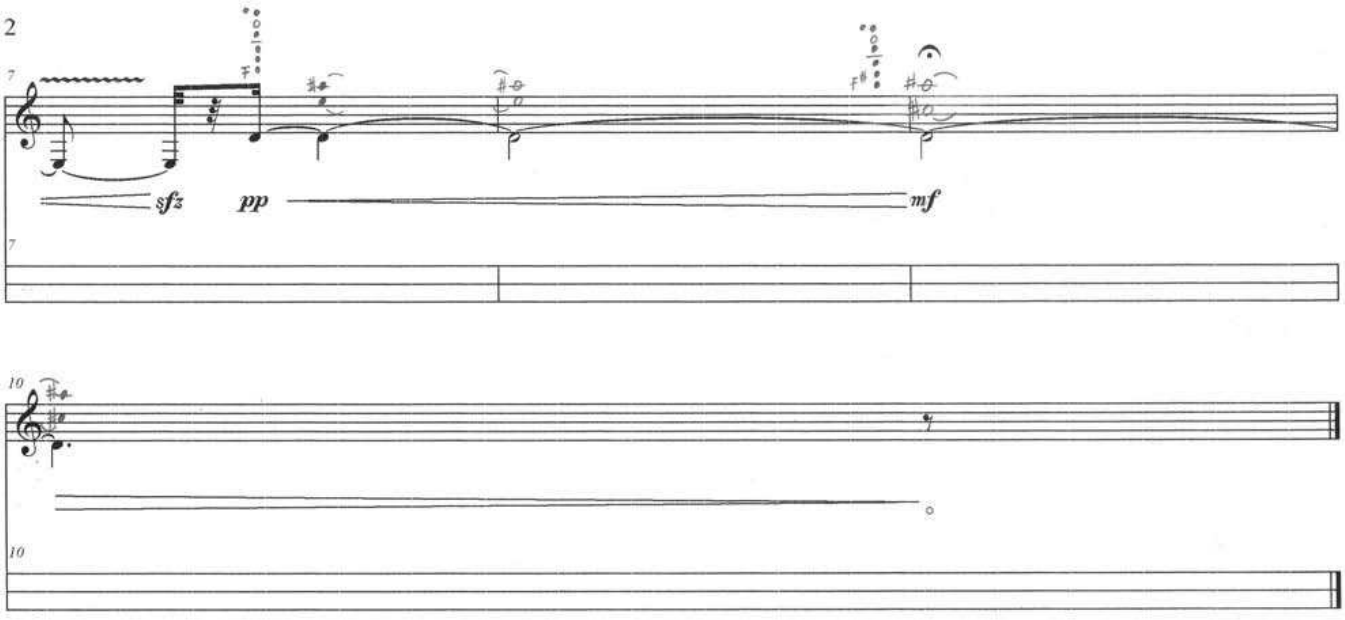
- module 00
- module 01
- module 03
- module 04
- module 05
- module 06
- module 10
- module 11
- module 12

MODULE 00

M.MEZA
2006

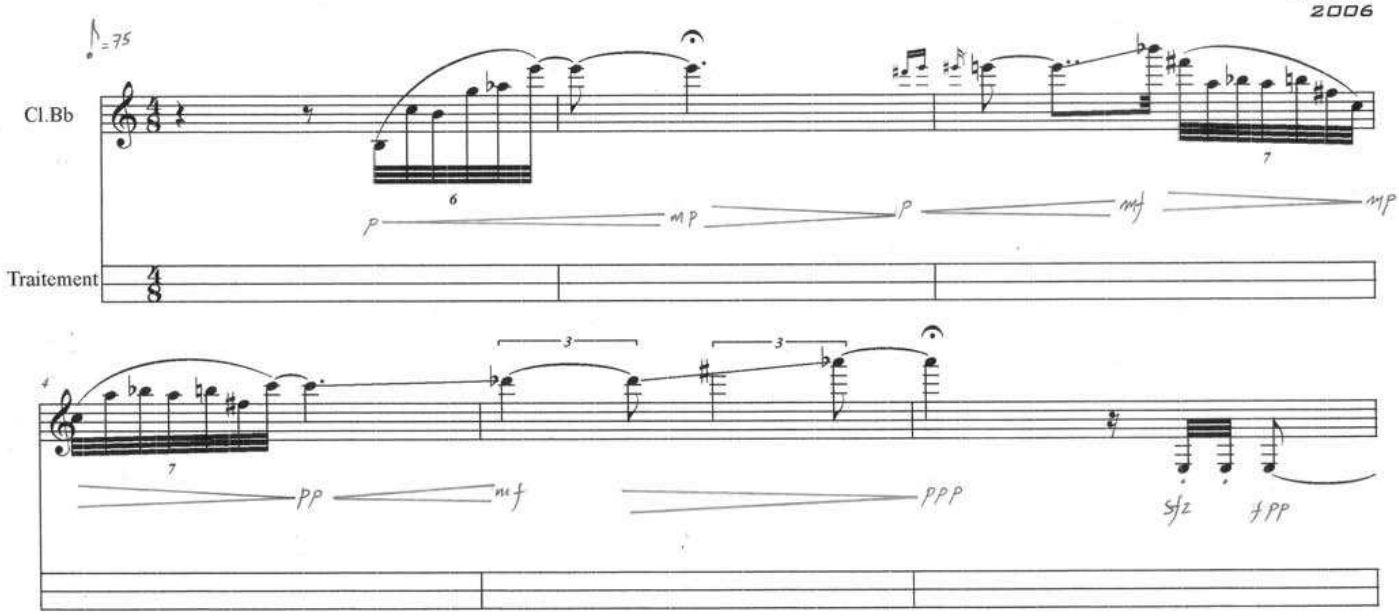


2

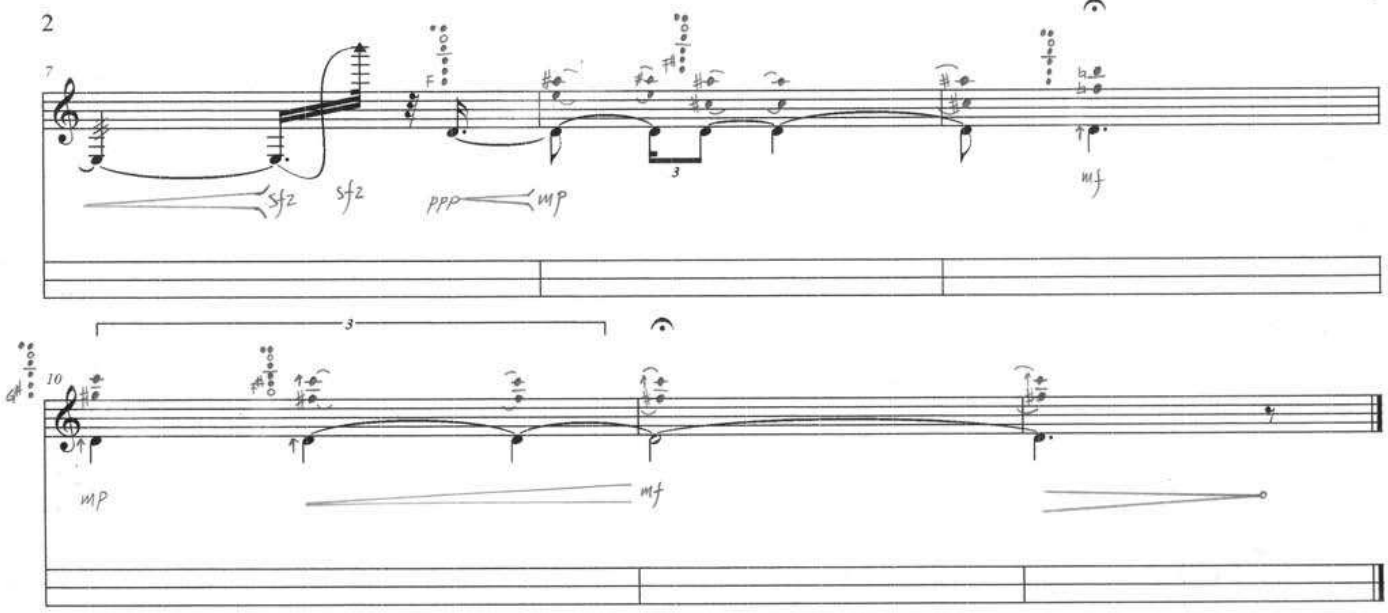


MODULE 01

M.MEZA
2006

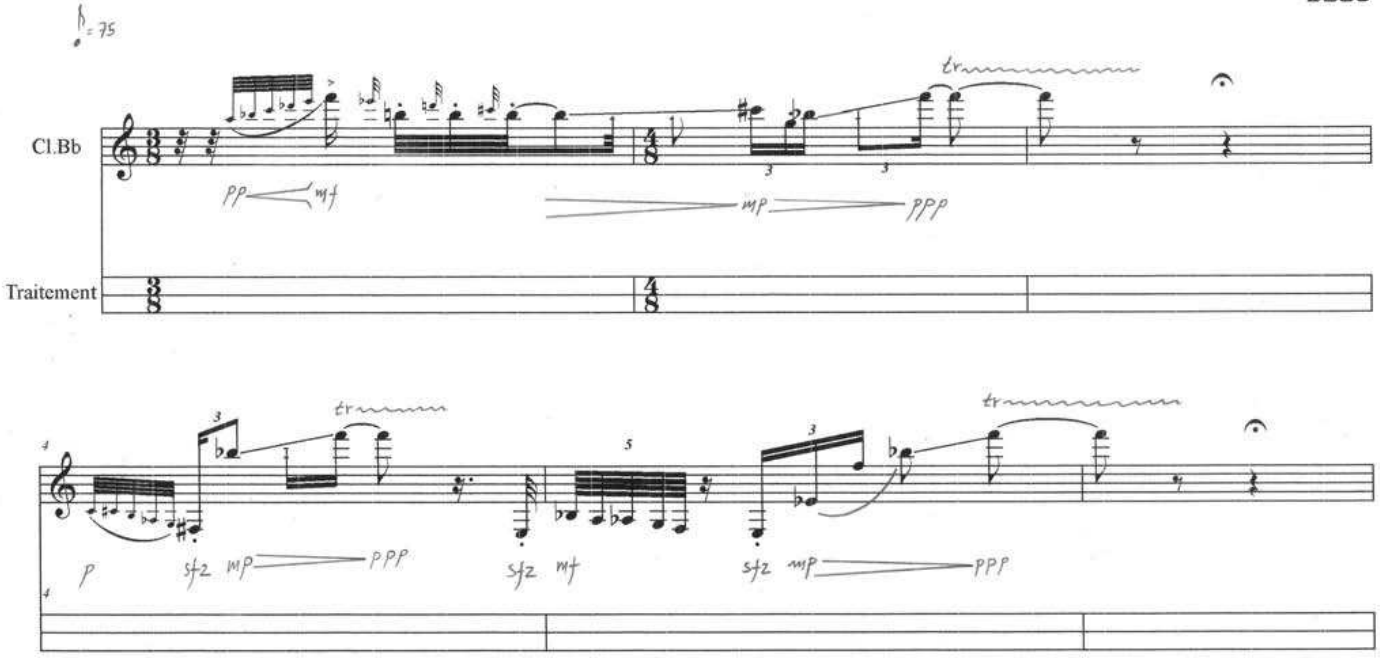


m. 01

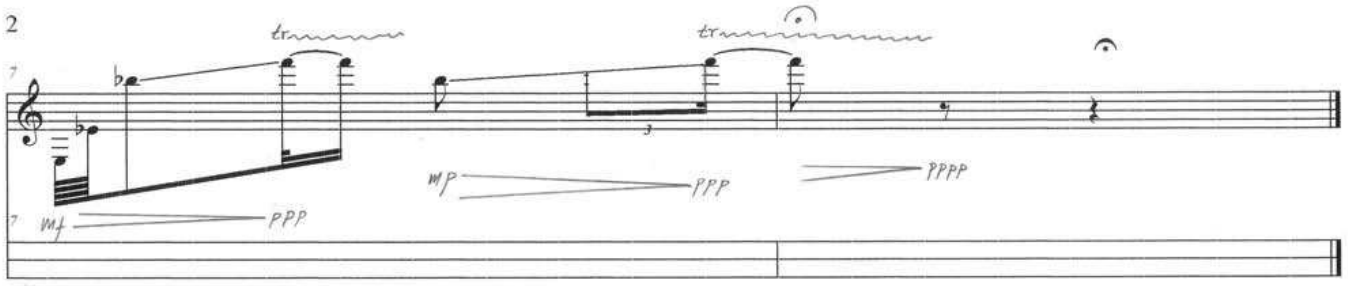


MODULE 03

M.MEZA
2006

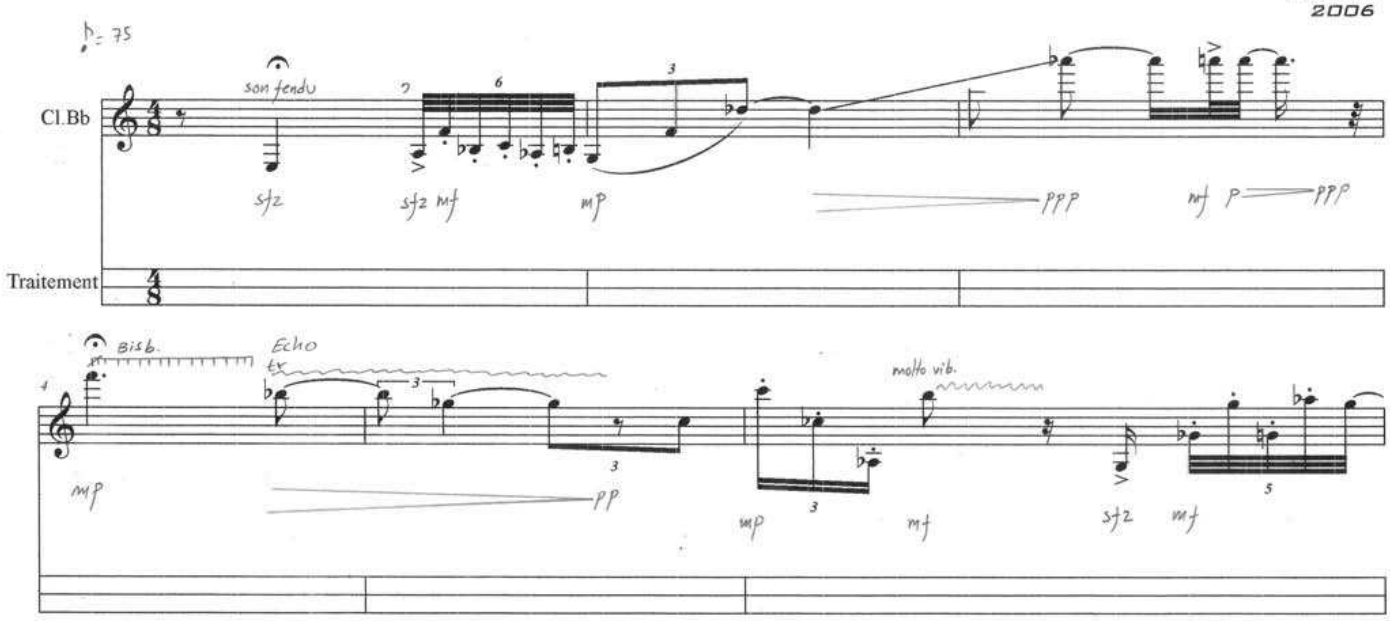


m. 03

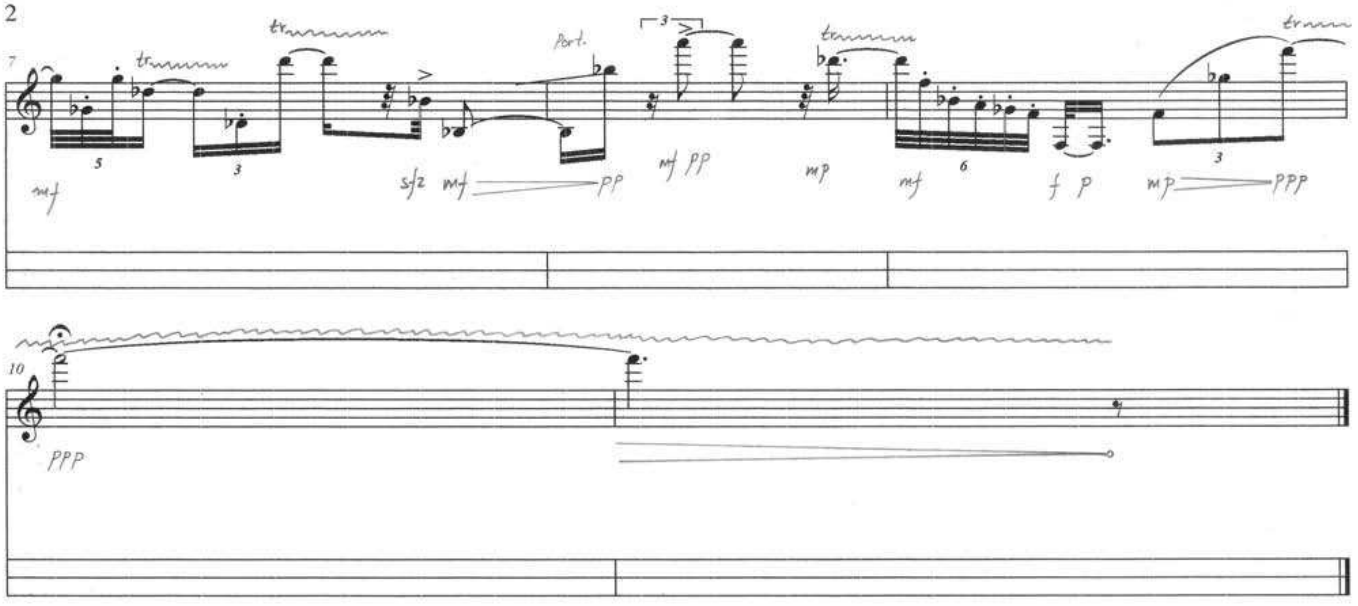


MODULE 04

M.MEZA
2006

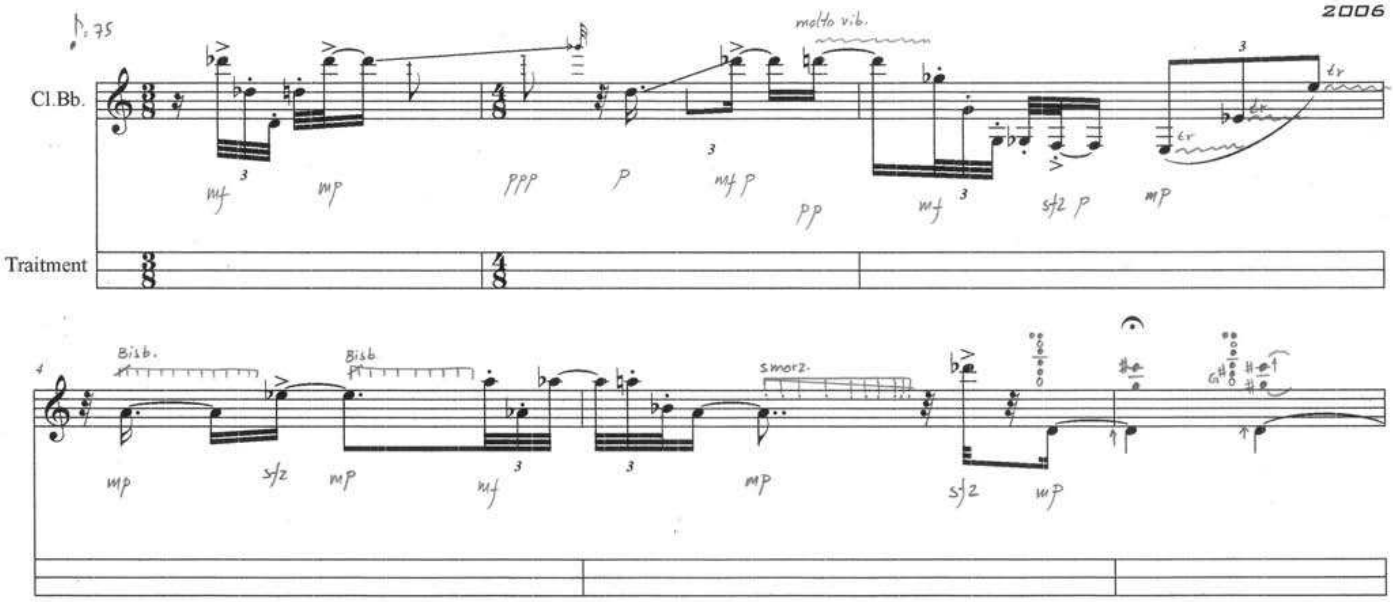


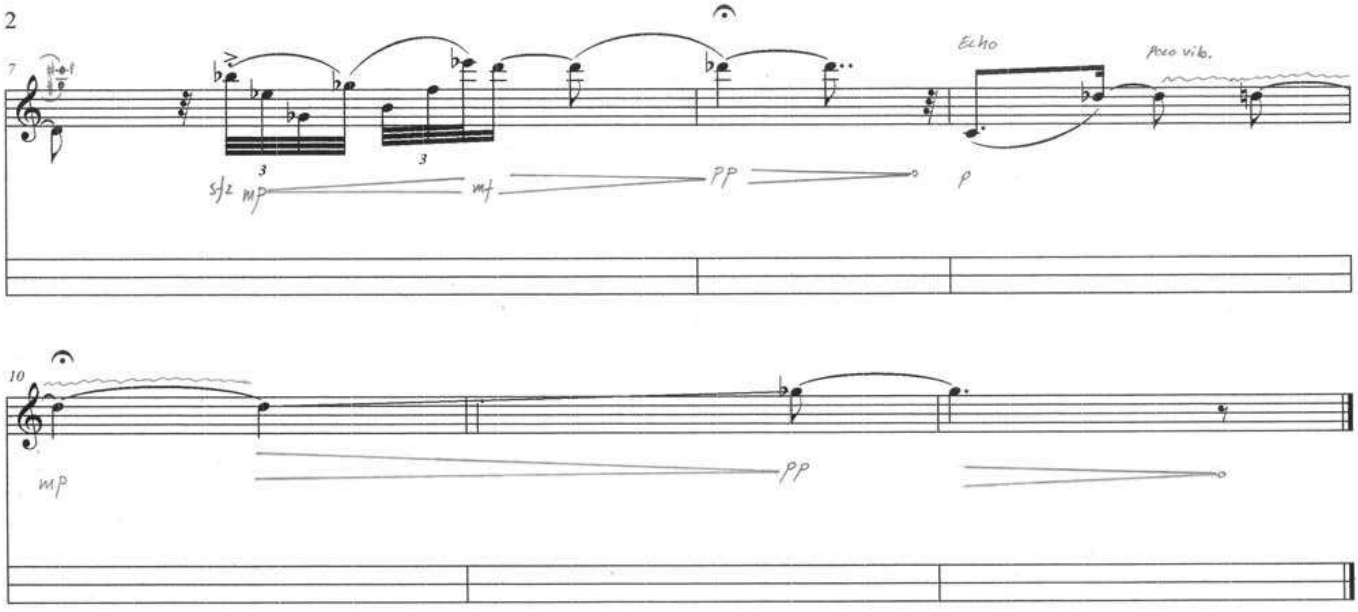
2



MODULE 05

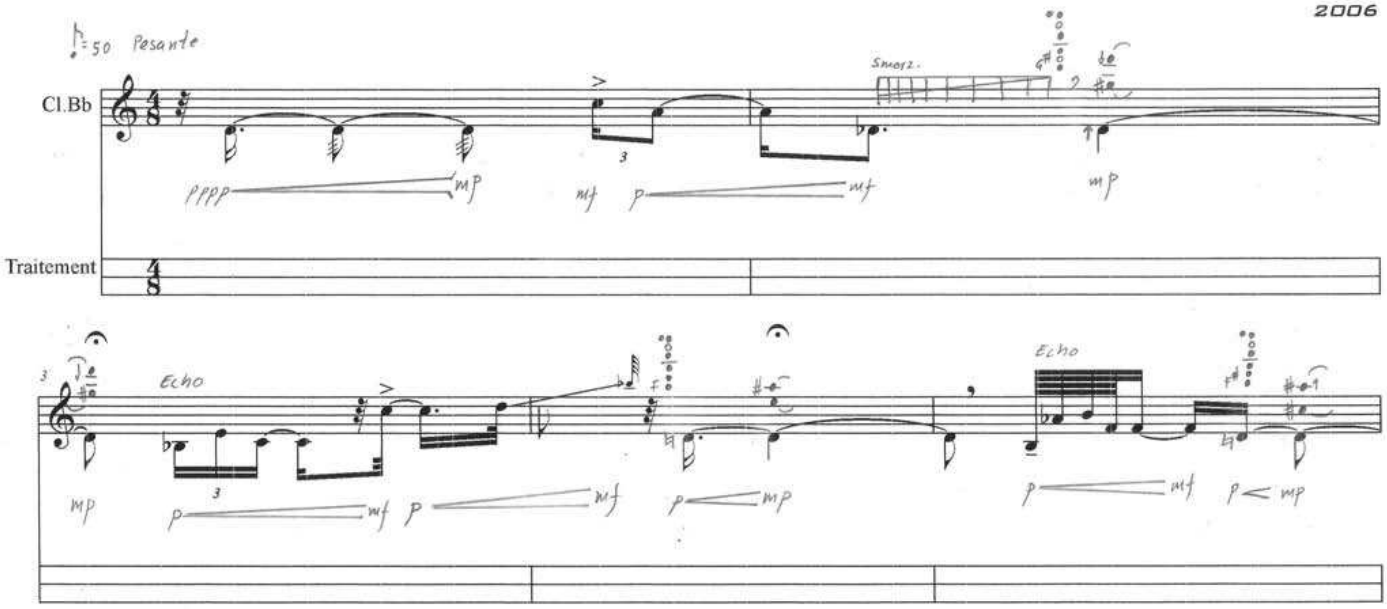
M.MEZA
2006

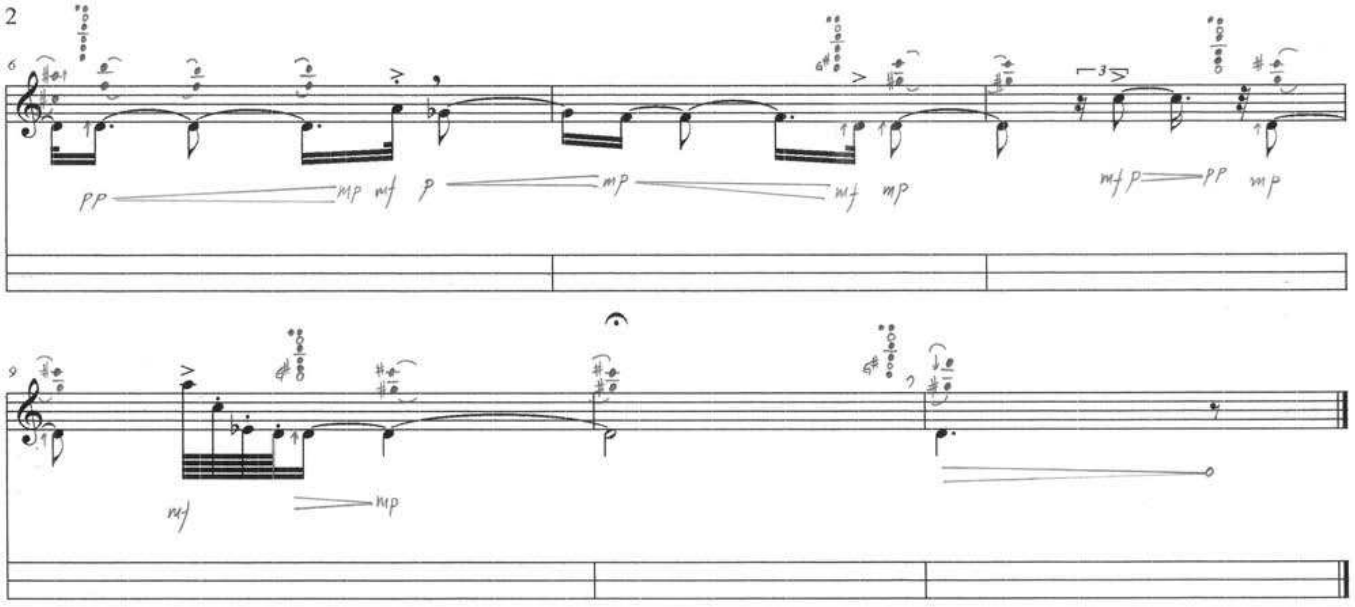




MODULE 06

M.MEZA
2006





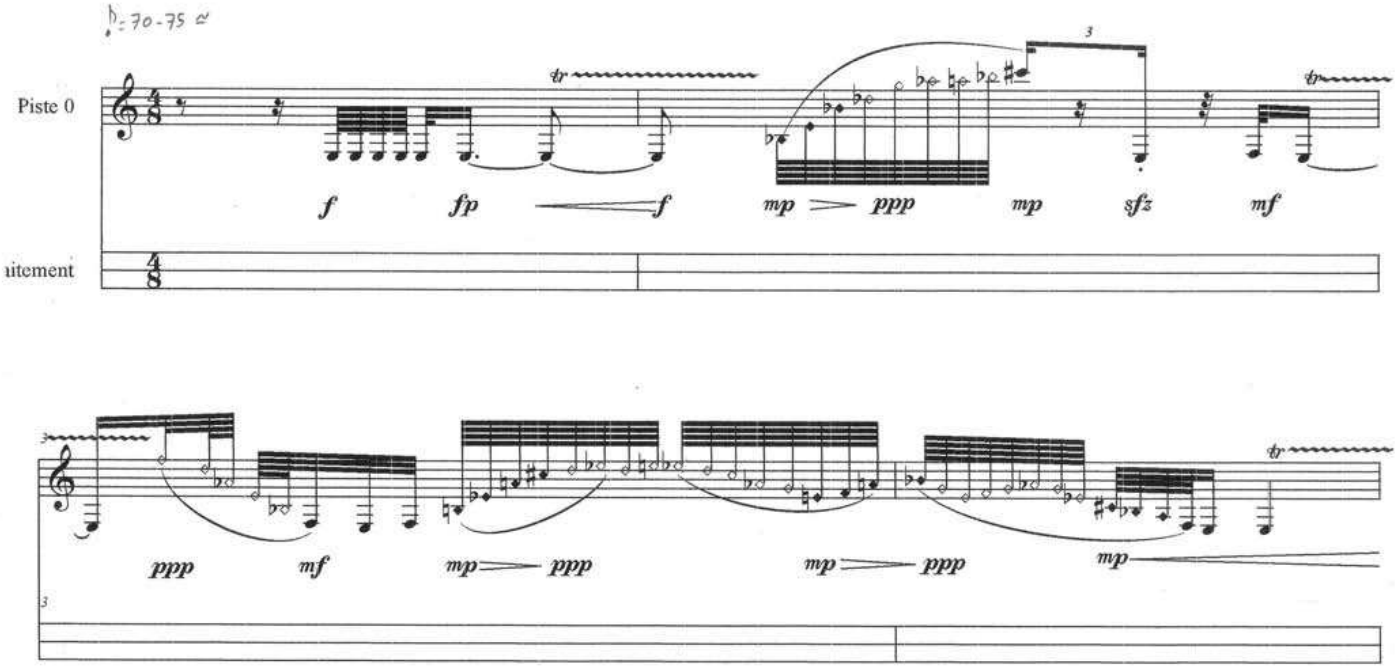
MODULE 10

M.MEZA
2006

♩= 70-75 ≃

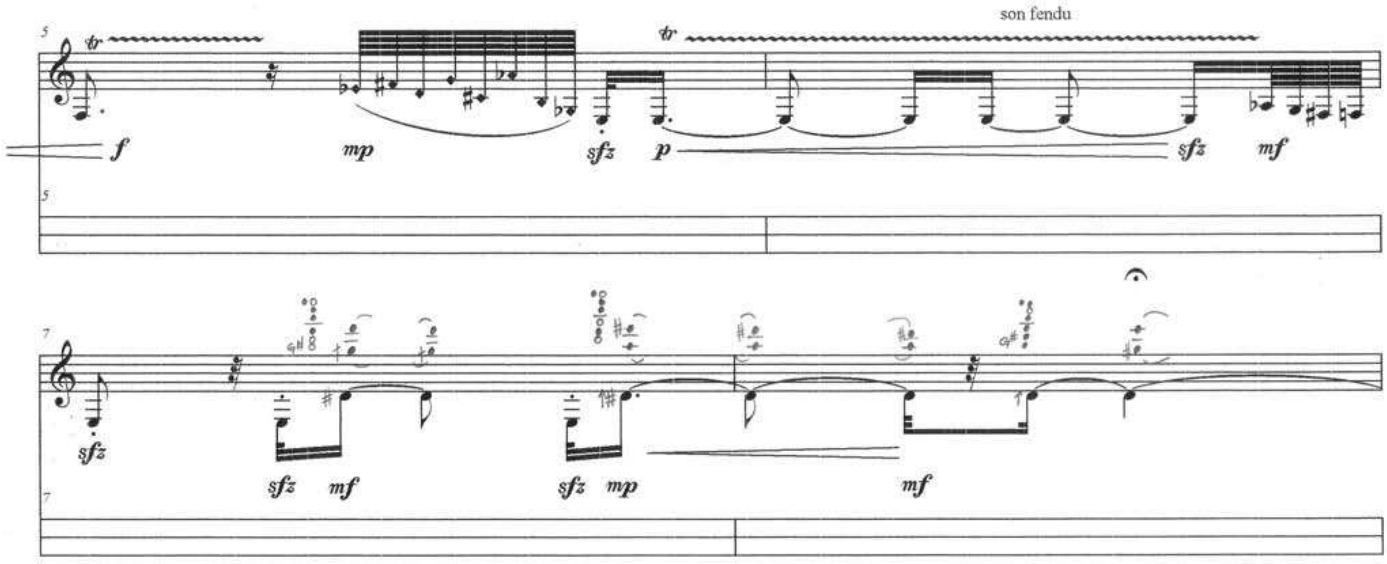
Piste 0

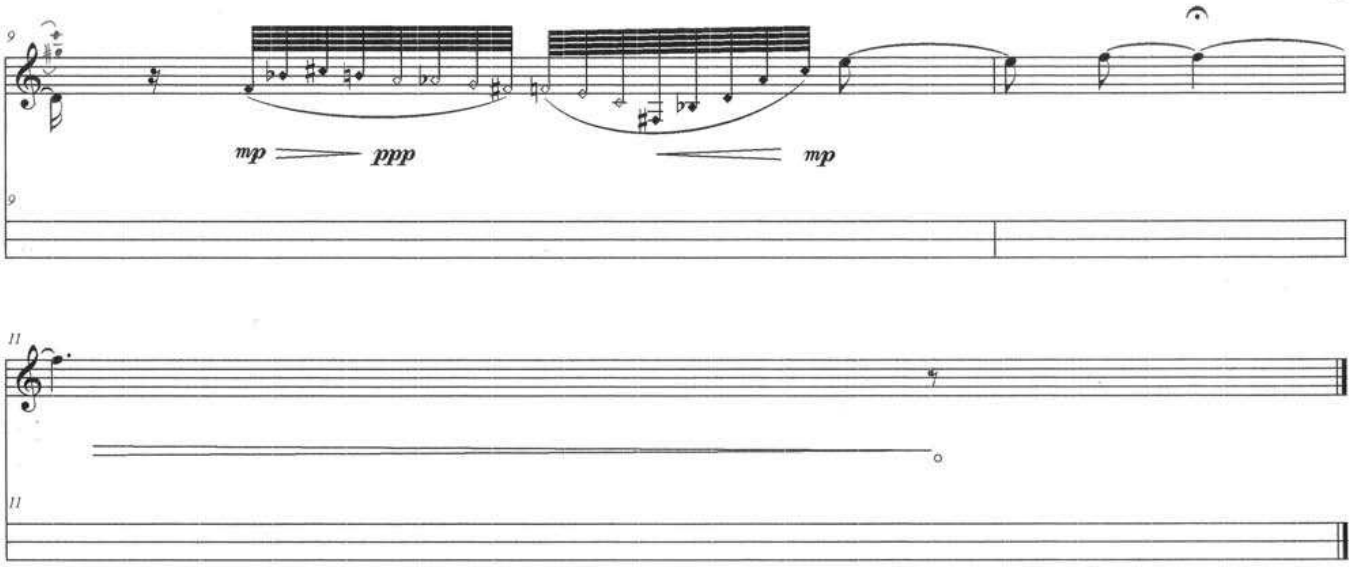
uitement



m 10

2

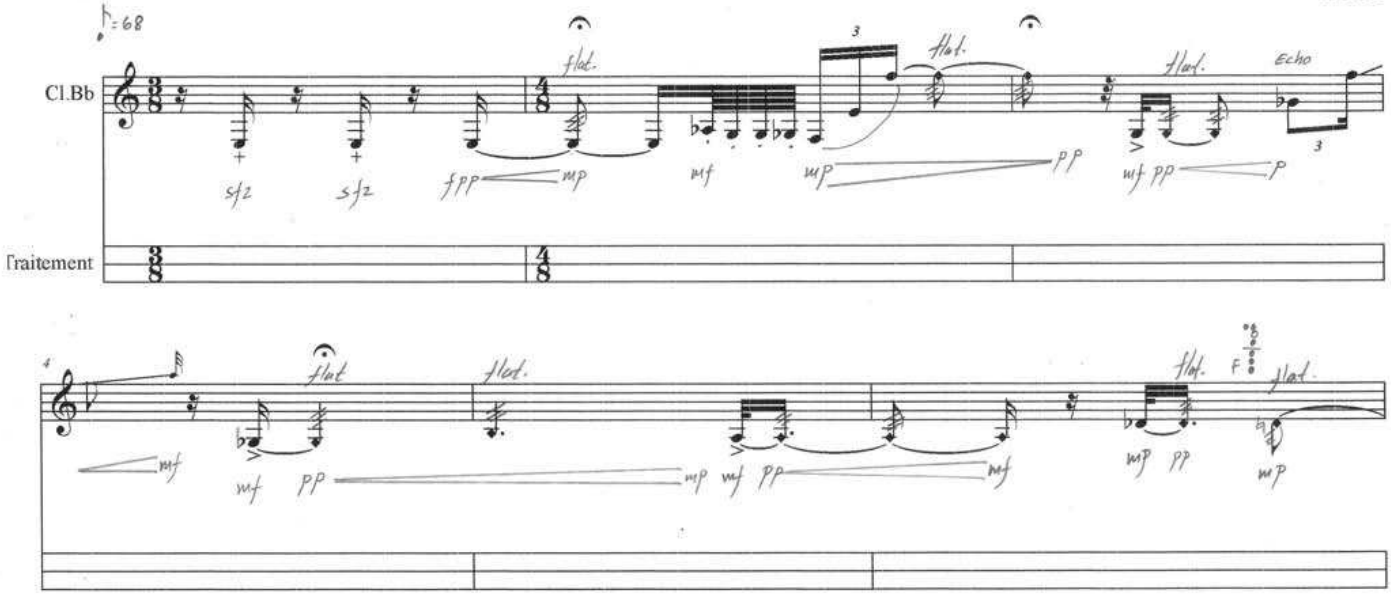



9
mp
ppp
mp
9

11
11

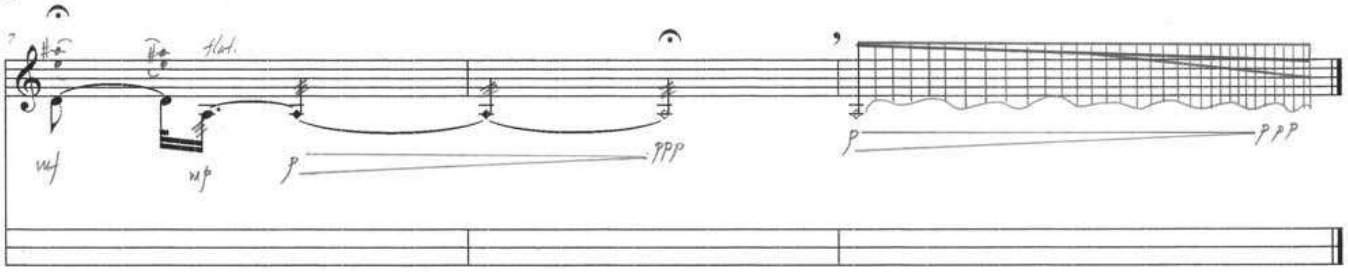
MODULE 11

M.MEZA
2006



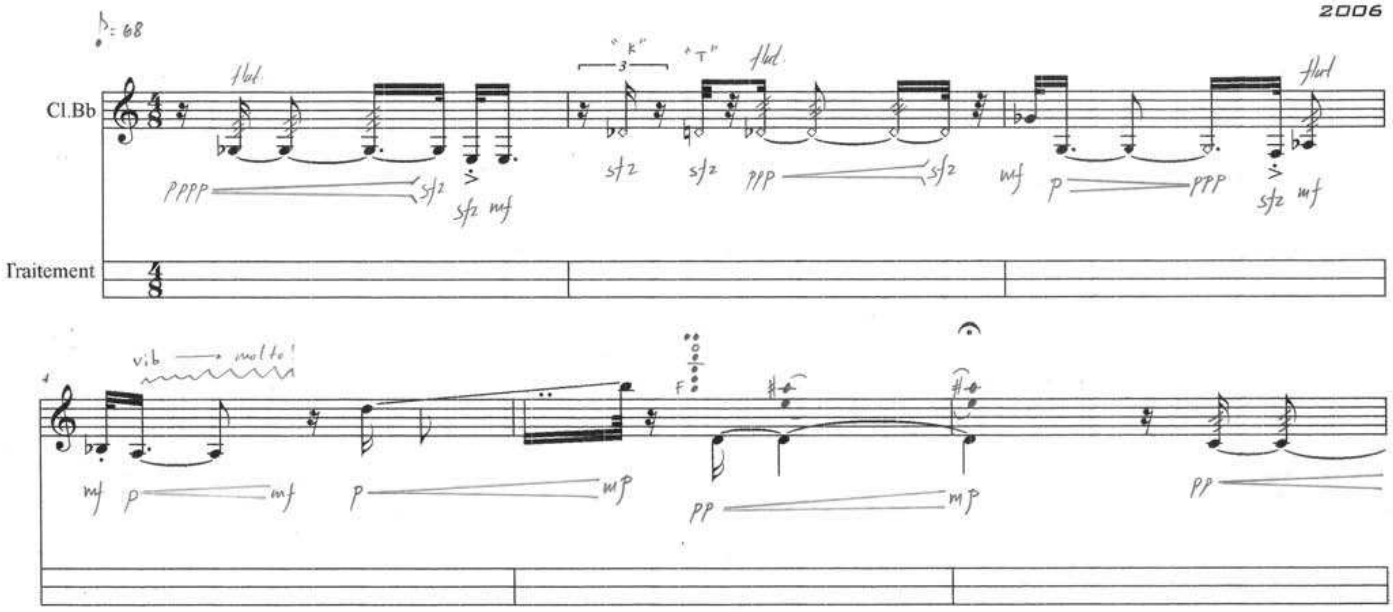
m. 11

2

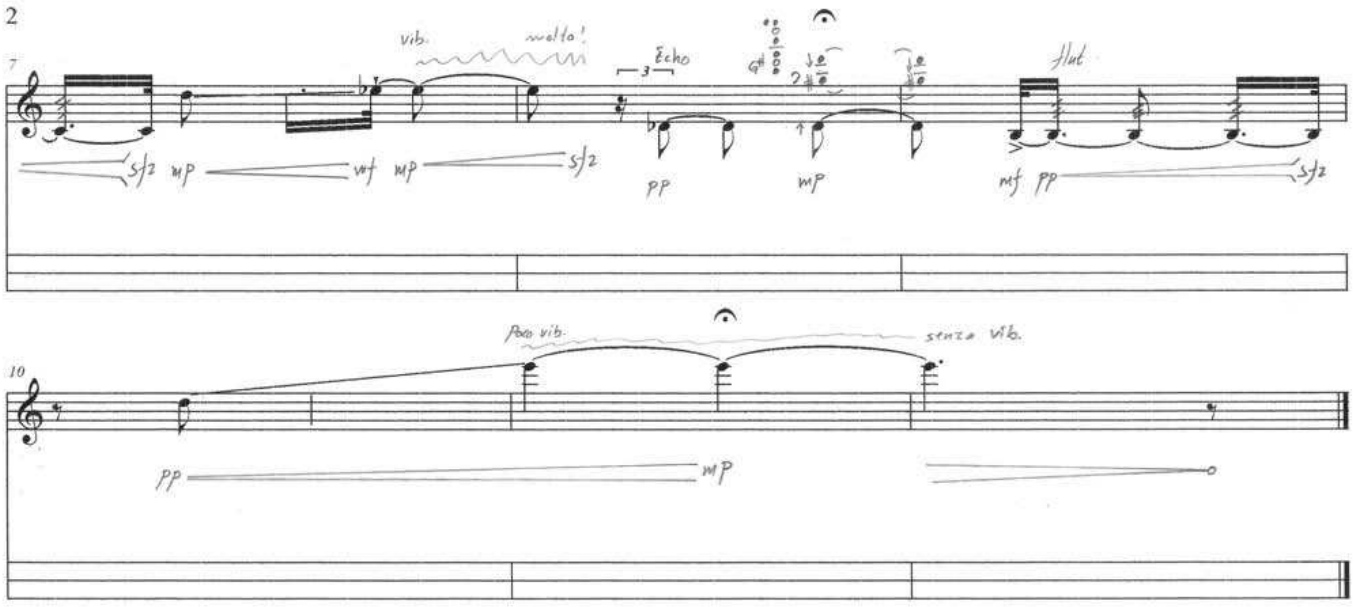


MODULE 12

M.MEZA
2006



m.12



Mots

La passage central de la pièce requière 12 fiches de 4 mots chacune, distribuées à l'interprète. Les originaux préparés par Mauricio utilisent un format A4 confortable par fiches, réduites à un sixième de leur surface pour cette annexe.

1.

Soleil

Gamin

Bleu

Croix

Woes-war-sollichswer-den

Mauricio Meza 2006

2.

Nuit

Rouge

Respiration

Aboyer

Woes-war-sollichswer-den

Mauricio Meza 2006

3.

Nuage

Éclat

Ombre

Arbre

Woes-war-sollichswer-den

Mauricio Meza 2006

4.

Traversée

Tête

Montagne

Machine

Woes-war-sollichswer-den

Mauricio Meza 2006

5.

Lumière

Futur

Gorge

Hallucination

Woes-war-sollichswer-den

Mauricio Meza 2006

6.

Voyage

Sort

Batôn

Chemin

Woes-war-sollichswer-den

Mauricio Meza

7.

Pouvoir Ignition

Abstraction Envol

8.

Anarchie Cascade

Absolu Tremblement

9.

Conscience Lutte

Impertinence Abîme

10.

Superstition Signifié

Renaissance Sobriquet

11.

Vampire Intuition

Espace Revolution

12.

Point Liberté

Rêve Incarnation

E.3 Chop Chich 2 de Pedro Castillo Lara

Ci-après, la partition de la pièce *Chop Chich 2* de Pedro Castillo Lara, pour clarinette basse, sons électroniques et transformation en temps réel. Les pages ont été réduites de moitié ici.

Le numéro des pédales sont indiqués au-dessus de la seconde portée (« Elec. »), encadrés dans des carrés. Ils sont systématiquement associés à une noire qui donne le moment exact du déclenchement, par rapport à la partition de la clarinette basse.

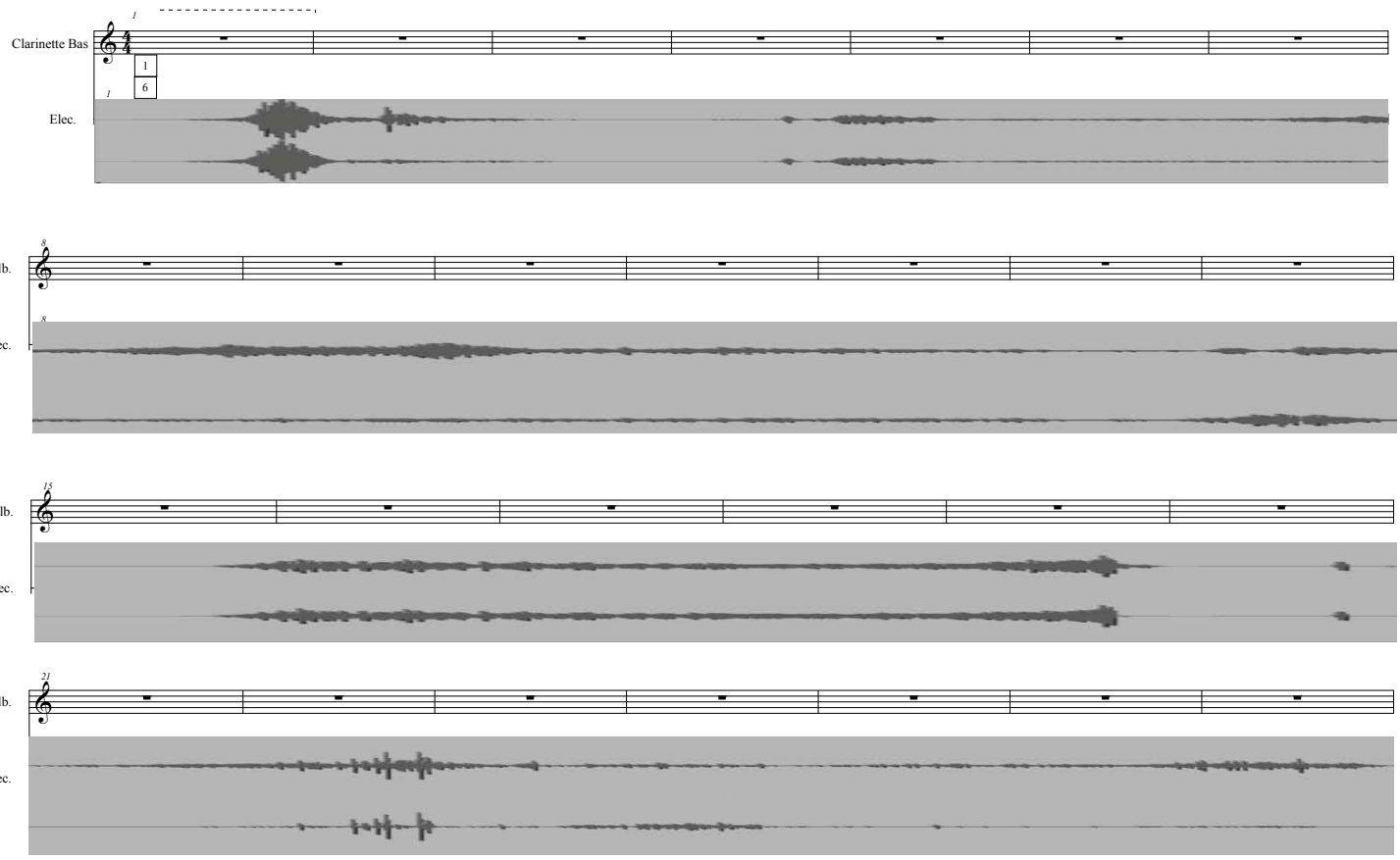
Les numéros de pédale vont par paire, sauf à la fin de la pièce, afin de laisser le choix à l'interprète entre deux comportements du logiciel de Pedro, dont j'ai réalisé l'interfaçage avec le pédalier. Les paires utilisées reprennent l'organisation du pédalier en deux lignes de cinq pédales (de 1 à 5 pour la première ligne, de 6 à 10 pour la seconde), soit les cinq paires suivantes : (1 ; 6), (2 ; 7), (3 ; 8), (4 ;9), et (5 ; 10).

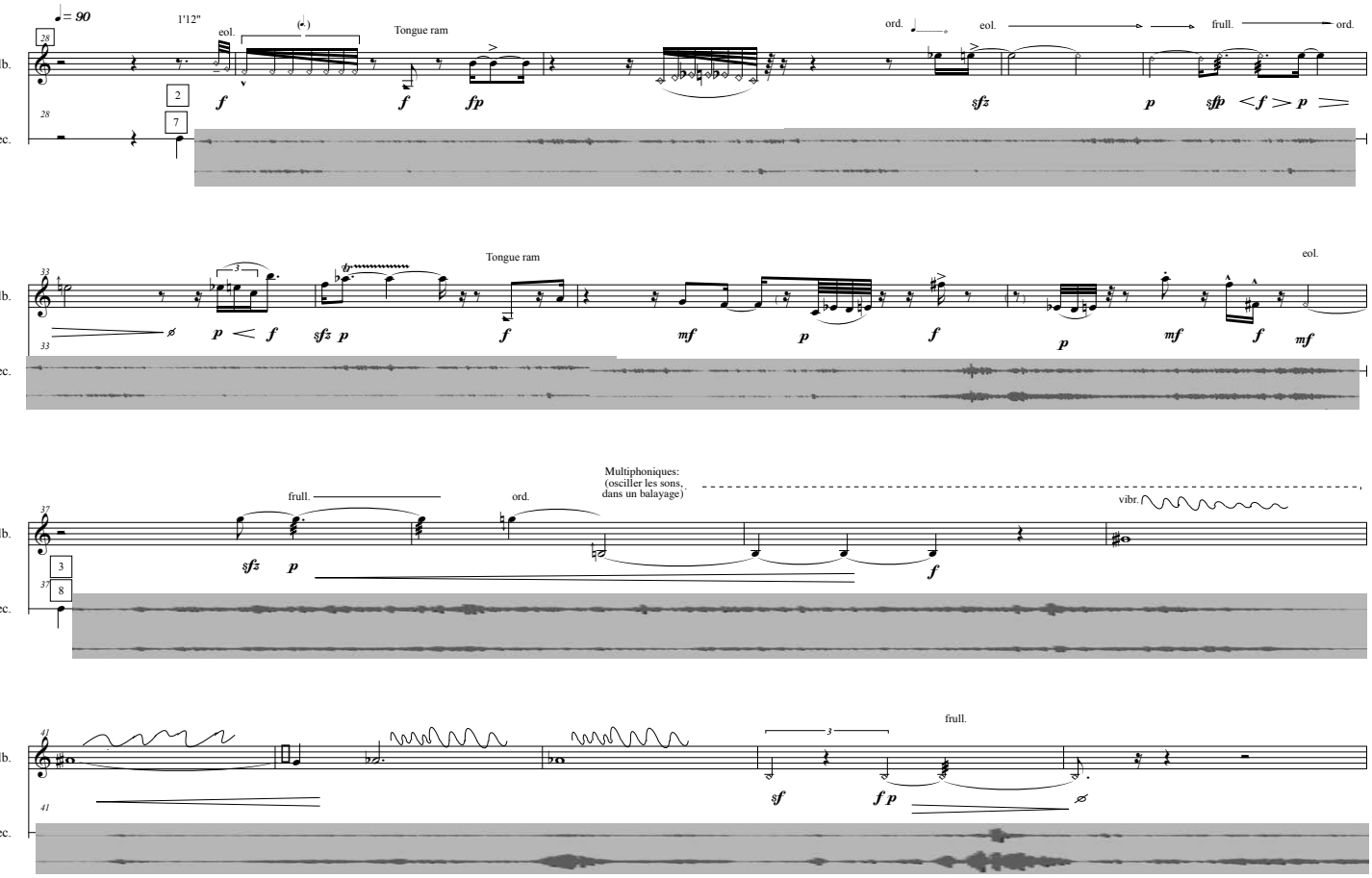
1	2	3	4	5
6	7	8	9	10

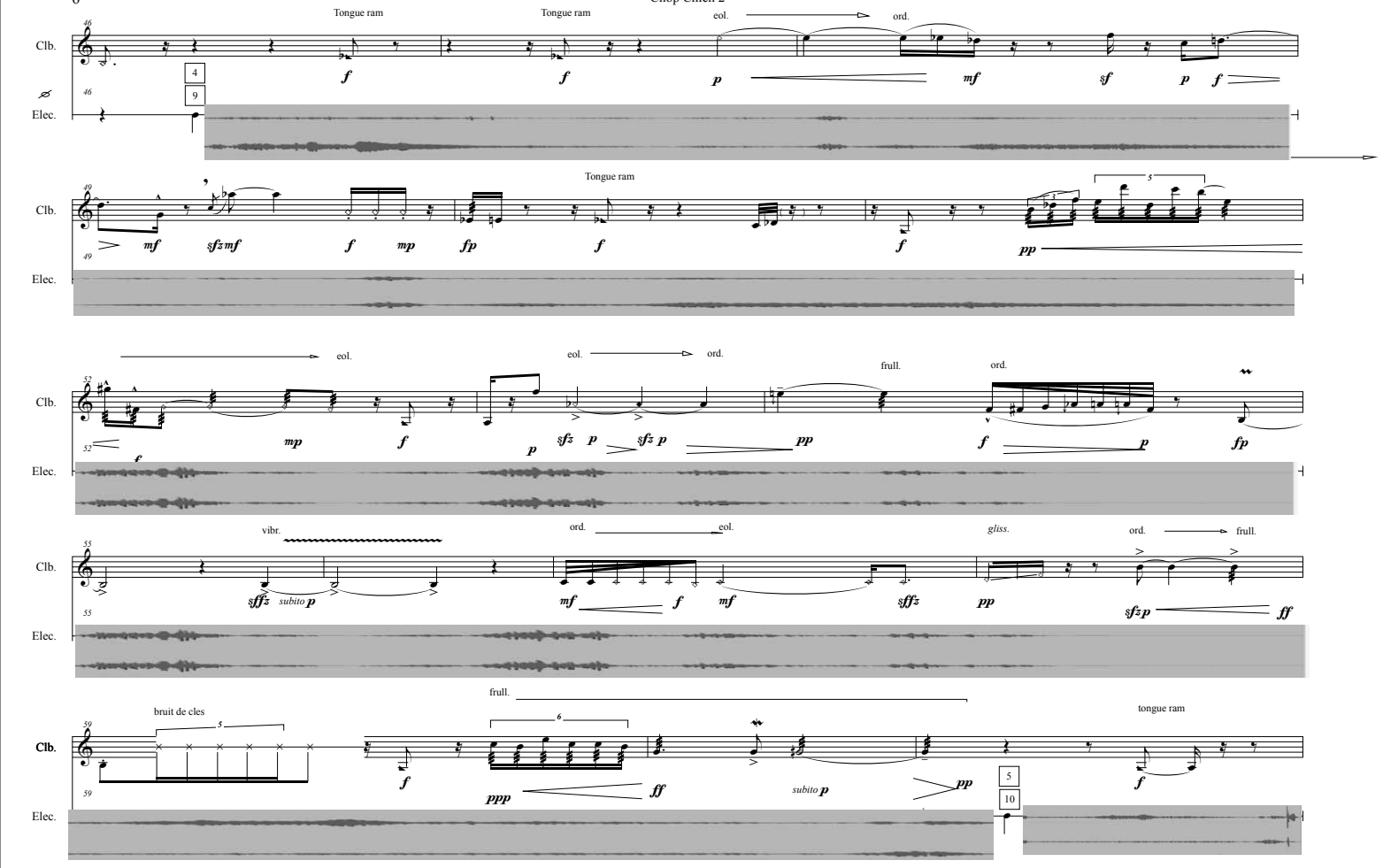
Ainsi, sur 10 déclenchements au total, l'interprète a le choix entre deux comportements différents pour les 8 premiers déclenchements.

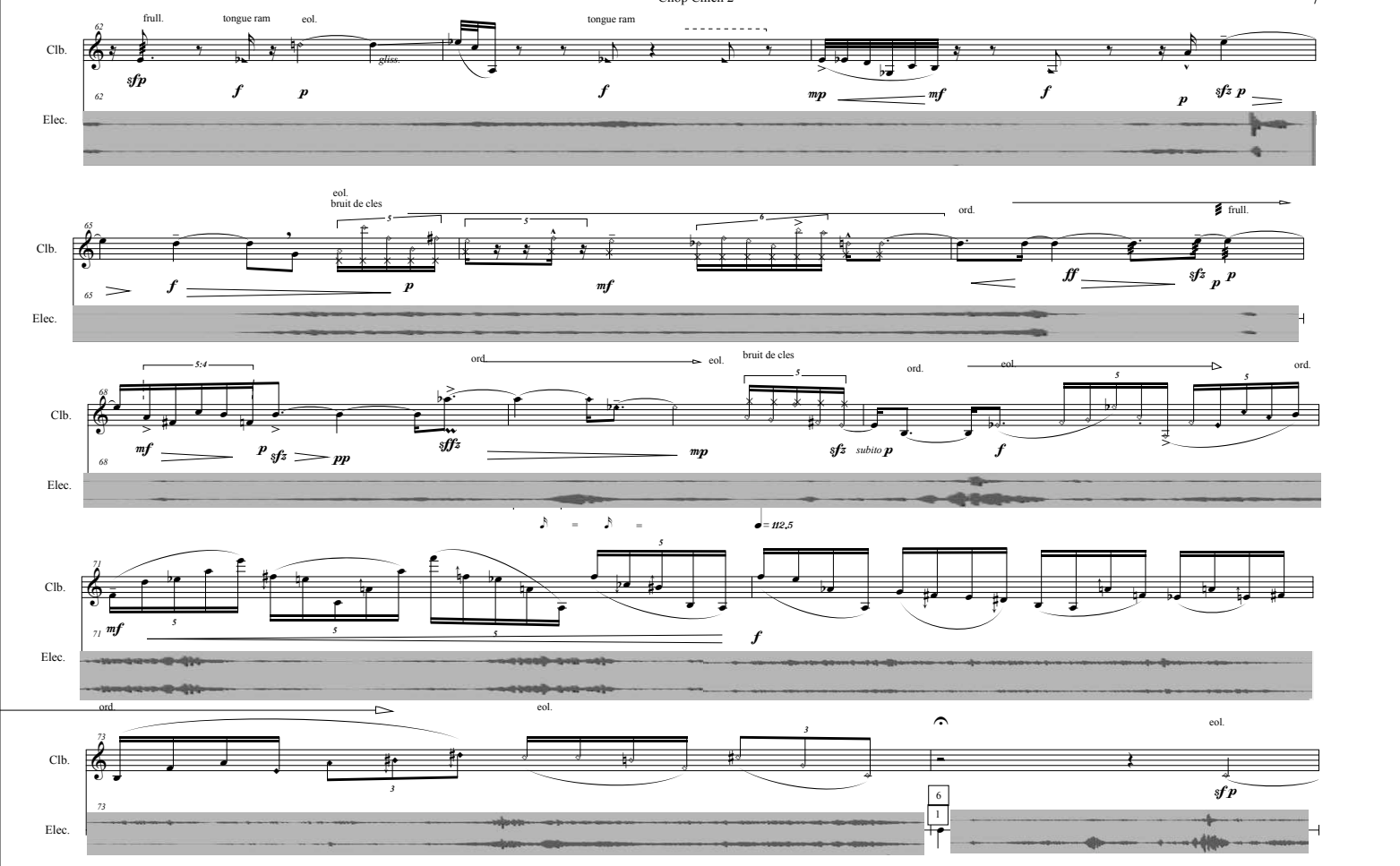
Chop Chich 2

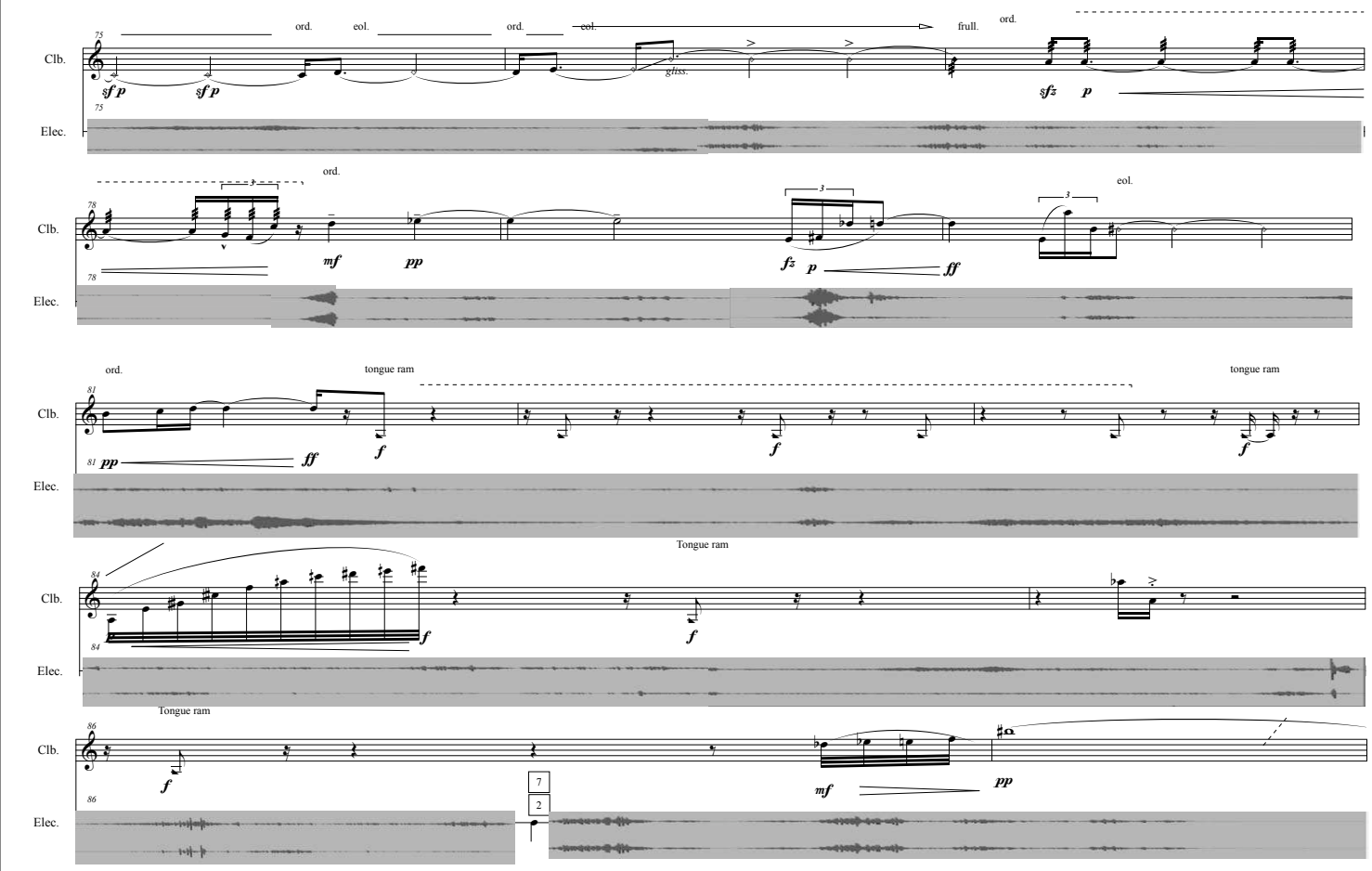
Pedro Castillo Lara

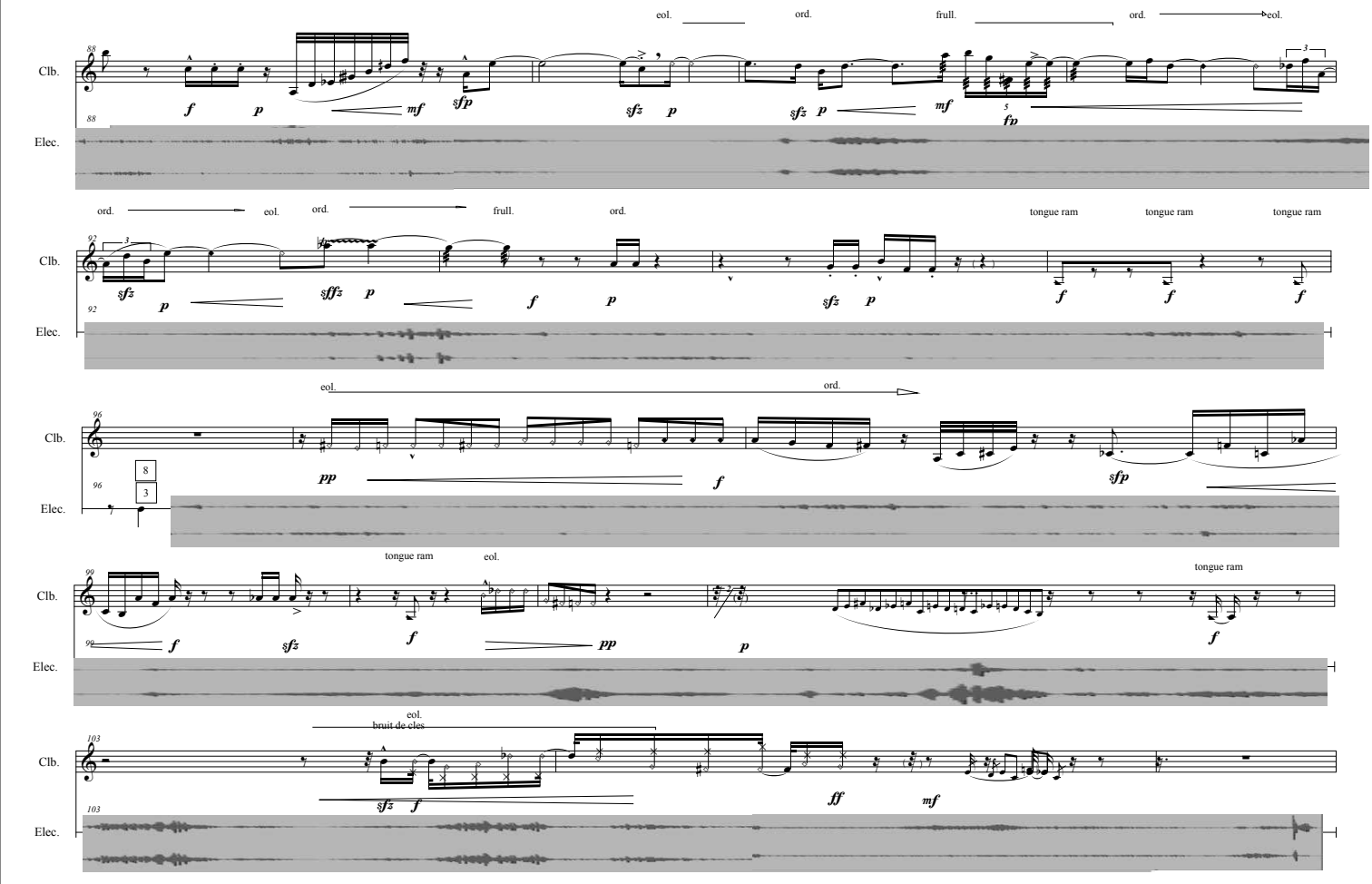


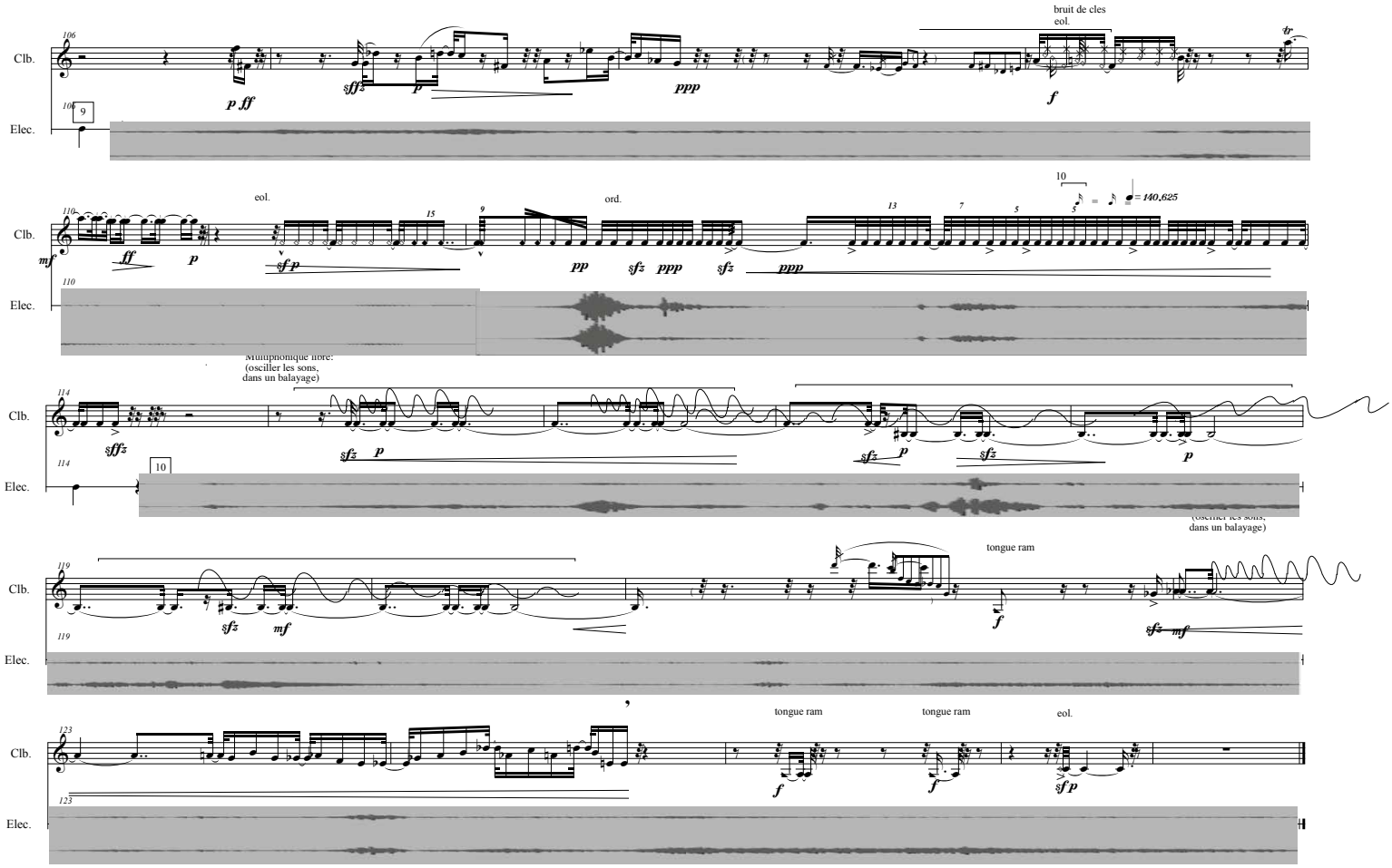


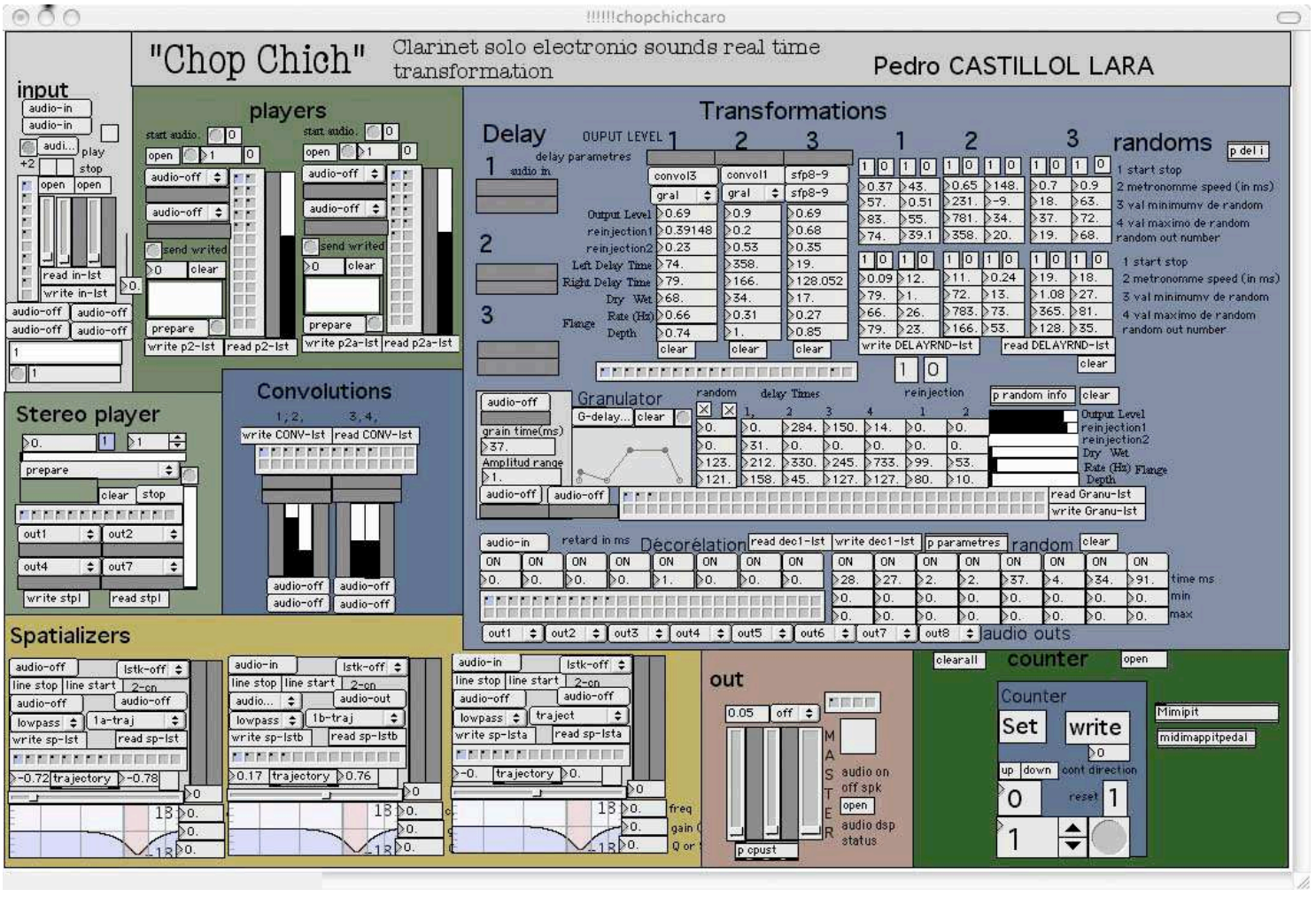

Tongue ram
Tongue ram
eol.
ord.
Clb.
Elec.
Tongue ram
eol.
eol.
ord.
frull.
ord.
vibr.
ord.
eol.
gliss.
ord.
frull.
bruit de cles
frull.
tongue ram


frull.
tongue ram
eol.
tongue ram
Clb.
Elec.
eol.
bruit de cles
ord.
frull.
ord
eol.
bruit de cles
ord.
eol.
ord.
= 112,5
ord.
eol.
eol.


Clb.
Elec.
ord.
eol.
frull.
gliss.
tongue ram
Tongue ram


Clb.
Elec.
ord.
eol.
frull.
tongue ram
bruit de clés


Clb.
Elec.
bruit de cles
eol.
eol.
ord.
= 140,625
Multiphonique libre:
(osciller les sons,
dans un balayage)
tongue ram
tongue ram
tongue ram
eol.


!!!!!!chopchichcaro
"Chop Chich"
Clarinet solo electronic sounds real time transformation
Pedro CASTILLOL LARA
input
players
Transformations
Delay
randoms
Stereo player
Convolutions
Granulator
Décorélation
Spatializers
out
counter
audio outs

Annexe F

Synthèse du cahier des charges de a2m

Les deux documents suivants, écrits par Mauricio Meza, présentent une synthèse du cahier des charges du logiciel a2m.

Cahier des charges : Génération automatique de structures (matériau pré-compositionnel) par le biais d'un outil logiciel de type CAO

Mauricio Meza
Octobre 2006

Ceci constitue une synthèse du cahier de charges qui a été soumis à Karim Barkati, doctorant à l'Université Paris 8, dans le cadre d'une collaboration visant à croiser nos recherches.

Mots-clés : traduction, structure temporelle, structure fréquentielle, MIDI, seuil d'information, saillance morphologique, homothétie.

Objectif du logiciel :

Traduction d'information. Au moyen d'une analyse sonore de type *X,* extraire des informations propres à la nature de l'objet analysé, puis transformer cette information en information symbolique musicale.

Parti pris :

Les objets analysés seront de nature musicale et sonore. Il s'agit de prendre comme modèle l'objet sonore d'une part et le geste instrumental d'autre part. Dans le contexte de la composition musicale nous visons à raccourcir la distance entre processus de création heuristique et processus de création formel. Par conséquent, l'un des usages courants de cette application sera lié à l'analyse des objets sonores issus des nos propres improvisations faites à l'instrument réel.

Démarche à suivre :

1 Sur la base d'un seuil prédéterminé à l'avance et avec les objets de Tristan Jehan (loudness, noisiness, pitch), mettre en place une application dévouée à la détection soit :

du contenu fréquentiel

des pics d'intensité

du contenu du bruit

2 Mettre en place une application capable de traduire les informations issues de l'analyse en données Midi.

Faire une correspondance entre :

Soit la valeur la plus haute par rapport à un *seuil* prédéterminé lors de l'analyse et la valeur en note Midi et durée Midi

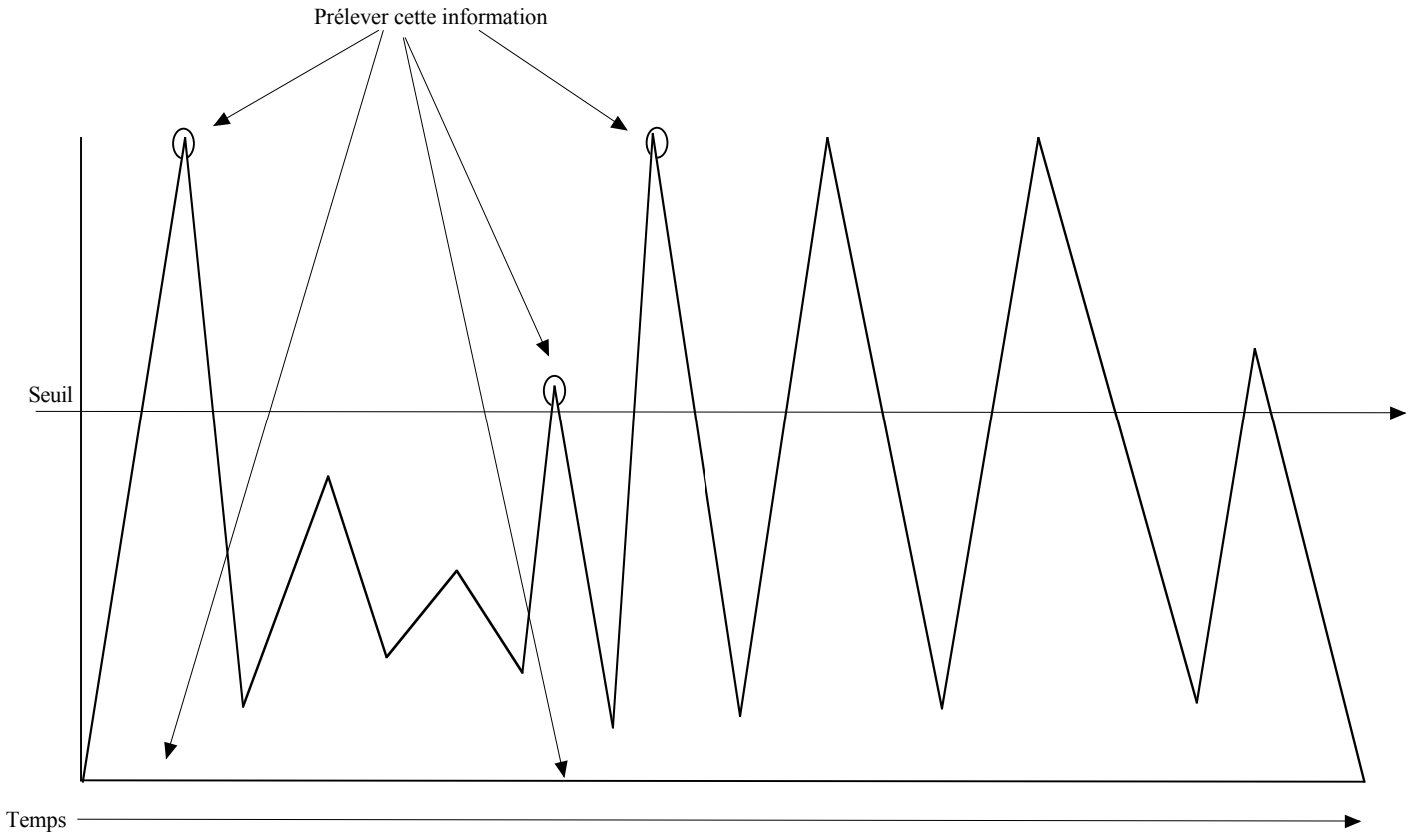
Soit le rapport intervallique entre une valeur *x* et une valeur *y* et la valeur Midi, note et / ou durée.

3 Mettre en place une application dévouée à la transformation des données Midi en symboles musicales.

4 Mettre en place une application dévouée à la transformation des structures issues de ces analyse selon un principe de multiplication / division (homothétie).

Cahier de charges : Génération automatique de structures (matériaux pré-compositionnel) par le biais d'un outil logiciel de type CAO.

ANNEXE 1 Prélèvement de saillances et occurrences temporelles. M. Meza



Annexe G

Journées professionnelles sur le métier de RIM

Nota Bene : ce texte est extrait de la brochure sur les journées de conférence tenues à l'Ircam les 23 et 24 juin 2007. Nous le reproduisons ici parce qu'il risque, en tant que brochure, de rapidement ne plus être trouvable.

Le métier de réalisateur en informatique musicale

Un compositeur, le plus souvent une singularité isolée, se retrouve avec un interlocuteur privilégié – le réalisateur en informatique musicale – au travail dans un studio. Quelles pratiques (temporalités, partage du travail, signature de l'œuvre, intégration des avancées technologiques...) et fonctions s'inventent ainsi ?

Les journées de rencontres internationales dessinent les contours de ce métier émergent, de ses pratiques et de ses formations spécifiques.

L'émergence du métier de réalisateur en informatique musicale (connu avant sous le nom d'assistant musical) au sein de l'Ircam dans les années 80 répondait à un certain nombre de besoins : libérer le chercheur d'une relation trop exclusive avec un compositeur et effectuer la traduction entre les mondes de la musique et de la recherche.

Avec l'augmentation progressive de la quantité des projets de production dans les années 90, le réalisateur devait aussi assumer de façon plus affirmée l'encadrement des compositeurs, la gestion les projets de production et, enfin, assurer, en collaboration avec l'ingénieur du son et le compositeur, l'exécution de l'œuvre. Ces besoins sont-ils toujours d'actualité ?

Le réalisateur en informatique musicale est-il une exclusivité de l'Ircam ? Vraisemblablement pas, puisque aujourd'hui dans tous les milieux où des créateurs travaillent avec des nouvelles technologies pour le son ou la musique – danse, théâtre, image et arts plastiques, musique –, on trouve, sous d'autres étiquettes – régisseur, ingénieur, etc. –, des professionnels qui maîtrisent des concepts, des technologies et des pratiques similaires.

En revanche, ce qui n'existe pas actuellement, serait une identité professionnelle partagée, une reconnaissance publique du métier et, conséquemment, une filière de formation

à même de garantir l'acquisition des connaissances et des compétences musicales et techniques nécessaires à l'exercice de ce nouveau métier.

Ces journées se proposent d'esquisser les contours de cette nouvelle profession dans ses différentes formes, pour ensuite aborder la question de la formation. Par ailleurs, une confrontation des méthodes de travail et pratiques professionnelles permettra d'envisager des formes de collaboration entre les diverses institutions (transmission des nouveautés technologiques, des moyens de production . . .).

* * *

Intervenants : Marc Battier, Cyril Béros, Peter Boehm, Andreas Breitscheidt, Éric Daubresse, Andrew Gerzso, Wolfgang Heiniger, Christophe Lebreton, Serge Lemouton, Mikhaïl Malt, Yan Maresz, Tom Mays, Olivier Pasquet, Manuel Poletti, André Richard, Rand Steiger, Daniel Zaley.