Université Paris 8 Vincennes à Saint-Denis

UFR ARTS, PHILOSOPHIE, ESTHETIQUE

Département de Musique

TRAITEMENTS TEMPS REEL ET ECRITURE

Vers un lexique musical de divers traitements sonores de base.

Tom MAYS

Mémoire de Master 2
sous la direction de
M. Horacio VAGGIONE

2009-2010

CD audio de la pièce :

Le patch bien tempéré numéro 1 – pour vibraphone et traitement temps réel

Enregistrée au concert de l'Atelier de Composition de José Manuel Lopez Lopez, le 25 juin 2010 à l'Institut Cervantes à Paris, jouée par le percussionniste Rémi Durupt.

Table de Matières

Remerciements

Au stade de ce mémoire j'aimerais simplement remercier mes professeurs à Paris 8 qui sont tous si engagés et passionnés – Horacio Vaggione, Anne Sedes, José Manuel Lopez Lopez et Michel Borne; ainsi que Francis Faber, Cort Lippe, Georges Gagneré et Yann Geslin qui ont été si volontaires pour relire; et mes collègues qui ont voulu participer si vivement aux réflexions – Eric Daubresse, Carl Faia, Serge Lemouton et Luis Naón (parmi d'autres).

Introduction

Le domaine du temps réel est extrêmement riche en recherches et développements des nouvelles techniques de traitement, des synthèses et des interfaces homme-machine, ainsi que des nouveaux environnements et plate-formes informatiques permettant de réaliser des traitements temps réel avec de plus en plus de certitude et de stabilité.

Cependant, nous croyons que l'heure est arrivée de se focaliser sur les aspects **musicaux** des traitements temps réel et sur leurs **écritures**. Quels sont les principes fondamentaux qui se dégagent de cette trentaine d'années de musique et traitements temps réel ? Quels sont les traitements *incontournables* et quels sont leurs comportements musicaux ?

Dans la première partie de ce mémoire, nous allons parler du temps réel en général, puis nous allons établir une liste de traitements et de catégories de traitements de base (principalement d'après notre expérience, mais en prenant aussi en compte les idées d'autres spécialistes dans le milieu) et ensuite nous travaillerons sur l'identification des fonctions musicales et des exemples de leurs utilisations – connues et/ou potentielles. Ensuite, nous choisirons la première catégorie (le temps) et nous l'explorerons profondément avec un grand chapitre, notamment en parlant du délai. Nous parlerons de son histoire, de ses caractéristiques et qualités, et des techniques musicales que nous pouvons employer.

La deuxième partie de ce mémoire concerne l'environnement que nous avons créé avec MaxMSP pour faciliter la création temps réel – *tapemovie*. Il y aura d'abord une discussion technique à propos de l'environnement, et ensuite un ensemble d'exemples d'utilisations, dans le style d'un tutoriel. Ce projet est un élément clé dans nos recherches autour des applications musicales des traitements temps réel.

En troisième et dernière partie, il y aura la présentation d'une pièce composée cette année dans le cadre de nos recherches temps réel, la première pièce d'une suite prévue pour le travail de doctorat. C'est une courte pièce pour vibraphone et temps réel, utilisant uniquement des délais – *Le patch bien tempéré numéro 1*. L'analyse de la pièce sera faite avec une attention particulière mise sur les utilisations musicales des délais.

Nous avons entamé ces recherches dans des cours au sein de la classe de composition au Conservatoire National Supérieur de Musique où nous explorons l'écriture d'études sur des principes de traitement temps réel – en commençant avec les plus simples (délai, modulation en anneau, enregistrement / relecture, freeze, etc…) et en insistant sur les fonctions musicales des transformations. Les résultats avec les étudiants sont très prometteurs. Ils commencent à se rendre compte que la bonne compréhension musicale de quelques transformations très simples permet une maîtrise du temps réel qui n'est pas possible autrement.

Notre souhait est d'approfondir ces recherches dans le cadre d'un doctorat, le but à long terme étant d'écrire un livre.

I Les traitements et la musique

1 Le temps réel sonore et les traitements de base

Dans ce chapitre, nous allons d'abord parler du temps réel en général, de sa place historique face au temps différé d'un côté et aux instruments analogique/électronique « live » de l'autre, et de ses avantages, inconvénients et particularités. Ensuite nous allons proposer une liste de catégories du traitement sonore de base avec une brève description de chacune et des traitements concernés que nous estimons constituer une *base* des plus utilisées. En troisième partie de ce chapitre, nous allons présenter les traitements de *contrôle*, les analyses du son donnant lieu à l'extraction de paramètres sonores et musicaux qui sont destinés à la gestion des paramètres des traitements sonores. L'élément de *contrôle* est essentiel pour parler de la création musicale dans les chapitres de chaque catégorie de traitement qui vont suivre.

1.1 Le monde particulier du temps réel

1.1.1 Historique

Avant de parler des aspects musicaux des traitements temps réel, il faudrait bien clarifier les termes et considérer l'historique.

Le terme *temps réel* s'applique à l'informatique qui calcule des traitements de données tellement rapidement que le temps nécessaire pour obtenir le résultat est imperceptible pour l'être humain (ou presque). Avec l'arrivée des premiers ordinateurs temps réel dans les années 60, il a fallu le différencier de l'ancien système informatique qui a pris le nom, par conséquence, de temps différé. Karim Barkati explore en profondeur les implications du temps réel et du temps différé dans sa thèse « *Entre temps réel et temps différé* ». Ici il parle des termes et de la relation à l'informatique :

> « 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. »[1]

L'arrivée du temps réel dans l'informatique musicale (premières expériences dans les années 70 et présence indiscutable dans les années 80) a instauré une séparation en deux grands axes, deux manières de travailler, voir même deux esthétiques : la musique pour instruments et traitements informatiques exécutés et contrôlés en temps réel, et la musique pour *support* (bande ou bande + instrument). Cette explication est simpliste, mais a l'avantage de pointer l'essentiel. Luc Rondeleux, dans un article intitulé « *Une histoire de l'informatique musicale, entre macroforme et microcomposition* », place l'arrivée du temps réel dans le contexte des formalistes pour qui ce nouvel outil, bien qu'excitant, ne pouvait pas satisfaire les besoins du contrôle des plus minutieux :

> « 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. »[2]

Aujourd'hui, nous nous interrogeons toujours sur les avantages et les inconvénients des deux mondes[3], mais il y a beaucoup de croisements et de transversalité: une composition travaillée en studio en temps différé utilise souvent des traitements temps réels pour générer de la matière. Une pièce temps réel va très souvent incorporer des sons ou des séquences qui ont été composés en studio en temps différé.

1.1.2 Avantages et inconvénients

Le traitement sonore par informatique en temps réel fait certaines choses très bien, et d'autres moins bien voir pas du tout. Par exemple, le temps réel, par sa nature, va toujours refléter les variations de la source dans le résultat – si l'instrumentiste joue différemment, le résultat est différent. Cette capacité rend le temps réel bien adapté à l'improvisation, mais donne aussi de la souplesse à la musique écrite. Le temps réel peut aussi permettre de varier la vitesse de

1 Barkati 2009
2 Rondeleux 1999
3 Pour plus de discusion sur ce thème, cf. Barkati 2009

déroulement d'une pièce, allant jusqu'à pouvoir suivre le jeu instrumental et avancer en fonction d'une connaissance préalable de la partition. Il permet également de relier les paramètres de traitement à des contrôleurs extérieurs ou à des paramètres extraits d'un jeu instrumental, créant ainsi des dispositifs interactifs et même des *instruments de traitement* qui peuvent, à leur tour, trouver une place dans des partitions futures. Le temps réel c'est la *réactivité*, l'*interactivité*, l'*instrumentalité*, l'*instantanéité*, et le rapport au *geste* et au corps humain.

Quant à ce que le temps réel fait moins bien ou pas du tout, il y a d'abord le grand inconvénient lié à son gros avantage : le temps. Le temps réel reste collé au temps. Il ne peut pas traiter un son AVANT que ce son n'ait existé, et il ne peut pas non plus le traiter plus vite que son déroulement temporel. Nous sommes ainsi « prisonniers » d'un rapport *causal* entre *source* et *traitement*. Par ailleurs, si la source sonore à traiter est un instrument acoustique ou une voix, il y a de fortes chances que l'instrumentiste qui le joue soit présent dans le lieu du concert, et par conséquence, le son direct aussi. Puisque les traitements temps réels en situation de concert contiennent quasiment toujours une part de son direct qui risque de masquer une partie du son traité, il faut toujours penser à une *séparation* avec la source vis a vis d'au moins un des paramètres sonores : le temps, la hauteur, le timbre ou l'espace. Un autre *inconvénient* du temps réel c'est la limite du nombre de paramètres disponibles en même temps et la difficulté d'exécuter un contrôle extrêmement fin sur beaucoup de paramètres de façon déterministe, comme explique Rondeleux plus haut[1]. Nous allons rajouter à cette liste les difficultés techniques liées à l'exécution d'une pièce temps réel, et les problèmes de pérennité des oeuvres dus à la rapidité d'évolution de l'informatique, à la fois le *hardware* et le *software*.

Nous proposons que tous ces *inconvénients* puissent devenir même des *qualités* si nous les connaissons, les acceptons, et les maitrisons assez bien.

1.1.3 Le rapport aux instruments électroniques non-informatiques

Quel est le rapport entre les instruments informatiques temps réel et les instruments ou

1 Bien que la rapidité et la complexité de l'informatique aujourd'hui, surtout reliée avec les interfaces gestuelles très fines, permettent une qualité et une sensibilité sonore non-imaginable dans les premières années du temps réel, cet *inconvénient* du temps réel a eu une grande importance historique.

appareils analogiques « traditionnels » qui ont beaucoup servi en concert, et qui font parfois des choses que nous ne faisons que REFAIRE avec les moyens informatiques d'aujourd'hui? Et quelle est leur place dans le discours musical que nous allons créer?

Bien que certaines utilisations du temps réel puissent être considérés comme des émulations de techniques analogiques, la malléabilité, la souplesse, le contrôle fin et l'interactivité rendus possibles par les implémentations temps réel ouvrent de véritables nouvelles possibilités de discours. Nous avons, cependant, beaucoup à apprendre des techniques analogiques, et des compositeurs et musiciens qui s'en sont servis et qui s'en servent encore. Le « temps » dégrade la mémoire et le « temps réel » fait croire parfois que tout est nouveau et tout est beau, mais d'autres ont bâti le chemin avant nous.

C'est pour cette raison que nous n'allons pas hésiter à puiser dans des exemples d'utilisations musicales à base d'instruments analogiques dans nos explorations des possibilités d'écriture et de jeu avec les différents traitements – quand ces utilisations sont pertinentes. Nous allons cependant toujours proposer des implémentations et des solutions du domaine du temps réel.

1.2 Les traitements temps réel de base par catégorie

1.2.1 La naissance d'une liste

Présenter tous les traitements sonores temps réel possibles aujourd'hui et explorer les techniques musicales de chacun est une gageure et un travail impossible à mettre à jour puisque la recherche en informatique musicale en invente régulièrement. Nous avons dû opérer un choix.

Après plus de seize années d'expérience en programmation du traitement son temps réel appliqué à la musique, nous avons pu constater une récurrence évidente de certains traitements qui semblent former une base incontournable dans la panoplie du possible, un ensemble de traitements bien connus et bien utilisés dans la composition et l'improvisation depuis parfois fort longtemps. Nous pensons que l'utilisation de ces traitements est arrivée à une certaine maturité qui nous permet de constater les tendances et de rassembler en un seul endroit l'essence de leurs applications musicales.

L'inventaire qui suit est donc le début d'un travail de constat et d'exploration qui forme ce mémoire et qui continuera dans le travail de doctorat. Cette liste est construite avec une certaine subjectivité, puisque qu'elle est issue en grande partie d'expériences personnelles (compositions, réalisations, cours) mais les avis d'autres experts dans le milieu confirment que l'essentiel y est présent. Nous espérons que la présentation et l'exploration de ces techniques de cette manière permettra, justement, d'aller beaucoup plus loin ensuite.

1.2.2 La liste des traitements par catégorie

Voici les traitements de base séparés en catégories qui, malgré un certain arbitraire personnel, pointe vers des applications musicales bien distinctes et permet d'en parler clairement. Ici, nous ne faisons qu'une brève description globale, puisque chacun fera le sujet d'un grand chapitre – soit dans ce mémoire, soit dans la thèse du doctorat qui va suivre.

Ils sont:

- Le traitement du temps
- Le traitement de la hauteur
- Le traitement de l'amplitude
- Le traitement du timbre par filtrage
- Le traitement du timbre par modulation
- Le traitement de l'espace
- Le traitement de la matière
- Le traitement dans le domaine fréquentiel

1.2.2.1 Le traitement du temps

Le traitement du *temps* est pour nous le point de départ par excellence, puisque le temps est nécessaire pour toute perception, et le traitement du temps est compris dans la quasi-totalité des autres traitements. L'amplitude, la fréquence, le timbre, le rythme, la forme, l'espace et la mémoire ont tous besoin du temps pour exister ou pour être détectés. Les « multiples échelles du temps » du compositeur Horacio Vaggione, constituent également un témoin de l'importance du temps et du traitement du temps dans la musique. Au sujet de ces échelles,

Anne Sedes dit dans son article *A propos du temps dans la musique d'Horacio Vaggione*[1] :

> « Elles peuvent être multiples, hétérogènes, disjointes, discontinues entre elles. Au sein d'un réseau d'objets, la composition va consister à articuler, à opérer entre ces niveaux des interactions, des réciprocités, des connexions, en quelque sorte à faire émerger et développer des propriétés porteuses de formes, des saillances, des parties, de moments et à qualifier le champ multiple du sonore comme le domaine à part entière de la composition musicale. »

Comment réaliser un traitement du temps et surtout comment le faire en *temps réel* ? Puisque nous devons respecter *l'ordre* du temps, ne pouvant pas faire entendre *l'effet* avant *la cause*, nous ne pouvons que *retarder* les événements sonores, les retenir un certain temps avant de les faire entendre. Nous faisons cela soit par une *ligne à retard* ou plus simplement un *délai* (décalage temporel prédéfini, volatile, limité dans le temps), soit par *enregistrement et relecture* (décalage temporel déconnecté du temps, non-volatile, théoriquement non-limité dans le temps).

Le délai est le plus courant et le plus simple à appliquer, bien que beaucoup plus puissant et complexe que nous pouvons imaginer au premier abord. Il peut agir sur l'espace, le timbre, le rythme, la forme, et même la hauteur. Il a une très longue histoire. Il a vu le jour comme un phénomène physique (l'écho) avant de se retrouver en tant que principe appliqué à la musique (le canon). Il a été recréé très habilement par les enregistreurs de bandes magnétiques, puis s'est fait numérisé et réincarné dans la mémoire de nos ordinateurs.

La technique d'*enregistrement/lecture* comme manière de décaler le temps permet de sortir d'un écoulement de temps prédéfini. Nous pouvons enregistrer un son sur une bande, sur un disque dur, ou bien dans la mémoire vive, puis le rejouer quand nous le voulons, après quelques millisecondes ou bien après quelques années – pourvu qu'il n'ait pas été effacé. Cette technique peut fonctionner sur des sons, des phrases ou des objets musicaux limité dans le temps, mais ne peut pas s'appliquer *en continu* comme le peut un délai.

En conclusion, le délai crée un décalage temporel limité sur une source sonore non limitée en durée, alors que la technique d'*enregistrement/relecture* crée un décalage temporel non-limité sur une source sonore limité en durée. Nous allons présenter en profondeur les caractéristiques et applications musicales de ces deux moyens de décalage temporel dans le

1 Sedes 2007, page 89

chapitre *Le traitement du temps*.

1.2.2.2 Le traitement de la hauteur

En temps réel, un traitement de hauteur s'appelle un *harmoniseur*, un *pitch shifter* ou simplement une *transposition de hauteur*. L'harmoniseur temps réel est construit avec un *délai à temps variable* en utilisant un principe proche de l'effet Doppler – quand le temps de délai se réduit progressivement, le résultat devient plus aigu, et quand le temps de délai s'augmente progressivement, le résultat devient plus grave. L'harmoniseur utilise deux délais variables en alternance qui se re-combinent par fenêtrage pour reconstituer un son continu avec une certaine transposition – l'enjeux technique étant de changer la hauteur sans modifier la durée du son. Le principe de l'harmoniseur existait avant l'ère du numérique temps réel par l'utilisation des magnétophones à bandes avec des têtes rotatives qui permettaient une transposition de la hauteur sans changer la durée, avec son contraire, un changement de durée ou de vitesse *sans* transposition. Des bons exemples historiques de ces fonctions se trouvent dans le *Phonogène Universal* du G.R.M à Paris[1], ou bien le *Tempophon* de Dennis Gabor[2].

L'utilisation musicale est, au départ très simple – nous jouons une note et l'harmoniseur génère une transposition qui sonne en même temps que la note *source* créant une certaine harmonie – ce qui explique l'origine du mot *harmoniseur*. Dans la composition et l'improvisation, nous pouvons rendre cet effet bien plus intéressant par le biais d'un contrôle fin et souple d'après des données extérieures ou d'après les caractéristiques du son source lui même (Cf. supra *Les analyses sonores et le contrôle des paramètres*).

1.2.2.3 Le traitement de l'amplitude

La troisième catégorie de traitement est celui de l'*amplitude* ou la *dynamique* du son. Nous pouvons considérer que le traitement d'amplitude est à la base une simple amplification sonore, mais que l'intérêt musical vient des possibilités du *contrôle* de cette amplitude : les *enveloppes* et les *modulations cycliques à basse fréquence* (lfo). Dans les deux cas nous

1 cf. text sur Pierre Shaeffer de Pierre Couprie sur le site internet de Leonardo / OLATS (http://www.olats.org/pionniers/pp/schaeffer/theorieSchaeffer.php), et la page Wikipedia en anglais sur la Musique Concrète (http://en.wikipedia.org/wiki/Musique_concrète), et la discussion détaillé du Phonogène du Jacques Poullin dans La revue Ars Sonora Numéro 08 (http://www.ars-sonora.org/html/numeros/numero09/09f.htm)

2 cf. Roads 1996, pp. 441-442

sortons du domaine de la *dynamique* seule pour intervenir dans le domaine de l'événement sonore et du rythme.

Le traitement d'amplitude serait le plus ancien de tous les traitements, sans doute parce qu'il existe comme phénomène acoustique simple dans la nature, mais aussi parce qu'il est parmi les paramètres sonores le plus simple à être manipulé par les premiers appareils et instruments électroniques, comme les consoles de mixage et les synthétiseurs analogiques.

1.2.2.4 Le traitement du timbre par filtrage

La méthode la plus habituelle de traiter le timbre est le filtrage, ce qui fait amplifier, atténuer, ou résonner certaines fréquences ou bandes de fréquences du spectre sonore. Cette catégorie représente les filtres du domaine temporel (amplitude qui varie dans le temps) et non pas ceux du domaine fréquentiel (d'après analyse FFT) qui sont dans la catégorie suivante. Le filtrage se trouve sous différentes formes pour différents utilisateurs : les filtres contrôlables et évolutifs qui conviennent souvent aux compositeurs, les filtres très précis que les ingénieurs son ont tendance à préférer, ou les filtres dits « tone » ou « treble/bass » que l'utilisateur grand public aura sur sa chaîne hifi.

Le filtrage pose un problème particulier dans le contexte du traitement temps réel des instruments acoustiques : le masquage du traitement par le son acoustique ou amplifié de l'instrument traité lui même. Ce problème est très prononcé quand le filtrage ne fait que *réduire* certaines fréquences ou bandes de fréquences, car le son non-traité recouvre le son filtré en remettant effectivement les fréquences manquantes. Même les filtres résonants qui amplifie certaines fréquences ont du mal à s'entendre puisque la source recouvre tout sauf les fréquences les plus fortes. Cela nous pousse fréquemment à sur-amplifier les traitements et à provoquer de la réinjection et de l'effet Larsen. La solution musicale, comme évoqué plus haut dans *Avantages et inconvénients* est de créer une séparation de la source, avant ou après filtrage, sur le plan du temps, de la hauteur, du timbre, ou de l'espace – ainsi rendant le son filtré plus indépendant du son source, et donc plus perceptible. C'est pour cela que les filtres se trouvent le plus souvent accompagnés d'autres effets dans une chaîne de traitements.

Les filtres sont très connus du monde analogique, comme les traitements de temps, hauteur, amplitude et timbre que nous avons discutés jusqu'à maintenant, alors qu'est-ce que les

techniques de l'informatique temps réel peuvent apporter? Comme pour les autres effets, ce sont les possibilités d'un contrôle extrêmement précis, dynamique, évolutif et interactif qui font la différence. Dans ce sens, nous voulons souligner une application très importante du filtrage contrôlé qui est le *filtrage par modèles de résonance* – technique qui configure une banque de filtres résonants d'après l'analyse antérieure d'un son qui en devient le *modèle* par lequel le son à traiter va passer, ainsi excitant les fréquences communes et effectuant une sorte d'hybridation[1].

1.2.2.5 Le traitement du timbre par modulation

Cette catégorie est la deuxième de celles relatives au timbre. Pour expliquer ce que nous entendons par *modulation*, nous citons Miller Puckette qui dit « Le terme *modulation* s'applique vaguement à toute technique qui altère systématiquement le contour d'une forme d'onde en tordant son graphe verticalement ou horizontalement. »[2]. Ainsi, en suivant son exemple, nous allons mettre ensemble *la modulation d'amplitude, la modulation en anneau* (*ring modulation* en anglais) avec sa cousine *la transposition de fréquence* (*frequency shifting*), et la distorsion non-linéaire (*waveshaping*).

Musicalement, la modulation d'amplitude, la modulation en anneau et la transposition de fréquence décalent les fréquences spectrales et créent de l'inharmonicité dans le timbre. La modulation en anneau et la transposition de fréquence sont les plus utilisés, ayant l'avantage d'éliminer la source à cause du fait qu'ils opère une modulation *bipolaire* alors que la modulation d'amplitude opère une modulation *unipolaire*[3]. Ces modulations sont capables d'effectuer des transformations très marquées et bien perceptibles dues au fait que le résultat ne contient pas les mêmes fréquences que la source. Elles sont aussi très pratiques pour rendre d'autres transformations plus évidentes (comme les filtrages, par exemple) ou pour modifier le spectre même légèrement et ainsi éviter des mauvais effets de Larsen. Egalement, la modulation en anneau de basses fréquences (< 20 Hz) crée un effet de trémolo très efficace.

La distorsion non-linéaire est une forme de modulation qui agit directement sur la forme d'onde. Un niveau d'amplitude en entrée passe par une fonction de forme d'onde qui détermine

1 Voir *Les modèles de résonance: Modèles de continuité entre synthèse et traitement*, J.-B. Barriere dans *Modèles physiques création musicale et ordinateur*, Maison des Sciences de l'Homme, 1995

2 « *The term "modulation" refers loosely to any technique that systematically alters the shape of a waveform by bending its graph vertically or horizontally.* ». Puckette 2007, p 119. C'est nous qui traduisons.

3 Roads 1996, pp. 215-224; Dodge 1985, pp. 80-85; Moore 1990, pp. 185-187; Dobson 1992, pp 135-6

le niveau d'amplitude correspondant : une forme d'onde est ainsi *modulée* par une autre (Roads 1985a; Dodge 1985, pp. 128-129). C'est une manière très efficace pour générer de la richesse dans le spectre, le plus connu de ces effets étant la *saturation* – longtemps utilisé dans le domaine analogique et la musique populaire.

1.2.2.6 Le traitement de l'espace

Par cette catégorie nous entendons l'espace *acoustique* du son – les spatialisations, virtuelles ou réelles, les traitements sonores qui créent l'impression que le son vient d'un certain endroit ou qu'il se déplace, qu'il vibre dans un certain type d'acoustique avec une qualité de réverbération et de réflexion. Le traitement de l'espace, comme le plupart des autres traitements, est beaucoup plus vieux que ses petits cousins du temps réel. Le son et la musique ont toujours existé dans l'espace et les hommes ont toujours voulu *occuper* l'espace de différentes façons – entre l'espace d'un tambour signalétique qui passe de colline en colline et une danse tribale avec la musique tournante des danseurs, ou l'emplacement et déplacement des musiciens dans les églises de la musique ancienne et les projections sonores dans les amphithéâtres. L'espace n'a pas toujours été un paramètre de composition comme la hauteur, le rythme et l'harmonie, mais il a toujours fait partie de la musique[1].

Il y a deux grandes écoles de spatialisation dans la musique électroacoustique et mixte : la première est la création d'un espace homogène où le son se place et se déplace librement parmi des haut-parleurs qui s'effacent devant la mobilité sonore qu'ils portent, et la seconde est la création d'un espace non-homogène occupé par différents haut-parleurs et types de haut-parleurs placés dans des endroits stratégiques qui ne définissent pas un espace virtuel et transparent, mais un espace où les haut-parleurs colorent et positionnent les sons par des assignations directes[2]. Dans les deux cas (et dans les cas « entre les deux ») ce sont des outils d'écriture d'espace très puissants, qui permettent, avec les possibilités du contrôle que le traitement temps réel amène, que l'espace s'écrive au même titre que les paramètres musicaux « traditionnels ».

1 Pour discussions sur les spatialisations sonores et revérerations, voir aussi Moore 1989 et Moorer 1979; Zelli 2009

2 Deshays 2006; Thigpen 2009

1.2.2.7 Le traitement de la matière

Le traitement de la matière est représenté par la synthèse granulaire qui offre une manipulation sonore inégalée dans le domaine temporel. L 'idée que nous pouvons recréer une énorme palette de sons en regroupant des petits grains de micro son a fait naitre une profusion d'applications et expériences musicales[3].

Les deux formes de synthèse granulaire les plus adaptées à un travail de traitement temps réel sont 1) la granulaire d'une mémoire dynamique, une ligne à retard qui est constamment renouvelé pour le son entrant, et 2) le *gel* sonore.

Dans la granulaire d'une ligne à retard, nous n'avons pas toutes les possibilités de voyager « dans le temps » que propose la synthèse granulaire d'un enregistrement *fixe*, mais l'aspect réactif et dynamique compense de manière largement suffisante. Ça permet de traiter un son en changeant tous ses paramètres – de l'intensité à la hauteur en passant par la densité et l'articulation. Nous pouvons faire d'un son faible un son fort, d'un son dense un son éparse, d'un son stable une mélodie et d'un son mouvant un son stable.

Le *gel* sonore (*freeze* en anglais) est simplement une forme dédiée du granulaire. Cela enregistre une petite mémoire avec un son, et ensuite le joue comme une masse continue et tenue. Nous le considérons être du temps réel car quand le *freeze* est implémenté, nous avons l'impression qu'il « émerge du son direct ».

1.2.2.8 Le traitement dans le domaine fréquentiel

Le domaine fréquentiel est une zone à part dans l'ensemble des traitements temps réels. Il est comme une dimension parallèle pour laquelle il y a un portail d'entrée et un portail de sortie : l'entrée étant représentée par différentes formes d'analyses spectrales, et la sortie par différentes formes de resynthèses.

L'idée centrale est que le son, qui est dans le domaine temporel, est analysé pour son contenu spectral (fréquences, amplitudes et phases) en suivant son évolution dans le temps. Une fois que le son passe par cette représentation de données du spectre, on dit qu'il est dans le domaine fréquentiel. Il est maintenant prêt à subir toute sorte de traitements en agissant directement sur ces paramètres : manipulation directe de l'amplitude ou de la phase de chaque

3 Roads 1985b, Roads 2006,

fréquence (modification directe du spectre et donc du timbre), croisement ou convolution avec les données spectrales d'un AUTRE son qui aurait également passé dans le domaine fréquentiel (synthèse croisée, empreinte de timbre, hybridation), traitement temporel sur différentes fréquences individuellement (délai spectral), etc. Une fois le traitement terminé, les données du spectre doivent passer par l'analyse inverse, ou bien la resynthèse, pour revenir dans le domaine temporel du départ.

La méthode d'analyse spectrale la plus répandue est la *Transformée de Fourier* qui a ses origines en 1822 quand Joseph Fourier a proposé qu'une fonction périodique pouvait se représenter par une somme d'un nombre fini de sinusoïdes. Après différents essais pour construire des analyseurs mécaniques et analogiques datant de la fin du 19e siècle et continuant durant le 20e, les années 1960 ont vu l'arrivée des analyses nettement plus précises en utilisant les outils informatiques. *La transformée de Fourier* a été rejointe par d'autres analyses comme l'*analyse par filtre hétérodyne* et le *vocodeur de phase*, puis a trouvé sa place dans la *transformée de Fourier discrète* avec ses fenêtres d'analyses qui avancent pas à pas dans le temps en faisant à chaque pas une *transformée de Fourier rapide* qui extrait les amplitudes et les phases d'un nombre de tranches ou *bins* du spectre qui correspond à la moitié de la taille de la fenêtre d'analyse (Roads 1996). Dans le langage courant, ce sont surtout le terme *FFT* (*Fast Fourier Transform*) ou son application le *vocodeur de phase* qui sont utilisés pour désigner cette analyse discrète avec des fenêtres à court terme, qui est impliquée dans la grande majorité des traitements dans le domaine fréquentiel[1].

Les outils musicaux dans le domaine fréquentiel sont nombreux et variés. Il y a le simple mais efficace *filtre fft* qui marche comme un égaliseur de spectre, une banque de filtres « tranchants » par *bin* de la FFT. Il y a également la famille des *synthèses croisées* incluant le *filtrage par convolution* ou *empreinte de timbre* (*timbre stamping* dans Puckette, 2007) où les magnitudes du spectre d'un son contrôlent les magnitudes du spectre d'un autre, ou bien la *synthèse croisée généralisée* qui peut échanger les magnitudes d'un spectre avec les phases d'un autre – ainsi créant des sons hybrides et même un effet d'interpolation de timbres (*morphing* en anglais)[2]. Un autre traitement qui a eu un certain succès ces dernières années est

1 Pour une excellente description abordable de l'analyse Fourier, voir l'appendice dans Roads 1996 avec Philip Greenspun. Pour une explication en profondeur du coté mathématique voir Puckette 2007, chapitre 9, *Fourier Analysis and Resynthesis*.

2 Pour des exemples de performances avec traitement vocal et instrumental dans le domaine spectral, voir les *Convolution Brothers*, ensemble live de trois pionniers du temps réel: Cort Lippe, Miller Puckette, et Zack Settel (http://www.music.buffalo.edu/lippe/convolutionbrothers/)

le *délai spectral* qui consiste à appliquer une ligne à retard séparée à différents groupes de *bins* de fréquences de la FFT (Pekonen, Välimäki, Abel, Smith 2009; Gibson 2009). Nous avons également la *compression/expansion temporelle* (changement de durée sans changer la hauteur) qui est une des applications du domaine fréquentiel la plus importante. La difficulté avec ce traitement dans un contexte temps réel et sa nature plutôt *hors temps* : on ne peut ni faire écouter un son plus vite qu'il ne se déploie dans le temps, ni ralentir un son indéfiniment car le moment « live » de la source sera de plus en plus loin du moment de la relecture au ralenti – occupant de plus en plus de mémoire. Nous pouvons aussi inclure les modifications du timbre d'après les décalages des *bins* d'analyse (*bin shifting* en anglais) ou bien la spatialisation spectrale qui consisterait à spatialiser séparément différentes bandes de *bins*.

Nous voyons clairement que, malgré le fait que le domaine fréquentiel donne un accès direct au spectre, nous pouvons traiter beaucoup d'autres choses que le timbre. D'ailleurs, cette catégorie touche tout simplement toutes les autres catégories, avec parfois des applications musicales très similaires[1].

1.3 Les analyses sonores et le contrôle des paramètres

1.3.1 Le principe d'analyse et du contrôle

Nous ne pouvons pas parler des applications musicales des traitements temps réel sans parler un peu de l'aspect du contrôle des paramètres d'après des analyses. C'est un des éléments le plus important qui distingue le potentiel du temps réel des outils analogiques historiques. Nous faisons donc une brève description des types d'analyses que nous trouvons aussi « basique » que la liste des traitements.

1.3.2 Les types d'analyses

1.3.2.1 Le suivi d'enveloppe et la détection des attaques

La technique de suivi la plus simple, rapide et « légère » à faire est sûrement le suivi

1 Pour des discussions d'applications de modificaitions timbrales, voir aussi Dolson 1989

d'enveloppe.[1] Ça se passe dans le domaine temporel et n'a besoin que de lisser les changements d'amplitude et d'extraire une valeur. Nous pouvons faire une détection de dépassement de seuil pour faire un *déclenchement* sur une attaque du son, ou bien suivre les changements d'amplitude sur plus ou moins de temps et les connecter à divers paramètres des traitements sonores. Pour voir une application musicale du suivi d'enveloppe, cf. II 2.3.2 LFO d'amplitude contrôlé par le suivi d'amplitude, page 61.

Il existe un autre modèle de détection d'attaque : par reconnaissance de timbre. Un système va *apprendre* un ensemble de timbres (enveloppes spectrales), et par la suite va *écouter* des nouveaux sons et les comparer avec les timbres en mémoire pour ensuite faire une identification. Ça marche surtout avec les sons de percussions, et surtout avec un *objet* pour MaxMSP créé par Miller Puckette qui s'appelle *bonk~*[2].

1.3.2.2 Suivi de hauteur

Le suivi de hauteur se fait par une analyse dans le domaine fréquentiel combinée avec une analyse approfondie des *pics* pour en déduire l'existence d'une fondamentale. C'est bien plus lourd qu'un simple suivi d'amplitude, et il y a une certaine latence intrinsèque. Cependant, il est très pratique pour modifier des traitements selon des registres du jeu (du moins pour les instruments monophoniques) et, peut-être plus significatif, il peut être appliqué à des algorithmes de suivi de partition – l'ordinateur qui compare le jeu de l'instrumentiste à la partition en mémoire pour pouvoir avancer dans sa liste d'événements et permettre à l'instrumentiste d'être le *maitre du jeu*. Un des exemples le plus réussi d'une composition pour traitements temps réel et suivi de partition est *Jupiter* de Philippe Manoury[3]. Pour voir une application musicale de la suivie de hauteur, cf. II 2.2.2 Harmoniseur contrôlé par la détection de hauteur, page 57.

1.3.2.3 Détection du centroïde spectral

L'analyse du centroïde spectral emploie une analyse fft pour, en quelque sorte, calculer le *centre de gravité* du spectre, autrement dit le *centre pondéré*. On obtient une indication de la brillance ou de la richesse dans le timbre. En pratique, il est parfois impossible de différencier

1 Settel 2001
2 http://crca.ucsd.edu/~tapel/software.html
3 Documentations et enregistrements de *Jupiter* de Philippe Manoury se trouvent à la Médiathèque de L'Ircam

un centroïde d'un son aigu et un son grave très riche en contenu aigu, mais nous pouvons par contre très facilement suivre l'ouverture d'un timbre sur une note unique (voix ou instrument). Pour voir une application musicale de la détection du centroïde, cf. II 2.5.2 Transposition de fréquence et contrôle par le centroïde, page 68.

1.3.2.4 Détection de transitoires

Une détection de transitoires peut se faire très simplement par une technique qui consiste à compter le nombre de fois qu'un signal traverse une valeur de « 0 » par laps de temps. Il se trouve que le bruit fait des traversées par zéro *beaucoup* plus souvent que les sons voisés[1]. Pour voir une application musicale de la détection de transitoires, cf. II 2.4.2 Filtre contrôlé par une détection de transitoires, page 65.

1 Settel 2001

2 Le traitement du temps

Nous allons ici poser les questions centrales : Qu'est-ce qu'un décalage temporel (délai) et comment s'en servir musicalement? Comment jouer et écrire de la musique instrumentale pour un délai?

2.1 Introduction

Dans ce chapitre, nous allons d'abord voir un peu l'histoire du délai, et les moyens techniques pour en faire. Ensuite nous allons étudier la perception du délai, par les temps courts, les temps moyens, les temps long, et les temps très longs. A la fin, il y a un guide à l'écriture qui propose des techniques d'écritures musicales qui devraient permettre de mieux profiter de nos délais par la suite.

2.2 Histoire et techniques du délai

2.2.1 L'écho acoustique et l'écho musical

Un écho est le résultat d'une réflexion du son contre une surface et sa caractéristique première est le temps qu'il faut pour que le son puisse parcourir la distance entre la source, la surface de réflexion et l'auditeur. Le vitesse du son étant environ 340 mètres par seconde à 15ºC[1], si la source est aussi l'auditeur, il faudrait qu'elle soit à 170 mètres d'une surface pour que le son revienne une seconde plus tard. Et dans quel état revient-il? Cette réflexion n'est jamais parfaite donc certaines fréquences se reflètent plus ou moins bien, ou bien se font absorber ou diffuser. A cause du rayonnement sonore de la source, une bonne partie de l'énergie du son part dans d'autres directions que la surface, ne laissant qu'un petit peu d'énergie disponible pour la réflexion. Après tous ces facteurs de rayonnement, absorption et diffusion, un écho plus ou moins petit et plus ou moins coloré revient à notre source/auditeur. Et la répétition? Il faudrait qu'il y a deux surfaces parallèles et que la source/auditeur soit très proche de l'une pour que la répétition puisse éventuellement se répéter à intervalles réguliers.

Dans la musique, nous ne cherchons pas à reproduire des phénomènes acoustiques, mais à

1 Cf http://fr.wikipedia.org/wiki/Vitesse_du_son

nous en inspirer pour les appliquer à d'autres règles, à d'autres exigences et à d'autres critères. Nous détournons les phénomènes naturels pour en extraire l'essence de leur musicalité au profit de notre musique. Tout comme la musique concrète cherchait souvent à rendre plus évident ou mettre en valeur une musicalité « cachée » dans les sons de notre environnement quotidien par l'enregistrement, le traitement et le montage de ces sons.

Est-ce que nous utilisons les délais électroniques pour imiter les retards ou répétitions du monde acoustique? Qu'est-ce qui fait qu'une utilisation d'un délai devient « musicale »? Comment faire de la musique avec un délai, et quels en sont les paramètres?

Avant l'arrivée des moyens électroniques, c'est plutôt le concept de la « répétition » qui constituait son application musicale. On répète une phrase ou une partie plus tard, éventuellement avec une nouvelle terminaison. L'idée même de versets d'une chanson incarne le concept de la répétition – du moins pour la musique.

Mais ces utilisations « manuelles » de la répétition sont très différentes d'un traitement artificiel d'un délai où « tout ce qui rentre ressort un certain temps après... »

2.2.2 le délai à bande magnétique

Les premiers vrais délais souples utilisés largement dans la création musicale étaient par bande magnétique. Ces implémentations des délais étaient beaucoup plus puissants et flexibles que nous pouvons être amener à penser – maintenant que nous avons des outils informatiques très puissants à notre disposition. Nous croyons souvent que ce qui existait avant n'était *pas* puissant en comparaison, mais l'époque informatisée a beaucoup à apprendre du passé.

Des délais par bande fonctionnaient d'abord par le temps passé entre la tête d'enregistrement et la tête de lecture, mais il y avait bien d'autres astuces. Un enregistreur deux pistes pouvait être branché pour mettre les pistes, et la distance entre les têtes, en *série*, ainsi *doublant* le temps disponible. Aussi on pouvait mettre deux machines à tourner la même bande, et donc avoir facilement plusieurs secondes de délai entre la tête d'enregistrement d'une machine et la tête de lecture de l'autre.

Il y avait aussi tous les effets de *flanging*, *phasing* et *chorusing* qui n'étaient que *simulés* par l'électronique et l'informatique plus tard. Dans le livre d'Allen Strange sur la musique électronique, il a écrit un passage définitif sur le *tape delay* qui décrit des circuits étonnants par leur ingéniosité et leurs résultats musicaux. (Strange 1972, pp 194-211)

2.2.3 Le délai informatique

2.2.3.1 Le principe

L'idée du délai informatique est relativement simple. Une mémoire est définie où les échantillons de l'audionumérique entrent d'un côté, se faisant repousser petit-à-petit par les échantillons suivants, et finissent par ressortir de *l'autre* côté de la mémoire et ainsi disparaissent. S'ils ressortent avant la fin de la mémoire, nous les entendons comme le son qui est rentré, mais un peu plus tard. Le gros avantage que les délais informatiques ont par rapport aux délais anciens c'est que la durée n'est qu'une fonction de mémoire, et puisque la mémoire n'est pas chère, les durées potentielles sont énormes.

2.2.3.2 La latence

Dans le cas spécifique de l'informatique, où il y a besoin d'une carte son, de convertisseurs analogiques/numériques d'un côté, de numériques/analogiques de l'autre, il y a un petit peu plus de délai ou « latence » dans notre « délai 0 ». Il s'agit de la latence due au regroupement des échantillons en *vecteurs* ou *listes* pour alléger le poids du calcul nécessaire à leur gestion. Ainsi, pour un vecteur d'entrée/sortie (*I/O Vector Size* dans Max/MSP) de 512 échantillons et un taux d'échantillonnage de 44100 échantillons par seconde, la latence théorique est 512/44100 = 11.6 ms en entrée et aussi en sortie : 23.2 ms au total. Cette valeur est déjà dans la zone perceptible des 20 à 50 millisecondes En testant avec Max5 et le patch « latency-test » qui se trouve dans le dossier *examples/utilities*, l'entrée/sortie sans traitement prend en moyenne environ 1202 échantillons (27,26 ms) avec un *I/O Vector Size* de 512 échantillons et un taux d'échantillonnage de 44100. Cela fait 1202 – 1024 = 178 échantillons de latence en plus de la théorique latence due au *I/O Vector Size* – ou environ 4ms. Pour des sons avec des attaques marquées, ce délai réel mesuré de 27 ms est perceptible (cf infra). Il est important d'essayer de réduire ce temps en réduisant cet *I/O Vector Size*. Si nous avons le choix avec

notre carte son de réduire à 256, ou mieux à 128, nous pouvons réduire le temps de délai minimum de façon significative. Dans un environnement informatique typique réglé à un *I/O Vector Size* de 128, le « délai 0 » serait 256 + 178 échantillons : environ 10 ms, bien plus acceptable que 27 ms.

Selon notre propre expérience, il nous semble qu'à partir d'environ 15 ms de latence de traitement, un léger retard est perceptible si le musicien écoute son propre son par casque et joue des sons percussifs. Nous avons testé en utilisant la chaîne: microphone de l'ordinateur → entrée son dans Max5 (adc~) → sortie son dans Max5 (dac~) → haut-parleur de l'ordinateur. En conclusion, pour avoir un « délai 0 » sans latence perceptible avec des sons percussifs, il faut pouvoir régler sa carte son sur 128 échantillons de vecteur d'entrée/sortie (*I/O Vector Size*) ou moins.

2.3 Le délai et la perception

La perception d'un délai ou d'un écho, et par conséquence sa fonction, varient selon les qualités, les caractéristiques et la proximité de la source, ainsi que selon le temps du délai et les qualités et la proximité de l'écoute. Dans ces recherches, nous nous sommes concentrés sur les traitements temps réel et l'écriture/l'improvisation de la musique mixte dans des contextes réels de perception en concert. Nous allons donc prendre en compte la disposition scénique, la diffusion et l'emplacement du public comme facteurs importants.

Nous allons prendre comme base le dispositif suivant d'un microphone, dans une carte son lié à l'ordinateur, puis la sortie de la carte son dans 2 haut-parleurs (ou bien dans 2 amplificateurs puis dans 2 haut-parleurs). Pour les programmes d'exemples, nous allons utiliser MaxMSP, ou *Tape* (tapemovie.org).

2.3.1 Le temps court (1-15 ms)

2.3.1.1 Phase et couleur

Délai provoquant un traitement sur le plan de la couleur du son : Ce sont les délais assez courts avec probablement un facteur de réinjection (*feedback*) peuvent créer un effet de

scintillement autour de la source, ou bien l'effet de filtre en peigne qui génère une hauteur « parasite » qui peut être perçue comme une pédale ou bien comme une coloration globale.

Transformations de timbre à base de délai : *Flanging*, *phasing* et *chorus*. Ces effets se servent (chacun à leurs manières) des délais cycliquement variables qui modifient légèrement la vitesse du déroulement du son créant ainsi des annulations spectrales liées à la phase quand le son source est re-combiné[1]. Ce sont clairement des effets qui servent à mettre en valeur un son, n'ayant que peu de conséquences directes sur l'écriture musicale. Si on pousse les valeurs de fréquence et de profondeur vers les extrêmes, on peut modifier la perception de la hauteur, mais on risque des effets de « sirène » ou « soucoupe volante »!

2.3.2 le temps moyen (15-50 ms)

2.3.2.1 Réflexion, espace

Délai qui fait un traitement de l'espace: C'est un délai qui s'entend comme une réflexion acoustique donnant à l'oreille un indice des caractéristiques acoustiques de la salle. Une multitude de ces « délais d'espace » crée ce qu'on appelle une réverbération. Cela peut aussi servir pour créer une perception de dédoublement d'un instrument. Il n'y a pas vraiment d'implications sur le plan purement musical de l'écriture, mais surtout sur la valorisation sonore – ce qu'on appelle affectueusement un effet. Les valeurs de délai concernées sont entre environ 15 ms et 50 ms, car au-delà, on perçoit plutôt un écho.

2.3.2.2 Épaississement

Délai créant un effet d'épaississement : L'effet d'épaississement peut se mettre en marche avec n'importe-quel temps de délai, pourvu qu'il y ait superposition avec le son source et que le son source soit de nature *stable*. Dès qu'il y a une nouvelle articulation ou un changement de hauteur ou de timbre (que le son source *change* de quelque manière), nous reconnaissons le temps de délai pour ces qualités d'espace, d'écho, de rythme ou de forme et mémoire.

1 Strange 1972

2.3.3 le temps long (>50 ms)

2.3.3.1 Écho, répétition

Délai qui crée un écho: une répétition entendue un certain temps après la source. On l'entend comme un élément séparé, bien distinct de la source. Il peut y avoir ou non du *feedback*, mais l'important c'est qu'il n'y ait pas vraiment de confusion avec la source. Il est difficile d' identifier le temps minimum ou maximum de délai pour cette catégorie, mais c'est en général plus que le délai de l'espace et moins que le délai rythmique qui dépend du tempo et de l'écriture.

2.3.3.2 Rythme, polyphonie

Délai qui fonctionne sur le plan rythmique: Le son qui revient après un certain temps contribue au rythme de l'instrument. Si un instrument joue des noires au tempo de 60, un délai de 500 ms crée un dédoublement à la croche. Si il joue des croches, un délai de 250 ms créera un dédoublement à la double croche, et ainsi de suite. S'il joue des noires dans un délai équivalent à la double croche (250 ms au tempo 60) le délai risque de s'entendre plus comme un écho que comme un élément rythmique. Encore une fois, tout dépend du rapport entre le temps de délai et l'écriture. La limite basse du temps qui peut fonctionner rythmiquement est probablement aux environs de 100 ms – ce qui représente une valeur de triple croche au tempo de 75 à la noire. Une écriture rythmique du délai requiert une grande précision de l'instrumentiste, et maintenir cette précision en dessous de 100 ms paraît délicat à demander à l'instrumentiste, voir même à l'auditeur.

Délai provoquant un effet de polyphonie : Une valeur de délai avec un temps *rythmique* et un *feedback* lui permettant de se répéter un certain temps ou même indéfiniment. Si nous coupons l'entrée de l'instrument dans le délai et nous mettons le feedback à 100%, nous créons effectivement une pulsation bouclée qui peut tolérer toutes sortes de confrontations polyrythmiques. Si nous continuons à laisser entrer le son instrumental, chaque nouvelle articulation régénère la pulsation, ce qui complexifie beaucoup le résultat.

2.3.4 Le temps très long (>10 sec)

2.3.4.1 Mémoire et forme

Les temps de délai trop longs pour fonctionner sur un plan d'épaississement, écho, contrepoint, et polyphonie fonctionnent probablement sur un plan de mémoire et de forme musicale. C'est bien moins complexe que le fonctionnement polyphonique, mais tout aussi important. Ces temps ne pouvaient pas vraiment exister comme phénomène acoustique naturelle car pour 10 secondes d'écho naturel le son ferait un tel trajet qu'il ne resterait pas beaucoup d'énergie pour l'écoute. Les temps de délai très longs sont donc plutôt une conséquence de la technologie.

2.4 Guide de l'écriture

Nous allons maintenant explorer les techniques d'écriture (applicables aussi à l'improvisation) liées aux différents temps et perceptions d'un décalage temporel. Nous pouvons voir cette partie comme une sorte de *traité* ou *méthode*, mais nous préférons voir cela plus comme un constat de certaines « évidences » qui ne se trouvent pas encore réunies en un seul endroit. Les compositeurs et improvisateurs doivent trouver pour eux-mêmes les applications musicales des outils en n'ayant appris que leurs fonctionnements. techniques. Nous estimons, cependant, que si ces « évidences » sont clairement expliquées l'écriture pour ces outils et pour ces « instruments » va s'améliorer, l'utilisation générale du temps réel va se solidifier et se sécuriser, et la recherche va s'accélérer pour tout ce qui est au-delà des limites que nous imaginons aujourd'hui.

2.4.1 Le domaine de l'espace

Nous allons commencer avec un effet de réflexion acoustique, qui donne à l'oreille un indice des caractéristiques acoustiques de l'espace réel et virtuel dans lequel nous écoutons le son. Les valeurs du délai qui créent cet effet sont environ entre 15 et 50 millisecondes. En dessous de 15 ms, l'oreille n'arrive pas à différencier la réflexion de la source et nous entendons une simple amplification. Au delà de 50 ms, le son s'entend comme un écho bien séparé de la source ainsi créant un nouvel objet musical et non pas un effet d'espace. Ces valeurs sont un

peu subjectives et dépendent aussi de la qualité d'attaque de la source. La fourchette 15-50 ms s'applique à une attaque bien franche comme un son de percussion, une consonne vocale ou un « slap » d'un instrument à vent. Une attaque plus douce comme un son de flûte non-articulé ou une voyelle chantée va augmenter ces valeurs par un certain facteur, allant jusqu'au double pour une attaque très molle.

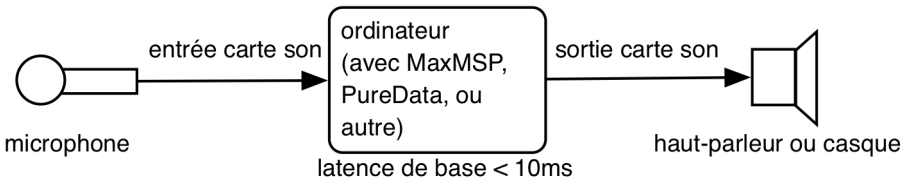


Figure 1: Circuit de traitement informatique

Nous pouvons tester et vérifier ce phénomène très simplement avec un circuit de traitement informatique : microphone → ordinateur (avec MaxMSP, PureData, ou autre application temps réel) → haut-parleur ou casque. Voir Figure 1: Circuit de traitement informatique

Dans l'ordinateur, nous allons mettre en marche un petit patch MaxMSP[1] constitué d'une entrée son, un délai et une sortie son, avec un contrôle manuel sur le temps du délai en millisecondes (Figure 2: Patch de délai simple). Il est important de noter que nous ne mélangeons pas l'entrée directe avec le signal retardé, pour que nous puissions avoir une comparaison nette entre le son acoustique et le même son après un temps de délai. Il est souvent souhaitable dans un traitement temps réel de bien séparer la source des traitements.

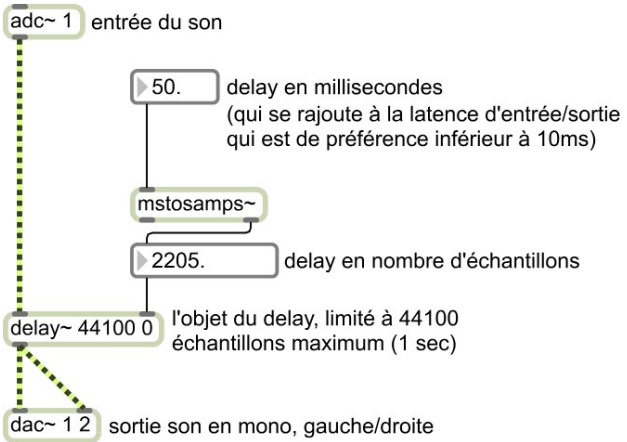


Figure 2: Patch de délai simple

1 MaxMSP est un produit de Cycling74 (www.cycling74.com). Le nom de la version actuelle s'appelle Max5.

Nous avons réglé les paramètres audio entre MaxMSP et la carte son pour avoir une taille de vecteur d'entrée/sortie (I/O Vector Size) de 128 échantillons pour que notre latence réelle d'entrée/sortie tombe en dessous de 10 ms (précisément 9.8 ms mesurée avec le patch « latency-test » qui se trouve dans les exemples de MaxMSP). Ainsi, nous savons qu'il faut rajouter 10 ms à la valeur que nous entrons dans le patch pour savoir la « vraie » valeur du délai.

Ensuite, en portant un casque sur une seule oreille et en gardant l'autre oreille libre pour entendre le son acoustique,, nous allons régler le délai sur 0 ms (ce qui fait 10 ms en considérant la latence) puis utiliser la voix ou un instrument pour jouer avec une attaque plutôt percussive la figure suivant répétée (Figure 3: Test pour un délai d'espace).

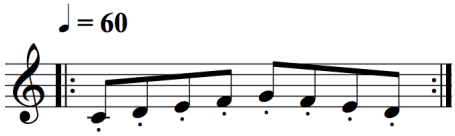


Figure 3: Test pour un délai d'espace

Nous entendons une simple amplification/diffusion de la source par la sortie casque, sans aucun délai effectif. Maintenant nous augmentons graduellement le temps de délai dans le patch MaxMSP. En arrivant entre 5 et 10 ms (15 à 20 ms réel), nous commençons à entendre un très léger temps de délai comme en reflet. L'espace est un petit peu élargi, mais il n'y a aucune interférence musicale – l'écoute du délai ne nous gêne pas pour jouer/chanter la phrase. Nous continuons à augmenter la valeur du délai jusqu'à ce que l'écoute du délai commence à interférer musicalement, jusqu'à ce que nous ayons un peu de mal à rester en rythme et à bien écouter ce que nous faisons. Nous avons trouvé que le temps de délai à ce moment là est autour de 50 ms, parfois légèrement plus parfois légèrement moins. Cette interférence indique que nous entendons le délai comme un écho, qui crée un objet sonore indépendant, capable d'interférer avec la source.

Cette valeur d'environ 50 ms est généralement acceptée comme temps auquel l'oreille détecte un élément séparé, phénomène qui s'appelle l'effet de précédence ou l'effet de Haas.[1] Mais, notre expérience suggère que les facteurs comme la qualité de l'attaque et la densité rythmique rentrent en jeu quant à nos perceptions du délai. Si nous jouons des notes plus rapprochées

1 Truax 1978

dans le temps, la valeur de délai en *écho* a tendance à baisser un peu car le temps de délai sera proportionnellement plus grand par rapport au rythme du jeu. De la même manière, si nous espaçons davantage les notes, l'interférence peut arriver à une valeur supérieur à 50 ms. La nature de l'attaque va également avoir de l'influence. Une attaque précise augmente la perception d'un délai alors qu'une attaque molle la diminue. Ainsi, si nous réduisons, voir éliminons l'attaque, nous allons aussi diminuer la perception du délai de l'espace.

Refaisons le même test qu'avant, mais cette fois-ci en jouant le figure suivante où l'attaque part du silence en crescendo et le relâchement arrive graduellement au silence par decrescendo (Figure 4: Test 2 pour un délai d'espace).

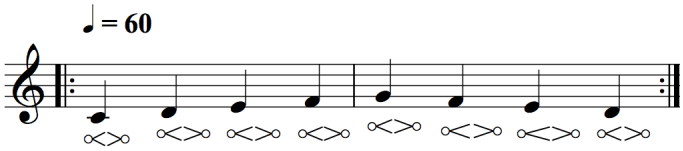


Figure 4: Test 2 pour un délai d'espace

Nous pouvons remarquer que nous devons augmenter d'avantage la valeur du délai pour l'entendre. Le début de l'effet *réflexion* devient difficile à identifier, se situant vers 30 ms plutôt que vers 15 ms comme l'exemple précédent. Le moment du passage vers *l'écho* qui interagit avec la musique ayant la capacité de perturber l'écoute ne s'active pas avant au moins 100 ms. Finalement, nous constatons que pour identifier un temps de délai (et par conséquence sa fonction) il faut entendre le début de la source. S'il n'y a aucun moment de changement rapide dans la source il est quasiment impossible d'identifier précisément la valeur du délai, et donc de le percevoir. Nous allons revenir souvent sur ce principe dans les recherches suivantes.

2.4.2 Le dédoublement et l'épaississement

Le décalage temporel peut fonctionner comme *dédoublement* musical ou *épaississement* sonore. Le *dédoublement* instrumental est le rajout d'un deuxième instrument qui joue la même musique que le premier, ce qui crée à l'oreille un effet d'*épaississement*. Cet effet est donné par des légers décalages temporels et des légères différences d'intonation et de timbre générés naturellement quand des êtres humains jouent des instruments acoustique ensemble. Si nous restons dans un discours uniquement du *temps*, nous pouvons dire que les valeurs de

délai qui donnent l'effet des réflexions spatiales (15 à 50 ms) sont parfaitement convenables pour créer un effet de dédoublement, puisque là aussi il faut élargir (par un instrument en plus) sans interférer dans la compréhension de la musique. Si nous incluons dans notre discours les effets de modification de hauteur et de timbre associés aux traitements de délai variable tels le *flanging*, le *phasing*, et le *chorusing*, nous avons beaucoup de ressources disponible pour doubler et épaissir le son.

Restant dans notre contexte de l'écriture et de l'improvisation musicale, nous allons parler de l'épaississement de manière plus large. Nous définissons l'épaississement comme étant un élargissement du son par l'empilement sonore avec lui-même. Au niveau de la perception, si nous reconnaissons un écho ou une répétition sonore, nous sortons du contexte d'un simple épaississement et nous passons dans une perception rythmique ou polyphonique (cf supra). Utiliser des valeurs de délai d'environ 15 à 50 ms est une bonne méthode pour éviter la perception de l'écho, mais il y a une autre méthode qui donne le pouvoir de la perception à la musique elle-même : les modes de jeu *stables*.

L'idée a été évoquée dans la section précédente : le fait qu'un son sans attaque ni relâchement évident empêchait la perception d'un temps de délai. Pour aller plus loin, nous pouvons dire que c'est la *stabilité* (tout ce qui change très lentement ou pas du tout) qui rend la perception d'un délai difficile. Par conséquence, l'*instabilité* (tout ce qui change rapidement) dévoile clairement la présence d'un délai et permet de mesurer facilement sa valeur de temps. Dans d'autres termes, l'identification d'un *instant* sonore permet d'identifier le même *instant* plus loin dans le temps. Un son *stable* ne propose pas de repère temporel.

Un son doit être stable ou ne changer que très lentement dans tout ses paramètres musicaux pour pouvoir effacer la perception d'un délai : son intensité, sa hauteur, son timbre, son rythme. Le temps du délai n'a aucune importance – le son *stable* efface un délai de n'importe-quelle durée, sans préjugé.

Voici quelques exemples en forme de partition pour donner des idées de ce que nous considérons être des sons *stables*. D'abord, il y a la *tenue,* avec crescendo depuis le silence et decrescendo jusqu'au silence (Figure 5: Son stable – clarinette). Les instruments à vent ont une bonne capacité à jouer ce type de figure, la clarinette ayant un don particulier pour le son niente. Ici, les changements d'intensité se font assez graduellement pour ne pas donner un

repère temporel.

Figure 5: Son stable – clarinette

Un son qui contient de l'activité peut avoir l'effet d'être *stable* si cette activité se répète sur une assez courte durée sans variations. Regardons l'exemple de tremolo sur une note de vibraphone – également avec un crescendo/decrescendo pour éviter la perception du début et de la fin de la phrase (Figure 6: Son stable – vibraphone).

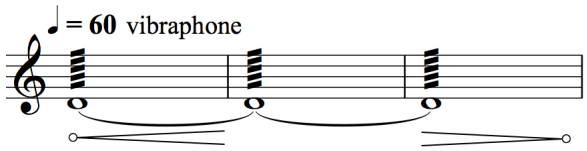


Figure 6: Son stable – vibraphone

Dans cet exemple pour flûte, la trille sur deux notes donne le même effet de stabilité (Figure 7: Son stable – flûte).

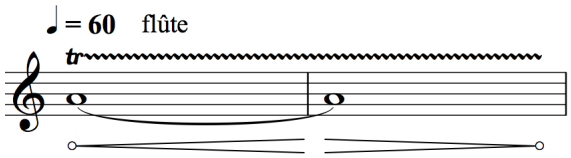


Figure 7: Son stable – flûte

Même un tremolo sur plusieurs notes se comporte comme *stable* si la vitesse est assez élevée (Figure 8: Son stable – piano).

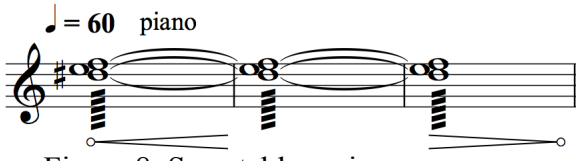


Figure 8: Son stable – piano

La continuité sonore n'est pas la seule méthode pour créer de la stabilité pour un délai et le rendre non-identifiable. Nous pouvons aussi répéter une note en pulsation dans un délai égal au temps de la pulsation ou bien un multiple de ce temps. Par exemple, avec une pulsation de 120 notes par minute (croche à noire = 60) dans un délai de 500 ms fait que chaque nouveau

son est superposé parfaitement avec le son précédent. Puisque les sons restent les mêmes (pas de changements de hauteur, de timbre, d'intensité ou de durée) il n'y a aucun repère temporel et l'auditeur entend un dédoublement ou un épaississement. Il y a pourtant le début et la fin du son qui pourrait dévoiler le délai et sa valeur, mais si nous faisons arriver et repartir le son depuis et vers le silence comme dans les autres exemples, nous masquons complètement le temps de délai(Figure 9: Son stable – pulse).

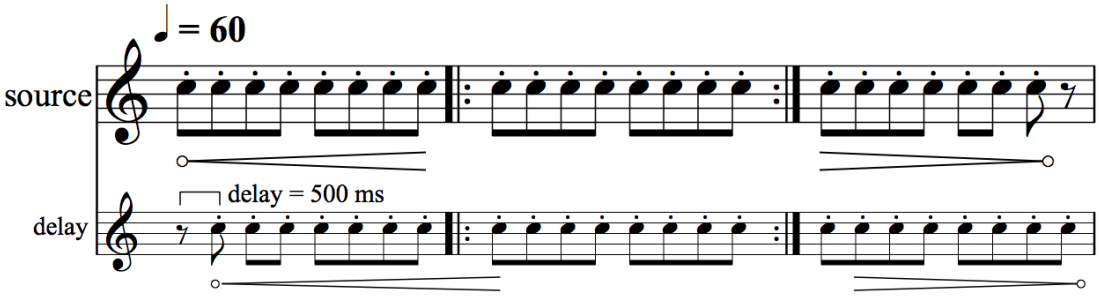


Figure 9: Son stable – pulse

Quel serait l'intérêt d'appliquer un délai que nous n'apercevons pas comme tel? Masquer la perception d'un délai permet, justement, de le démasquer à des moments précis – d'avoir une valeur de délai que nous pouvons faire apparaître et disparaître uniquement par le jeu instrumental. Tout ce qui se démarque du son *stable* crée un repère temporel qui permet de le détecter après le temps de délai, activant la perception qui correspondrait normalement au temps de délai choisi (i.e. Espace, écho, rythme, mémoire...). Nous avons ainsi l'impression qu'il y a un traitement pour la partie *stable* et un AUTRE traitement pour les événements ponctuels. Par exemple, dans notre pulsation en *do*, un *ré* va générer un *ré* en écho une croche plus tard. Une note jouée plus fort serait facilement repérable après le temps de délai, et ainsi de suite – accélération / décélération, changement d'articulation ou de mode de jeu, changement de timbre... Regardons l'exemple suivant (Figure 10: Stable avec éléments ponctuels) où les sons qui se démarquent dans le délai sont encadrés.

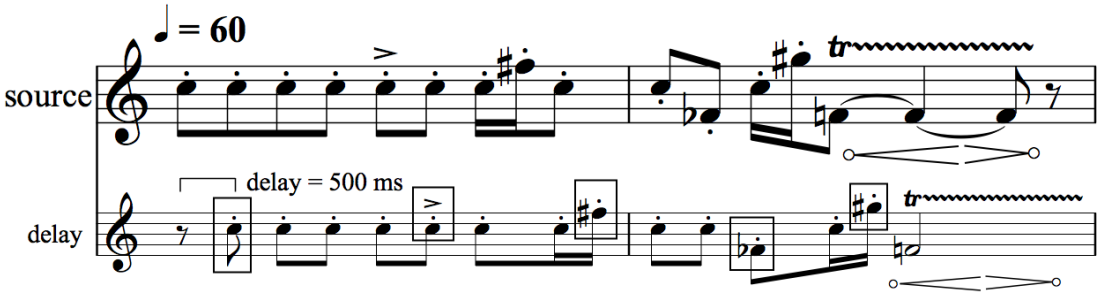


Figure 10: Stable avec éléments ponctuels

2.4.3 L'écho et la répétition

L'écho et la répétition font partie des effets de délai *long* (cf le temps long (>50 ms)), donc supérieur à 50 ms, environ. Comme expliqué plus haut, c'est un effet qui ne se confond pas avec la source – nous entendons clairement un son, une note ou une phrase se répéter. Il peut se superposer avec une source qui continue, mais il y a toujours assez de repères temporels pour pouvoir identifier à tout moment que l'écho revient après un certain temps. C'est potentiellement l'application du délai le plus dangereux musicalement, car en ayant compris assez rapidement la règle du jeu, notre oreille en perd l'intérêt, et l'ennui ou la fatigue d'écoute peut s'installer. Cela se différentie du délai de polyphonie et contrepoint que nous allons explorer juste après.

La manière la plus simple pour entendre un délai en tant qu'écho ou répétition est de jouer une note ou une phrase qui est plus courte que le temps de délai. Ainsi, il est facile de comprendre que *nous avons une note ou une phrase, puis la note ou la phrase se répète*. Regardons l'exemple (Figure 11: Exemple d'écho).

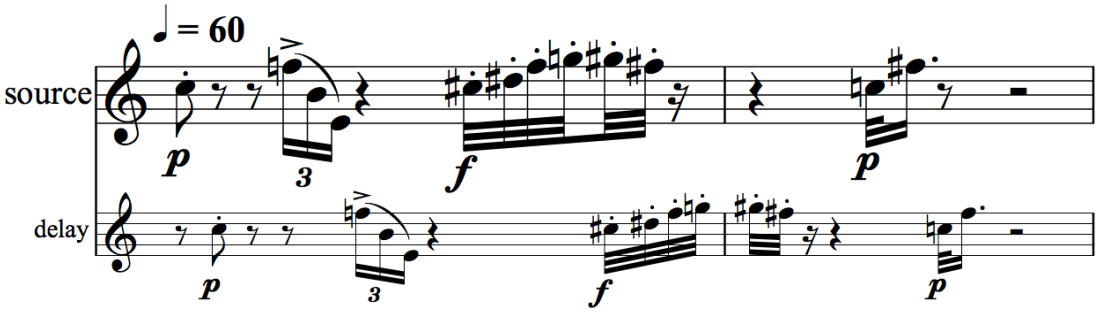


Figure 11: Exemple d'écho

2.4.4 La polyphonie et le contrepoint

La polyphonie et le contrepoint font aussi partie des effets de délai *long*, comme l'*écho et la répétition*. La différence est plutôt une conséquence de l'*écriture*[1] elle-même. Contrairement à l'écho (cf. infra) où les phrases musicales sont généralement plus *courtes* que le temps de délai, dans un délai en contrepoint, les phrases musicales sont généralement plus longues. Allen Strange donne un exemple d'une écriture en contrepoint ou *canon* (Strange 1972, page 200) en jouant des noires, des croches et des triolets de croches au tempo de 60 dans un délai équivalent à la valeur d'une croche (500 ms). Le délai n'est absolument pas entendu comme un

1 Par *écriture* nous entendons aussi *improvisation* dans ce contexte.

écho, mais comme une deuxième voix polyphonique – ou nous avons parfois une difficulté à reconnaître « qui joue quoi ». Strange mention aussi un compositeur qui a beaucoup travaillé le contrepoint avec les délais, Terry Riley. Ces compositions/performances *A Rainbow in Curved Air* et P*ersian Sergery Dervishes* sont des classiques du genre.

Pour d'autres exemples et des discussions sur les délais en tant que polyphonie et contrepoint, lisez la troisième section de ce mémoire sur *Le patch bien tempéré numéro 1*.

2.4.5 Le rappel, la mémoire et la forme

Il est probable qu'avec un délai *très long* de plus de 5 ou 10 secondes nous avons de la difficulté a maintenir une fonction de contrepoint, et nous allons commencer à glisser dans un perception de *mémoire* ou de *forme*. Comme nous y faisons souvent allusion, tout dépend de la *stabilité* dans le jeu ou dans l'écriture. Si le son ou la phrase musicale garde un même caractère ou ne change pas trop durant le temps de délai, il est possible que le délai soit entendu comme un contrepoint, mais si le jeu musical change plus radicalement durant le temps de délai, et le délai est très long, nous allons surement être dans une fonction de *rappel*, de *mémoire*, ou de *forme.* Globalement, cela veut dire que nous avons le temps de penser « tiens, j'ai déjà entendu cela. ». A ce sujet, nous ne pouvons pas parler d'absolu, mais d'idées à considérer, et à prendre en compte quand nous sommes dans un processus de création.

Le compositeur Alvin Lucier a utilisé un *très long* délai à des fins à la fois *formelles* et *timbrales* dans sa pièce *I'm sitting in a room*. Dans cette pièce, un très long délai de plusieurs minutes fait revenir dans un haut parleur le texte qu'il venait de dire. Le micro reprend encore une fois le texte qui sera encore rediffusé dans l'enceinte – et ainsi de suite. Avec chaque apparition du texte nous avons la mémoire du texte et de comment on l'avait entendu, mais aussi nous avons le traitement de filtrage par résonance qui devient de plus en plus évident avec chaque réenregistrement du texte.

II Tapemovie et la programmation d'outils pour la création temps réel

1 *Présentation de* tapemovie *et sa partie audio :* tape

Tapemovie[1] est un projet de développement collectif à l'initiative de l'auteur de ce mémoire, et il reste son environnement de choix pour toute création autour des traitements temps réel. Durant l'année de ce mémoire il a subit de nombreuses améliorations et formalisations, ainsi qu'une « sortie officielle » en tant qu'application *open source* accompagnée d'une présentation d'article écrit avec le co-développeur de tapemovie, Renaud Rubiano, lors des Journées d'Informatique Musicale à Rennes ce mai dernier (Mays et Rubiano 2010). La partie de cet article qui concerne l'architecture globale et les traitements audio se retrouve ici, permettant une bonne compréhension de *tapemovie* avant de procéder à la partie d'exemples qui suit.

1.1 Introduction

Tapemovie est un environnement logiciel pour la création temps réel intermédia. Il a été écrit dans Max/MSP/Jitter[2] et existe à la fois sous forme de patch et sous forme d'application autonome fonctionnant uniquement sous MacOSX – dans les 2 cas sous licence libre LGPL[3]. Ses champs d'action sont : la composition et l'improvisation musicale temps-réel, les dispositifs intermédia interactifs, et l'art numérique pour le spectacle vivant. Il vise un utilisateur prêt à acquérir une certaine compétence en informatique musicale et visuelle, et qui cherche un environnement solide et puissant lui permettant des réalisations exigeantes, sans avoir à programmer dans Max/MSP/Jitter. Il est le fruit de 20 ans d'expérience dans le temps réel de Tom Mays, de l'expérience dans l'art visuel et le spectacle vivant interactif de Renaud Rubiano[4], des connaissances, tests et documentations du réalisateur sonore Olivier Pfeiffer, et du soutien de la plateforme *didascalie.net*[5]. Il est utilisé depuis 2007 dans les projets de ses auteurs et dans ceux de *didascalie.net*, ainsi que par divers étudiants en composition au CNSMDP[6]. Tapemovie a participé également au projet de recherche Virage[7] autour des interfaces de contrôle et d'écriture.

1 cf. site de *tapemovie*, http://www.tapemovie.org
2 de cycling74, http://www.cycling74.com
3 Lesser GNU Public Licence
4 http://renaudrubiano.com/
5 http://didascalie.net/
6 Conservatoire National Supérieur de Musique et Danse de Paris
7 cf. site de Virage (http://www.virage-platform.org)

1.2 Historique

A partir des premiers environnements intégrés de contrôle et de traitement de signal, comme Max/FTS à l'Ircam pour la NeXT et les cartes ISPW au début des années 90, en passant par PureData, puis MSP qui se rajoute à Max en 1997 (Manning 2004, pp 361-374), chaque utilisateur expert cherchait petit à petit à faciliter et optimiser son travail de développement en créant des bibliothèques de fonctions (abstractions et objets) et en réutilisant des structures de patchs globaux. L'auteur, ayant fait de nombreuses créations et réalisations avec Max MIDI, Max/FTS à l'Ircam et Max/MSP depuis 1991, a commencé à formaliser de cette manière son environnement de traitement audio temps réel au début des années 2000. Il y avait déjà l'architecture modulaire, la séparation des moteurs de traitement et leurs interfaces, une matrice centrale qui permettait de connecter tout à tout, et un séquencement d'événements en forme de qlist[1]. L'arrivée de Jitter en 2003 (Manning 2004, pp 361-374) a permis de créer un environnement vidéo basé sur les mêmes principes.

A partir de 2005 ces environnements ont commencé à servir dans les créations des spectacles intermédia de la Compagnie *Incidents Mémorables* (devenue *didascalie.net* en 2008). Les membres de l'équipe expérimentant sur divers logiciels ont voulu partir de ce travail existant pour réaliser un environnement commun répondant à leurs différentes pratiques.

Il leur fallait donc un nom. Le premier nom par défaut, *l'environnement de Tom*, à été remplacé en 2007 par l'introduction des noms actuels pour les deux parties, audio et vidéo (avec une touche d'ironie) : *tape* (*Tom's Audio Patch Environnement*) et *movie* (*MOdular Video Environment*). En 2008 les deux patches *tape* et *movie* et leurs préférences ont été réunis en tant que *plugins*[2] à l'intérieur du patch *tapemovie* – en introduisant la notion de *projet* et celle de construction dynamique des instances des modules (fonction du *build*, cf. 1.3.3)

Tapemovie version 1 est le sujet de cette présentation. Elle est programmée dans Max 4, bien qu'elle puisse tourner dans Max5. La version d'application autonome permet un fonctionnement complet sans licence Max.

1 cf. *Max Object Reference*, www.cycling74.com
2 Extensions au programme de base

1.3 Architecture globale

Que nous utilisions *tapemovie* dans sa version *autonome* ou dans sa version de *patch* le fonctionnement reste le même. Nous allons donc plutôt parler de la version *autonome*.

Quand on ouvre l'application *tapemovie*, on voit une fenêtre qui représente le *patch* principal (Figure 12).

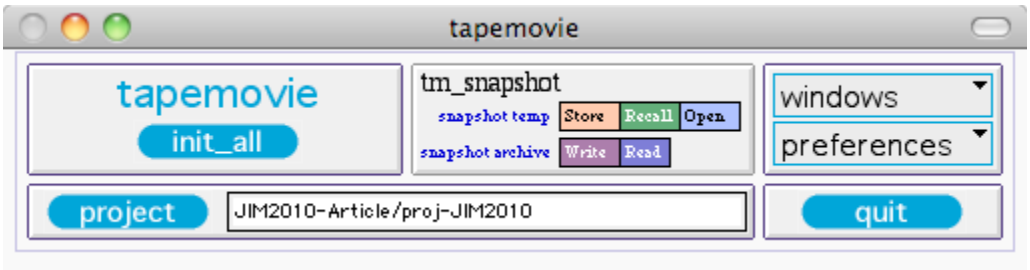


Figure 12: La fenêtre principale de tapemovie

Ce *patch* est une sorte de noeud qui sert à gérer le choix du *projet* (cf. 1.3.1 *le projet*), accéder aux réglages de toutes les configurations (cf 1.3.2 les configs), initier la construction par script des configurations choisies (cf 1.3.3 le *build*), faire un *snapshot*[1] global de tous les paramètres, accéder aux diverses fenêtres de l'application, et également, quitter *tapemovie*. Ce patch est le niveau supérieur de l'arborescence. Il permet d'ouvrir des sous-environnements (*plugins*) *tape* et *movie*, ainsi que d'instancier divers modules communs aux deux environnements comme des contrôleurs externes, les objets de réseau, et l'event manager. (cf. 1.3.2.1 tm_config). On configure d'abord *tapemovie* puis *tape* et *movie* (si la configuration le nécessite), chacun ayant encore sa propre configuration avec ses propres modules audio et vidéo.

1.3.1 Le projet

Le *project* est un dossier qui contient des sous-dossiers qui rassemblent toutes les préférences utilisateur et les données associées à un "projet" spécifique.

Le dossier */config* contient toutes les données des choix des modules pour l'environnement, plus les données spécifiques à certains modules de traitement, ainsi que les fichiers

1 Stockage d'un ensemble de paramètres en un « clic » de la souris

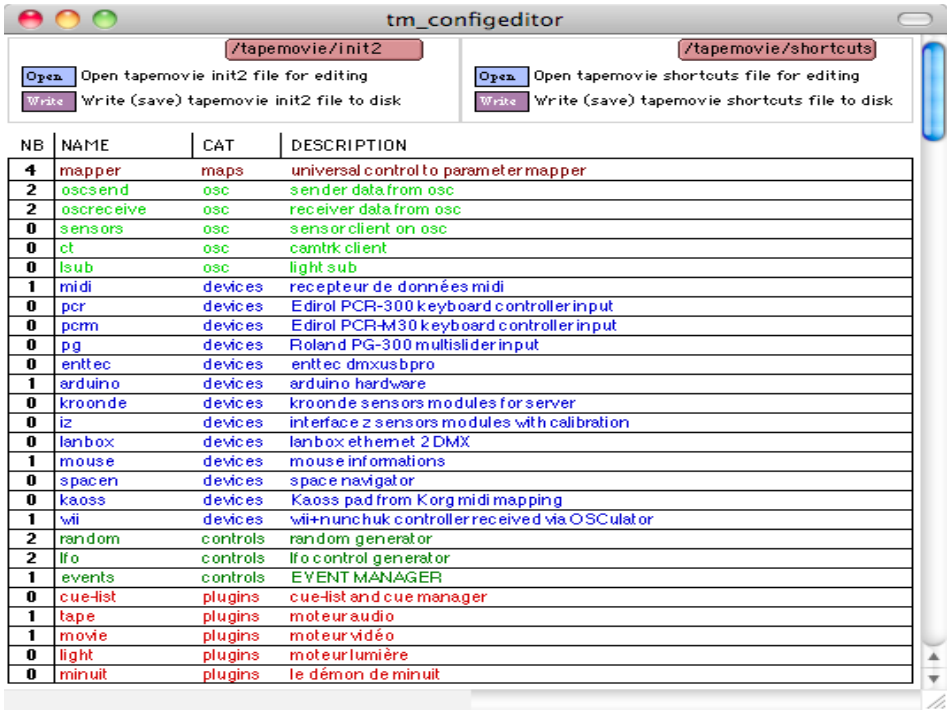
temporaires (*snapshots*). Le dossier */events* contient uniquement des fichiers des événements et des séquences. Le dossier */instruments* contient le patch *instruments* qui est un espace de programmation "libre" interfaçable avec le reste de l'environnement (cf. 1.3.5). Le dossier */media* contient tous les fichiers des images et des sons par catégorie.

Par le biais du *project* nous pouvons entièrement changer les caractéristiques de l'environnement de manière souple et dynamique sans quitter le programme. Un même *project* est compatible à la fois avec la version *patch* et la version autonome de *tapemovie*.

1.3.2 Les config

Les *config* sont les pages de configuration de chaque partie de l'environnement – *tapemovie*, *tape*, et *movie*. Ils sont accessibles par le menu *preferences* de tapemovie en choisissant *tm_config*, *t_config*, et *m_config* respectivement.

1.3.2.1 tm_config



NB	NAME	CAT	DESCRIPTION
4	mapper	maps	universal control to parameter mapper
2	oscsend	osc	sender data from osc
2	oscreceive	osc	receiver data from osc
0	sensors	osc	sensor client on osc
0	ct	osc	camtrk client
0	lsub	osc	light sub
1	midi	devices	recepteur de données midi
0	pcr	devices	Edirol PCR-300 keyboard controller input
0	pcrm	devices	Edirol PCR-M30 keyboard controller input
0	pg	devices	Roland PG-300 multislider input
0	enttec	devices	enttec dmxusbpro
1	arduino	devices	arduino hardware
0	kroonde	devices	kroonde sensors modules for server
0	iz	devices	interface z sensors modules with calibration
0	lanbox	devices	lanbox ethernet 2 DMX
1	mouse	devices	mouse informations
0	spacen	devices	space navigator
0	kaoss	devices	Kaoss pad from Korg midi mapping
1	wii	devices	wii+nunchuk controller received via OSCulator
2	random	controls	random generator
2	lfo	controls	lfo control generator
1	events	controls	EVENT MANAGER
0	cue-list	plugins	cue-list and cue manager
1	tape	plugins	moteur audio
1	movie	plugins	moteur vidéo
0	light	plugins	moteur lumière
0	minuit	plugins	le démon de minuit

Figure 13: La fenêtre tm_configeditor

Le *tm_config* est édité dans la fenêtre *tm_configeditor*. Nous l'utilisons principalement pour choisir la quantité de chaque modules disponibles que nous souhaitons avoir dans notre

tapemovie, spécifique à un projet. Les modules sont organisés par catégorie : *maps*, *osc*, *devices*, *controls* et *plugins* (Figure 13).

Les *maps* sont les *mappeurs* centraux qui permettent de connecter n'importe quel contrôle (objet *send* ou *forward* dans Max) à n'importe quel paramètre contrôlable en local ou sur le réseau (cf. 1.5). Les *osc* sont des modules d'envoi et/ou réception des messages OSC via UDP, comme le module minuit[1] qui permet la communication avec le séquenceur Virage[2]. Dans cette catégorie sont aussi les clients "sensors" et "camtrk" qui reçoivent et distribuent les données provenant d'une application de gestion et d'analyse de capteurs et de suivi de caméra. Les *devices* implementés dans *tapemovie* sont des contrôleurs comme *midi*, *arduino*, *kroonde*, *kaoss*, *wii* et *mouse*. Les *controls* sont des générateurs de *lfo* ou de *random*, ainsi que l'*event manager* qui est capable de stocker et d'organiser l'ensemble des paramètres. Les *plugins* sont *tape*, *movie* et *light* (gestion de lumière via DMX) On peut choisir jusqu'à 20 instances de modules normaux, mais uniquement une instance pour chaque *plugin*.

Dans le *tm_configeditor* nous pouvons également éditer une séquence d'initialisation personnalisée qui s'appelle *init2*, puis aussi un fichier de définition des *shortcuts* permettant d'ouvrir les fenêtres par une touche du clavier de l'ordinateur.

1.3.2.2 t_config

Le *t_config* fonctionne exactement sur le même principe que le *tm_config*, mais pour les modules *dsp* du sous environnement *tape* (cf. 1.4 pour plus info sur *tape*). Il y a, pourtant, un paramètre de plus: le choix global du dispositif de spatialisation (Figure 14). Il existe actuellement 3 dispositifs : spat4 (4 haut parleurs en quadriphonie), spat41 (spat4 + un centre) et spat441 (4 plus 4 plus un centre). (cf. 1.4.4 La Spatialisation)

Figure 14: choix de dispositif de spatialisation

1 http://www.plateforme-virage.org/?p=1444
2 cf. site de Virage (http://www.virage-platform.org)

1.3.3 Le build[1]

Une fois que nous avons réglé les configurations, nous lançons la construction de l'environnement par *script*. Cette fonction s'appelle un *build* et elle est lancée en cliquant sur le bouton *init* dans la fenêtre *tapemovie*. Cet étape peut prendre de 10 secondes à plus d'une minute selon les configurations.

Durant le *build*, les fenêtres *tape* et *movie* s'ouvrent (dans le cas où ils étaient choisis dans le *tm_config*). Le *build* a fini lorsque "tapemovie ready" s'affiche dans la fenêtre "status" (*standalone*) ou "max" (patch Max). Le patch a été crée automatiquement suivant les choix de configuration de l'utilisateur.

1.3.4 La séparation des moteurs et des éditeurs

Un des principes forts dans *tapemovie* est la séparation des moteurs de traitement de leurs éditeurs qui ne sont pas essentiels. On peut donc ne pas les créer pour des raisons d'optimisation, si on souhaite lors d'un *build.*

Il est notre souhait dans le futur proche de s'inspirer de l'architecture modèle/vue/contrôleur pour améliorer ce système. L'architecture *Modèle/Vue/Contrôleur* (MVC) est une façon d'organiser une interface graphique d'un programme (Reenskaug 2003). Elle consiste à distinguer trois entités distinctes qui sont, le *modèle*, la *vue* et le *contrôleur* ayant chacun un rôle précis dans l'interface[2]. Pour l'instant, on accède aux éditeurs par les menus *windows* de chaque partie de l'environnement.

1.3.5 Le patch instruments

Dans l'optique de faire de *tapemovie* un outil prêt à l'emploi bien que toujours ouvert à une utilisation experte, il fallait un "endroit" où l'on pouvait faire de la programmation Max librement. Le patch *instruments* se trouve dans le projet dans le dossier */instruments* (Figure 15). C'est un patch, pour ainsi dire, vide. On l'ouvre par le menu *windows* de *tapemovie*, ou bien automatiquement par le biais de l'init2 – l'init "user" (cf. 1.3.2.1).

1 Construction de l'ensemble des modules et interconnections par *script*.
2 Le site du *Laboratoire d'Informatique Algorithmique*, la page sur l'architecture MVC : (http://www.liafa.jussieu.fr/~carton/Enseignement/InterfacesGraphiques/MasterInfo/Cours/Swing/mvc.html)

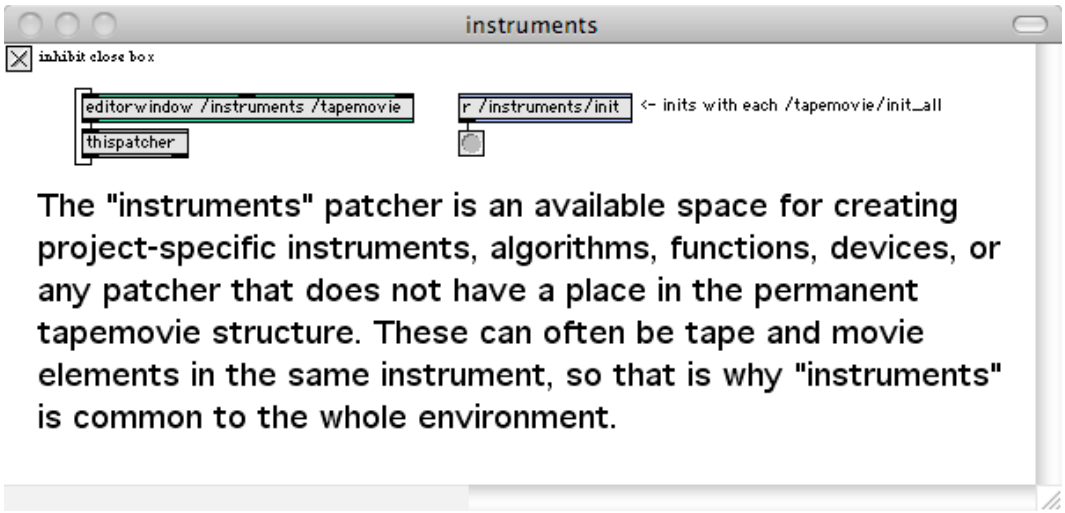


Figure 15: Le patch instruments

Il peut communiquer avec le reste de *tapemovie* par le moyens de *send/receive*[1] traditionnels, et en audio par les *send~/receive~*[2] associés aux modules *aux* dans *tape* (cf. 1.4.2). Ainsi nous pouvons profiter de l'architecture de *tapemovie* tout en gardant la liberté et l'efficacité de programmer en parallèle dans Max– pour ceux qui le souhaitent.

1.3.6 Le fonctionnement en réseau

Tapemovie incorpore les bases du protocole OSC[3] pour pouvoir contrôler les paramètres en dehors de l'application, ou bien en dehors de la machine. Chaque paramètre est capable également d'envoyer sa valeur en OSC par le réseau. Cet envoi de paramètres est très utile pour échanger des valeurs entre différents logiciels (compatible OSC) et aussi pour créer des interfaces sur des contrôleurs externes tels le Lemur[4]).

Il y a également une implémentation de Minuit[5] par le module *minuit*. Celui ci à été créé pour pouvoir tester le Séquenceur Virage[6].

1 cf. documentation de *Max* de Cycling74 (www.cycling74.com)
2 id.
3 cf. site de "open sound control" (http://opensoundcontrol.org)
4 cf. site de JazzMutant (http://www.jazzmutant.com)
5 cf. le site du plateforme-virage et la page spécifique à *minuit*. (http://www.plateforme-virage.org/?p=1444)
6 cf. site de Virage (http://www.virage-platform.org)

1.3.7 Le SDK

Il est essentiel pour l'avenir de *tapemovie* qu'un utilisateur puisse incorporer ses propres patchs dans l'environnement. Dans ce but, il existe un guide de développement qui documente le fonctionnement et la réalisation d'un module *tapemovie*. Ce SDK permet à un programmeur Max de créer ses propres modules *tapemovie*, *tape* ou *movie*.

1.4 La partie audio : *tape*

Si le *plugin tape* a été choisi dans la *config* de tapemovie (cf. infra 1.3.2.1), une fois le *build* effectué, une fenêtre *tape* va s'ouvrir (Figure 16).

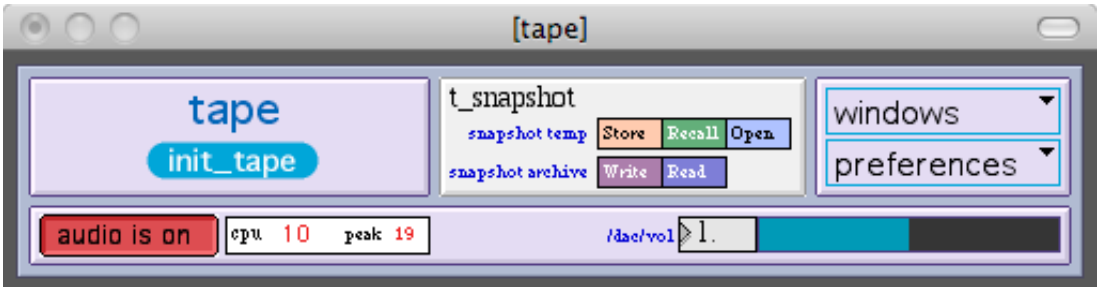


Figure 16: La fenêtre principale de tape

Cette fenêtre ressemble beaucoup à la fenêtre *tapemovie*. Elle contient un bouton *init_tape* qui initialise ou construit uniquement la partie *tape*, un *snapshot* de tous les paramètres *tape*, un accès par menu (ou *shortcuts*) de tous les éditeurs *tape*, un menu pour accéder à la configuration de *tape* (*t_config*), un bouton pour allumer/éteindre l'audio, un affichage de la cpu utilisée, puis enfin un contrôle de volume global de l'audio.

1.4.1 L'architecture

Les entrées et sorties audio (*adc, dac*) ainsi que tous les modules de traitement ou de synthèse, passent par une matrice centrale. Toutes les connections et tous les paramètres sont manipulables par messages envoyés (cf. 1.6 *event manager*), ce qui permet une souplesse virtuellement infinie de configurations possibles.

1.4.2 Les modules audio

Les modules audio livrés en standard avec *tapemovie* représentent un certain choix de simplicité en se reposant sur le principe que nous pouvons construire des effets et des instruments complexes en connectant des modules entre eux par le biais de la matrice (Tableau 1), et en créant des mises en correspondance entre paramètres via les mappeurs (cf. 1.5).

adc	adc input	afol	amplitude follower
src	mono source	pt	pitch analysis
src2	stereo source	cntrd	spectral centroid analysis
filt	standard filter	zcross	zerocross analysis
del	standard delay	delfib	8 tap delay with feedback based on fibonacci
fshift	freq shift/ringmod	fftfilt	fft filter 253 bands
harm	harmonizer module	fftx	fft generalized cross-synthesis - 2 ins 1 out
gizmo	gizmo spectral domain harmonizer module	fftsf	fft source-filter cross-synthesis - 2 ins 1 out
disto	tanh~ distortion module	freeze	mono freeze
mng	munger module	gran	granular synthesis
sfm	mono soundfile	reson	model-based resonating filter bank
sfst	stereo soundfile	sndmass	soundmass fm synthesizer
sf4	quad soundfile	synth	simple polyphonic fm synthesis
sf5	5 channel soundfile (L C R Ls Rs)	csynth	click synthesizer
smp	mono one-shot sampler	nsynth	noise synthesizer
smp4	mono 4 channel spat one-shot sampler with channel 5 to rev	rewire	8 tracks rewire mixer
vst	mono vst module	aux	auxilary send/return (mono)
rev	reverb module based on gigaverb	dac	dac output
ngate	noise gate	subb	dac output for sub

Tableau 1. La liste des modules *tape*

1.4.3 La matrice

Une partie de la souplesse de *tape* réside dans sa matrice (éditeur: *mtrx*). Toutes les entrées audio (*adc*), les modules de traitement et d'analyse, ainsi que les sorties audios (*dac*) y passent. L'éditeur de la matrice affiche le nom d'entrée, la valeur en dB et le nom de sortie dans chaque case. Dans Figure 17, nous voyons que adc.1 est connecté à filt.1 avec une valeur de -5 dB, puis les autres connections sont toutes éteintes, à une valeur de -127 dB

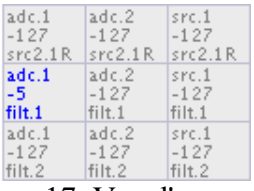


Figure 17: Vue d'une petite partie de l'éditeur de la matrice

1.4.4 La spatialisation

Chaque instance de chaque module de la configuration choisie peut accéder à la spatialisation. Ceci est rendu possible grâce au passage de tous les modules par la matrice, qui peuvent ainsi rejoindre les sorties (*dacs*) et la réverbération. Le calcul des amplitudes envoie automatiquement les bonnes valeurs à la matrice pour effectuer une spatialisation. Pour un exemple d'un éditeur de spat, dans un dispositif de 441 (cf. 1.3.2.2 t_config), regardons dans le spat du module de l'adc (Figure 18).

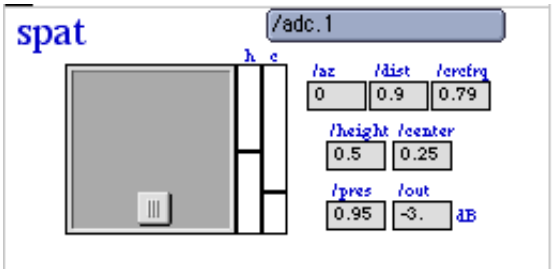


Figure 18: l'éditeur d'un spat441

Il y a les paramètres azimut et distance, plus un moteur de rotation (crcfrq = fréquence de cercle), avec aussi un paramètre de hauteur (*height*) pour la quadriphonie haut, et centre pour le haut-parleur de centre. Le paramètre "pres" est pour la présence qui va effectuer le dosage du son direct et du son réverbéré.

1.4.5 Les modules d'analyse et le contrôle

Parmi les modules de *tape* il y a des modules d'analyse permettant le contrôle de paramètres. Il y a des modules indépendants ainsi que des modules dédiés aux adc. Par exemple, chaque module adc contient un détecteur de niveau (*noise gate*), une détection de hauteur (*pt*), un suivi d'amplitude (*afol*), un suivi du centroïde, (*cntrd*), et un zérocross pour la détection du bruit (Figure 19).

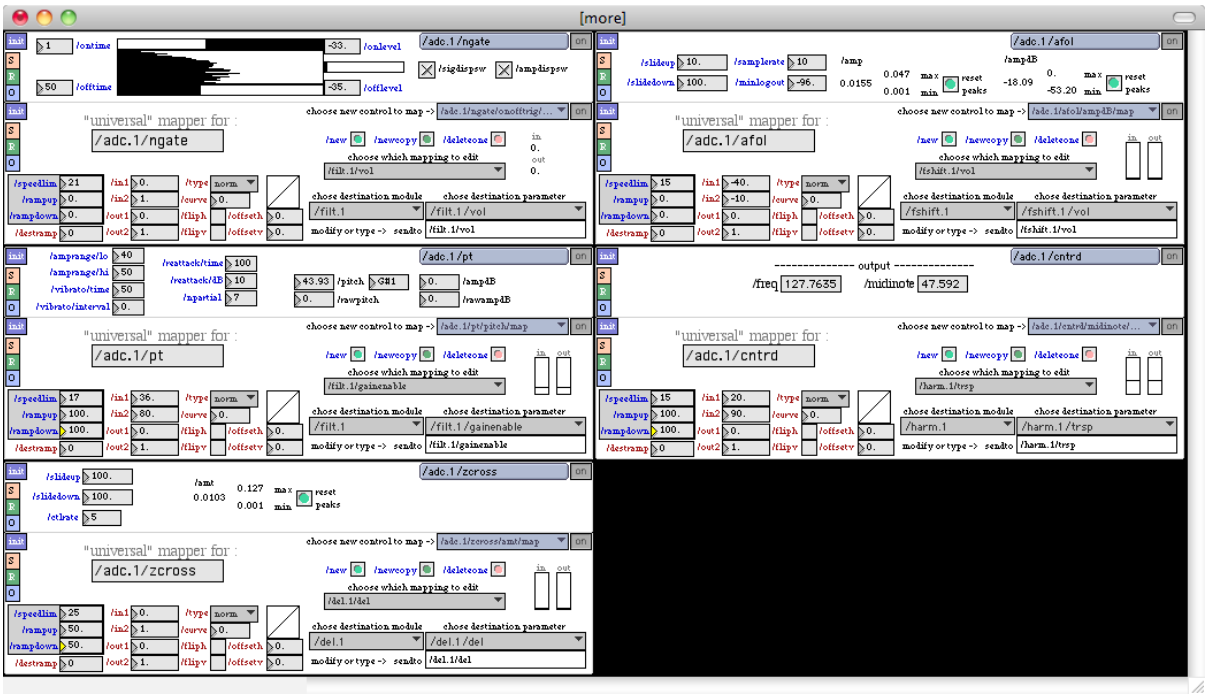


Figure 19: La fenêtre des modules d'analyse audio

1.5 Le contrôle et les *mappeurs*

Tous les *controleurs*, qu'ils viennent des entrées OSC ou MIDI, ou bien depuis les analyses internes (comme les suivis de hauteur ou d'amplitude se trouvant dans les *adc*) ont la possibilité de faire un "multi-mapping" (*one to many*) pour chaque paramètre de contrôle. Ces *mappeurs* ont été conçus et éprouvés par la pratique.

Ils sont situés à deux niveaux. Le premier est au niveau des modules *tapemovie*. Ce sont les *mappeurs centraux* et nous en choisissons le nombre dans la configuration pour *tapemovie* (Figure 20). Dans ces *mappeurs centraux* nous déterminons un contrôleur d'entrée (à *mapper*) puis un paramètre de destination. Dans les deux cas nous avons l'aide de menus listant les modules de contrôle ou de traitement disponible, puis un sous-menu avec les contrôleurs ou paramètres de ce module.

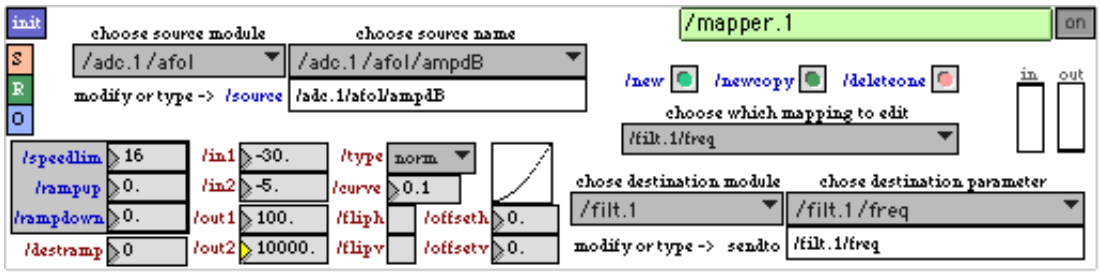


Figure 20: L'éditeur d'un mappeur central

Nous réglons évidemment l'échelle d'entrée et de sortie de chaque *mapping* mais également les choix de courbes entre exponentiel-linéaire-logarithmique, symétrique autour de zéro et bosse/creux (*up/down*). Des fonctions de *invert* et *offset* permettent de rapidement configurer la courbe selon l'échelle.

Quand nous avons un *mapping* il suffit de cliquer sur *new* ou *new copy* pour en créer un autre avec le même contrôleur – vers une destination différente. En principe il n'y a pas de limite au nombre de *multi-mappings*.

Le deuxième niveau des *mappeurs* est le niveau associé aux contrôleurs directement. Dans chaque éditeur de contrôleur ou module d'analyse, il y a un éditeur unique de *mappeur* permettant de choisir dans un menu le contrôle spécifique du module en question. Le réglages de ces *mappeurs* se font de la même manière que les *mappeurs centraux*.

1.6 L'event manager

Pour stocker des états de paramètres en créant et en organisant des événements, on choisit au minimum une instance du module "events" dans le *tm_config*. Ce module permet de créer un événement du groupe *tapemovie* (tous les paramètres), du groupe *tape* (paramètres d'audio) ou du groupe *movie* (paramètres du traitement de l'image). Les "events" sont stockés dans le dossier *events* du projet en tant que fichier texte, mais on peut aussi créer des sous-dossiers pour les différencier. Une fois créé, on peut faire "play", "edit", "rename" ou "delete" d'un événement. (Figure 21).

Pour en exécuter un, on le sélectionne dans le menu *play*. A ce moment, le fichier de l'événement sera lu depuis le disque. Puisque ils sont relus avant chaque lecture, on peut les

modifier avec n'importe quel éditeur de texte à l'extérieur de tapemovie, et l'event manager va relire la version modifiée automatiquement depuis le disque.

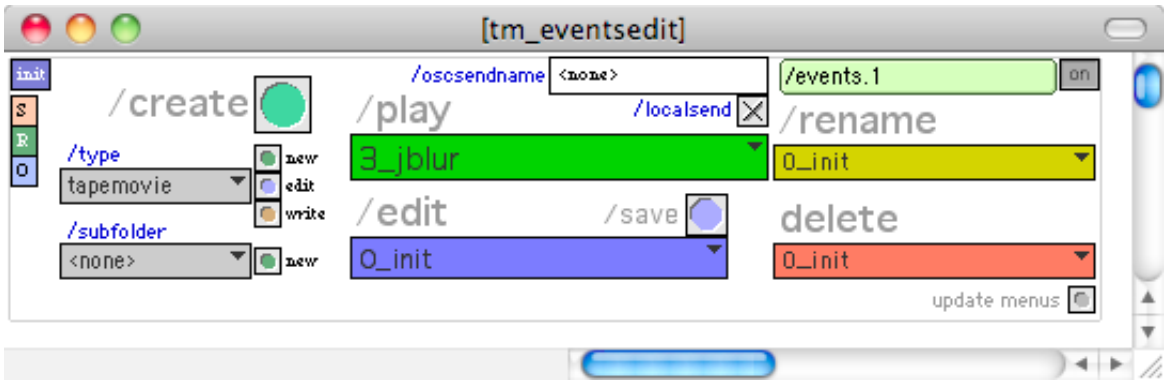


Figure 21: l'éditeur des events

1.7 La pérennité des créations

Une des questions les plus importantes dans la création avec les outils informatiques temps réel est comment rejouer l'oeuvre dans le futur. Les pièces du répertoire doivent subir des réactualisations (portages) régulièrement, sinon elles risquent de ne plus pouvoir être jouées à cause des supports de stockage, l'obsolescence des standards techniques, l'absence de soin dans la préservation des parties électroniques des œuvres, les conditions pratiques de la performance, et la confection souvent empirique des programmes (Bonardi et Barthélemy 2008).

Tapemovie aborde le sujet de la pérennité des créations par son recours aux fichiers texte qui définissent les événements. Chaque événement est une description d'un état ou une action incluant tous les paramètres nécessaires à son exécution. Ces fichiers textes sont lisibles en dehors de l'application par un éditeur de texte.

Si on combine la lisibilité des événements avec la documentation de l'environnement de *tapemovie* on peut dire que dans plusieurs années il sera encore possible d'étudier la documentation de tapemovie puis déchiffrer les configurations et événements d'un *project* pour, avec les fichiers sons, re-créer l'oeuvre dans un environnement futur – sans jamais avoir à faire tourner *tapemovie*.

1.8 Conclusions

Tapemovie est utilisé dans la composition, l'improvisation et le spectacle vivant. Il peut servir à la fois le créateur individuel et toute une équipe avec des ordinateurs en réseau. Il a une capacité d'extension par sa nature open-source et son utilisation par différentes équipes dans des spécialités variées. En servant une pluralité de projets, tapemovie s'augmente aussi en terme de configurations possibles.

Les réflexions entre les développeurs ont commencé pour les futures versions crées en Max 5. La prochaine est en préparation en collaboration avec les développeurs de Jamoma[1].

Tapemovie est un environnement de création qui tend la main vers une utilisation intuitive, tout en gardant la spécificité d'un outil expert, solide et ouvert.

1 cf. site officiel Jamoma (www.**jamoma**.org)

2 Utiliser tapemovie pour créer les traitements de base

Dans ce chapitre nous allons revoir les traitements par catégorie en réalisant quelques exemples concrets à l'aide de tapemovie et sa partie audio, tape. Dans une méthodologie tutorielle, nous verrons la fabrication et la syntaxe d'un événement, ainsi que les éditeurs des modules d'effets, de contrôle et de suivi.

2.1 Le temps

2.1.1 Délai simple

Commençons avec une configuration très simple dans laquelle l'entrée du son passe par un retard (délai) de 500 ms avec spatialisation, re-mélangé avec le son direct (Figure 22).

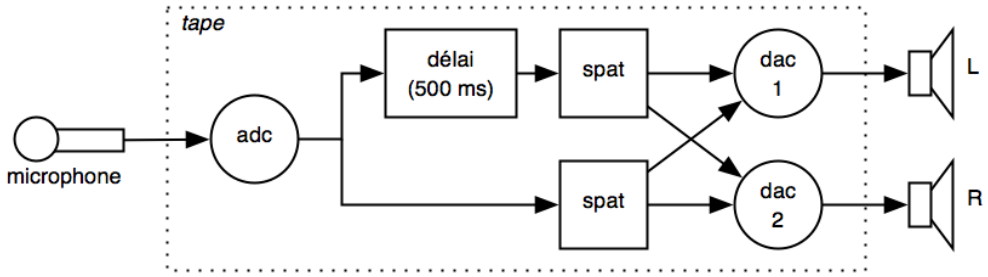


Figure 22: Schéma du délai à 500 ms

Dans *tape* nous devons connecter le module *adc* au module de délai[1] (*del*) par la matrice[2] Dans Figure 23, *adc.1* est connecté à *del.1* avec un niveau de 0 dB

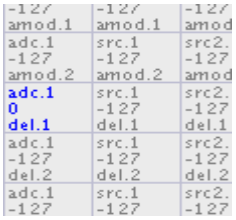


Figure 23: adc.1 vers del.1 dans la matrice

1 Dans *tape* le module de délai est basé sur les objets *tapin~* et *tapout~*, avec un temps de délai maximum de 30 secondes. Cf. références MaxMSP : http://www.cycling74.com/docs/max5/refpages/msp-ref/tapin~.html et http://www.cycling74.com/docs/max5/refpages/msp-ref/tapout~.html

2 cf. II 1.4.3 La matrice

Nous devons ensuite activer l'*adc.1*, puis la spatialisation. A la Figure 24 l'éditeur de *adc.1* est activé (interrupteur *on*), avec un niveau de sortie de 0 dB (paramètre */out*), un positionnement *devant-gauche* (azimut 45 dégrées et distance 1.42) et une *présence* de 0.99 (sur 1.0) ce qui donne une très légère réverbération[1].

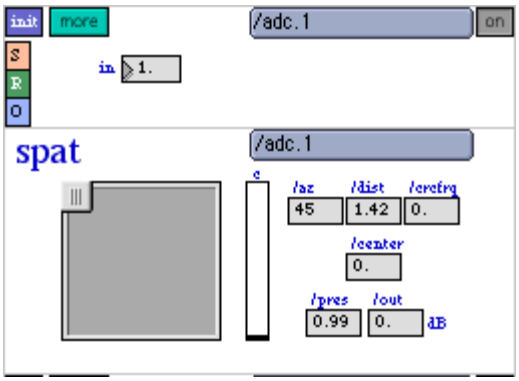


Figure 24: Editeur de l'adc.1

Puis finalement, nous configurons l'éditeur du délai, *del.1*. Le module *del.1* doit également être activé (comme pour tout module *tape et tapemovie*). Voici l'éditeur allumé avec un réglage du délai (*/del*) de 500 ms et une spatialisation *devant-droite* à 0 dB et une *présence* (*/pres*) *de* 0.97 ce qui lui donne un petit peu plus de réverbération que le son direct pour un léger effet de profondeur (Figure 25).

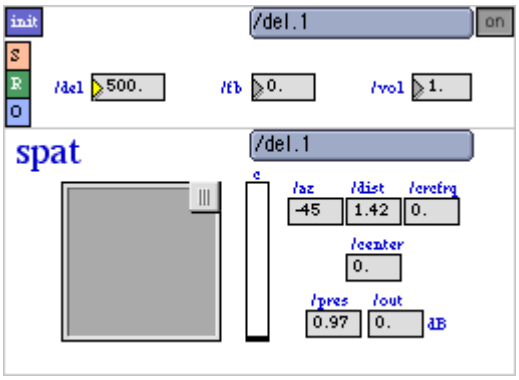


Figure 25: Editeur du del.1

Le traitement fonctionne exactement selon le schéma de la Figure 22, et nous pouvons le tester. Notez que les connexions entre les *spat* et les *dac* sont gérées automatiquement dans la matrice, sans que cela n'apparaisse dans l'éditeur de la matrice elle-même[2]. Nous allons maintenant créer une mémoire que nous pouvons rappeler facilement grâce au module *events*

1 cf. II 1.4.4 La spatialisation
2 id.

(cf. Figure 21: l'éditeur des events, page 45). Nous cliquons sur le bouton */create* et nous créons un événement avec un nom *01.1-del* ce qui crée un fichier dans notre dossier de *projet*, au niveau du dossier */events* (cf. II 1.3.1 Le projet, page35). Le texte de ce fichier contient le paramétrage nécessaire pour configurer notre délai. Le contenu en est le suivant :

```
** replace for comment;
** ---------------------------------------- /mtrx connections ---------------------------------------- ;
/mtrx connect adc.1 del.1 0 ;
** ---------------------------------------- /adc.1 ---------------------------------------- ;
/adc.1/sw 1 ;
/adc.1/in 1.000 ;
/adc.1/az 45 ;
/adc.1/dist 1.420 ;
/adc.1/crcfrq 0.000 ;
/adc.1/center 0.000 ;
/adc.1/pres 0.990 ;
/adc.1/out 0.000 ;
/adc.1/grain 50 ;
** ---------------------------------------- /del.1 ---------------------------------------- ;
/del.1/sw 1 ;
/del.1/del 500.000 ;
/del.1/fb 0.000 ;
/del.1/vol 1.000 ;
/del.1/az -45 ;
/del.1/dist 1.420 ;
/del.1/crcfrq 0.000 ;
/del.1/center 0.000 ;
/del.1/pres 0.970 ;
/del.1/out 0.000 ;
/del.1/grain 50 ;
```

Les lignes commençant avec « ** » ne sont que des *commentaires* et ne seront pas exécutées, mais toutes les autres lignes seront exécutées dans l'ordre, chaque ligne représentant un seul paramètre d'un module de traitement. Il y a d'abord la partie */mtrx connections* où nous trouvons toutes les connexions de la matrice – dans notre cas, uniquement la connexion entre *adc.1* et *del.1* à 0 dB. Ensuite il y a la partie */adc.1* dans laquelle nous retrouvons les mêmes paramètres et valeurs proposées dans l'éditeur du *adc* (cf. Figure 24). Enfin, la partie */del.1* exécute les paramètres ajustés dans l'éditeur du *del* (cf. Figure 25).

Nous pouvons maintenant repositionner tous les paramètres à l'état initial de tout l'environnement *tapemovie* avec le bouton *init* dans la fenêtre principale de *tapemovie* (cela prend plusieurs secondes) ou bien seulement initialiser les paramètres de *tape* par le bouton *init* dans la fenêtre principale de *tape* (ce qui est un peu plus rapide que l'initialisation complète de *tapemovie*) ou bien initialiser uniquement les modules que nous avons utilisés en cliquant sur les boutons *init* de chaque éditeur (*mtrx, adc* et *del*). Un fois l'initialisation effectuée, nous pouvons lancer notre événement dans l'éditeur *events.1* pour activer notre traitement *(II 1.6 L'event manager, page 44)*. Nous choisissons *01.1-del* dans le menu */play*

et nous relâchons pour le lancer[1]. Notre traitement par délai se met en place instantanément.

Si nous voulons obtenir une arrivée progressive du traitement (crescendo), le plus rapide, dans notre exemple, est d'automatiser le paramètre du volume d'entrée de l'*adc* (*/adc.1/in*). Pour faire cela, il faut d'abord modifier la valeur de ce dernier dans la partie de l'événement que nous avons déjà réalisé et le mettre à « 0 ». Choisissons *01.1-del* dans le menu */edit* pour ouvrir l'événement à l'édition et modifions la ligne */adc.1* :

```
/adc.1/in 0.;
```

Maintenant, à la fin de l'événement il faut ajouter une ligne de texte pour le */adc/in* avec une valeur de « 1 » (volume normal) suivie d'une valeur de temps en millisecondes pour la durée du crescendo (*fade in*[2]) avant de terminer la ligne avec le « ; ». Juste avant cette ligne, nous pouvons rajouter une ligne de commentaire avec « ** » pour améliorer la lisibilité. Le résultat pour l'événement complet avec un *fade in* en 2 secondes donne le texte suivant :

```
** simple délai de 500 ms avec fade in;
** ---------------------------------------- /mtrx connections ---------------------------------------- ;
/mtrx connect adc.1 del.1 0 ;
** ---------------------------------------- /adc.1 ---------------------------------------- ;
/adc.1/sw 1 ;
/adc.1/in 0.;
/adc.1/az 45 ;
/adc.1/dist 1.420 ;
/adc.1/crcfrq 0.000 ;
/adc.1/center 0.000 ;
/adc.1/pres 0.990 ;
/adc.1/out 0.000 ;
/adc.1/grain 50 ;
** ---------------------------------------- /del.1 ---------------------------------------- ;
/del.1/sw 1 ;
/del.1/del 500.000 ;
/del.1/fb 0.000 ;
/del.1/vol 1.000 ;
/del.1/az -45 ;
/del.1/dist 1.420 ;
/del.1/crcfrq 0.000 ;
/del.1/center 0.000 ;
/del.1/pres 0.970 ;
/del.1/out 0.000 ;
/del.1/grain 50 ;
** fade in;
/adc.1/in 1. 2000;
```

Notons que la conséquence d'avoir choisi d'intervenir sur le paramètre du niveau d'entrée de l'*adc* est que le son direct ET le traitement sont contrôlés par le *fade*. Si nous voulons garder le son direct en place et ne faire le *fade* qu'au niveau du délai, il faudra gérer le paramètre

1 Il existe des méthodes pour créer une séquence d'événements dans *tapemovie* avec déclenchement par pédale ou autre contrôleur, mais ces fonctions ne seront pas traitées dans ce mémoire.

2 Nous employons le terme anglais *fade* pour désigner une entrée ou une sortie graduelle d'un volume – *fade in* pour une entrée et *fade out* pour une sortie.

/del.1/vol à la place de */adc.1/in*.

Nous pouvons par la même manière dynamiser la quasi-totalité des paramètres dans *tape*, à l'exception des paramètres binaires (*on/off*), alpha-numériques ou sous forme de *liste*. Nous avons encore la possibilité de créer des temps d'attente à l'intérieur de l'événement pour retarder l'arrivée d'autres paramètres et actions. Cela se fait par l'ajout d'une ligne comprenant un temps en millisecondes. Si nous voulons par exemple laisser notre traitement de délai en place pendant 5 secondes avant de le faire disparaître par decrescendo, il suffit d'ajouter les lignes suivantes à notre événement :

```
** attendre 5 secondes avant fade out du del en 3 sec;
5000;
/del.1/vol 0. 3000;
```

Cette fois-ci le son direct reste présent car nous appliquons ce *fade out* uniquement au module *del.1*. Si nous voulons ensuite ré-initialiser le module *del.1* et déconnecter la matrice, nous faisons :

```
** attendre 5 secondes avant fade out du del en 3 sec;
5000;
/del.1/vol 0. 3000;
3000;
/mtrx disconnect adc.1 del.1;
/del.1/init bang;
```

Le 3000 dans une ligne à part crée un retard dans l'exécution équivalent des 3000 ms du *fade out*. Le message */mtrx disconnect* est là pour déconnecter la matrice, puis le message */del.1/init bang* initialise le *del.1*.

Au lieu d'organiser des séquences de messages dans un même événement, nous pouvons également créer des événements vides et y mettre les messages ou séquences que nous souhaitons.

2.1.2 Délai à temps court avec réinjection

Nous allons réaliser cet exemple de délai à temps court avec beaucoup de réinjection (*feedback*) pour générer un effet de scintillement autour d'une hauteur définie par le temps de délai – une méthode simple pour créer un effet de *filtre en peigne* (cf. supra). Nous allons éliminer l'amplification du son directe pour mieux entendre le traitement (Figure 26).

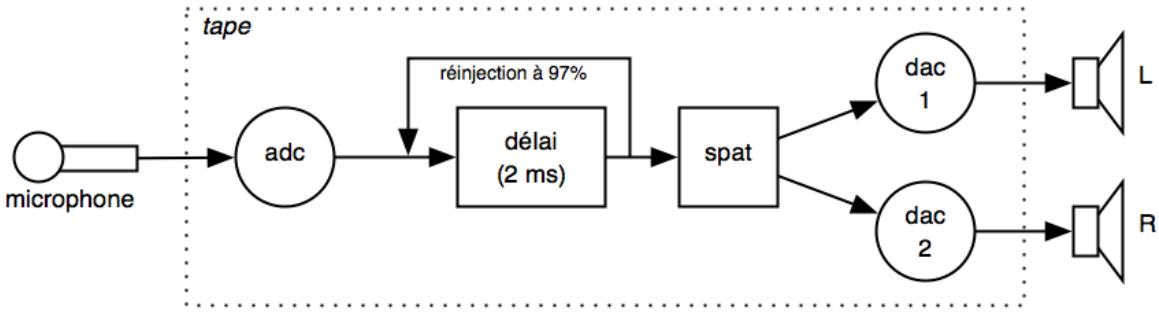


Figure 26: Schéma du délai à 2 ms avec réinjection

La matrice est la même que dans II 2.1.1 Délai simple (cf. infra), mais les paramètres du délai sont différents et il n'y a plus de diffusion du son direct. De plus, puisque le son traité est seul, sans le son direct, nous avons remis l'azimut à 0 dégrés, la distance à « 1 » et la présence à « 0.99 » pour n'avoir qu'une petite dose de réverbération (Figure 27).

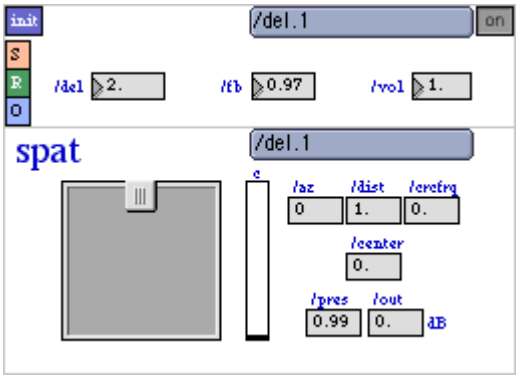


Figure 27: Editeur du del.1

Voici le contenu de l'événement :

```
** délai à temps court avec feedback;
** ---------------------------------------- /mtrx connections ---------------------------------------- ;
/mtrx connect adc.1 del.1 0 ;
** ---------------------------------------- /adc.1 ---------------------------------------- ;
/adc.1/sw 1 ;
/adc.1/in 1.000 ;
/adc.1/az 0 ;
/adc.1/dist 1.000 ;
/adc.1/crcfrq 0.000 ;
/adc.1/center 0.000 ;
/adc.1/pres 1.000 ;
/adc.1/out -127.000 ;
/adc.1/grain 50 ;
** ---------------------------------------- /del.1 ---------------------------------------- ;
/del.1/sw 1 ;
/del.1/del 2.000 ;
/del.1/fb 0.970 ;
/del.1/vol 1.000 ;
/del.1/az 0 ;
/del.1/dist 1.000 ;
/del.1/crcfrq 0.000 ;
/del.1/center 0.000 ;
```

```
/del.1/pres 0.990 ;
/del.1/out 0.000 ;
/del.1/grain 50 ;
```

Il faut toujours un module *adc* présent et allumé, mais puisque le paramètre */adc.1/out* est au minimum (-127 dB), l'*adc* n'est pas spatialisé directement. Le délai de 2 ms avec réinjection (*feedback*) très élevée (0.97 sur 1 ou 97%) crée un effet de scintillement à 500 Hz (1/0.002 secondes) ce qui produit une hauteur un peu plus élevée que Si3. Ce circuit est une version simplifiée du *filtre en peigne* (cf. I 2.3.1.1 Phase et couleur, page 20).

2.1.3 Délai à temps court contrôlé par un LFO

Pour obtenir un effet de déphasage (I 2.3.1.1 Phase et couleur, page 20) il faut avoir un délai court (1 à 15 ms environ) et recombiner la source avec le son retardé. La différence de phase entre les deux copies du son provoque des annulations et des renforcements qui créent un effet de filtrage. Si nous modulons le temps de délai par un LFO (*low frequency oscillator* ou *oscillateur de basse fréquence*), nous pouvons entendre des mouvements dans le filtrage, comme si les annulations glissaient à travers le spectre, produisant une variation du déphasage[1].

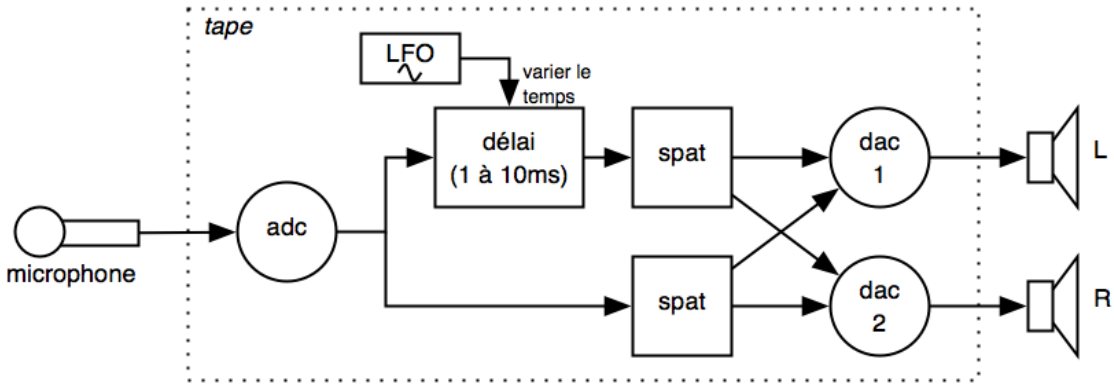


Figure 28: Schéma du délai qui varie par un LFO

Ici nous reprenons le délai simple avec diffusion du son direct, mais nous réduisons le délai jusqu'à de 1 à 10 ms, et nous connectons un module de *lfo*. Le *lfo* se trouve dans le menu *windows* de *tapemovie* (et non de *tape*) car c'est un contrôleur généralisé pour tout l'environnement – et pas uniquement pour l'audio. Ce *lfo* est capable de générer des formes sinusoïdales ou circulaires, et de les envoyer par le système de *mapping* à n'importe quel

1 Pour plus d'information sur les effets d'annulation de phase avec les délai courts : cf. Roads 1996, pp 436-440 ou Strange 1972, pp. 202-204

paramètre de l'environnement (cf. II 1.5 Le contrôle et les mappeurs, page 43). Voyons à la Figure 29 l'éditeur du module *lfo*. Dans le haut de la partie du générateur, nous avons réglé la fréquence du *sinus* (/freq) à 0.28 Hz et nous avons coché l'activation de la boucle (*/loop*) afin que les valeurs montent ET descendent. Regardons ensuite la partie *mapping* située en bas de la fenêtre. Notre *lfo* sinusoïdal envoie des valeurs comprise entre -1 et 1; nous réglons donc les valeurs d'entrée minimum et maximum du *mapping* en conséquence (*/in1* et */in2*). Nous réglons ensuite la sortie du *mappping* aux valeurs de délai que nous souhaitons : par exemple de 1.5 ms à 10 ms (*/out1* et */out2*). Nous choisissons une valeur de courbe légèrement exponentielle (puisque la perception du temps est exponentielle) en mettant 0.1 dans */curve*. Finalement, nous devons déterminer la destination, le nom du paramètre qui va recevoir la valeur du *lfo*. Dans ce cas c'est le */del.1/del*, le temps de délai du module *del.1*.

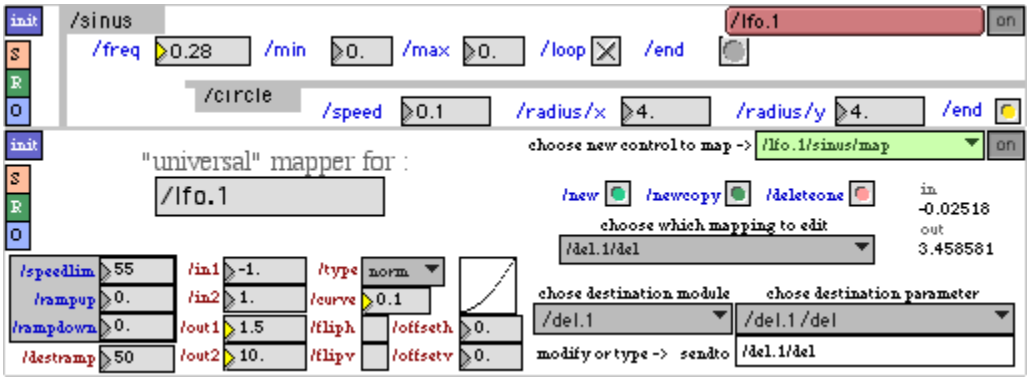


Figure 29: L'´éditeur du module lfo dans tapemovie

Le texte de l'événement est le suivant, avec la ligne */map* qui contient le nom de la destination ainsi que la liste des paramètres que l'on trouve en rouge dans l'éditeur du *mappeur* :

```
** délai temps court - phasing par lfo;
** ---------------------------------------- /lfo.1 ---------------------------------------- ;
/lfo.1/sw 1 ;
/lfo.1/sinus/min -1.000 ;
/lfo.1/sinus/max 1.000 ;
/lfo.1/sinus/freq 0.280 ;
/lfo.1/sinus/loop 2 ;
/lfo.1/sinus/map/sw 1 ;
/lfo.1/sinus/map/speedlim 55 ;
/lfo.1/sinus/map/rampup 0.000 ;
/lfo.1/sinus/map/rampdown 0.000 ;
/lfo.1/sinus/map /del.1/del -1.000 1.000 1.500 10.000 norm 0.100 0 0 0.000 0.000 50 ;
** ---------------------------------------- /mtrx connections ---------------------------------------- ;
/mtrx connect adc.1 del.1 0 ;
** ---------------------------------------- /adc.1 ---------------------------------------- ;
/adc.1/sw 1 ;
/adc.1/in 1.000 ;
/adc.1/az 0 ;
/adc.1/dist 1.000 ;
/adc.1/crcfrq 0.000 ;
/adc.1/center 0.000 ;
/adc.1/pres 1.000 ;
```

```
/adc.1/out 0.000 ;
/adc.1/grain 50 ;
** ---------------------------------------- /del.1 ---------------------------------------- ;
/del.1/sw 1 ;
/del.1/del 4.935 ;
/del.1/fb 0.000 ;
/del.1/vol 1.000 ;
/del.1/az 0 ;
/del.1/dist 1.000 ;
/del.1/crcfrq 0.000 ;
/del.1/center 0.000 ;
/del.1/pres 1.000 ;
/del.1/out 0.000 ;
/del.1/grain 50 ;
```

Notons que les spatialisations du *adc* et du *del* sont toutes deux réglées à 0 dégrées d'azimut avec présence à 1 (ce qui fait un signal *mono* sans réverbération) pour qu'ils se mélangent autant que possible et ainsi rendre l'effet d'annulations par déphasage audible.

2.2 La hauteur

Il y a dans *tape* deux modules pouvant réaliser une transposition de hauteur ou *harmonisation* : le module « traditionnel » *harm,* qui est basé sur la technique de délai variable et l'effet Doppler[1] et construit sur une modification de *harmv2*[2], et le module *gizmo* basé sur l'objet *gizmo~* de MSP[3] fonctionnant sur le principe d'une transposition dans le domaine fréquentiel[4]. Le module d'harmoniseur avec délai variable est efficace sur les petites transpositions de +/- 2 ou 3 demi tons, tandis que le module *gizmo* supporte mieux les transpositions plus importantes grâce à une méthode prenant en compte l'enveloppe formantique de la source. Mais, en dépit d'une meilleure qualité, l'analyse *FFT* de *gizmo* introduit une latence qui peut parfois être gênante. Il faut faire le bon choix pour le matériau que l'on souhaite traiter.

2.2.1 Harmoniseur simple

Dans la Figure 30 nous voyons le schéma de base de l'harmoniseur réalisé dans *tape*. Celui-ci est identique au schéma du délai simple, mais avec un harmoniseur à la place du délai.

1 cf. Puckette 2007, pp. 202-208; Roads 1996, p 444

2 harmv2 se trouve dans la bibliothèque d'objets *Jimmies* pour Max/FTS écrit par Zack Settel (et al), finalisé en 1993 (bien que basé en partie sur des objets écrits pour la 4X dans les années 80). Il a a été porté par Richard Dudas pour Max/MSP en 1998. Cf. la documentation de l'Ircam *Jimmies for MSP handbook, 1998*

3 cf. documentation de Max/MSP en ligne : http://www.cycling74.com/docs/max5/refpages/msp-ref/gizmo~.html

4 cf. Roads 1996, pp. 444-445, *Time/Pitch Changing with the Phase Vocoder*

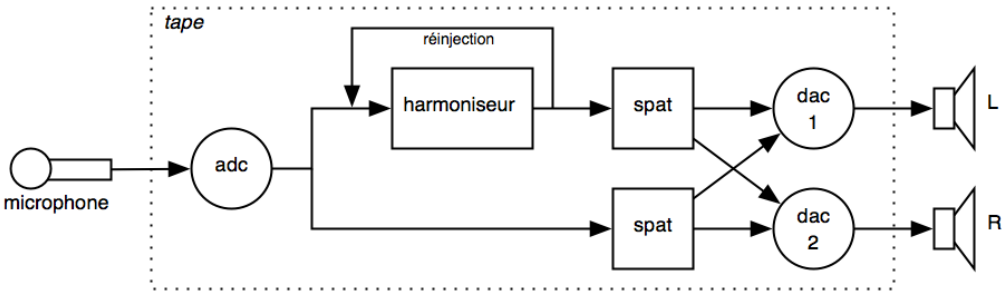


Figure 30: Schéma de l'harmoniseur avec spatialisation

Voici l'éditeur du module *harm* réglé sur une transposition de 300 cents (3 demi-tons). Notez les paramètres de délai (*/del*) et de réinjection (*/fb*). (Figure 31)

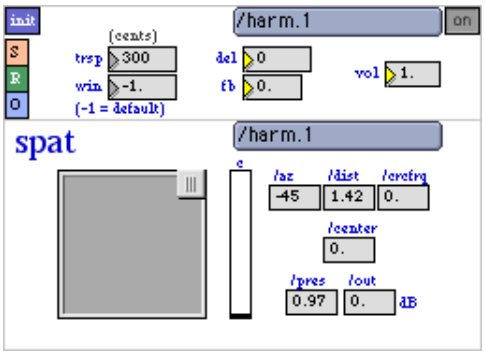


Figure 31: Editeur du module harm

Les modules *harm* et *gizmo* contiennent un délai avec réinjection placé AVANT la transposition – capable de créer un effet dans lequel les sons transposés sont à nouveau transposés à chaque passage dans le délai, en fonction du taux de réinjection. Cet effet s'appelle *tunneling* (mise en tunnel) selon Allen Strange[1], mais on peut aussi parler d'arpégiation du son.

Evénement pour l'harmoniseur simple avec les réglages de Figure 31 :

```
** harmoniseur simple - harm;
** ---------------------------------------- /mtrx connections ---------------------------------------- ;
/mtrx connect adc.1 harm.1 0 ;
** ---------------------------------------- /adc.1 ---------------------------------------- ;
/adc.1/sw 1 ;
/adc.1/in 1.000 ;
/adc.1/az 45 ;
/adc.1/dist 1.420 ;
/adc.1/crcfrq 0.000 ;
/adc.1/center 0.000 ;
/adc.1/pres 0.990 ;
/adc.1/out 0.000 ;
/adc.1/grain 50 ;
** ---------------------------------------- /harm.1 ---------------------------------------- ;
```

1 Allen Strange disait cela dans les cours de composition électroacoustique à San José dans les années 80. Nous n'avons pas trouvé de document qui le confirme aujourd'hui, mais nous continuons à le redire.

```
/harm.1/sw 1 ;
/harm.1/trsp 300 ;
/harm.1/win -1.000 ;
/harm.1/del 0.000 ;
/harm.1/fb 0.000 ;
/harm.1/vol 1.000 ;
/harm.1/az -45 ;
/harm.1/dist 1.420 ;
/harm.1/crcfrq 0.000 ;
/harm.1/center 0.000 ;
/harm.1/pres 0.970 ;
/harm.1/out 0.000 ;
/harm.1/grain 50 ;
```

Pour transformer cette configuration en un effet de *tunneling*, mettons le temps de délai sur une valeur d'ordre plutôt rythmique et la réinjection sur une valeur plutôt élevée, 0.6 à 0.8. Exemple :

```
/harm.1/del 250.000 ;
/harm.1/fb 0.700 ;
```

L'éditeur et les paramètres de *gizmo* ressemblent beaucoup à l'éditeur et les paramètres de *harm*. Il n'y manque que le paramètre de la taille de la fenêtre du délai variable (*win*) qui est sans utilité dans *gizmo* (Figure 32) :

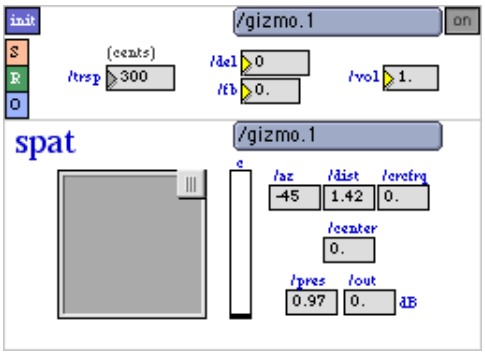


Figure 32: L'éditeur de gizmo

2.2.2 Harmoniseur contrôlé par la détection de hauteur

Voyons maintenant l'exemple d'un harmoniseur contrôlé par la détection de hauteur . La détection de hauteur se fait par un module appelé *pt* (*pitch track* en anglais, suivi de hauteur en français). Ce module est basé sur l'objet de détection de hauteur de Miller Puckette, *fiddle~*[1].

En voici le schéma *(Figure 33)*.

1 cf. article *Real-time audio analysis tools for Pd and MSP*, Miller S. Puckette, Theodore Apel, David D. Zicarelli, International Computer Music Conference, 1998; http://www-crca.ucsd.edu/~tapel/icmc98.pdf

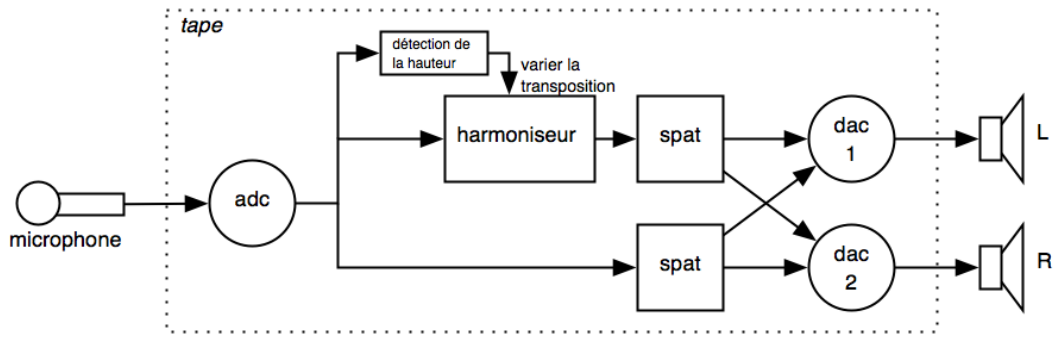


Figure 33: Schéma d'un harmoniseur contrôlé par une détection de hauteur

Le module *pt* est intégré dans le module *adc;* pour en voir l'éditeur, il faut cliquer sur le bouton « more » de l'éditeur *adc*. Une fois dans l'éditeur de *pt*, il faut l'allumer, puis choisir et activer un paramètre à *mapper* sur notre module d'harmoniseur. En suivant la même logique que le *mapping* du délai contrôlé par un *lfo* (cf. infra), nous pouvons déterminer une hauteur minimum et maximum en entrée du *mapping* et aussi les valeurs de transposition en sortie. Nous choisissons une détection de hauteurs comprises entre la note 36 (Do1 en notes MIDI) et la note 60 (Do3), *mappées* de -700 cents (une quinte plus bas) à +700 cents (une quinte plus haut). Nous plaçons une courbe symétrique légèrement exponentielle afin que les données restent stables au point milieu (transposition 0). Enfin, nous choisissons comme destination la transposition de l'harmoniseur (*/harm.1/trsp*). Voir Figure 34.

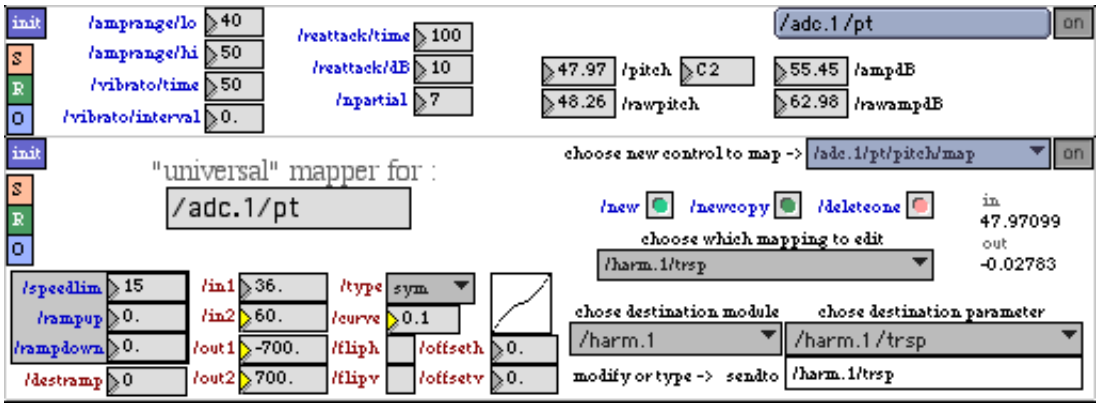


Figure 34: L'éditeur du suivi de hauteur de l'adc

Avec un mapping de cette sorte, une source jouant ou chantant une hauteur Do2 produit en traitement une transposition nulle (0). Quand la source s'éloigne de la hauteur Do2, l'intervalle de transposition augmente progressivement, jusqu'à atteindre une quinte supérieure pour une source à Do3 et une quinte inférieure pour Do1.

L'événement sous forme de texte pour cette configuration est le suivant :

```
** Harmoniseur contrôlé par la détection de hauteur;
** ---------------------------------------- /mtrx connections ---------------------------------------- ;
/mtrx connect adc.1 harm.1 0 ;
** ---------------------------------------- /adc.1 ---------------------------------------- ;
/adc.1/sw 1 ;
/adc.1/in 1.000 ;
/adc.1/az 45 ;
/adc.1/dist 1.420 ;
/adc.1/crcfrq 0.000 ;
/adc.1/center 0.000 ;
/adc.1/pres 0.990 ;
/adc.1/out 0.000 ;
/adc.1/grain 50 ;
** ---------------------------------------- /adc.1/pt ---------------------------------------- ;
/adc.1/pt/sw 1 ;
/adc.1/pt/amprange/lo 40 ;
/adc.1/pt/amprange/hi 50 ;
/adc.1/pt/vibrato/time 50 ;
/adc.1/pt/vibrato/interval 0.500 ;
/adc.1/pt/reattack/time 100 ;
/adc.1/pt/reattack/dB 10 ;
/adc.1/pt/npartial 7 ;
/adc.1/pt/pitch/map/sw 1 ;
/adc.1/pt/pitch/map/speedlim 15 ;
/adc.1/pt/pitch/map/rampup 0.000 ;
/adc.1/pt/pitch/map/rampdown 0.000 ;
/adc.1/pt/pitch/map /harm.1/trsp 36.000 60.000 -700.000 700.000 sym 0.100 0 0 0.000 0.000 ;
** ---------------------------------------- /harm.1 ---------------------------------------- ;
/harm.1/sw 1 ;
/harm.1/trsp 0.000 ;
/harm.1/win -1.000 ;
/harm.1/del 0 ;
/harm.1/fb 0.000 ;
/harm.1/vol 1.000 ;
/harm.1/az -45 ;
/harm.1/dist 1.420 ;
/harm.1/crcfrq 0.000 ;
/harm.1/center 0.000 ;
/harm.1/pres 0.970 ;
/harm.1/out 0.000 ;
/harm.1/grain 50 ;
```

2.3 L'amplitude

La catégorie d'amplitude comprend tout d'abord le contrôle du volume et sa dynamisation. Ceci se fait dans tout module par le paramètre vol (volume). Ce paramètre peut recevoir des messages de valeur associés avec une valeur de temps pour créer des interpolations (cf infra, explication du fade in et fade out du délai simple.). Un enchaînement de message d'interpolation peut produire des enveloppes. Un autre traitement d'amplitude simple est le LFO d'amplitude, dont voici maintenant l'exemple.

2.3.1 LFO d'amplitude simple

Le *LFO* d'amplitude peut produire un simple trémolo. Pour ce traitement, nous ne diffusons pas de son direct, qui pourrait nuire à la perception du trémolo (Figure 35).

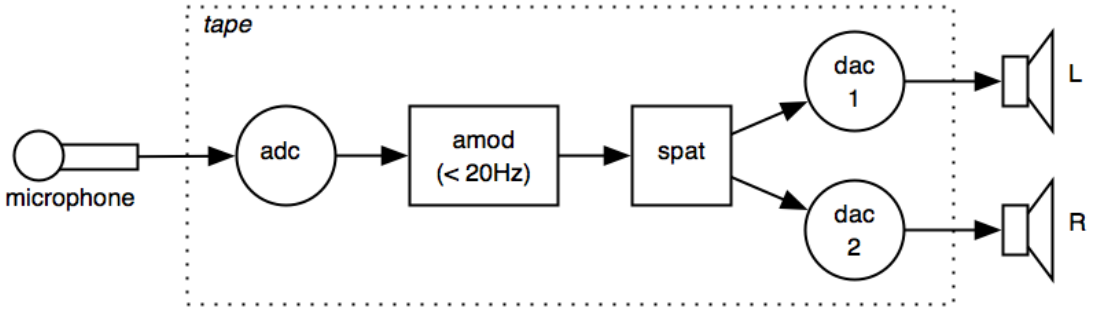


Figure 35: Schéma du lfo d'amplitude

Nous utilisons le module *amod* (modulation d'amplitude) ajusté dans des basse fréquences, de moins de 20 Hz. Il faut connecter l'*adc* dans l'*amod* par la matrice. Nous ouvrons ensuite l'éditeur de l'*amod* pour régler les valeurs de fréquence (*/freq*), amplitude (*/amp*) et les paramètres du *spat* (Figure 36).

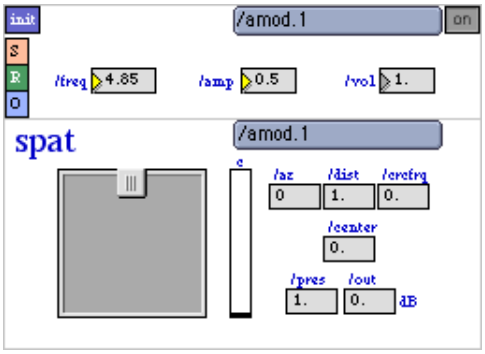


Figure 36: L'éditeur de l'amod

Nous réglons le paramètre *freq* sur 4,85 Hz pour obtenir un temps de trémolo « moyen », et le paramètre *amp* sur 0,5 pour une profondeur du lfo également « moyenne ». Une valeur *amp* à 0 fait passer tout simplement le son sans en modifier l'amplitude (multiplication par 1). Une *amp* de 0,5 module entre 0,5 et 1, une *amp* de 1, entre 0 et 1.

L'événement sous forme de texte est le suivant :

```
** basic amod;
** ---------------------------------------- /mtrx connections ---------------------------------------- ;
/mtrx connect adc.1 amod.1 0 ;
** ---------------------------------------- /adc.1 ---------------------------------------- ;
/adc.1/sw 1 ;
/adc.1/in 1.000 ;
/adc.1/az 0 ;
/adc.1/dist 1.000 ;
/adc.1/crcfrq 0.000 ;
/adc.1/center 0.000 ;
/adc.1/pres 1.000 ;
/adc.1/out -127.000 ;
/adc.1/grain 50 ;
```

```
** ------------------------------------- /amod.1 ------------------------------------- ;
/amod.1/sw 1 ;
/amod.1/freq 4.850 ;
/amod.1/amp 0.500 ;
/amod.1/vol 1.000 ;
/amod.1/az 0 ;
/amod.1/dist 1.000 ;
/amod.1/crcfrq 0.000 ;
/amod.1/center 0.000 ;
/amod.1/pres 1.000 ;
/amod.1/out 0.000 ;
/amod.1/grain 50 ;
```

2.3.2 *LFO d'amplitude contrôlé par le suivi d'amplitude*

Reprenons l'exemple précédent en ajoutant un module de suivi d'amplitude/enveloppe (afol) pour créer un contrôleur modifiant à la fois la fréquence du lfo et sa profondeur (amp). L'afol est construit à l'aide d'objets standard de la distribution MSP.

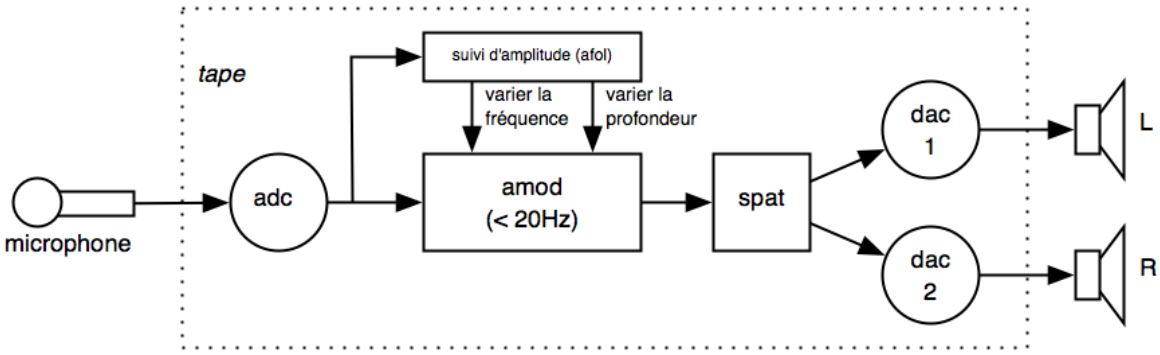


Figure 37: Schéma du LFO d'amplitude contrôlé par le suivi d'amplitude

Dans l'éditeur de l'*adc*, cliquons sur « more » pour afficher l'éditeur de suivi d'amplitude (*afol*). Il faut activer ce dernier, puis observer la mesure d'amplitude en dB en faisant entrer des sons faibles et forts afin d'établir les bornes pour le mapping. (Figure 38).

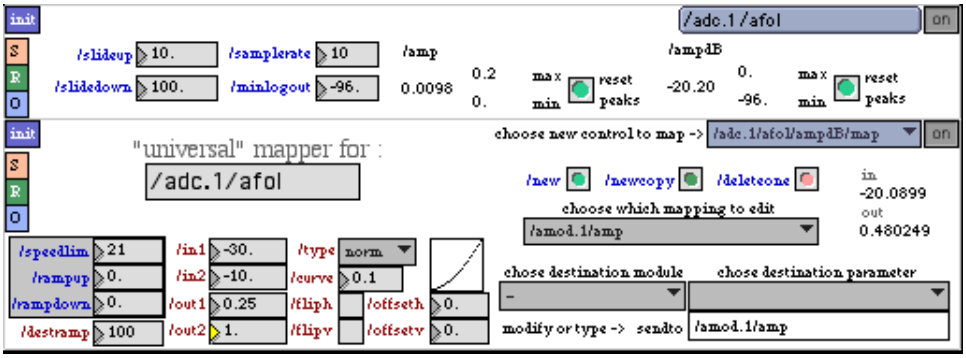


Figure 38: L'éditeur du afol avec son mappeur

Décidons d'utiliser -30 dB comme repère pour un amplitude minimum, et -10 dB comme repère pour l'amplitude maximum. Dans la partie du mapper de la même fenêtre, choisissons /adc.1/afol/ampdB/map dans le menu de choix de contrôleur, et donnons -30 comme valeur in1 et -10 comme valeur in2. Nous devons d'abord contrôler le paramètre amp : pour la valeur minimum et maximum de sortie nous choisissons 0.25 et 1.0. Mettons 100 comme valeur de temps d'interpolation (ramp) de destination, puis une valeur de courbe (curve) de 0.1. Enfin, indiquons /amod.1/amp comme nom de paramètre de destination (sendto) et finissons par allumer la connexion en plaçant l'interrupteur du mapper sur on. L'amplitude du son source détermine maintenant la profondeur du lfo d'amplitude.

Pour que l'amplitude du son source détermine la fréquence du lfo d'amplitude, il faut créer une connexion supplémentaire dans le mappeur. Pour cela, nous cliquons sur le bouton « newcopy » (nouveau mapping en tant que copie d'un autre), puis changeons la destination en /amod.1/freq et les valeurs de destination (out1 et out2) en 0.5 et 10 (Figure 39). Dès lors, un son faible est traité avec un lfo d'une profondeur d'amplitude de 0.25 et une fréquence de 0.5 Hz, et un son fort est traité avec une profondeur de 1 et une fréquence de 10 Hz.

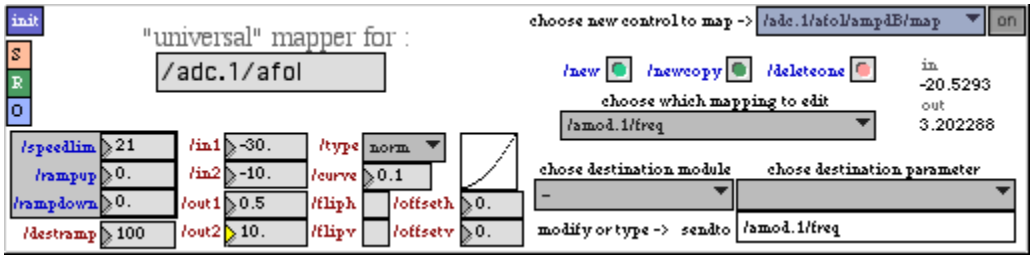


Figure 39: Mapping vers le paramètre /amod.1/freq

Nous créons un *event* pour stocker cet état, qui le texte suivant :

```
** amod with amplitude control – amp et freq;
** ---------------------------------------- /mtrx connections ---------------------------------------- ;
/mtrx connect adc.1 amod.1 0 ;
** ---------------------------------------- /adc.1 ---------------------------------------- ;
/adc.1/sw 1 ;
/adc.1/in 1.000 ;
/adc.1/az 0 ;
/adc.1/dist 1.000 ;
/adc.1/crcfrq 0.000 ;
/adc.1/center 0.000 ;
/adc.1/pres 1.000 ;
/adc.1/out -127.000 ;
/adc.1/grain 50 ;
** ---------------------------------------- /adc.1/afol ---------------------------------------- ;
/adc.1/afol/sw 1 ;
/adc.1/afol/slideup 10.000 ;
/adc.1/afol/slidedown 100.000 ;
```

```
/adc.1/afol/samplerate 10 ;
/adc.1/afol/minlogout -96.000 ;
/adc.1/afol/ampdB/map/sw 1 ;
/adc.1/afol/ampdB/map/speedlim 21 ;
/adc.1/afol/ampdB/map/rampup 0.000 ;
/adc.1/afol/ampdB/map/rampdown 0.000 ;
/adc.1/afol/ampdB/map /amod.1/amp -30.000 -10.000 0.250 1.000 norm 0.100 0 0 0.000 0.000 100 ;
/adc.1/afol/ampdB/map /amod.1/freq -30.000 -10.000 0.500 10.000 norm 0.100 0 0 0.000 0.000 100 ;
** ------------------------------------- /amod.1 ------------------------------------- ;
/amod.1/sw 1 ;
/amod.1/freq 0.500 ;
/amod.1/amp 0.250 ;
/amod.1/vol 1.000 ;
/amod.1/az 0 ;
/amod.1/dist 1.000 ;
/amod.1/crcfrq 0.000 ;
/amod.1/center 0.000 ;
/amod.1/pres 1.000 ;
/amod.1/out 0.000 ;
/amod.1/grain 50 ;
```

2.4 Le timbre par filtrage

Les traitements par filtrage peuvent être difficiles à faire sonner car un son filtré est facilement masqué par le son acoustique de l'instrument, ou par sa sonorisation (cf I 1.2.2.4 Le traitement du timbre par filtrage, page 9). Pour cette raison, dans un contexte de traitement temps réel d'instruments acoustiques passant par des micros aériens, les filtrage sont rarement utilisés seuls. Nous pouvons employer différentes manières pour *séparer* la source du traitement, mais la plus simple est sûrement par décalage temporel – le délai. Ainsi, le filtrage avec délai est beaucoup plus perceptible qu'un filtrage sans délai.

2.4.1 Filtre simple avec délai

Faisons passer une source dans un délai avant qu'elle n'entre dans un filtre. Nous plaçons le filtre *après* le délai afin que les éventuelles modifications du filtrage soient entendues au moment du contrôle et non pas *après* le temps de délai (Figure 40).

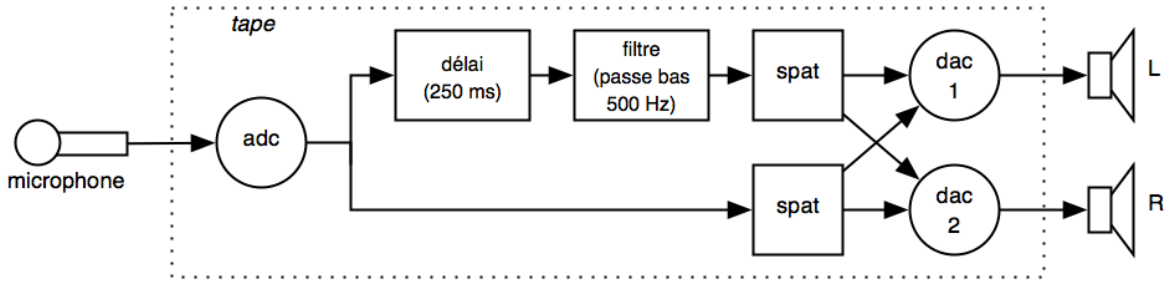


Figure 40: Schéma d'un filtre précédé d'un délai

Il faut d'abord connecter la matrice. Notons qu'il faut connecter l'*adc* au *del* (délai), et ensuite connecter le *del* au *filt* (filtre). La figure 41 nous montre les connections dans l'éditeur de la matrice.

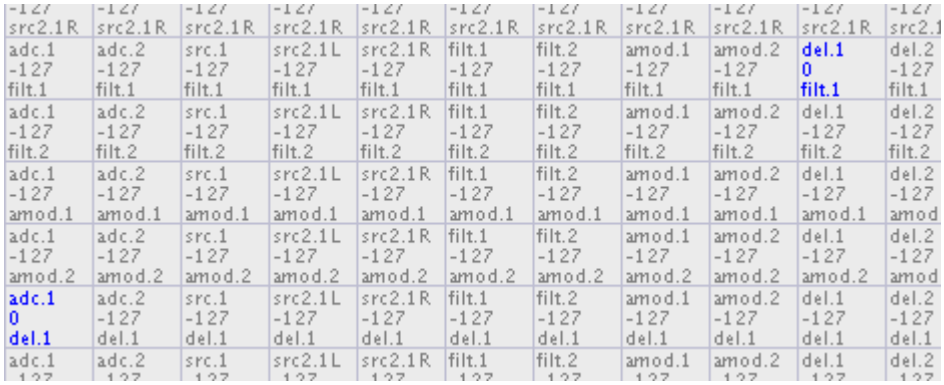


Figure 41: Connexions de la matrice (adc->del->filt)

Réglons le délai à 250 ms et un filtre passe bas à 500 Hz. Puisque le son est maintenant retardé, nous pouvons spatialiser le signal direct pour produire un espace sonore plus cohérent entre la source et le traitement. Mettons un peu de résonance (*q* d'une valeur de 2.) dans le filtre passe-bas pour accentuer un peu plus le filtrage. L'éditeur du filtre affiche ces valeurs (Figure 42).

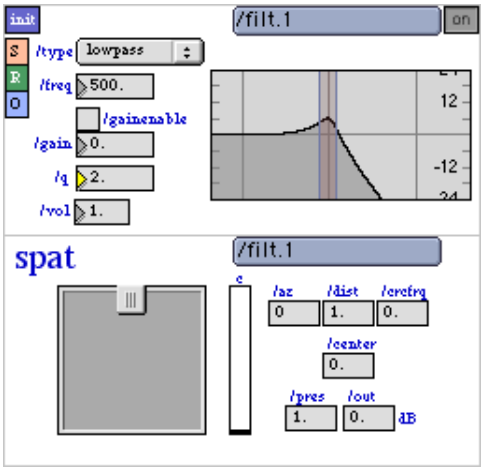


Figure 42: L'éditeur du module filt

Voici le texte de l'événement de cette configuration :

```
** del + filt;
** ------------------------------------- /mtrx connections ------------------------------------- ;
/mtrx connect adc.1 del.1 0 ;
/mtrx connect del.1 filt.1 0 ;
** ------------------------------------- /adc.1 ------------------------------------- ;
/adc.1/sw 1 ;
/adc.1/in 1.000 ;
/adc.1/az 0 ;
```

```
/adc.1/dist 1.000 ;
/adc.1/crcfrq 0.000 ;
/adc.1/center 0.000 ;
/adc.1/pres 1.000 ;
/adc.1/out 0.000 ;
/adc.1/grain 50 ;
** ---------------------------------------- /del.1 ---------------------------------------- ;
/del.1/sw 1 ;
/del.1/del 250.000 ;
/del.1/fb 0.000 ;
/del.1/vol 1.000 ;
/del.1/az 0 ;
/del.1/dist 1.000 ;
/del.1/crcfrq 0.000 ;
/del.1/center 0.000 ;
/del.1/pres 1.000 ;
/del.1/out -127.000 ;
/del.1/grain 50 ;
** ---------------------------------------- /filt.1 ---------------------------------------- ;
/filt.1/sw 1 ;
/filt.1/type lowpass ;
/filt.1/gainenable 0 ;
/filt.1/freq 500.000 ;
/filt.1/gain 0.000 ;
/filt.1/q 2.000 ;
/filt.1/vol 1.000 ;
/filt.1/az 0 ;
/filt.1/dist 1.000 ;
/filt.1/crcfrq 0.000 ;
/filt.1/center 0.000 ;
/filt.1/pres 1.000 ;
/filt.1/out 0.000 ;
/filt.1/grain 50 ;
```

2.4.2 *Filtre contrôlé par une détection de transitoires*

Conservons le filtre avec son délai, et ajoutons-lui un contrôle par détection de transitoires. Cette détection se fait par le module *zcross* (*traversée de zéro* ou *zero cross* en anglais) construit à partir de l'objet MSP *zerox~*[1] (Figure 43).

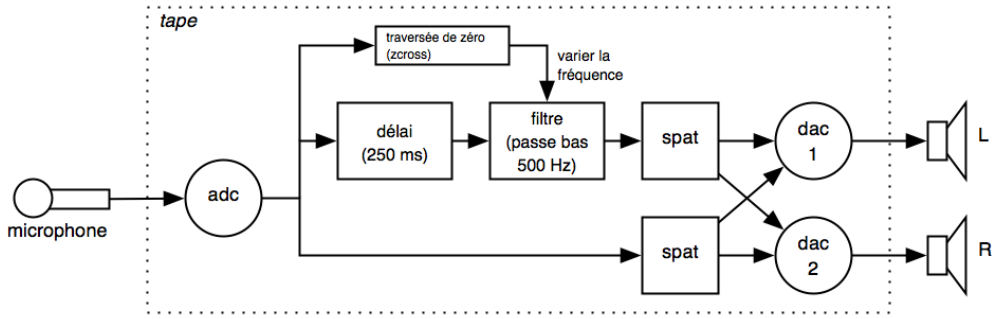


Figure 43: Schéma du filt contrôlé par zcross

Le module comprend les paramètres *slide up* (glissement des valeurs qui augmentent) et *slide down* (glissement des valeurs qui diminuent) pour régler la réactivité de la détection des transitoires ; mais nous pouvons laisser ces paramètres sur leurs valeurs par défaut de 100 ms.

1 cf. documentation MaxMSP : (http://www.cycling74.com/docs/max5/refpages/msp-ref/zerox~.html)

Le module *zcross* compte le nombre de fois que le signal traverse le « 0 » dans un laps de temps choisi ; il délivre une valeur proche de « 0 » pour un son comprenant peu de traversées du zéro (sans transitoires, ou sans contenu suraigu) et une valeur plus proche de « 1 » quand il détecte de nombreuses traversées de zéro (beaucoup de transitoires ou de contenu suraigu).

Dans cet exemple, nous utilisons les valeurs de *zcross* comprises entre 0. et 0.9 pour changer la fréquence du filtre entre 200 Hz et 12 Khz, avec une courbe relativement exponentielle (*curve* 0.2) (Figure 44).

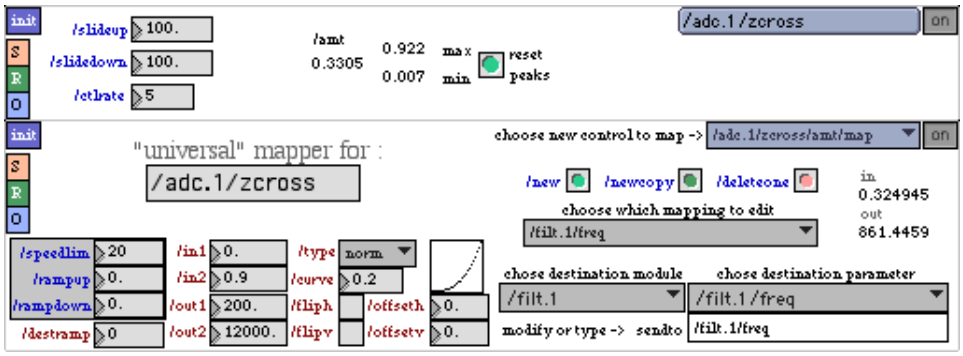


Figure 44: L'éditeur du zcross avec son mappeur

Ainsi, plus l'attaque du son contient du bruit, plus le filtre s'ouvre dans l'aigu. Ceci donne un effet de coloration vivante, réactive et musicale.

Le texte de ce contrôle est le suivant :

```
** ------------------------------------- /adc.1/zcross ------------------------------------- ;
/adc.1/zcross/sw 1 ;
/adc.1/zcross/slideup 100.000 ;
/adc.1/zcross/slidedown 100.000 ;
/adc.1/zcross/ctlrate 5 ;
/adc.1/zcross/amt/map/sw 1 ;
/adc.1/zcross/amt/map/speedlim 20 ;
/adc.1/zcross/amt/map/rampup 0.000 ;
/adc.1/zcross/amt/map/rampdown 0.000 ;
/adc.1/zcross/amt/map /filt.1/freq 0.000 0.900 200.000 12000.000 norm 0.200 0 0 0.000 0.000 ;
```

2.5 Le timbre par modulation

Dans cette catégorie, nous allons voir un exemple de transposition de fréquence pouvant également produire une modulation en anneau, ainsi qu'un exemple de distorsion non-linéaire donnant un effet de saturation.

2.5.1 *Transposition de fréquence et modulation en anneau*

Le module qui réalise à la fois la transposition de fréquence et la modulation en anneau s'appelle *fshift*, construit avec l'objet MSP *freqshift~* (d'après *frequency shift* ou *transposition de fréquence* en français)[1].

Ce module comprend un délai avec réinjection (*feedback*), suivi de la fonction de transposition de fréquence ; celle-ci peut produire, soit les bandes latérales hautes (somme du spectre avec la fréquence de modulation), soit les bandes latérales basses (différence du spectre avec la fréquence de modulation), soit les deux en même temps (la somme ET la différence), ce qui est équivalent à une modulation d'anneau. Contrairement à l'harmoniseur, la réinjection n'inclue pas ici le traitement de transposition de fréquence dans la boucle ; voir Figure 45.

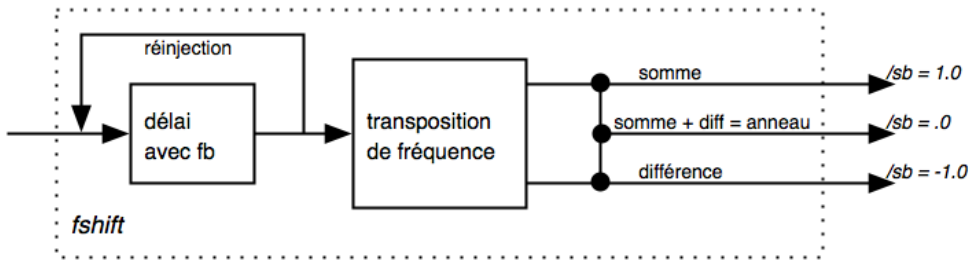


Figure 45: Schéma du module fshift

Pour obtenir un effet plus prononcé, nous préférons le plus souvent ne pas spatialiser le son direct quand nous utilisons le *fshift*, comme dans la Figure 46. Nous réalisons la connexion entre l'*adc* et le *fshift* de la même manière que les connexions précédentes.

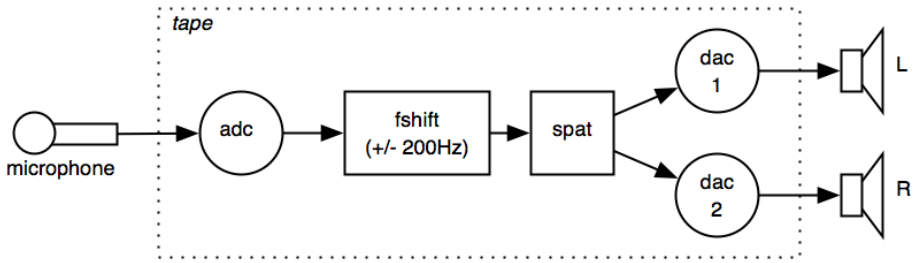


Figure 46: Schéma d'un simple traitement par fshift

L'éditeur du fshift comprend le paramètre de *freq* pour la fréquence de modulation, et un paramètre *sb* pour *side band* (bandes latérales), qui contrôle le mix de la somme et la

1 cf. documentation MaxMSP : http://www.cycling74.com/docs/max5/refpages/msp-ref/freqshift~.html

différence. A la valeur de 1 nous n'avons que la somme, à -1 nous avons la différence, et à 0 nous avons les 2. L'éditeur propose aussi les valeurs de *del* et *fb* (feedback – réinsertion) pour permettre de retarder le signal en amont du vrai traitement de *fshift* (Figure 47).

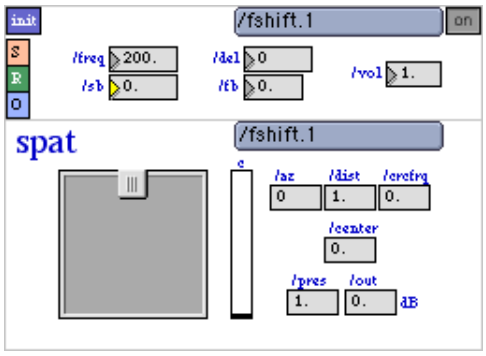


Figure 47L'éditeur du fshift

Le texte de l'événement de cette config est le suivant :

```
** simple fshift config;
** ---------------------------------------- /mtrx connections ---------------------------------------- ;
/mtrx connect adc.1 fshift.1 0 ;
** ---------------------------------------- /adc.1 ---------------------------------------- ;
/adc.1/sw 1 ;
/adc.1/in 1.000 ;
/adc.1/az 0 ;
/adc.1/dist 1.000 ;
/adc.1/crcfrq 0.000 ;
/adc.1/center 0.000 ;
/adc.1/pres 1.000 ;
/adc.1/out -127.000 ;
/adc.1/grain 50 ;
** ---------------------------------------- /fshift.1 ---------------------------------------- ;
/fshift.1/sw 1 ;
/fshift.1/freq 200.000 ;
/fshift.1/sb 0.000 ;
/fshift.1/del 0.000 ;
/fshift.1/fb 0.000 ;
/fshift.1/vol 1.000 ;
/fshift.1/az 0 ;
/fshift.1/dist 1.000 ;
/fshift.1/crcfrq 0.000 ;
/fshift.1/center 0.000 ;
/fshift.1/pres 1.000 ;
/fshift.1/out 0.000 ;
/fshift.1/grain 50 ;
```

2.5.2 Transposition de fréquence et contrôle par le centroïde

Nous allons maintenant contrôler un des paramètres du *fshift* par une autre analyse proposée dans l'*adc*, l'analyse du *centroïde spectral* et l'objet *centroid~*[1]. Cette analyse donne une indication de la brillance ou de la richesse du timbre par calcul du centre pondéré du spectre à

1 L'objet *centroid~* n'est pas dans la distribution de MaxMSP. Il a été programmé par Ted Apel, John Puterbaugh, and David Zicarelli; disponible à http://crca.ucsd.edu/~tapel/software.html. Pour plus d'info sur le centroïde spectral en anglais cf. : http://en.wikipedia.org/wiki/Spectral_centroid

l'aide d'une *fft* (cf. I 1.2.2.8 Le traitement dans le domaine fréquentiel, page 12). Voici le schéma des connexions (Figure 48) :

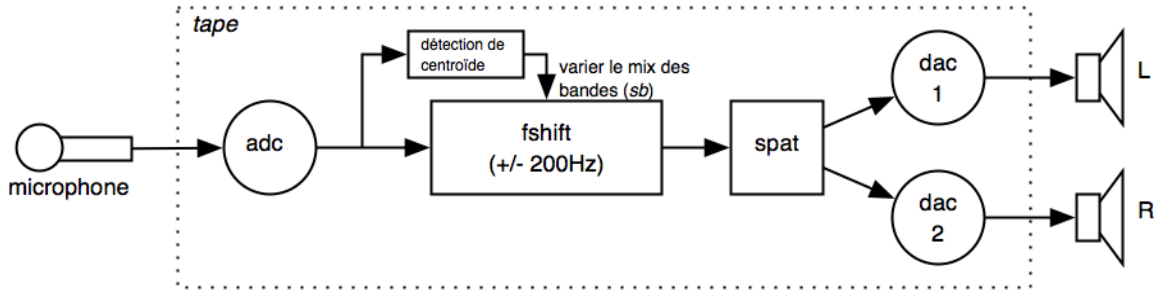


Figure 48: Schéma du fshift contrôlé par l'analyse du centroïde

Il n'y a dans l'éditeur aucun paramètre ajustable pour le module du centroïde (nommé *cntrd*), mais uniquement la sortie de l'analyse en deux unités différentes : la fréquence (*freq*) et la note selon la référence des notes MIDI (*midinote*) (Figure 49).

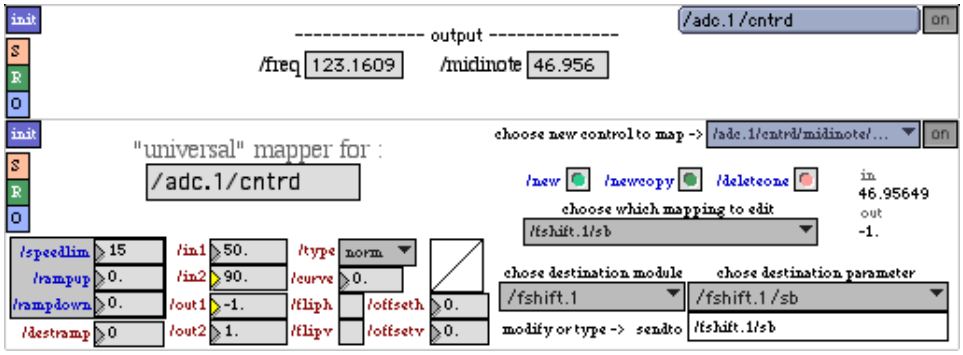


Figure 49: L'éditeur du centroïde (cntrd) avec son mappeur

Nous pensons que la conversion en note MIDI de la valeur de centroïde spectral rend le *mapping* de ce paramètre plus intuitif car directement lié à la perception de la hauteur ; mais il ne faut pas pour autant confondre le centroïde spectral avec la détection de hauteur de la fondamentale. Le centroïde spectral d'un son sinusoïdal à la note 50 (Ré2) donne environ 123 Hz comme résultat de *freq* et 50 comme *midinote* ; mais la même note avec une forme d'onde complexe et un timbre brillant peut donner une valeur supérieur à la note 90 (1500 Hz), voire plus.

Dans cet exemple, des valeurs du centroïde spectral comprises entre midinote 50 et midinote 90 envoient des valeurs de résultats comprises entre -1. et 1. au mix des bandes latérales (sb) du module fshift. Ceci a pour conséquence que les sons plutôt purs ou « étouffés » font sonner

les bandes latérales basses du fshift, alors que les sons riches et brillants font sonner les bandes latérales hautes. Voici le texte de l'événement :

```
** fshit et centroid;
** ---------------------------------------- /mtrx connections ---------------------------------------- ;
/mtrx connect adc.1 fshift.1 0 ;
** ---------------------------------------- /adc.1 ---------------------------------------- ;
/adc.1/sw 1 ;
/adc.1/in 1.000 ;
/adc.1/az 0 ;
/adc.1/dist 1.000 ;
/adc.1/crcfrq 0.000 ;
/adc.1/center 0.000 ;
/adc.1/pres 1.000 ;
/adc.1/out -127.000 ;
/adc.1/grain 50 ;
** ---------------------------------------- /adc.1/cntrd ---------------------------------------- ;
/adc.1/cntrd/sw 1 ;
/adc.1/cntrd/midinote/map/sw 1 ;
/adc.1/cntrd/midinote/map/speedlim 15 ;
/adc.1/cntrd/midinote/map/rampup 0.000 ;
/adc.1/cntrd/midinote/map/rampdown 0.000 ;
/adc.1/cntrd/midinote/map /fshift.1/sb 50.000 90.000 -1.000 1.000 norm 0.000 0 0 0.000 0.000 ;
** ---------------------------------------- /fshift.1 ---------------------------------------- ;
/fshift.1/sw 1 ;
/fshift.1/freq 200.000 ;
/fshift.1/sb -0.305 ;
/fshift.1/del 0.000 ;
/fshift.1/fb 0.000 ;
/fshift.1/vol 1.000 ;
/fshift.1/az 0 ;
/fshift.1/dist 1.000 ;
/fshift.1/crcfrq 0.000 ;
/fshift.1/center 0.000 ;
/fshift.1/pres 1.000 ;
/fshift.1/out 0.000 ;
/fshift.1/grain 50 ;
```

2.5.3 La distorsion / saturation

L'implémentation de la distorsion non-linéaire (cf. I 1.2.2.5 Le traitement du timbre par modulation, page 10) dans *tape* est réalisée par un module de saturation appelé *disto*. Ce module est construit avec l'objet MSP *tanh~* (tangente hyperbolique) qui donne pour des valeurs entre -pi et pi des valeurs de -1 et 1 (Figure 50).

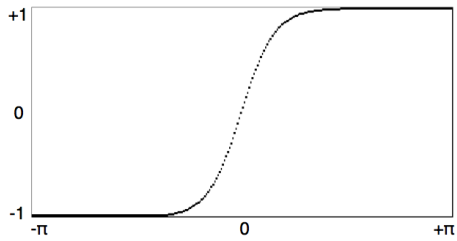


Figure 50: Courbe de tangente hyperbolique

L'éditeur du module *disto* contient principalement le paramètre de *quantité de distorsion* (*amt*, ou *amount* en anglais), outre le volume et la partie du spat (présents dans tous les modules).

Le paramètre *amt* est ajustable entre 1 et 100 ; c'est un facteur de sur-amplification du signal entrant, provoquant la saturation (Figure 51). Dans l'exemple, l'*amt* est ajusté à 100 et nous avons aussi réduit la présence à 0.9, afin d'éloigner un peu le son et créer un espace légèrement réverbéré.

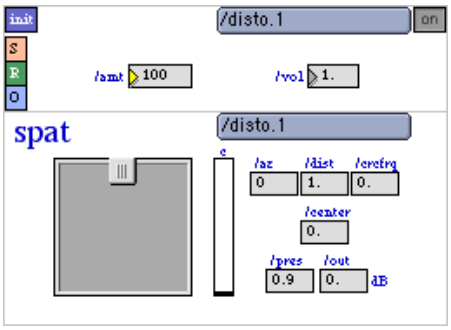


Figure 51: L'éditeur du module disto

Puisque ce traitement fonctionne par *sur-amplification*, il y a de très fortes qu'un effet de Larsen se produise. Également, un bruit de souffle bien prononcé peut apparaître durant les « silences » de la source. Pour ces raisons, il est fortement conseillé d'insérer ce que l'on appelle un *noise gate* (littéralement *porte à bruits*) qui coupe le signal entrant en dessous d'un certain seuil, et ceci avec un certain temps de réactivité dans l'ouverture et la fermeture de la *porte*.

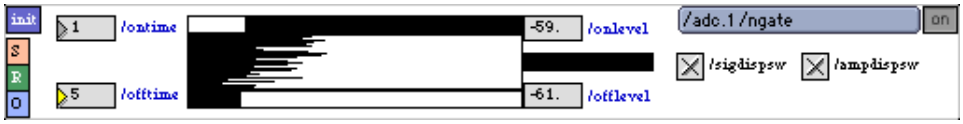


Figure 52: L'éditeur du noise gate (ngate) qui fait partie de l'adc

L'éditeur du noise gate (ngate) se trouve dans l'éditeur de l'adc avec tous les autres modules de détection (suivi); il faut cliquer sur le bouton more dans la fenêtre de l'éditeur de l'adc. Il suffit d'allumer le module pour le rendre actif avec les valeurs par défaut; nous avons choisi toutefois de réduire le off time (temps de fermeture de la porte) à 5 ms pour que la coupure du son soit plus nette qu'elle n'est avec la valeur par défaut de 50 ms.

Le schéma de cette configuration complète est donnée à la figure 53.

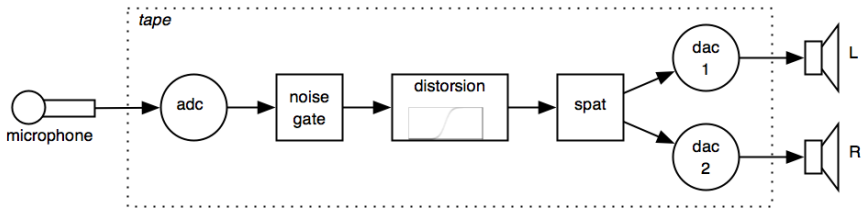


Figure 53: Schéma de la distorsion avec noise gate

Le texte d'un événement *disto* avec *noise gate* pourrait prendre la forme suivante :

```
** disto with noise gate;
** ---------------------------------------- /mtrx connections ---------------------------------------- ;
/mtrx connect adc.1 disto.1 0 ;
** ---------------------------------------- /adc.1 ---------------------------------------- ;
/adc.1/sw 1 ;
/adc.1/in 1.000 ;
/adc.1/az 0 ;
/adc.1/dist 1.000 ;
/adc.1/crcfrq 0.000 ;
/adc.1/center 0.000 ;
/adc.1/pres 1.000 ;
/adc.1/out -127.000 ;
/adc.1/grain 50 ;
** ---------------------------------------- /adc.1/ngate ---------------------------------------- ;
/adc.1/ngate/sw 1 ;
/adc.1/ngate/onlevel -59.000 ;
/adc.1/ngate/offlevel -61.000 ;
/adc.1/ngate/ontime 1 ;
/adc.1/ngate/offtime 5 ;
** ---------------------------------------- /disto.1 ---------------------------------------- ;
/disto.1/sw 1 ;
/disto.1/amt 100 ;
/disto.1/vol 1.000 ;
/disto.1/az 0 ;
/disto.1/dist 1.000 ;
/disto.1/crcfrq 0.000 ;
/disto.1/center 0.000 ;
/disto.1/pres 0.900 ;
/disto.1/out 0.000 ;
/disto.1/grain 50 ;
```

2.6 L'espace

Le traitement de l'espace dans *tape* se fait par une combinaison de panoramique multi haut-parleurs et un dosage de réverbération avec le son direct. Comme expliqué dans II 1.4.4 (La spatialisation) tous les modules ont la possibilité d'avoir une spatialisation indépendante qui n'est qu'une question d'envoi de valeurs à la matrice. L'éditeur de chaque module *sonore* contient un éditeur *spat* permettant de placer le son dans l'espace des haut-parleurs, puis de régler le dosage de la réverbération par le biais du paramètre de *présence* (*pres*).

2.6.1 La réverbération

La réverbération dans *tape* est généré par le module *1* des modules de réverbération (*rev.1*); on ne peut donc pas avoir une configuration *tape* sans au moins *un* module *rev*. L'objet de réverbération, lui même, est un objet non compris dans la distribution MSP; il s'appelle *gigaverb* et est proposé par Olaf Matthes[1]. L'éditeur (Figure 54) montre les paramètres de la réverbération ainsi que son éditeur du *spat*. Les valeurs ici présentes sont les valeurs par

1 *Gigaverb~* est distribué sous licence GPL. Un lien se trouve à : http://www.maxobjects.com/

défaut – dans tous les haut-parleurs à -10 dB, sans *présence*[1] (Voir l'éditeur de *rev.1* Figure 54).

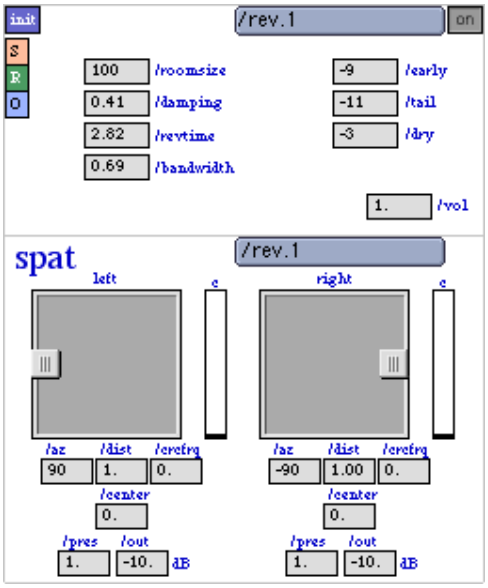


Figure 54: L'éditeur de rev.1

2.6.2 Le panoramique

Il y a actuellement trois configurations possibles pour le panoramique de haut-parleurs, et il faut faire le choix au moment de créer une configuration (cf. II 1.3.2.2 t_config, page 37). Nous proposons une configuration pour 4 haut-parleurs en quadriphonie (*spat4*), pour 4 haut-parleurs en quadriphonie plus un centre (*spat41* pour 5 hp), et une pour 4 haut-parleurs bas plus 4 haut-parleurs hauts plus un haut-parleur au centre (*spat441* pour 9 hp). Le son se place dans les 4 haut-parleurs de la même manière dans tous les cas – par des paramètres d'azimut (*az*) et de distance (*dist*). La valeur du *centre* (pour les *spat41* et *spat441*) est un contrôle entre 0 et 1 qui diminue le niveau dans les 4 hp pour monter celui du centre (paramètre *center*). Dans le cas du *spat441* un paramètre de hauteur (*height*) déplace progressivement le son entre les 4 hp du bas vers les 4 hp du haut.

Le principe du panoramique de 4 haut-parleurs est un cercle à l'intérieur d'un carré. Le carré est définit par l'emplacement des haut-parleurs, et nous dessinons un cercle à l'intérieur définissant la distance « 1 ». L'emplacement *devant-centre* aura ainsi un azimut de 0 et une

1 Effectivement, le module *rev* peut lui-même passer dans la réverbération grâce à son paramètre *pres*, mais il n'est pas d'usage de le faire.

distance de 1. Pour déplacer le son sur le cercle on change l'azimut en gardant la distance à 1, et nous constatons de cette manière que le son n'est jamais *uniquement* dans un seul haut-parleur. Ceci permet plus de homogénéité d'espace et un meilleure effacement de la perception des haut-parleurs eux-mêmes. Quand nous voulons placer un son *uniquement* dans un haut-parleur, il faut augmenter la distance jusqu'à la valeur de la racine carré de 2 (1.414), d'après le théorème de Pythagore[1]. Dans Figure 55 nous voyons que :

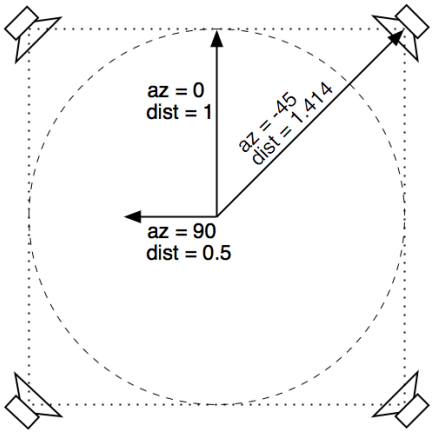


Figure 55: Le carré des 4 haut-parleurs

• le son a*vant-centre* correspond à *az = 0* et *dist = 1*

• le haut-parleur *avant-droit* à *az = -45* et *dist = 1,414*

• le son au milieu de l'espace légèrement à gauche correspond à *az = 90* et *dist = 0,5*.

2.7 La matière

La synthèse granulaire est l'archétype du traitement de la matière. Elle permet de modifier tous les paramètres du son et de reconstruire un autre son sur la matière du premier. Dans tape il y a trois modules en relation avec les techniques de synthèse granulaire :

- Une synthèse granulaire *traditionnelle* qui joue des grains dans une mémoire statique dans laquelle nous aurions chargé ou enregistré un son (nom de module : *gran*)
- Une synthèse granulaire qui joue des grains dans une mémoire dynamique de type *ligne à retard* qui reste lié au son directe (nom de module : *munger* ou *mng*)
- Une synthèse granulaire proposant un gel sonore (nom de module : *freeze*)

1 Dans un triangle rectangle, le carré de la longueur de l'hypoténuse (c) est égal à la somme des carrés des longueurs des deux autres côtés (a et b). Ça fait $a^2 + b^2 = c^2$. Puisque dans notre cas a et b sont égales à 1, c = la racine carrée de 2.

2.7.1 Granulaire dans une mémoire dynamique : munger

Ceci est un exemple du deuxième type, à l'aide du module *munger* (*mng*). Le module *munger* est construit avec l'objet *munger~* faisant partie d'une collection gratuite d'objets programmés par Dan Trueman et Luke DuBois et appelée PeRColate[1].

Le principe du *munger* est que le son entre dans une mémoire dynamique de type *ligne à retard* ; le moteur de type *granulaire* joue des grains à l'intérieur de cette mémoire. Ainsi, la granularisation se fait sur du son légèrement retardé vis-à-vis du direct. Ce procédé fonctionne particulièrement bien pour produire une *ombre de grains* autour d'un son instrumental, issue du son instrumental lui-même. Les paramètres du module *mng* de *tape* correspondent à ceux définis par Trueman et DuBois dans leur objet *munger~*[2]. Il y a les paramètres de base d'une synthèse granulaire traditionnelle, en un peu moins développé. Nous pouvons les voir dans l'éditeur de *mng* (Figure 56). Notons également qu'un *spat* stéréo – spatialisation indépendante de la gauche et de la droite – est proposé.

Dans notre exemple, nous définissons une longueur de ligne à retard de 500 ms (*/del*), une intervalle de temps des grains de 50 ms (*/rate*), une variation dans l'intervalle de temps des grains de 50 ms (*/ratevar*), une taille des grains de 200 ms (*/size*), une variation dans la taille des grains de 100 ms (*/sizevar*), une transposition de 0 (*/trsp*) avec une variation dans la transposition (*/trspvar*) de 50% d'une gamme de transposition en demi-tons (*/scale*) de « 0 2 4 5 7 9 11 », un élargissement du champs stéréo des grains (*/stereospread*) de 100%, un gain des

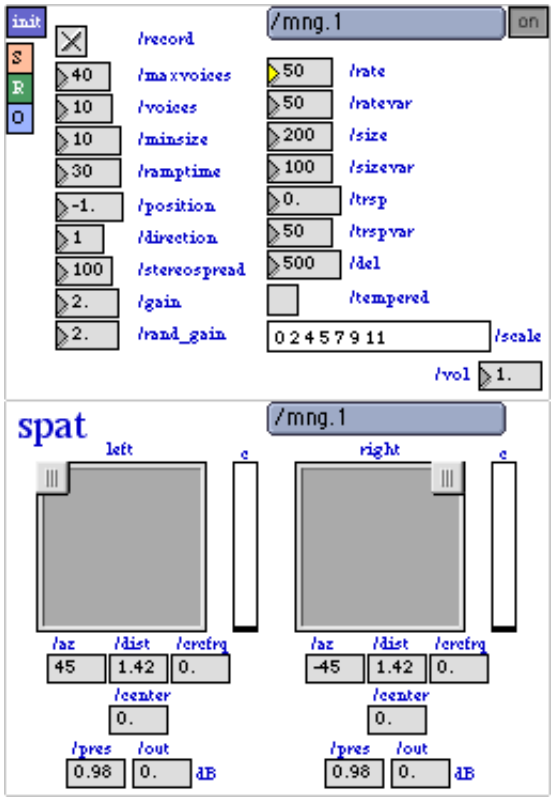


Figure 56: L'éditeur du munger (mng)

1 cf. le site internet de PeRColate : http://music.columbia.edu/percolate/ . Dan Trueman est dans le département de musique à l'Université de Princeton, et Luke DuBois dans le *Computer Music Center* de l'Université de Columbia.

2 id.

grains de 2 (*/gain*), et un ambitus de gain aléatoire de 2 (*/rand_gain*). Pour la spatialisation, nous avons mis le canal gauche sur le haut-parleur avant-gauche et le canal droite sur le haut-parleur avant-droite, avec un petit peu de réverbération des deux canaux (*pres = 0.98*).

Comme dans l'exemple de la distorsion (cf. Infra) nous pouvons activer un *noise gate* afin d'éliminer le bruit de fond dans un environnement qui n'est pas bien silencieux. Nous retrouvons donc le schéma final suivant (Figure 57) :

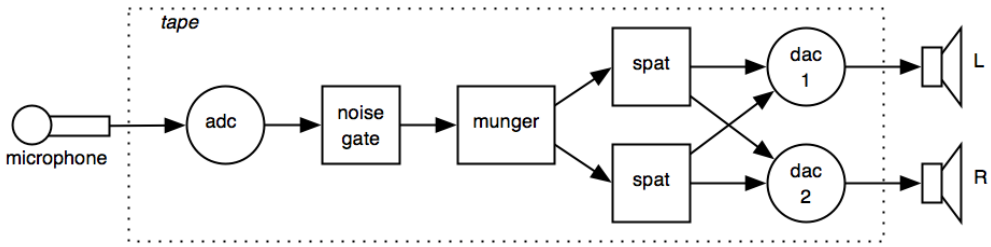


Figure 57: Schéma du munger avec noise gate

Le texte de l'événement qui correspond à cette configuration est le suivant :

```
** basic mng;
** ---------------------------------------- /mtrx connections ---------------------------------------- ;
/mtrx connect adc.1 mng.1 0 ;
** ---------------------------------------- /adc.1 ---------------------------------------- ;
/adc.1/sw 1 ;
/adc.1/in 1.000 ;
/adc.1/az 0 ;
/adc.1/dist 1.000 ;
/adc.1/crcfrq 0.000 ;
/adc.1/center 0.000 ;
/adc.1/pres 1.000 ;
/adc.1/out -127.000 ;
/adc.1/grain 50 ;
** ---------------------------------------- /adc.1/ngate ---------------------------------------- ;
/adc.1/ngate/sw 1 ;
/adc.1/ngate/onlevel -59.000 ;
/adc.1/ngate/offlevel -61.000 ;
/adc.1/ngate/ontime 1 ;
/adc.1/ngate/offtime 50 ;
** ---------------------------------------- /mng.1 ---------------------------------------- ;
/mng.1/sw 1 ;
/mng.1/record 1 ;
/mng.1/maxvoices 40 ;
/mng.1/voices 10 ;
/mng.1/minsize 10 ;
/mng.1/ramptime 30 ;
/mng.1/rate 50 ;
/mng.1/ratevar 50 ;
/mng.1/size 200 ;
/mng.1/sizevar 100 ;
/mng.1/trsp 0.000 ;
/mng.1/trspvar 50 ;
/mng.1/del 500 ;
/mng.1/position -1.000 ;
/mng.1/direction 1 ;
/mng.1/stereospread 100 ;
/mng.1/gain 2.000 ;
/mng.1/rand_gain 2.000 ;
/mng.1/vol 1.000 ;
/mng.1/tempered 0 ;
/mng.1/scale 0 2 4 5 7 9 11 ;
/mng.1/L/az 45 ;
/mng.1/L/dist 1.420 ;
/mng.1/L/crcfrq 0.000 ;
```

```
/mng.1/L/center 0.000 ;
/mng.1/L/pres 0.980 ;
/mng.1/L/out 0.000 ;
/mng.1/L/grain 50 ;
/mng.1/R/az -45 ;
/mng.1/R/dist 1.420 ;
/mng.1/R/crcfrq 0.000 ;
/mng.1/R/center 0.000 ;
/mng.1/R/pres 0.980 ;
/mng.1/R/out 0.000 ;
/mng.1/R/grain 50 ;
```

2.8 Le domaine fréquentiel

Le domaine fréquentiel est une catégorie riche comprenant de nombreuses possibilités de traitements et touchant tous les aspects de la musique. Il y a dans tape plusieurs modules fonctionnant dans le domaine fréquentiel. Ils sont :

- *gizmo* – transposition de hauteur ou *harmoniseur* d'après analyse spectrale. Nous l'avons traité dans II 2.2 La hauteur, page 55.

- *pt* – détection de hauteur d'après analyse des pics et estimation de la fondamentale. Nous l'avons traité dans II 2.2.2 Harmoniseur contrôlé par la détection de hauteur, page 57

- *cntrd* – suivi du centroïde spectral d'après analyse fft. Nous l'avons traité dans II 2.5.2 Transposition de fréquence et contrôle par le centroïde, page 68

- *fftfilt* – un filtre qui agit directement sur les amplitudes des *bins* de l'analyse fft, fonctionnant ainsi comme un égaliseur de spectre – une banque de filtres « tranchants » (cf. I 1.2.2.8 Le traitement dans le domaine fréquentiel, page 12).

- *fftx* – la synthèse croisée généralisé où deux sons analysés par fft (vocodeur de phase) subissent une hybridation en combinant les amplitudes de l'un avec les fréquences de l'autre – intensifié par un facteur de convolution (cf. I 1.2.2.8 Le traitement dans le domaine fréquentiel, page 12)

- *fftsf* – la synthèse *source-filtre* ou *empreinte de timbre* où les magnitudes du spectre d'un son (par analyse fft) contrôlent les magnitudes du spectre d'un autre (cf. I 1.2.2.8 Le traitement dans le domaine fréquentiel, page 12)

2.8.1 Le filtre fft

Le module *filtre fft* (*fftfilt*) est extrêmement utile. Il est dérivé d'un patch d'exemple classique dans Max/MSP, qui date même de Max/FTS, créé par Zack Settel et Cort Lippe – forbidden-planet[1]. Les magnitudes d'une entrée son sont extraites par une analyse fft, puis multipliées par une liste de valeurs de magnitudes dessinées à la main (ou générées par d'autres moyens) pour imprimer son empreinte au son traité. Ceci est obtenu avec un filtre de 512 bandes de fréquences espacées d'environ 47 Hz (si nous échantillonnons à 48 Khz : fréquence Nyquist, 24Khz, divisée par le nombre de bandes). A l'époque de la programmation de ce module (avant la version 5 de Max de 2008) il y avait une limitation dans les longueurs des listes et il fallait diviser la liste de 512 en 2 fois 253. C'est pour cela que cet éditeur contient une moitié pour les *bins* 0-252 et une autre pour les *bins* 253-505 (les *bins* 206-211 ne peuvent pas être édités, mais ne sont pas audibles non plus...).

Le signal peut provenir d'une entrée *adc,* de n'importe quel autre module via la matrice, ou bien d'un générateur de bruit interne (paramètre */noise/vol*). Il est possible de dessiner un spectre à la main(cf. Figure 58), de générer des spectres interpolés par un algorithme aléatoire (cf. Figure 59), ou de gérer l'amplitude d'un groupe de *bins* adjacents (cf. Figure 60).

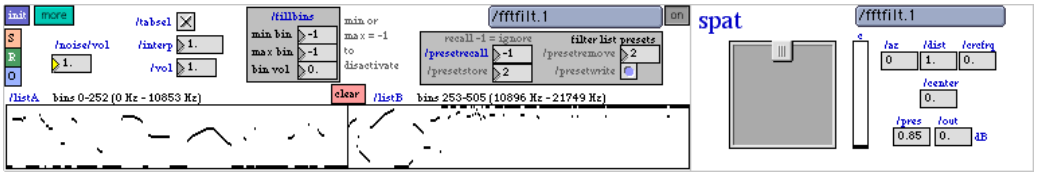


Figure 58: L'éditeur du filtre fft – dessiné à la main

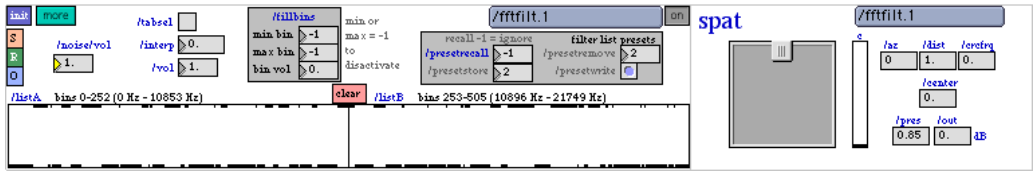


Figure 59: L'éditeur du filtre fft – spéctre aléatoire

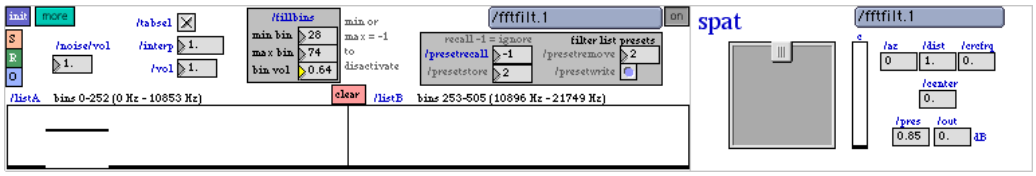


Figure 60: L'éditeur du filtre fft – remplissage d'un groupe de bins

1 Ce patch est encore inclus dans les exemles de Max5, /Max5/examples/fft-fun/forbidden-planet.maxpat

Lors du stockage des données du fftfilt dans un événement, le profil du filtre est stocké avec l'ensemble des paramètres; mais nous pouvons également stocker des mémoires des *bins* dans la partie *filter list presets.*

2.8.2 La synthèse croisée généralisée

Pour obtenir une synthèse croisée généralisées (basée sur une simple implémentation des objets d'analyse fft dans MSP), il faut d'abord connecter une entrée son à l'entrée « gauche » du module (entrée A), puis un autre à l'entrée « droite » (entrée B) (Figure 61).

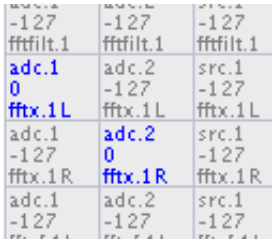


Figure 61: La matrice et fftx

Ensuite, il faut activer le module dans l'éditeur. Par défaut, les amplitudes du son gauche (A) sont combinées avec les phases du son droite (B), avant d'être re-synthétisée à la sortie de l'effet; ceci devrait fonctionner dès qu'il y a du son. Nous pouvons changer ce rapport, ou simplement alterner les entrées par le paramètre /swapLR. Également, le paramètre /comod crée une co-modulation entre les 2 entrées.

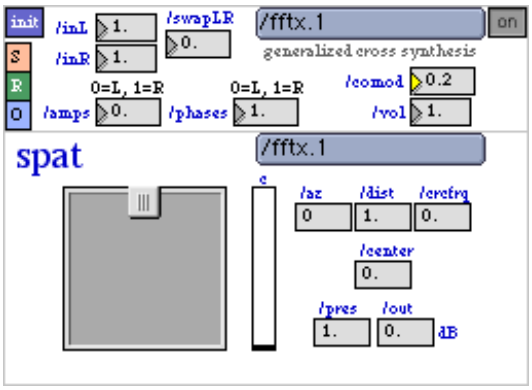


Figure 62: L'éditeur du fftx

2.8.3 la synthèse *source-filtre*

Le dernier module que nous présentons très brièvement ici est la synthèse source-filtre réalisant l'*empreinte spectrale d'*un son sur un autre. Ce traitement suit le même principe que le *filtre fft*, mais entre deux sons et non entre un éditeur et un son. Connectons la matrice comme dans fftx – un son à l'entrée gauche (A) et l'autre à l'entrée droite (B) (Figure 63).

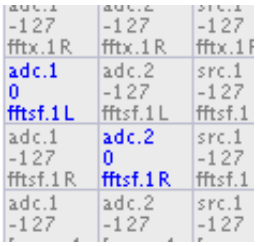


Figure 63: La matrice et fftsf

Dans l'éditeur, les seuls paramètres proposés sont les niveaux des inL et inR, ainsi qu'un swapLR pour tester les deux sons dans l'autre sens sans changer les connexions à la matrice.

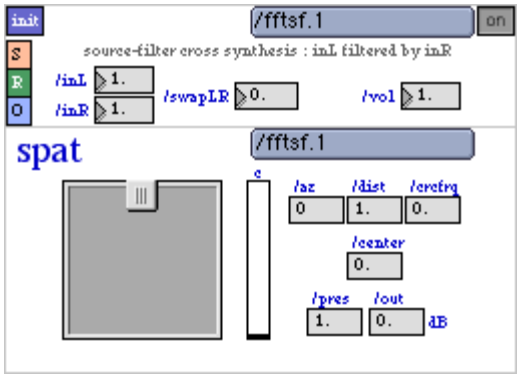


Figure 64: L'éditeur du fftsf

2.9 Conclusion

Des modules de tous les catégories de traitements sont disponibles dans tape et tapemovie, depuis des modules de traitement sonore jusqu'à des modules de traitements d'analyse destinés au contrôle des paramètres. La conception par matrice permet des configurations sans limite de complexité, tandis que la facilité de stockage des événements ainsi que la puissance dans leurs édition en format texte donnent une grande souplesse de construction et de composition.

III La création musicale avec les traitements de base

1 Le Patch Bien Tempéré Numéro 1

Dans le contexte du travail de master/doctorat, J'ai[1] envisagé de créer une série de 8 à 10 courtes pièces sous le titre *Le patch bien tempéré*, chacune pour un instrument seul avec un traitement temps réel tiré des catégories définies dans ce mémoire (cf. I 1.2.2 La liste des traitements par catégorie, page 6). Au stade de ce mémoire, la première de ces pièces existe, composée pour vibraphone et *traitement du temps par délai*. Elle a été créée lors des concerts de l'Atelier de Composition de José Manuel Lopez Lopez en juin 2010, jouée par le percussionniste Rémi Durupt.

Je vais présenter ici la pièce *Le patch bien tempéré numéro 1* – pour vibraphone et traitement temps réel (délai), en parlant de ses principes générales, puis des applications musicales précises liées au traitement du temps.

1.1 Principes générales

Pour la première pièce de la série, il me semblait évident qu'il fallait commencer avec le temps, et plus précisément avec le *délai*. J'avais envie d'employer plusieurs délais pour faire un travail poussé au niveau de la construction polyphonique issue des superpositions de rythmes et de hauteurs. Pour le coté rythmique, je voulais la précision d'attaque et des variations dans les *types* d'attaques que l'on trouve chez les percussions, ce qui a amené aux claviers pour le coté des hauteurs. Le vibraphone a été choisi parmi les claviers pour la neutralité de son timbre qui support très bien des multiples superpositions, et pour la grande souplesse possible dans les attaques avec des différentes mailloches – allant jusqu'à la possibilité de jouer *sans attaque* avec un archet.

1.1.1 Le dispositif et les valeurs du temps des délais

L'écriture instrumentale dans cette pièce est volontairement assez simple pour que la complexité puisse vraiment venir de la superposition polyphonique que les délais produisent.

1 Dans cette section, l'empoi de la première personne du singulier semble mieux adapté à une discussion d'une création musicale personnelle.

Il faut noter, cependant, que cette pièce est très exigeante au niveau de la précision rythmique demandé à l'instrumentiste, et c'est pour cela qu'un *clic* que l'interprète écoute par oreillette est essentiel à son bon déroulement.

Pour réaliser la densité polyphonique souhaitée, j'ai mis en place (en utilisant l'environnement *tape*) un dispositif de 6 délais. Pour les valeurs du temps de ces délais, j'ai déterminé 3 suites de 6 nombres – une basé sur une division rythmique en 3 temps, une en 4 temps et une en 5 temps. Les qualités de ces suites de nombres devraient être : 1) que la valeur la plus grande soit environ 10 à 11 fois la valeur la plus petite pour qu'il y a une cohérence temporelle entre les différentes suites et une distance temporelle significative, et 2) et que la recombinaison de toutes les valeurs d'une suite remplisse les subdivisions définies pas sa base.

Par exemple, la suite construite pour une subdivision en **3** (extraite de la suite de Fibonacci) est :

3 – 5 – 8 – 13 – 21 – 34

...ce qui représente des multiples d'un tiers de la valeur d'un temps :

3 * 1/3 = 1 temps
5 * 1/3 = 1 temps et 2/3
8 * 1/3 = 2 temps et 2/3
13 * 1/3 = 4 temps et 1/3
21 * 1/3 = 7 temps
34 * 1/3 = 11 temps et 1/3

La superposition rythmique de ces valeurs de temps fait que le 3 et le 21 tombent sur le temps, le 5 et le 8 tombent sur le 2/3 de temps, et le 13 et le 34 tombent sur le 1/3 de temps – ainsi remplissant la subdivision en 3. Le facteur entre la plus grande et la plus petite valeur est 11,33.

La suite construite pour une subdivision en **4** est :

4 – 9 – 15 – 22 – 31 – 44

...ce qui représente des multiples d'un quart de la valeur d'un temps :

4 * 1/4 = 1 temps
9 * 1/4 = 2 temps et 1/4
15 * 1/4 = 3 temps et 3/4

22 * 1/4 = 5 temps et 2/4
31 * 1/4 = 7 temps et 3/4
44 * 1/4 = 11 temps

La superposition rythmique de ces valeurs de temps fait que le 4 et le 44 tombent sur le temps, le 9 tombe sur le 1/4 de temps, le 22 tombe sur le 2/4 de temps, et le 15 et le 31 tombent sur le 3/4 de temps – ainsi remplissant la subdivision en 4. Le facteur entre la plus grande et la plus petite valeur est 11.

La suite construite pour une subdivision en **5** est :

5 – 11 – 18 – 27 – 39 – 55

...ce qui représente des multiples d'un cinquième de la valeur d'un temps :

5 * 1/5 = 1 temps
11 * 1/5 = 2 temps et 1/5
18 * 1/5 = 3 temps et 3/5
27 * 1/5 = 5 temps et 2/5
39 * 1/5 = 7 temps et 4/5
55 * 1/5 = 11 temps

La superposition rythmique de ces valeurs de temps fait que le 5 et le 55 tombent sur le temps, le 11 tombe sur le 1/5 de temps, le 27 tombe sur le 2/5 de temps, le 18 tombe sur le 3/5 de temps, et le 39 tombe sur le 4/5 de temps – ainsi remplissant la subdivision en 5. Le facteur entre la plus grande et la plus petite valeur est 11.

Le principe de l'écriture rythmique sera de choisir une valeur de base temporelle et de calculer les temps des 6 délais selon une des trois suites de nombres, puis d'écrire les rythmes du vibraphone soit dans la même base rythmique que les délais (ce qui va renforcer et souligner ce rythme) soit dans une *autre* base / subdivision pour créer un rapport polyrythmique.

1.1.2 Le système des hauteurs

1.1.2.1 Les suites des valeurs du temps

J'ai eu un peu de difficulté pour choisir un système de hauteurs pour cette pièce qui donnerait une cohérence dans l'ensemble en permettant beaucoup de souplesse dans les combinaisons possibles. Je ne voulais pas tomber dans la suggestion d'une quelconque tonalité, mais je ne

voulais pas non plus un esthétique où toutes les notes étaient parfaitement égales. Je voulais assez de rigueur sans pour autant éliminer la possibilité de trier et de choisir pour des raisons toute à fait musicales et subjectives.

Finalement, je suis arrivé à la solution la plus logique et évidente : utiliser les suites des valeurs du temps pour générer le système des hauteurs. Je voulais créer des groupes ou des « gammes » de huit notes pour pouvoir éventuellement regrouper en 2 fois 4. Dans ce cas, les suites de 6 valeurs rythmiques ne suffisaient pas pour générer 8 hauteurs. Il fallait donc calculer la continuation des suites.

La suite en 3, basé sur la suite de Fibonacci, était facile – il suffisait de continuer en additionnant les deux dernières valeurs pour calculer la nouvelle. J'ai ensuite choisi de baser la suite des hauteurs sur la *différence* entre les valeurs du temps. Donc, à partir de la suite [3 – 5 – 8 – 13 – 21 – 34] ça donne la suite des *différences* [2 – 3 – 5 – 8 – 13] et par extension pour arriver à 8 valeurs : [**2 – 3 – 5 – 8 – 13 – 21 – 34 – 55**]. C'est cette suite de différences qui sera la base des gammes de hauteurs pour la série sur 3.

Pour continuer la suite en 4, il fallait regarder la suite des *différences*, puis les *différences des différences*. La suite en 4 étant [4 – 9 – 15 – 22 – 31 – 44], les différences sont donc [5 – 6 – 7 – 9 – 13] et les différences des différences est [1 – 1 – 2 – 4]. J'ai additionne les 3 précédentes valeurs (quand il y en a 3) pour avoir la suite. Ainsi, 1+1=2 et 1+1+2=4, donc 1+2+4=7, 2+4+7=13, et 4+7+13=24. Ça fait la suite prolongée des différences des différences [1 – 1 – 2 – 4 – 7 – 13 – 24], ce qui fait que la suite des différences est [**5 – 6 – 7 – 9 – 13 – 20 – 33 – 57**]. C'est cette suite de différences qui sera la base des gammes de hauteurs pour la série sur 4.

Pour la suite en 5, [5 – 11 – 18 – 27 – 39 – 55], les différences sont [6 – 7 – 9 – 12 – 16] et les différences des différences [1 – 2 – 3 – 4]. Il suffit d'augmenter par 1 chaque intervalle de différence pour prolonger la suite des différences : **[6 – 7 – 9 – 12 – 16 – 21 – 27 – 34**]. C'est cette suite de différences qui sera la base des gammes de hauteurs pour la série sur 5.

1.1.2.2 Les gammes

La conversion en hauteurs s'est fait de 8 manières différentes, donnant lieu à 8 « gammes », chacune basées sur la référence Do.

1) intervalles relatives ascendantes
2) intervalles relatives descendantes
3) intervalles relatives ascendantes/descendantes
4) intervalles relatives descendantes/ascendantes
5) intervalles absolues ascendantes
6) intervalles absolues descendantes
7) intervalles absolues ascendantes/descendantes
8) intervalles absolues descendantes/ascendantes

Prenons par exemple la suite de nombres déduite de la série de valeurs de délai sur 3 (2–3–5–8–13–21–34–55). Pour les *intervalles relatives ascendantes*, j'ai commencé à une référence 0 (*do, qui ne fera pas partie de la gamme*) et j'ai ajouté la première valeur de 2 ce qui donne *ré*. Ensuite, j'ai ajouté 3 pour arriver à 5 (*fa*), puis 5 pour arriver à 10 (*sib*), puis 8 pour arriver à 18 ce qui donne 6 (*fa#*) quand on replie sur les 12 demi-tons de l'octave, puis encore 13 pour *sol* (19 modulo 12 = 7), 21 pour *mi* (28 modulo 12 = 4), 34 qui donne *ré* (38 modulo 12 = 2), et encore 55 qui donne *la* (57 modulo 12 = 9). J'ai ensuite généré la gamme relative descendante de la même manière mais dans la soustraction. (cf. Figure 65).

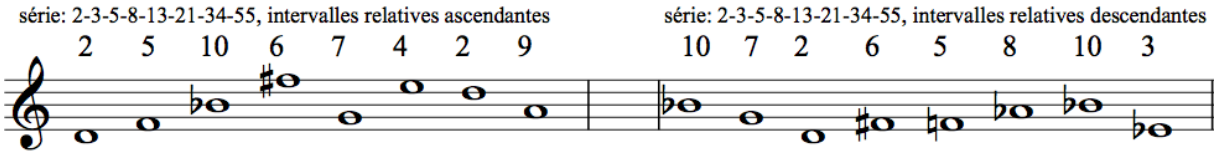


Figure 65: Série sur 3, relative ascendante et relative descendante

Pour les *intervalles relatives ascendantes/descendantes,* j'ai fait la même chose, mais en alternant addition/soustraction à chaque intervalle. Pour *descendantes/ascendantes,* le contraire – soustraction/addition (cf. Figure 66).

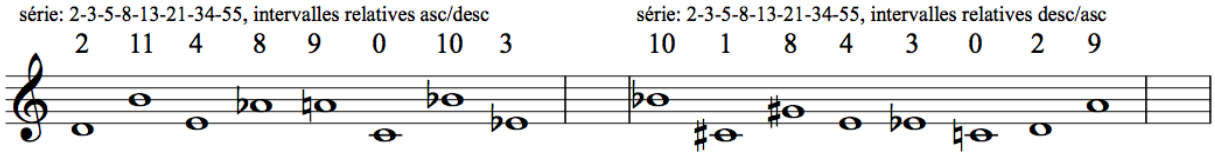


Figure 66: Série sur 3, relative ascendante/descendante et descendante/ascendante

Pour les intervalles absolues chaque valeur a été additionnée ou soustraite avec ou depuis la note de référence 0. Pour voir les 4 combinaisons en absolue, cf. Figures 67 et 68).

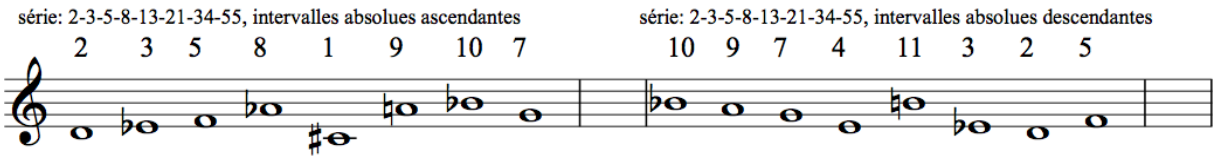


Figure 67: Série sur 3, absolue ascendante et absolue descendante

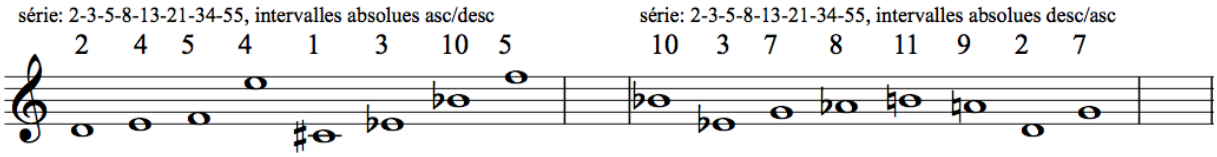


Figure 68: Série sur 3, absolue ascendante/descendante et descendante/ascendante

Par extension, j'ai calculé les 8 gammes de la série sur 4 (5–6–7–9–13–20–33–57) et les 8 de la série sur 5 (6–7–9–12–16–21–27–34), donnant lieu à 24 gammes parmi lesquelles faire des choix de l'écriture.

Ensuite, j'ai fait des statistiques pour chaque classe de hauteur pour voir combien de fois chacune était présente dans l'ensemble des gammes. *Mib* et *Fa#* étaient présentes plus que les autres avec 18 apparitions, ensuite *La* y était 17 fois, suivi de *Ré* et *Sib* (14 fois), *Fa* et *Sol* (13 fois), *Do* (12 fois), *Mi* et *Lab* (11 fois), et à la fin *Do#* et *Si* (10 fois). Ceci a permit de déterminer certains « pôles fortes » ou hauteurs avec plus de poids que d'autres – capables de donner une certaine couleur sans pour autant se faire sentir comme une *tonalité*. Par exemple, ces informations ont déterminé la première et la dernière hauteur de la pièce – Mib et Fa# respectivement.

Pour conclure cette discussion sur les hauteurs, j'ai enchaîné les séries de façon très libre, parfois même en rétrograde, un peu comme un labyrinthe, pour partir sur une certaine hauteur et finir une partie de la pièce sur une autre que j'aurais choisi selon les statistiques ou un goût d'enchaînement. Je me suis permit également de sauter des valeurs, répéter des valeurs, ou même remonter en arrière – le tout géré par le résultat musical.

1.2 Les usages du délai dans la pièce

1.2.1 Première partie (mes. 1-39)

Au début de la pièce, le vibraphoniste joue *arco*, autant que possible sans attaques pour pas que les attaques soient marqués par les délais. Les six délais sont en marche dans la série de valeurs sur 4, et chaque valeur correspond à un multiple de double-croches au tempo noire = 99. Ainsi, la série 4-9-15-22-31-44 devient : 4 double-croches (1 temps ou 606 ms), 2 temps et 1 double-croche (1364 ms), 3 temps et 3 double-croches (2273 ms), 5 temps et 2 double-croches (3333 ms), 7 temps et 3 double-croches (4697 ms), et 11 temps (6667).

Ces délais servent musicalement à envelopper le vibraphone dans un *nuage* de ces propres notes sans avoir de franche repère temporelle grâce aux attaques non-timbrées et non marquées de la mode du jeu à l'archet[1]. Parlons du rapport entre le jeu de l'instrument et la distribution du temps des délais. Quand le vibra joue le Mi bécarre à la mesure 3, délai 1 aura tout juste fini de jouer le Mi bémol du début, délais 2-3-4-5 le jouent encore, et délai 6 ne le commencera que 2 temps plus tard.

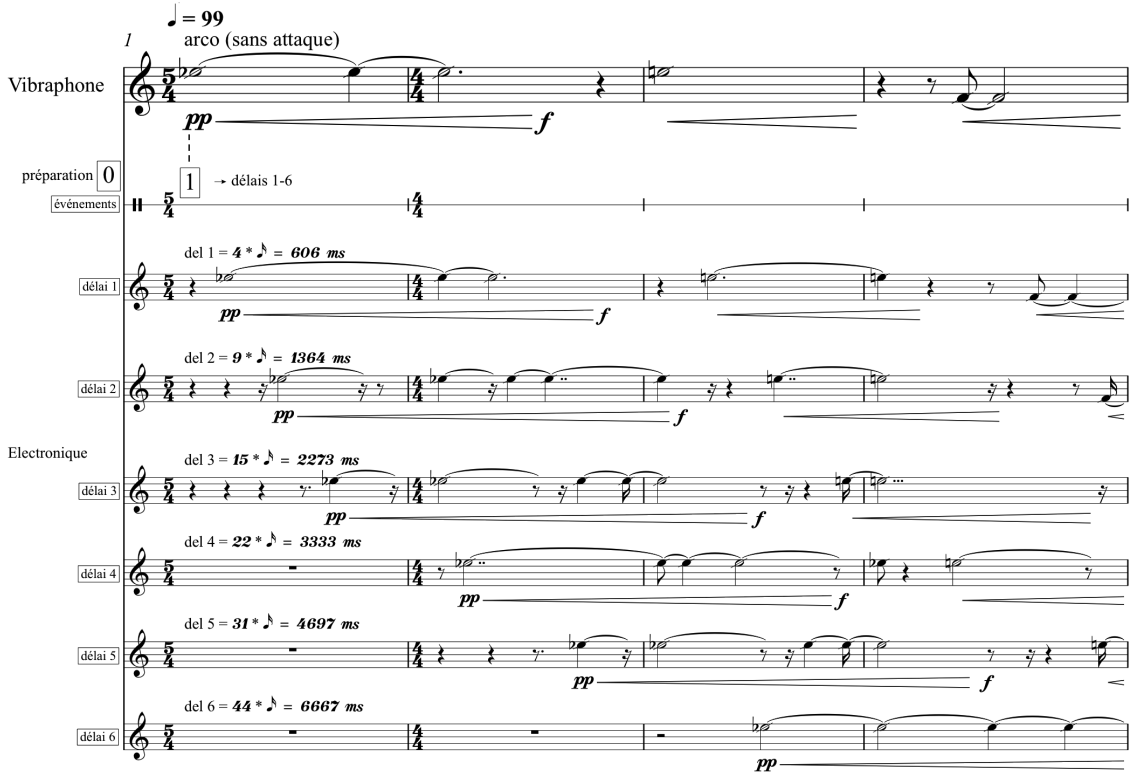


Figure 69: Mesures 1-4

1 cf. la discussion sur la *stabilité* et *l'instabilité* dans I 2.4.2 Le dédoublement et l'épaississement, page 26

Au deuxième temps de la mesure 3, délai 1 rejoint le vibra sur Mi bécarre, mais délai 2 arrête le Mi bémol, ne laissant que délais 3-4-5. Pendant le troisième temps, délai 2 rejoint délai 1 et le vibra sur Mi bécarre, alors que délai 3 termine le Mi bémol et délai 6 ne le fait que commencer (cf. Figure 69).

Ces superpositions et ce *nuage* de hauteurs continuent jusqu'à la mesure 11 où l'entrée dans les délais est coupée par l'événement 2 et le vibra joue deux coups de dé à coudre sans reprise par les délais. L'entrée das les délais s'ouvre de nouveau à la mesure 12 quand le vibra reprend l'archet sur le Si bémol. Les délais 5 et 6 maintiennent jusqu'à là le Ré de la mesure 10, et nous ne perdons pas la continuité des notes de l'archet (cf. Figure 70).

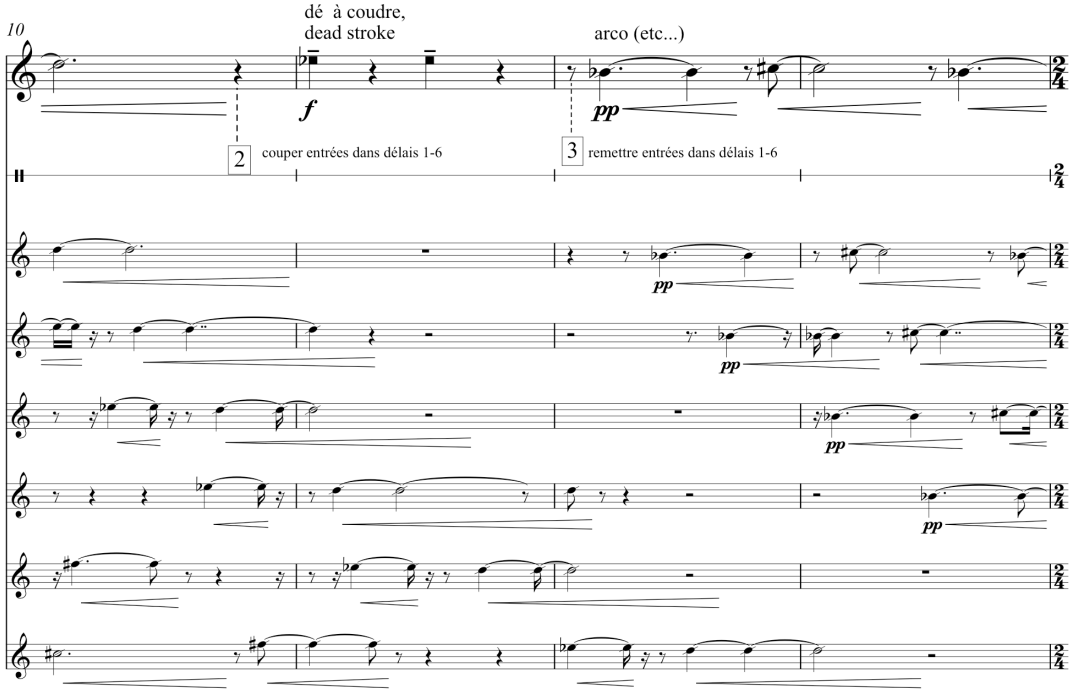


Figure 70: Mesures 10-13

Ces deux attaques de dé à coudre *sans* incidence dans les délais sont là pour tromper la perception et ne pas encore dévoilé le développement rythmique qui va suivre. A la mesure 15 deux nouvelles attaques de dé à coudre sont joués au Mi bémol, cette fois-ci dans tous les délais sauf le 2 et le 3. Les entrées dans délais 2 et 3 restent fermées pour ne pas générer trop de répétitions de courte durée rythmique qui vont trop rapidement dévoiler le mystère du processus et nuire à la construction de la polyphonie. Ainsi, quand le vibra revient au dé à coudre à la mesure 16, toujours sur le Mi bémol, il est accompagné par des échos des deux coups de la mesure 15 repris par les délais 4-5-6. J'instaure maintenant une alternance entre

notes à l'archet et leur effet de « nuage de notes », et les attaques de dé à coudre avec leur effet de pulsation et contrepoint (cf. Figure 71).

Il est important à noter que, malgré l'arrivée des sons du dé à coudre avec les attaques marquées, les notes à l'archet qui continuent en alternance restent sans référence vis-à-vis du temps de délai. C'est un exemple d'une écriture double dans un même dispositif. Les attaques sont perçues comme des échos dans les délais ayant une incidence dans la construction rythmique autour de la pulsation, et les sons d'archet *sans* attaque s'entend comme une sorte de nuage amorphe ayant un rôle d'enveloppement et d'épaississement et non pas de rôle rythmique précis.

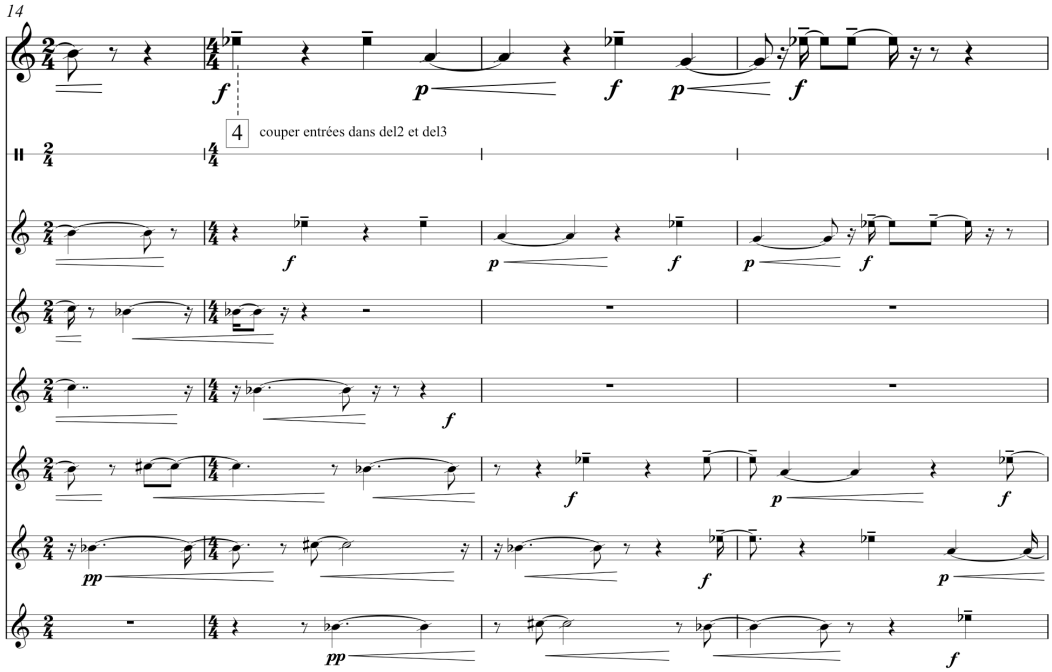


Figure 71: Mesures 14-17

Les attaques de dé à coudre deviennent de plus en plus denses mais restent sur le Mi bémol, ainsi réduisant la possibilité de trouver des repères temporelles qui permettent d'identifier les temps d'écho. L'écriture polyphonique est concernée par la combinaison du vibra direct et les délais – un contrepoint très actif. Il est souvent difficile même de séparer dans la perception les coups du vibra de ceux des délais.

A la mesure 21 le dé à coudre joue enfin une note différente, un La, et à partir de ce moment, les notes des attaques deviennent mobiles et la polyphonie devient plus mélodique, tout en gardant le *fond* des notes de l'archet en deuxième plan. L'événement 5 ouvre de nouveau

l'entrée dans les délais 2 et 3 qui peuvent maintenant contribuer à la polyphonie (cf. Figure 72).

Ce jeu d'alternance entre dé à coudre et archet dans une polyphonie à 6 voix de délais continue jusqu'à la mesure 40.

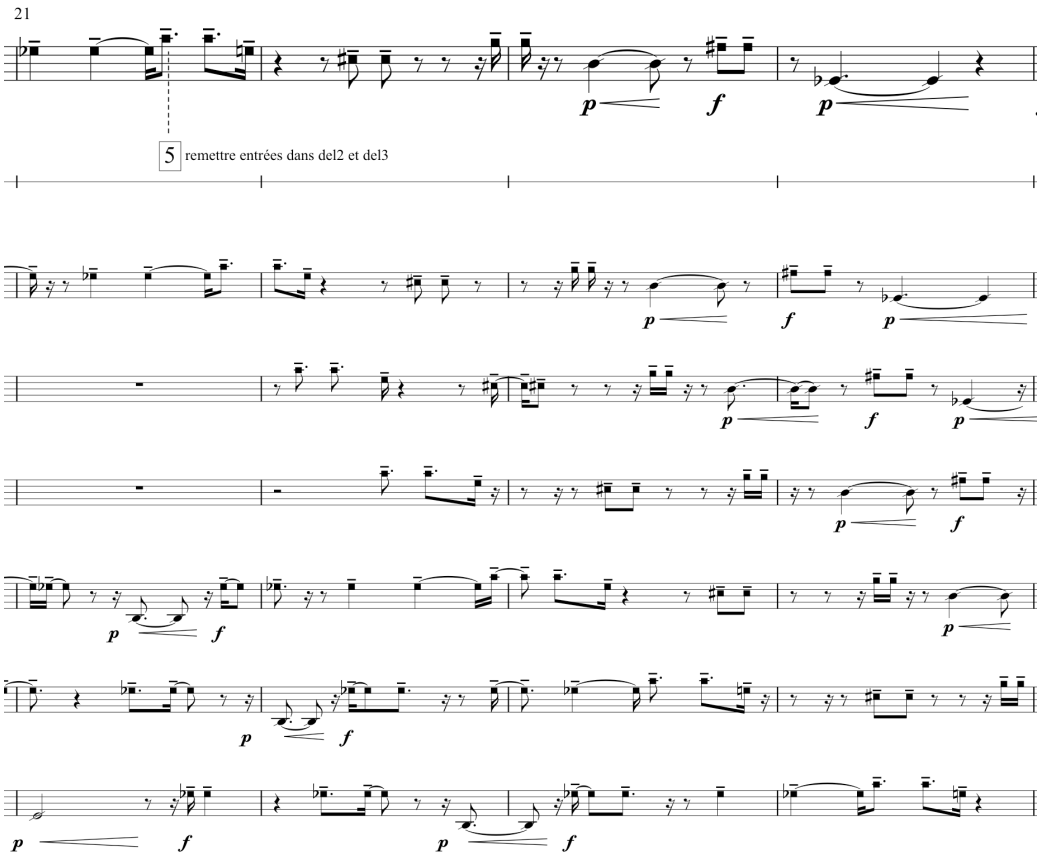


Figure 72: Mesures 21-24

1.2.2 Deuxième partie (mes. 40-74)

A l'arrivée de la deuxième section à la mesure 40, il y a une modulation métrique où la valeur de 5 double croches au tempo de 99 devient la valeur d'une noire au nouveau tempo de 79,2. Les nouveaux temps de délai se mettent en place par un fondu enchainé sur quatre secondes. La suite des valeurs de temps reste la même, étant 4-9-15-22-31-44 double croches, mais les temps changent à cause du nouveau tempo un peu plus lent. Cela donne : 750 ms, 1705 ms, 2841 ms, 4167 ms, 5871 ms, et 8333 ms pour les 6 délais respectivement. Le vibra qui joue des noires au nouveau tempo scande la nouvelle pulsation et laisse le temps pour les délais de se remplir avec le nouveau matériau (cf. Figure 73).

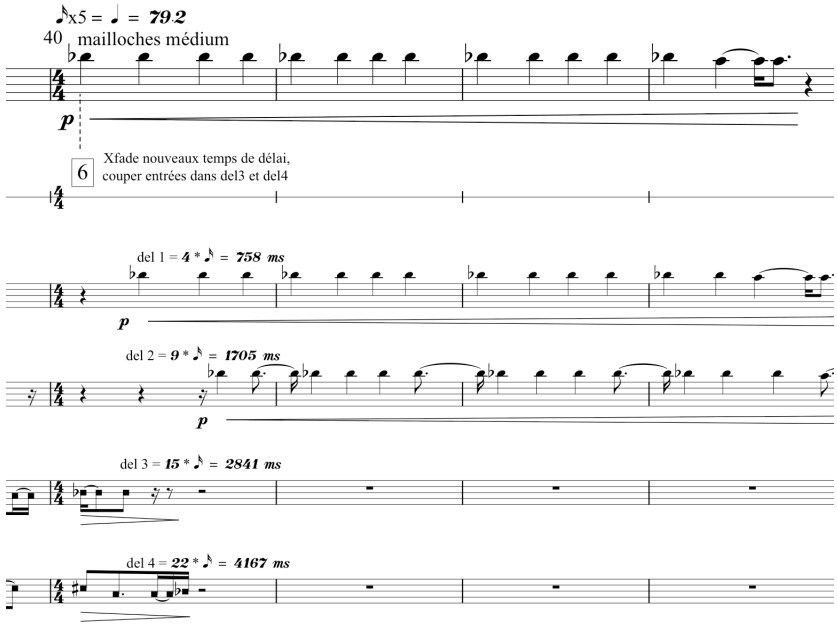


Figure 73: Mesures 40-43

L'instrumentiste pose l'archet est le dé à coudre pour prendre en mains quatre mailloches – deux *médium*, une *dure* et une *très douce*. Ainsi, il va y avoir un jeu de *timbres d'attaques* qui crée des variations dans la construction polyphonique. L'événement 6 (à la mesure 40) coupe les entrées du délai 3 et 4 pour diminuer la densité. Plus tard, le 3 sera ouvert de nouveau à la mesure 49 et le 4 à la mesure 58.

A la mesure 51, nous pouvons voir la notation des trois types d'attaques (cf. Figure 74).

Figure 74: Mesures 51-53

La polyphonie et le contrepoint continuent pendant toute cette section, avec le première élément perturbateur, des triolets de croches, qui arrive à la mesure 66. La série des valeurs du temps des délais s'alignent sur les divisions de la double croche, et quand elle est confronté avec un triolet nous entendons une sorte de *dissonance* rythmique. Ces perturbations se développent entre la mesure 66 et 68 pour devenir les triolets de double croches répétés en crescendo entre la mesure 69 et l'apogée de la fin de la mesure 71. A ce moment là, le vibra s'arrête, laissant sonner les 6 délais qui lâchent les triolets un par un, ayant eu les entrées déconnectées. Il y a encore une modulation métrique qui prend 4 triolets de double croche pour en faire une noire au tempo 59,4. Le vibra joue une série de 4 notes en trémolo dans ce tempo plus lent, sans traitement, superposé avec les triolets des délais. Il termine son decrescendo à la fin de la quatrième note en même temps que le délai 6 fini de jouer les triolets (cf. Figure 75).

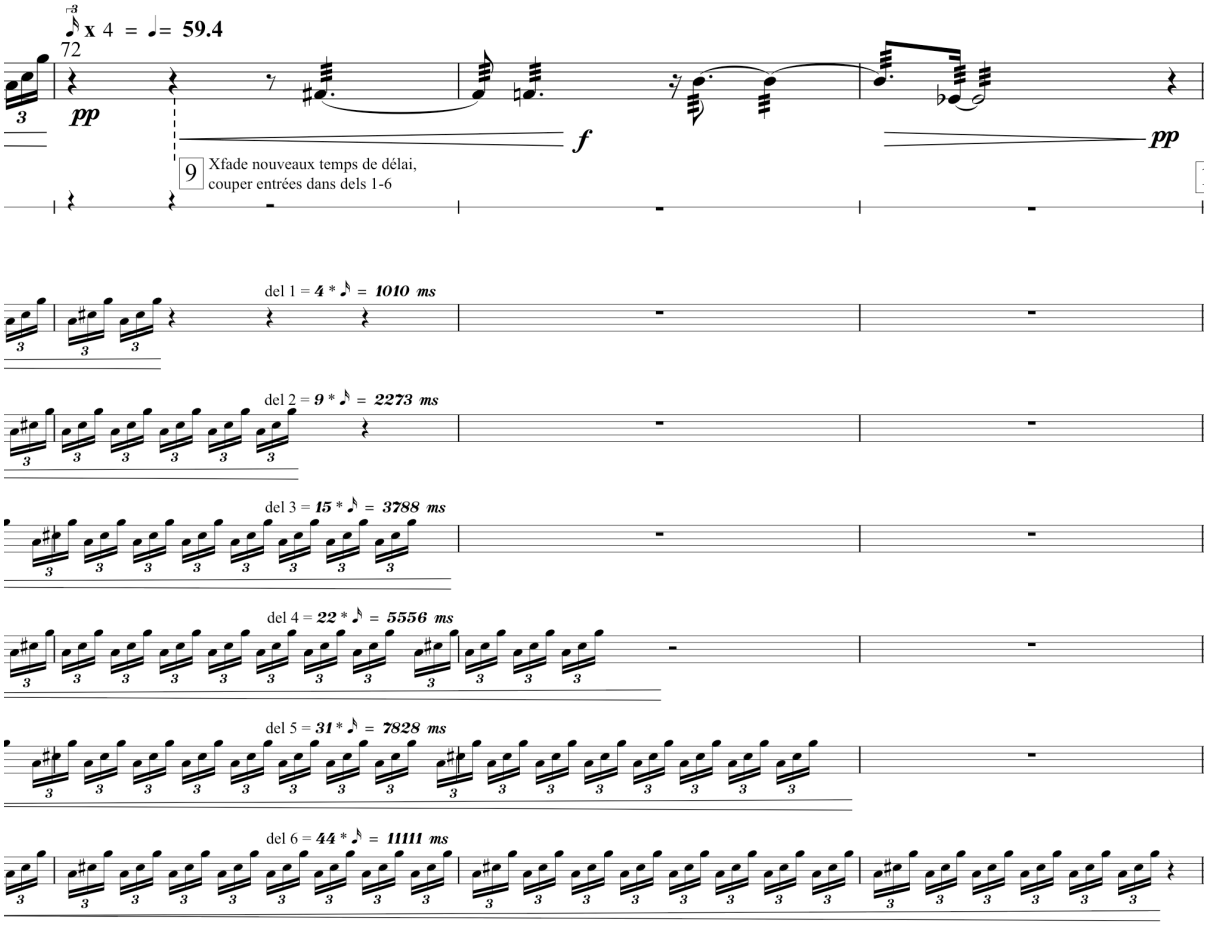


Figure 75: Mesures 72-74

1.2.3 Troisième partie (mes. 75-93)

La fin de la mesure 74 et le début de la 75 (événement 10) représente le premier vrai arrêt de la pièce, et la signifiance de cet arrêt est un changement radicale dans le comportement des délais. Les 6 délais deviennent des *boucles de notes répétées* (cf. Figure 76 pour la discussion suivante).

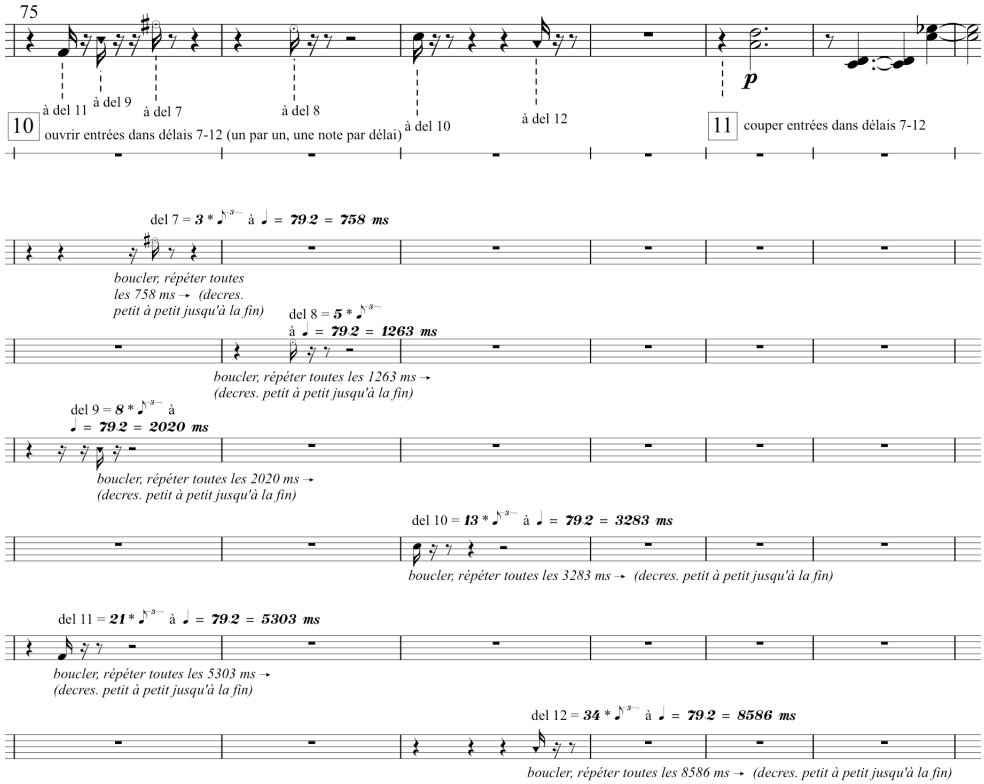


Figure 76: Mesures 75-80

Chaque attaque déclenche une sorte *d'enregistrement* dans un délai. Quand une attaque est détectée, l'entrée dans le délai s'ouvre pendant un petit temps d'enveloppe puis referme aussitôt. La réinjection (*feedback*) de chaque délai étant proche de « 1 », le son pris par cette ouverture d'enveloppe se met à boucler – le temps de la boucle étant égale au temps de délai.

Les temps des délais pour cette partie ont été déterminé par une nouvelle suite de valeurs du temps, la suite basé sur « 3 » (3-5-8-13-21-34) avec une base de temps de 757.58 millisecondes qui correspond à un tempo de 79,2. Les triolets de croches générés par cette suite en « 3 » ont la même valeur rythmique que la double croche du tempo du vibraphone qui est 59,4 (757.58 / 3 = 252.53, et 252.53 * 4 = 1010 ms ce qui correspond à la noire à 59,4). Si

nous regardons à partir de la mesure 75, le Fa rentre en boucle dans le délai 11 (le cinquième délai du deuxième groupe de six) avec un temps de 21 triolets de croche (5303 ms), le Si rentre en boucle dans le délai 9 (3 du deuxième groupe) avec un temps de 8 triolets de croche (2020 ms), le Fa# boucle à 3 triolets de croche (758 ms), le Mi à 5 triolets de croche (1263 ms), le Do à 13 triolets de croche (3283 ms) et le La à 34 triolets de croches (8586 ms). Le tout crée une polyrythmie répétée qui continue en autonome avec un lent decrescendo jusqu'à la fin de la pièce.

Sur cette base, le vibra joue des accords pour la première fois dans la pièce, entre la mesure 79 à 83, sans traitement par les délais.

A la mesure 84, le vibra retourne dans les délais 1-6 (alors que les délais du deuxième groupe en boucle continuent à sonner) (cf. Figure 77).

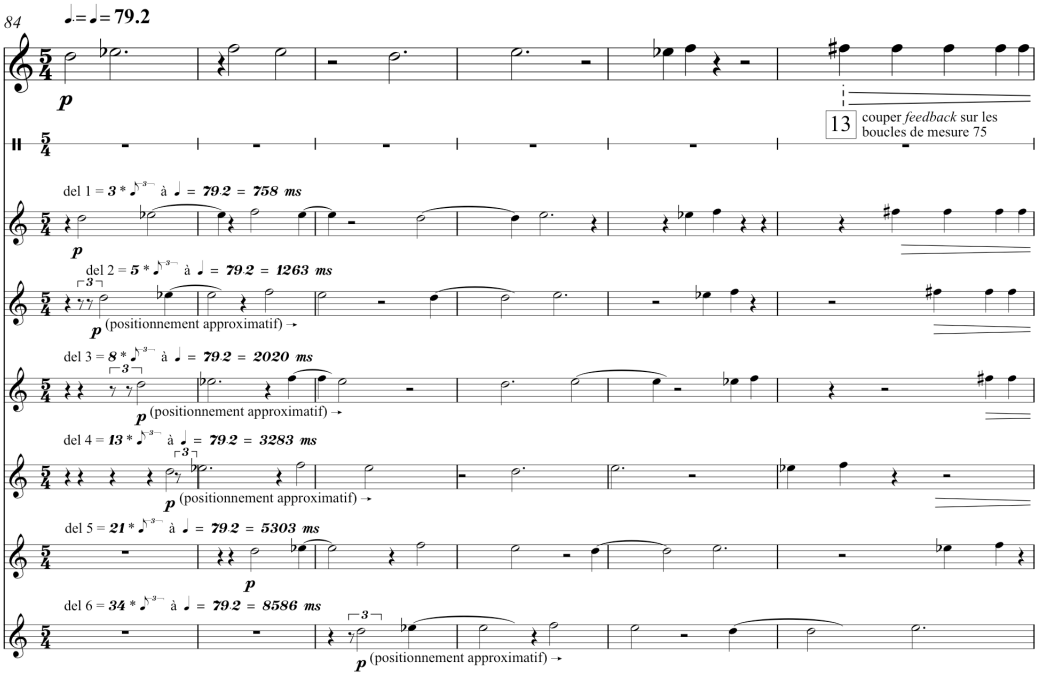


Figure 77: Mesure 84-89

Il y a une nouvelle modulation métrique où la noire pointée devient la noire au nouveau tempo de 79.2. La suite des valeurs du temps pour les délais est identique à celle des boucles, soit 3-5-8-13-21-34. Cette suite du temps en « 3 » soulignent les triolets de croches qui résonne avec les boucles autonome (qui sont toujours audibles).

À la mesure 89, le vibra joue un Fa# répété à la noire pour terminer la pièce dans un

decrescendo lent. Ce decrescendo masque les repères temporelles et le son du vibra se fusionne avec les délais, ainsi faisant une prolongation très organique.

Les boucles disparaissent en même temps que le decrescendo du vibra, et la pièce se termine par un arrêt imperceptible de chaque délai un par un.

Conclusion

Dans ce mémoire nous avons voulu commencer un travail d'exploration des aspects musicaux des traitements temps réel, et nous avons commencé à le faire. Nous avons défini huit catégories de traitement temps réel : le *temps*, *l'amplitude*, la *hauteur*, le *timbre par filtrage*, le *timbre par modulation*, *l'espace*, la *matière*, et le *domaine fréquentiel*. Nous nous sommes lancés dans une première version d'un *traité d'écriture du délai* basé sur de nombreuses expériences personnelles d'écriture et de travail collaboratif dans le temps réel.

Nous avons développé un environnement informatique avec Max/MSP pour la réalisation des traitements temps réel dans le contexte des créations, qui s'appelle *tapemovie,* et nous avons montré cet environnement, par une manière technique et par une manière tutoriale. Nous avons ainsi clarifié le raisonnement derrière les huit catégories de traitement car nous avons montré des exemples de chaque catégorie dans l'environnement.

Nous avons commencé la suite de pièces qui s'appelle *Le patch bien tempéré* en faisant la première qui est le *numéro 1 pour vibraphone et délai*. Nous avons soutenu nos propositions de méthodes d'écriture pour le délai en les appliquant ainsi directement à la composition, et nous avons décrit le processus avec beaucoup de détails en présentant l'analyse de la pièce.

Nous souhaitons maintenant continuer ce travail en poursuivant un doctorat.

Bibliographie

Appleton, Jon H.; Perera, Ronald C.; editeurs, 1975, *The Development and Practice of Electronic Music*, Prentice-Hall Inc, Englewood Cliffs, New Jersey, 384 pages

Barkati, Karim, 2009, *Entre temps réel et temps différé – Pratiques, techniques et enjeux de l'informatique dans la musique contemporaine*, Thèse de Doctorat de l'Université de Paris 8, 332 pages

Battier, Marc; Nelson, Peter; Osborne, Nigel; editeurs, 1999, *Aesthetics of Live Electronic Music*, Contemporary Music Review, Volume 18 Part 3, Harwood Academic Publishers, Singapore, 136 pages

Bonardi, Alain; Barthélemy, Jérôme, 2008, *Le patch comme document numérique : support de création et de constitution de connaissances pour les arts de la performance*, CIDE [En ligne], CIDE 10, Session Document culturel, mis à jour le : 16/09/2008, URL : http://172.16.128.67:50010/cide/index.php?id=298

Deshays, Daniel, 2006, *Pour une écriture du son*, Editions Klincksieck, Paris, 192 pages.

Dobson, Richard, 1992, *A Dictionary of Electronic & Computer Music Technology – Instruments, Terms, Techniques*, Oxford University Press, New York, 238 pages

Dodge, Charles; Jerse, Thomas A.; 1985, *Computer Music – Synthesis, Composition and Performance*, Schirmer Books, New York, 383 pages

Dolson, Mark, 1989, *Fourier-Transform-Based Timbral Manipulations*, *Current Directions in Computer Music Research*, The MIT Press, Cambridge, Massachusetts, pp. 105-112

Garnett, Guy E., 2001, *The Aesthetics of Interactive Computer Music*, article dans *Computer Music Journal*, 25:1, Massachusetts Institute of Technology, Cambridge, Massachusetts, pp. 21–33

Gibson, John, 2009, *Spectral Delay as a Compositional Resource*, publié dans eContact! 11.4, Toronto Electroacoustic Symposium 2009, http://cec.concordia.ca/econtact/11_4/gibson_spectraldelay.html

Manning, Peter, 2004, *Electronic and Computer Music*, Oxford University Press, New York, 474 pages

Mathews, Max V.; Pierce, John R.; editeurs, 1989, *Current Directions in Computer Music Research*, The MIT Press, Cambridge, Massachusetts, 432 pages

Mays, Tom; **Rubiano**, Renaud, 2010, *Tapemovie: un environnement logiciel pour la création intermédia*, Journées d'Informatique Musicales 2010, Rennes (http://jim10.afim-asso.org/actes/81mays.pdf)

Moore, F. Richard, 1989, *Spatialization of sounds over loudspeakers*, *Current Directions in Computer Music Research*, The MIT Press, Cambridge, Massachusetts, pp. 89-103

Moore, F. Richard, 1990, *Elements of Computer Music*, Prentice-Hall Inc, Englewood Cliffs, New Jersey, 560 pages

Moorer, James A., 1979, A*bout this reverberation business*, originally published in Computer Music Journal 3(2):13-28, reprinted in Roads, Strawn 1985

Oliveros, Pauline, 1969, *Tape Delay Techniques for Electronic Music Composers*, *The Composer*, Volume I, No. 3, December1969 (obtenu directement depuis l'auteur).

Pekonen, Jussi; **Välimäki**, Vesa; **Abel**, Jonathan S.; **Smith**, Julius O., 2009, Proc. of the 12th Int. Conference on Digital Audio Effects (DAFx-09), Como, Italy, September 1-4, 2009

Puckette, Miller, 2007, Version en ligne du 30 décembre 2006, *Theory and Techniques of Electronic Music*, World Scientific Publishing Co. Pte. Ltd., Singapore, 337 pages

Reenskaug, Trygve, 2003, The Model – View - Controller (MVC) Its Past and Present, article publié par INTEGRATED DOMAIN SERVICES, http://heim.ifi.uio.no/~trygver/2003/javazone-jaoo/MVC_pattern.pdf

Roads, Curtis, 1985a, *A tutorial on Nonlinear Distortion or Waveshaping Synthesis*, Article in *Foundations of Computer Music*, The MIT Press, Cambridge, Massachusetts (5e Edition 1991), pp 83-94

Roads, Curtis, 1985b, Granular *Synthesis of Sound*, Article in *Foundations of Computer Music*, The MIT Press, Cambridge, Massachusetts (5e Edition 1991), pp 145-159

Roads, Curtis, 1996, *The computer music tutorial*, The MIT Press, Cambridge, Massachusetts, 1234 pages

Roads, Curtis, 2006, *The evolution of granular synthesis: an overview of current research*, article présenté à : International Symposium on The Creative and Scientific Legacies of Iannis Xenakis, 8-10 June 2006, University of Guelph, Toronto, Canada, 14 pages

Rondeleux, Luc, 1999, *Une histoire de l'informatique musicale – entre macroforme et microcomposition*, Journées d'Informatique Musicale 19999, http://www.ai.univ-paris8.fr/~jim99/actes_html/RondeleuxJIM99.htm

Sedes, Anne, 2007, *A propos du temps dans la musique d'Horacio Vaggione*, dans *Espaces Composables – essais sur la musique et la pensée musicale d'Horacio Vaggione*, L'Harmattan, Paris, pages 89-98

Settel, Zack, 2001, *The use of real-time interactive music systems in music composition and performance, Thèse de Doctorat, Université de Montréal, 107 pages*

Strange, Allen, 1972 (1983 2e édition), *Electronic Music – Systems, Techniques and Controls*, Wm. C. Brown Company Publishers, Dubuque, Iowa, 274 pages

Thigpen, Benjamin, 2009, *Spatialization Without Panning*, publié dans eContact! 11.4, Toronto Electroacoustic Symposium 2009, http://cec.concordia.ca/econtact/11_4/thigpen_spatialization.html

Truax, Barry, éditeur, 1978, *Handbook for Acoustic Ecology*, A.R.C. Publications, Vancouver, British Columbia, 166 pages

Zelli, Bijan, 2009, *Space and Computer Music : A SURVEY OF METHODS, SYSTEMS AND MUSICAL IMPLICATIONS*, publié dans eContact! 11.4, Toronto Electroacoustic Symposium 2009, http://cec.concordia.ca/econtact/11_4/zelli_space.html

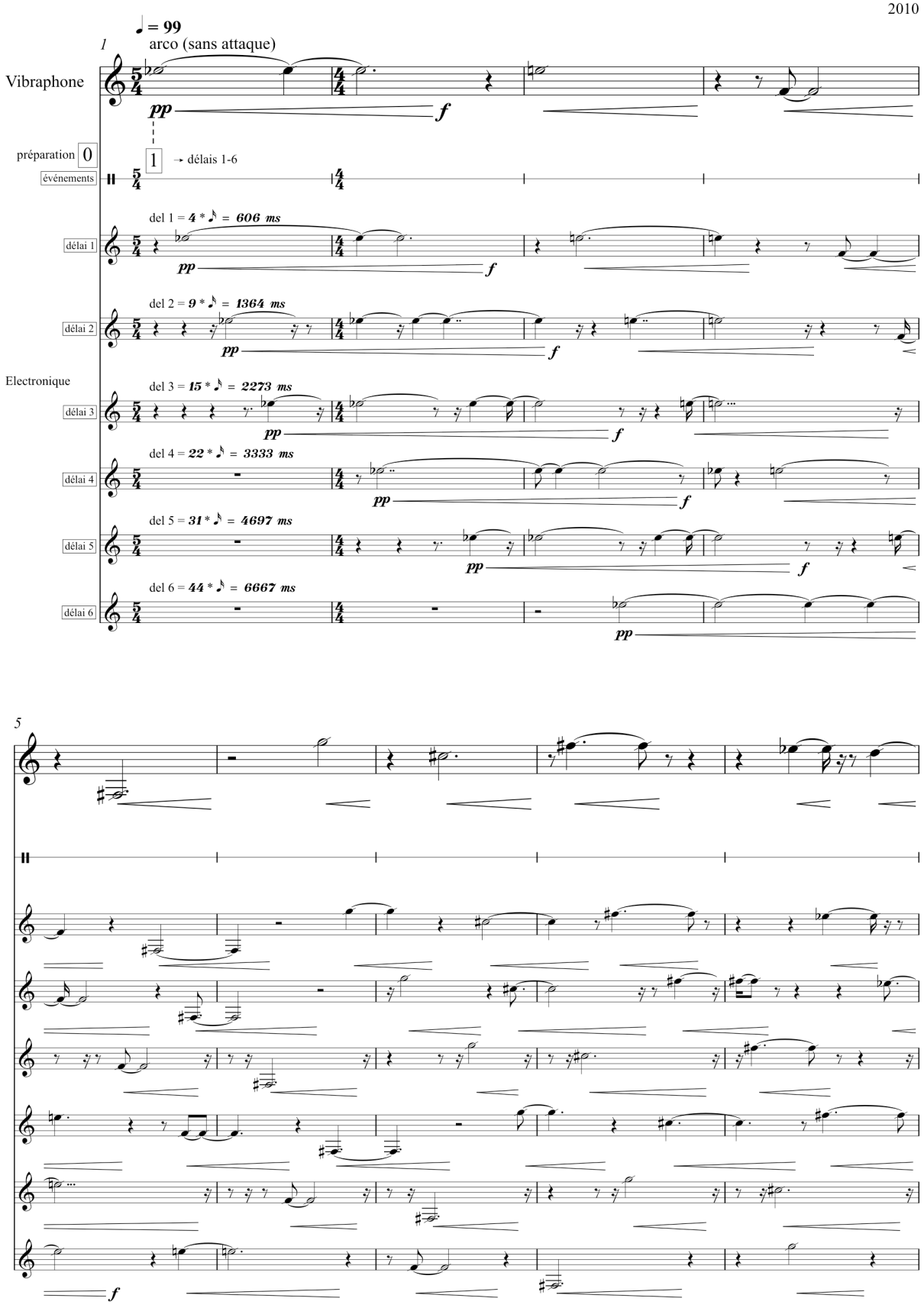
Annexe

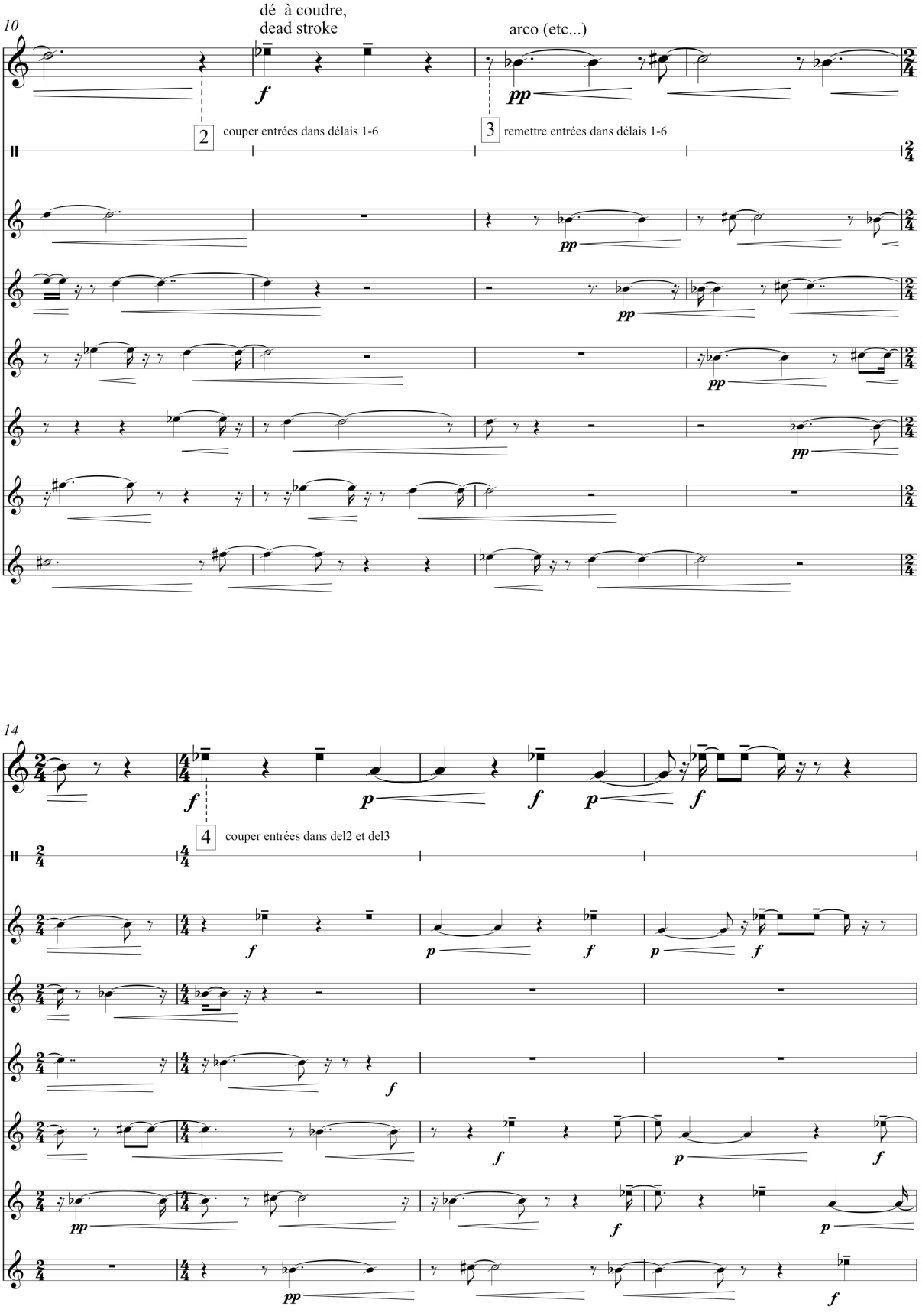
La partition complète du Patch bien tempéré numéro 1

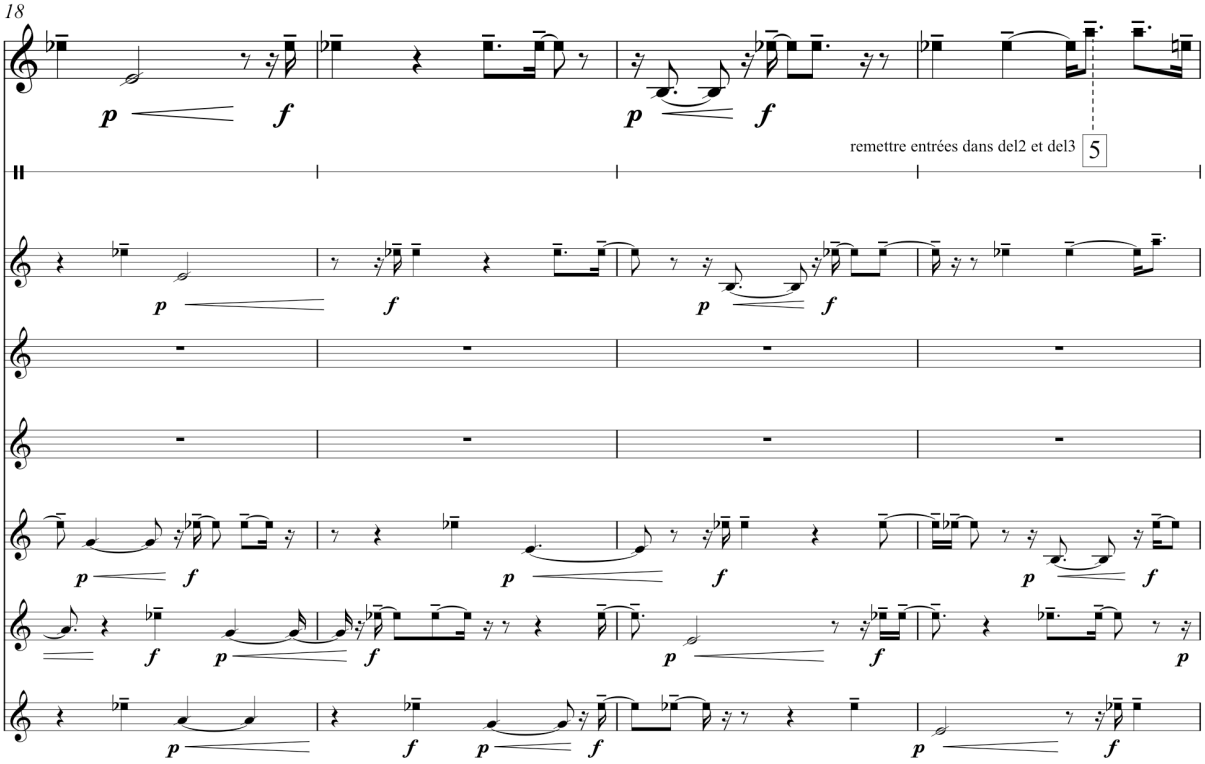
Le patch bien tempéré numéro 1

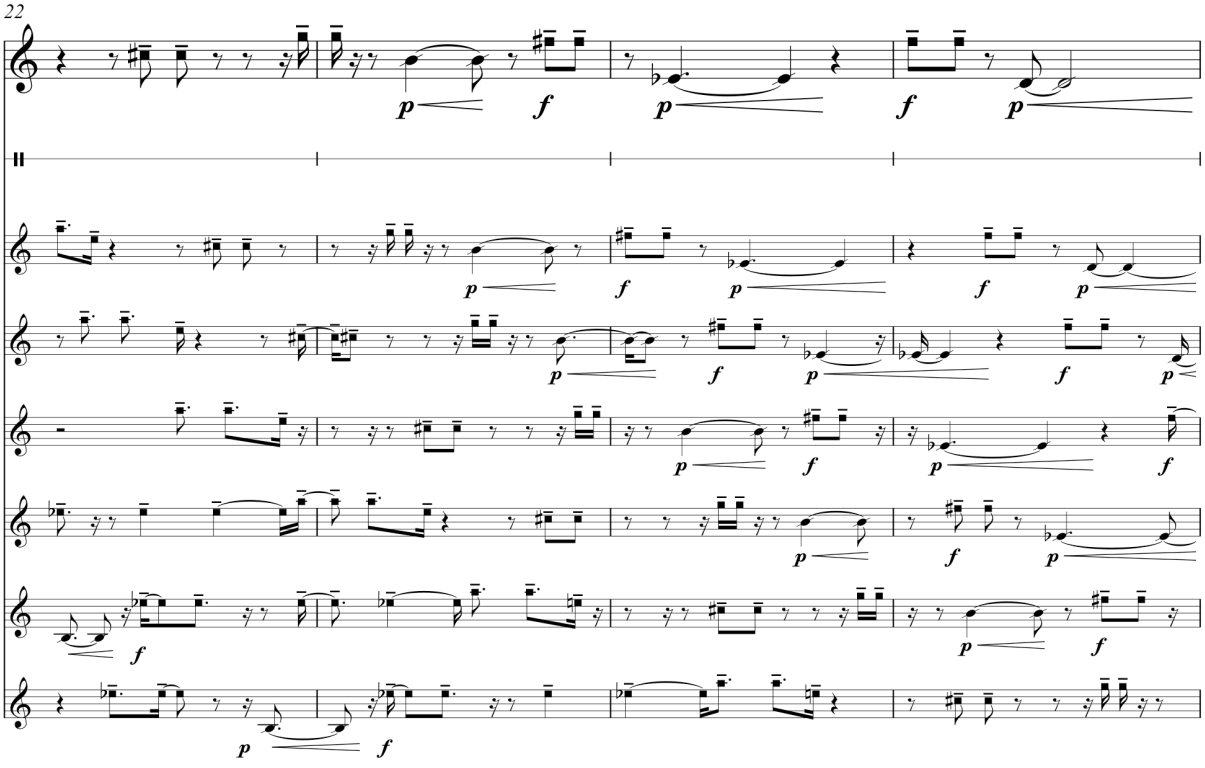
pour vibraphone et traitement temps réel (délai)

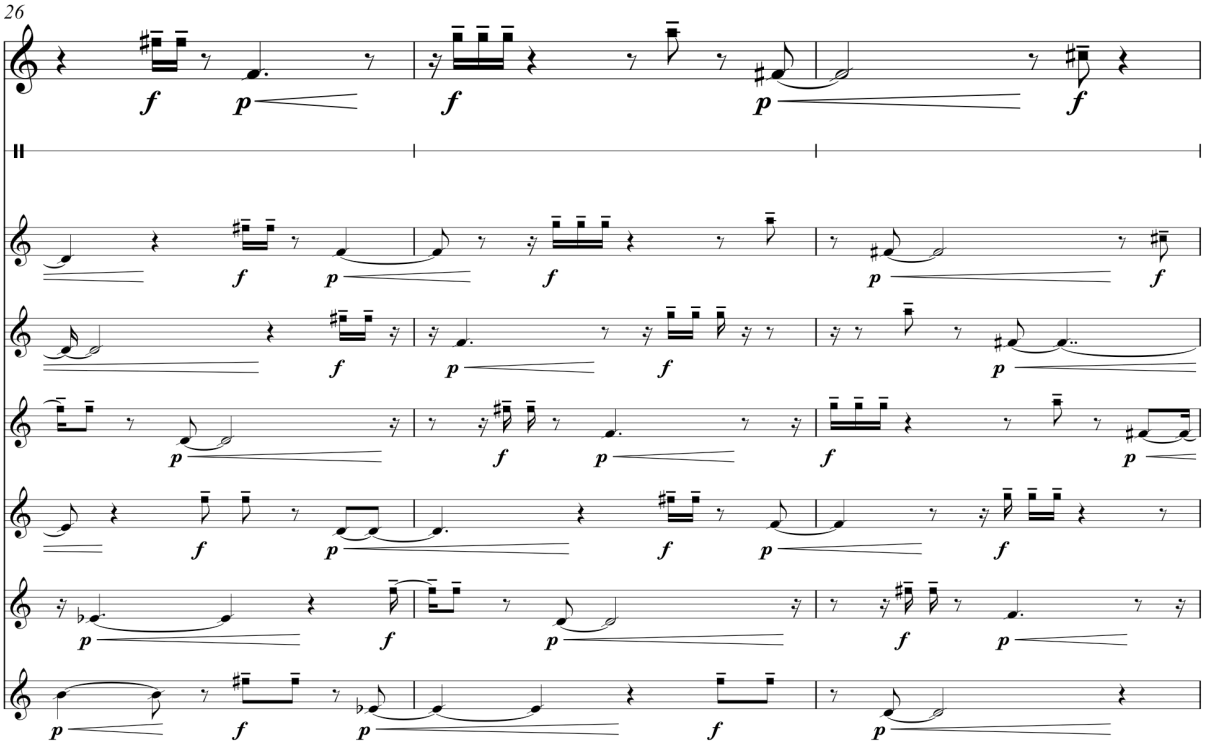
Tom Mays
2010

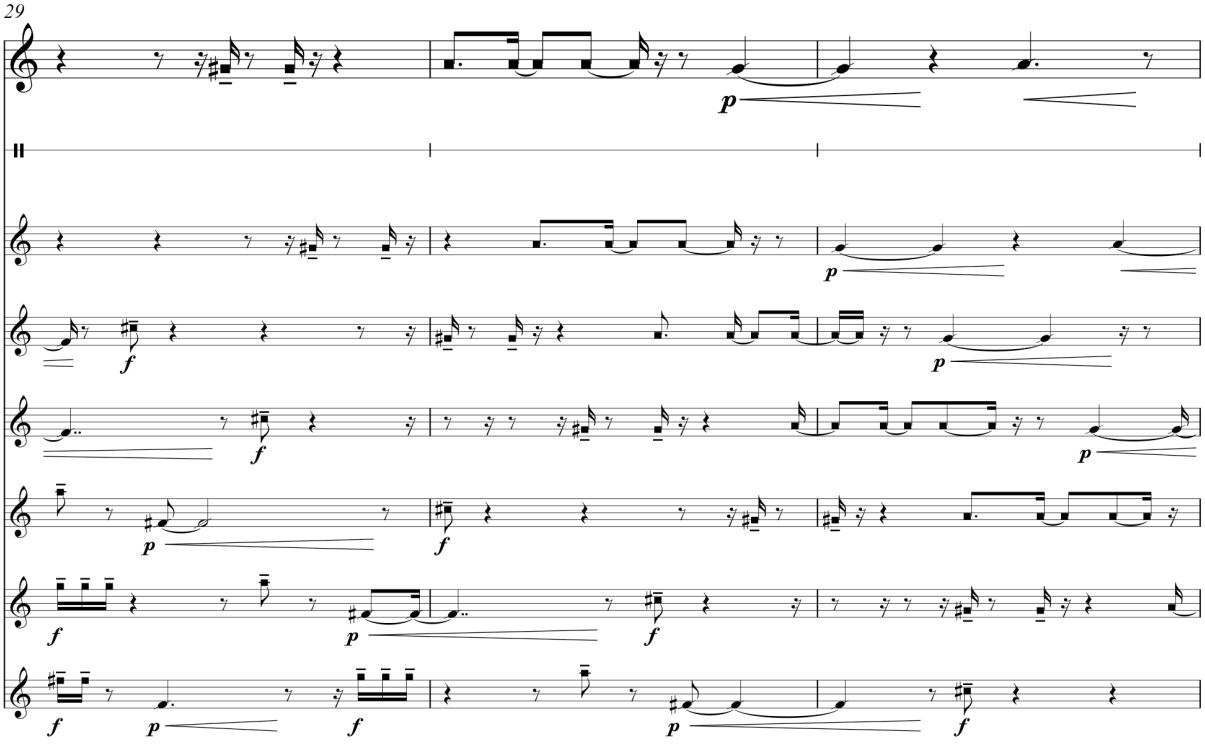


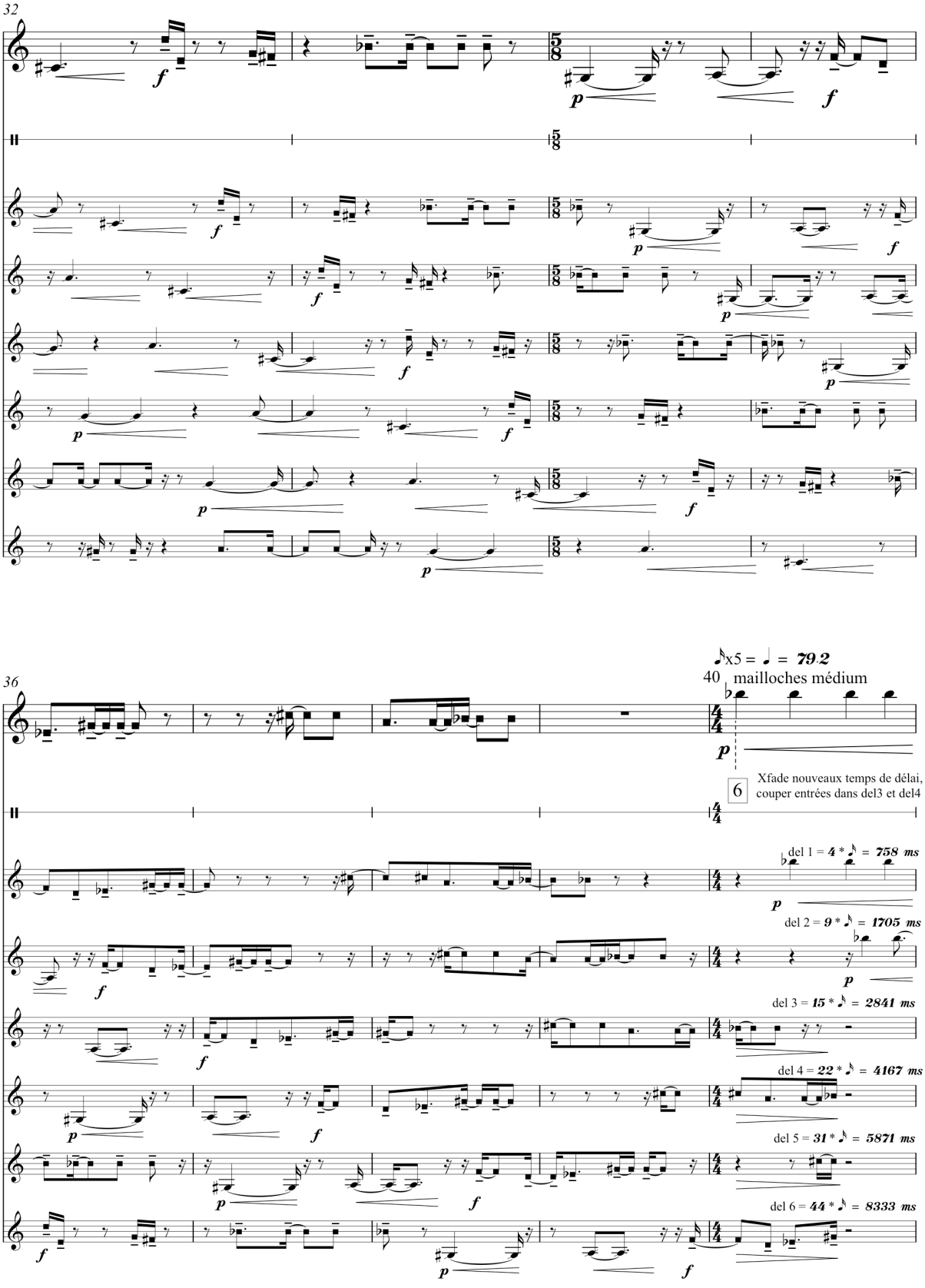

10
dé à coudre,
dead stroke
arco (etc...)
f
pp
2
couper entrées dans délais 1-6
3
remettre entrées dans délais 1-6
14
4
couper entrées dans del2 et del3

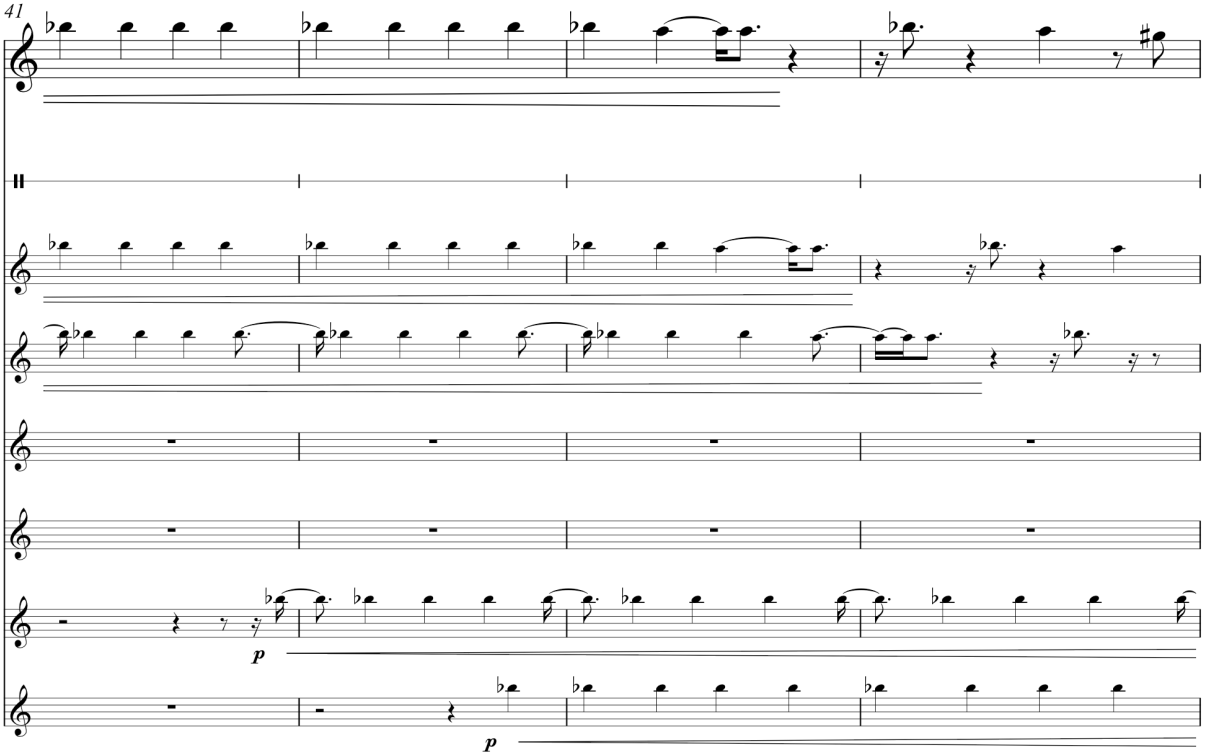

18
remettre entrées dans del2 et del3
5

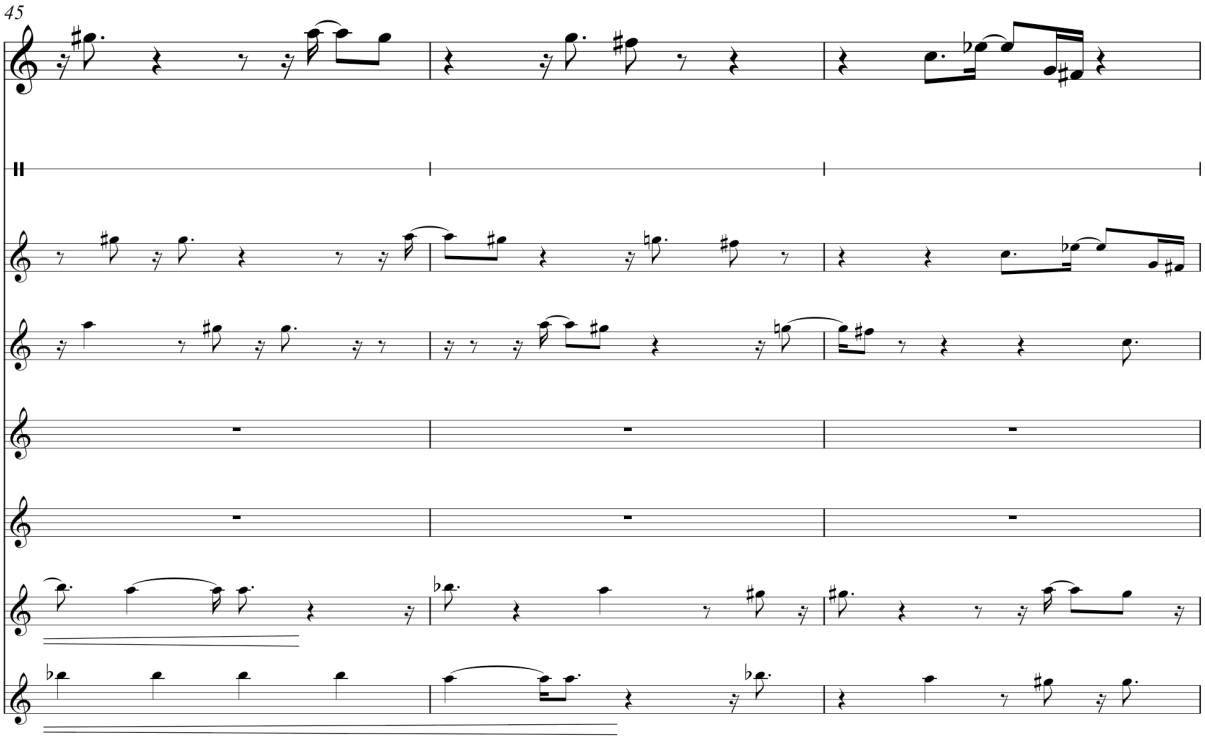

22

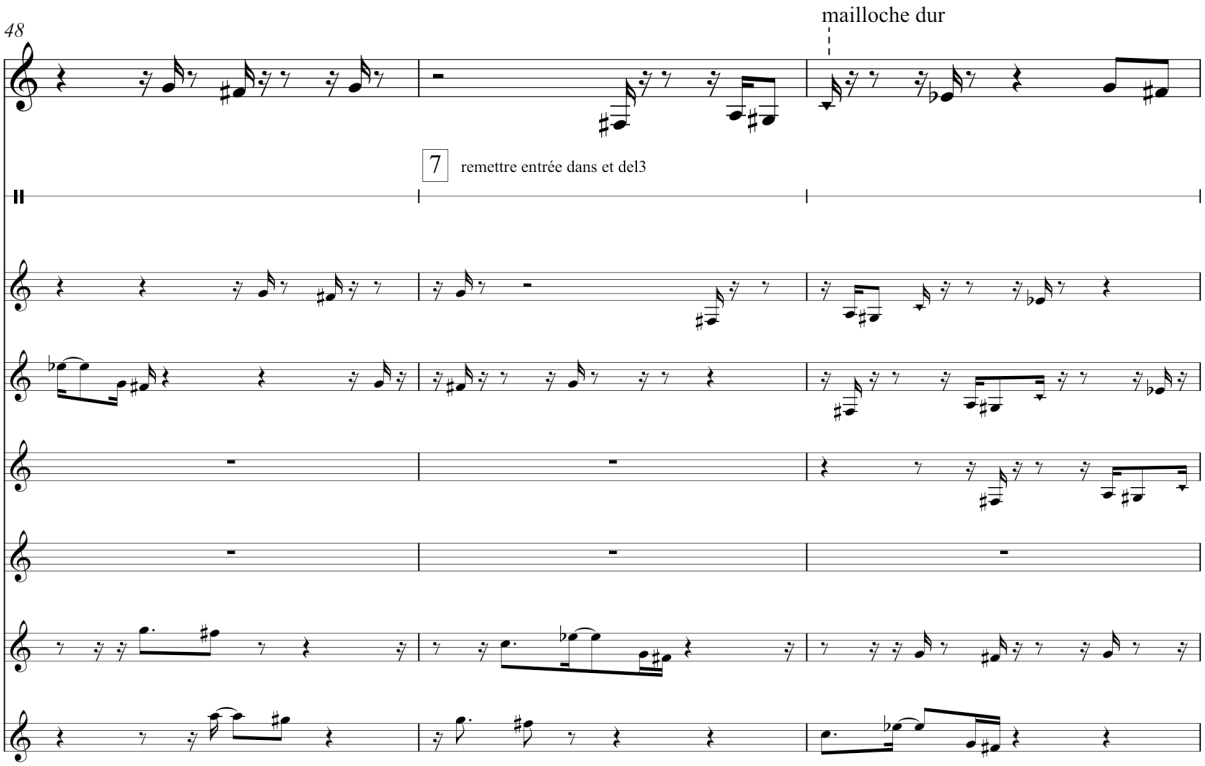

26

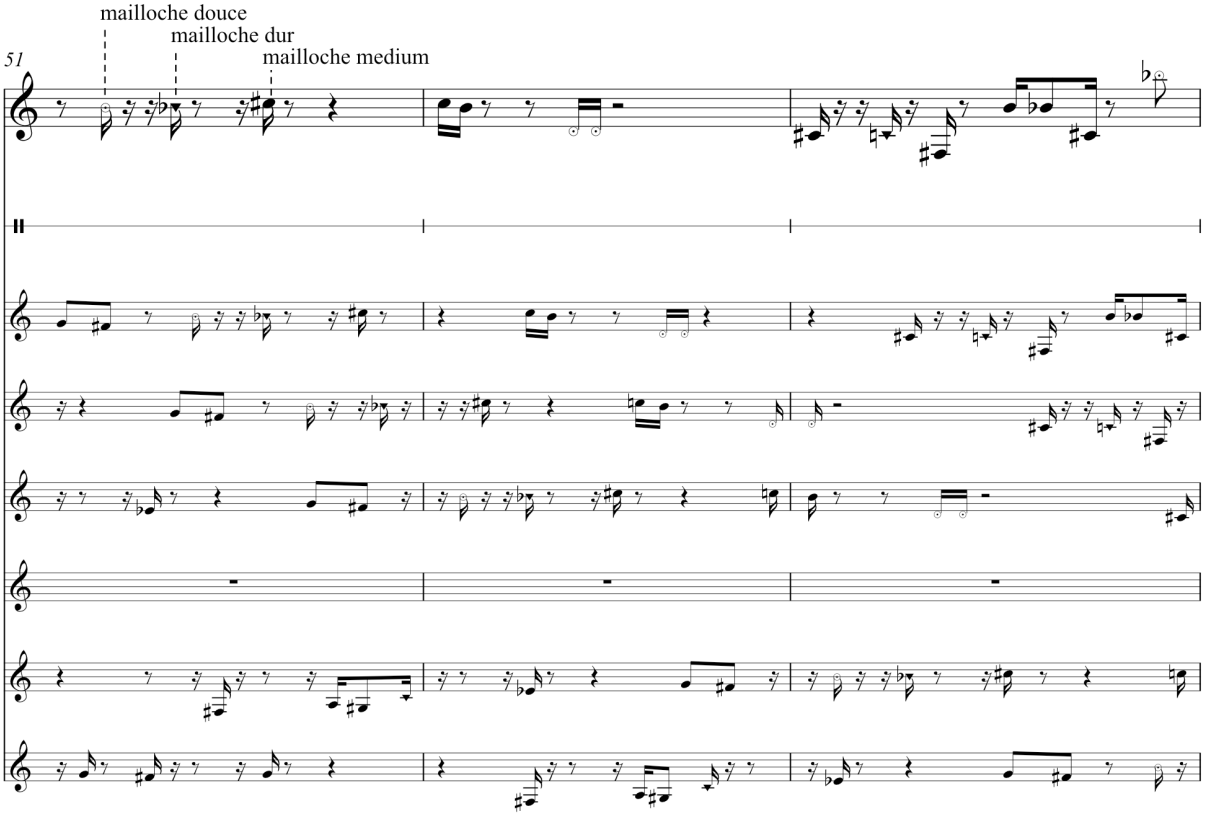

29

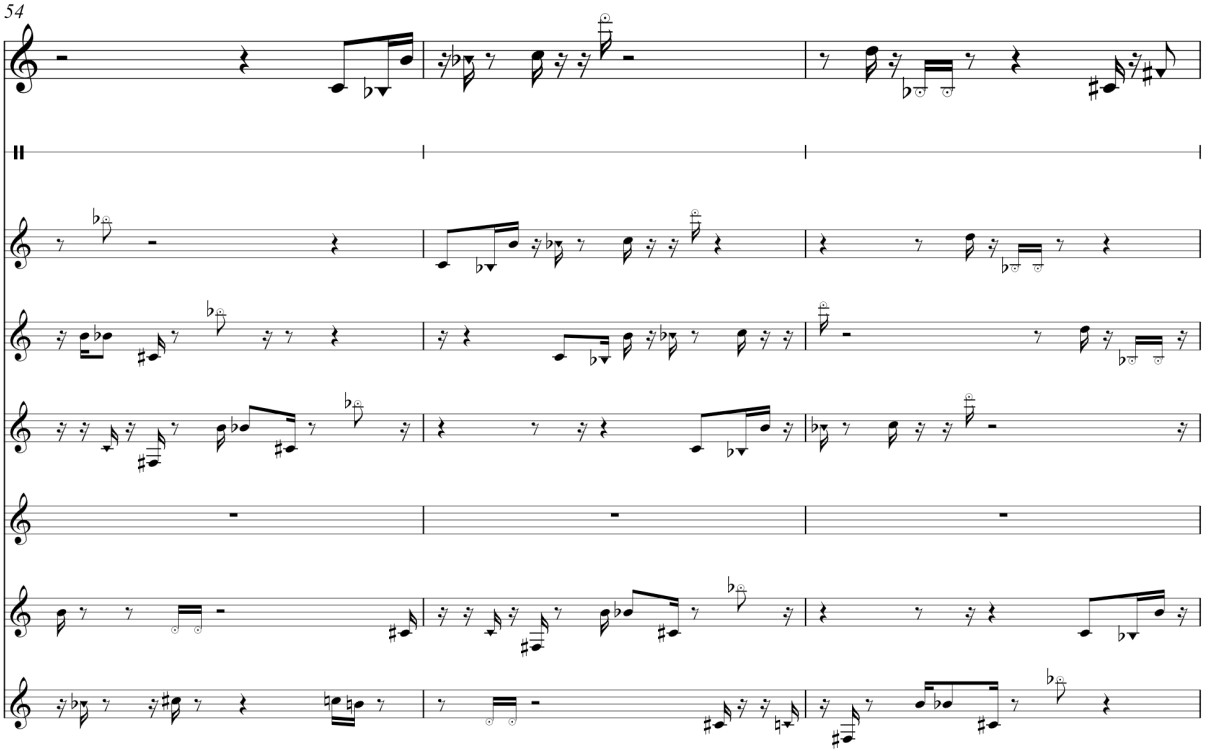

32
36
40
x5 = = 79.2
mailloches médium
6
Xfade nouveaux temps de délai,
couper entrées dans del3 et del4
del 1 = 4 * = 758 ms
del 2 = 9 * = 1705 ms
del 3 = 15 * = 2841 ms
del 4 = 22 * = 4167 ms
del 5 = 31 * = 5871 ms
del 6 = 44 * = 8333 ms

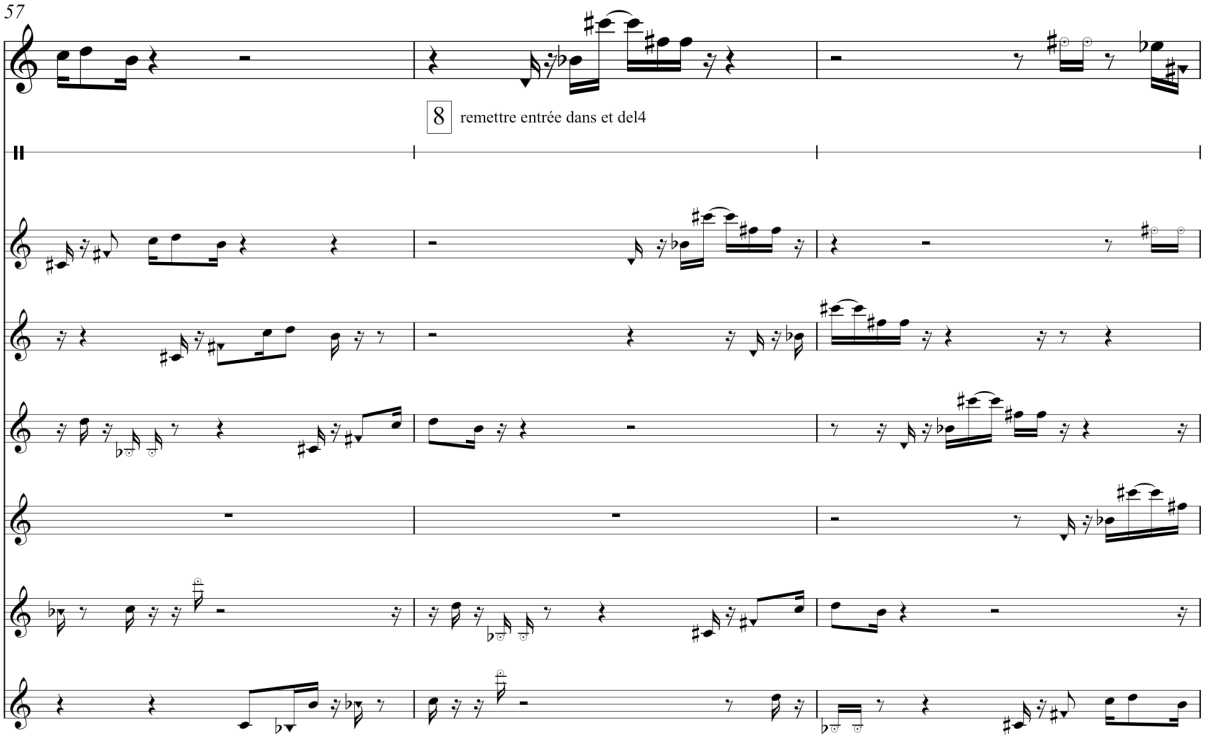

41
p
p

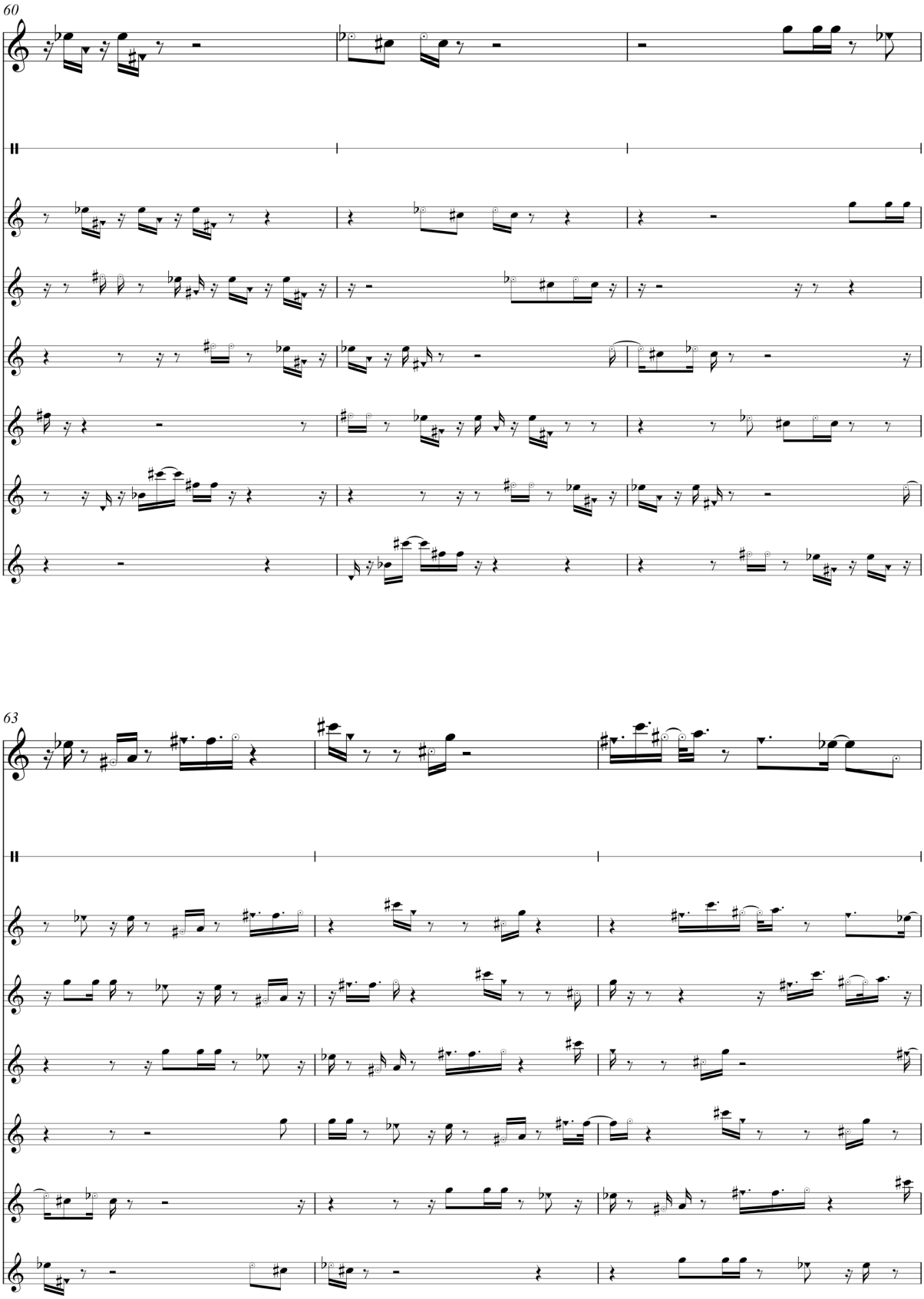

45

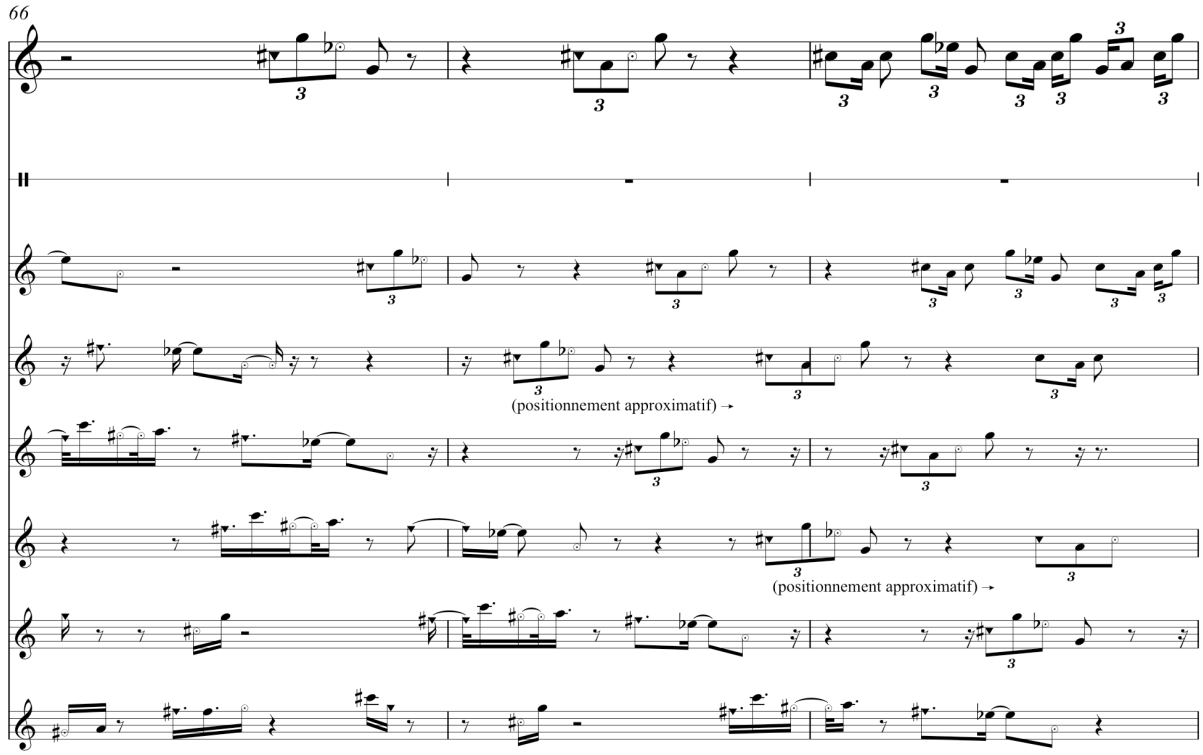

48
mailloche dur
7
remettre entrée dans et del3

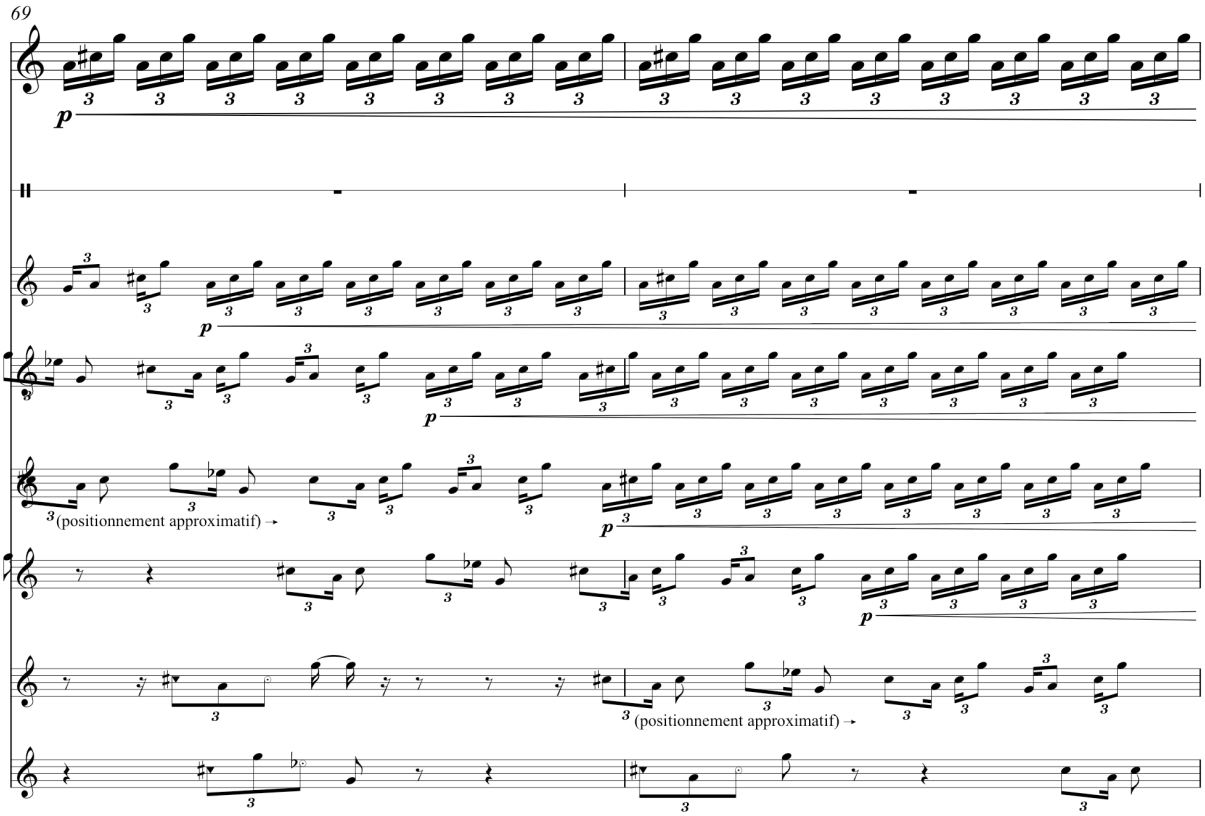

51
mailloche douce
mailloche dur
mailloche medium

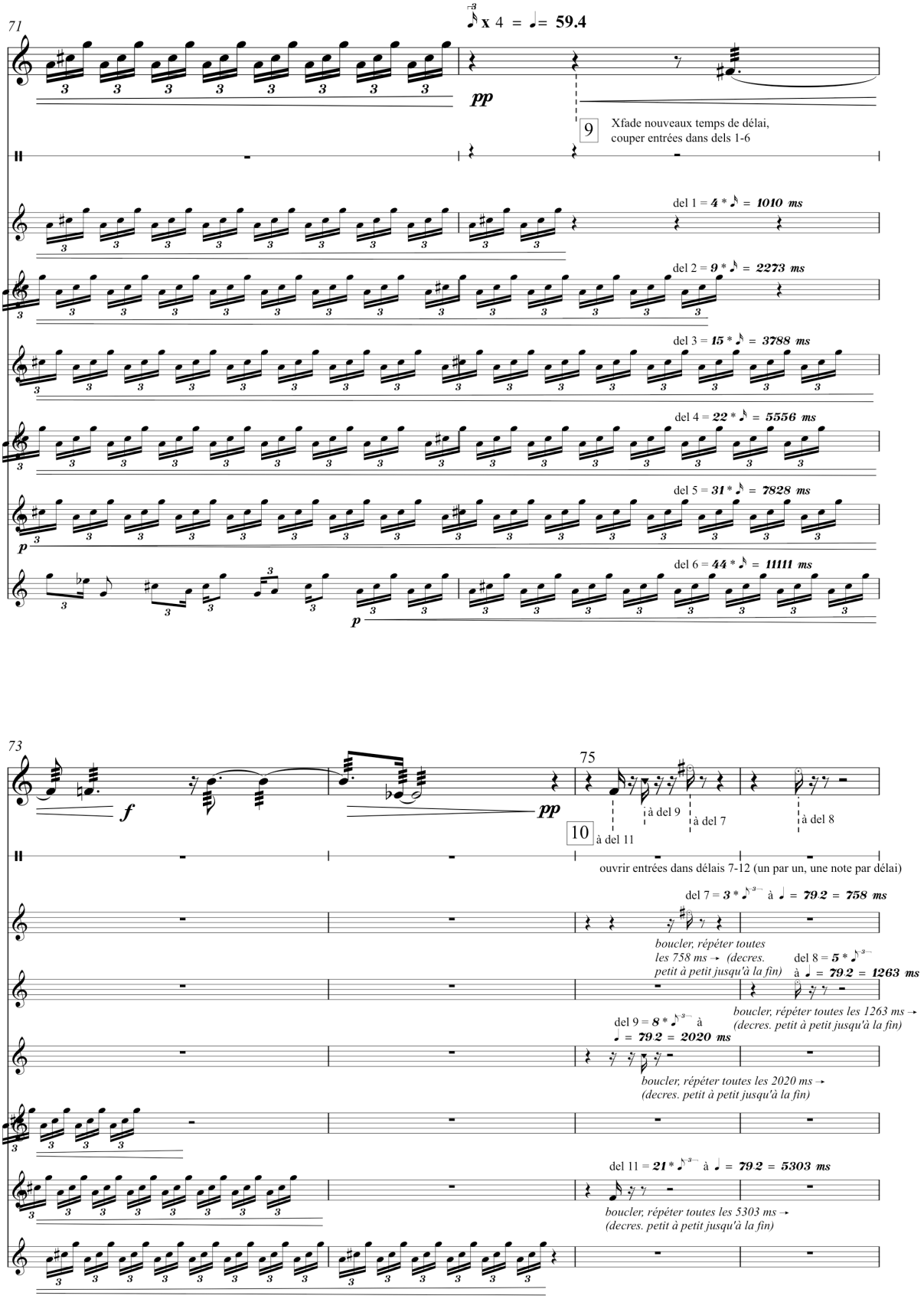

54

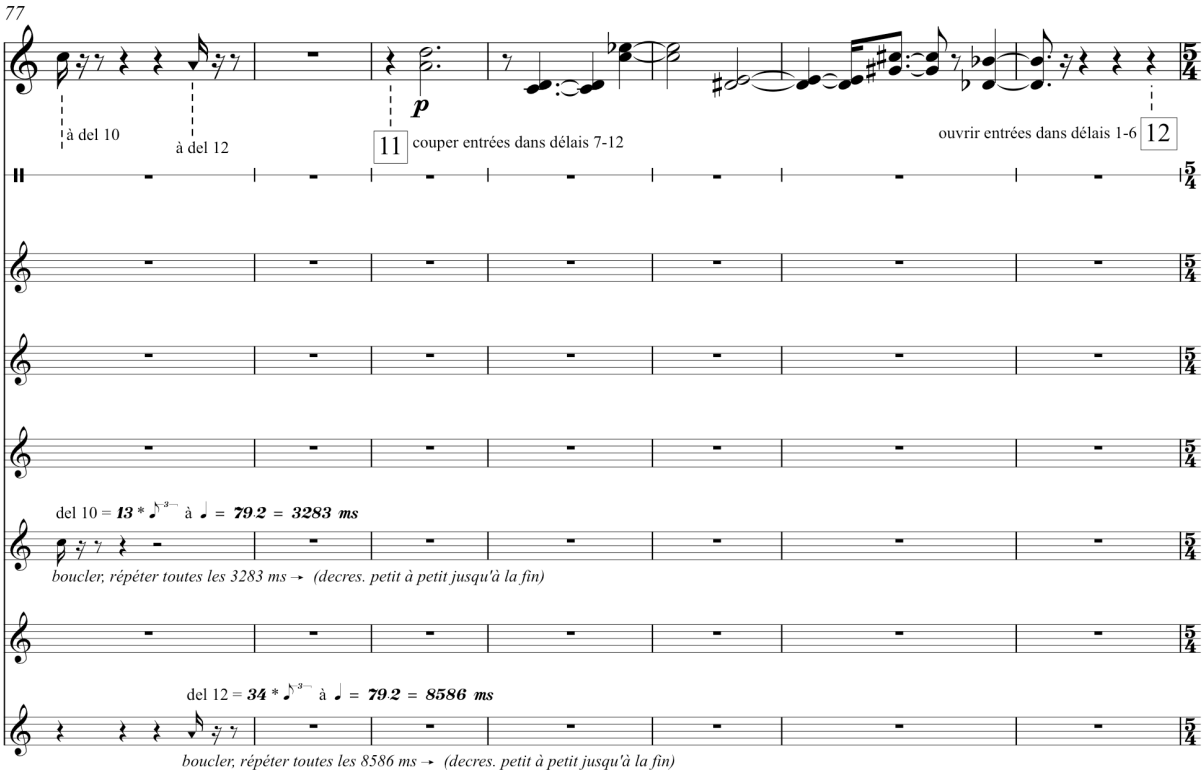

57
8
remettre entrée dans et del4

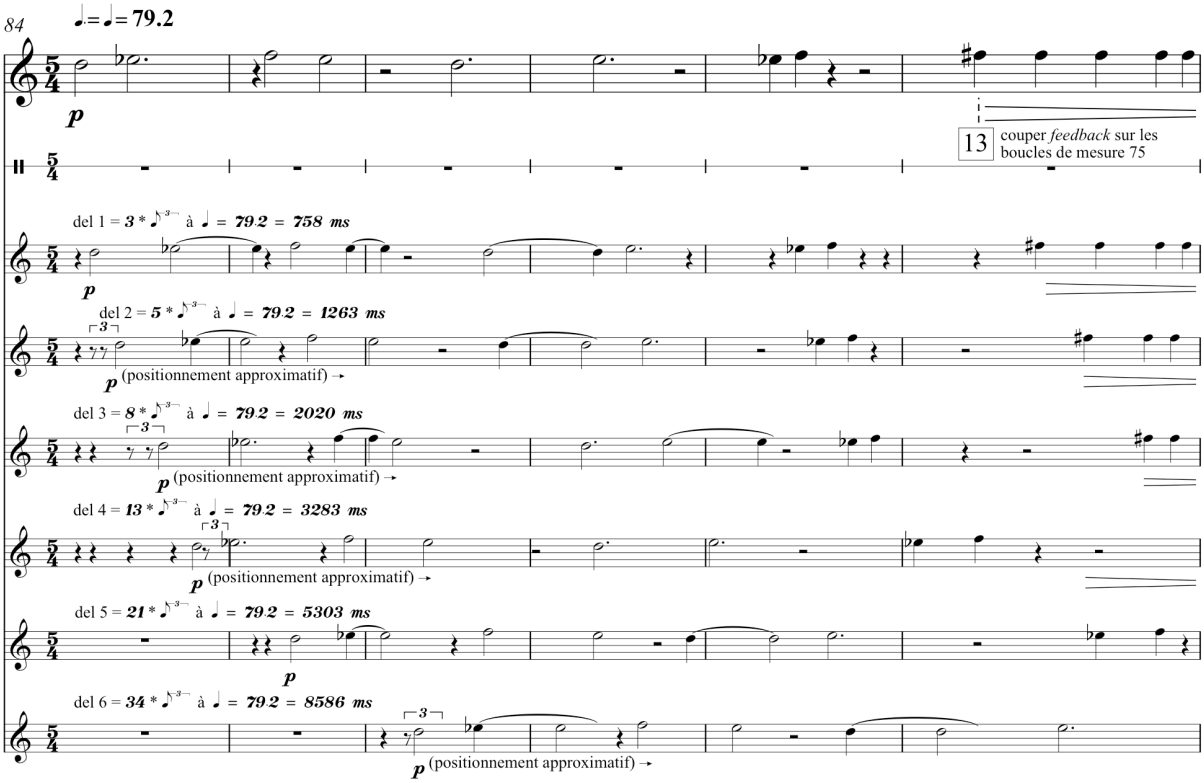

60
63

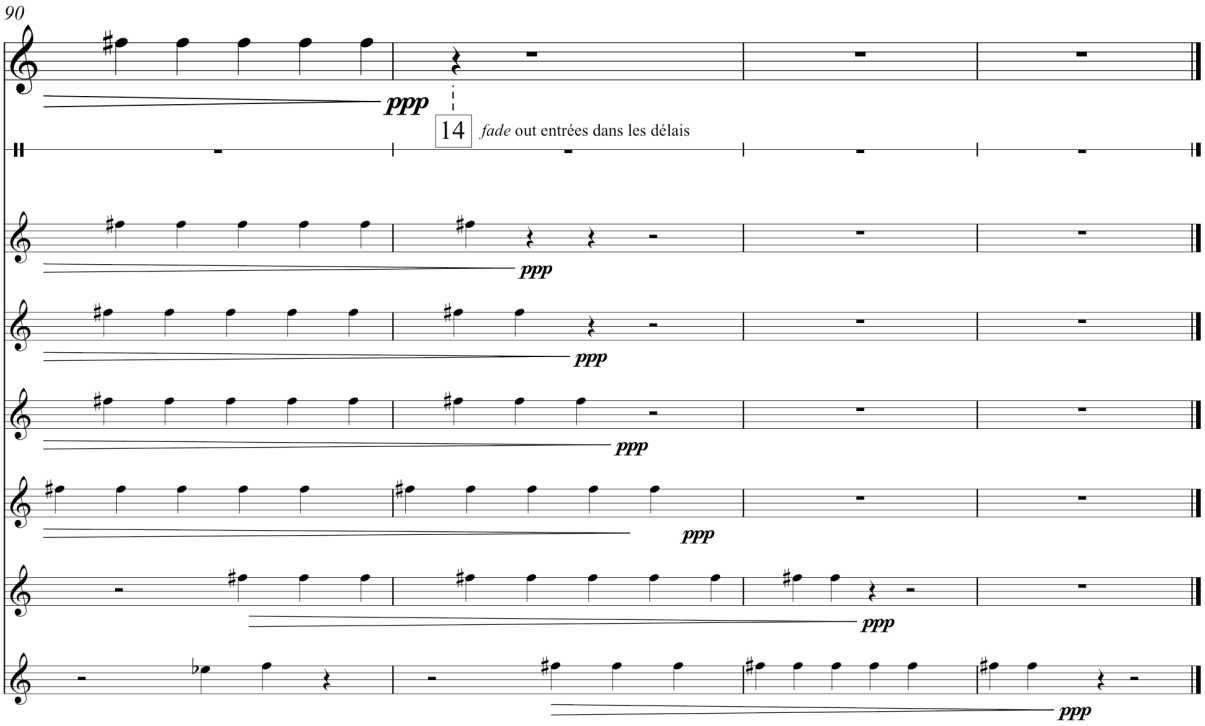

66
(positionnement approximatif) →
(positionnement approximatif) →


69
p
(positionnement approximatif) →
(positionnement approximatif) →


71
x 4 = = 59.4
pp
9
Xfade nouveaux temps de délai,
couper entrées dans dels 1-6
del 1 = 4 * = 1010 ms
del 2 = 9 * = 2273 ms
del 3 = 15 * = 3788 ms
del 4 = 22 * = 5556 ms
del 5 = 31 * = 7828 ms
p
del 6 = 44 * = 11111 ms
p
73
f
pp
75
10
à del 11
à del 9
à del 7
à del 8
ouvrir entrées dans délais 7-12 (un par un, une note par délai)
del 7 = 3 * à = 79.2 = 758 ms
boucler, répéter toutes
les 758 ms → (decres.
petit à petit jusqu'à la fin)
del 8 = 5 *
à = 79.2 = 1263 ms
boucler, répéter toutes les 1263 ms →
(decres. petit à petit jusqu'à la fin)
del 9 = 8 * à
= 79.2 = 2020 ms
boucler, répéter toutes les 2020 ms →
(decres. petit à petit jusqu'à la fin)
del 11 = 21 * à = 79.2 = 5303 ms
boucler, répéter toutes les 5303 ms →
(decres. petit à petit jusqu'à la fin)


77
à del 10
à del 12
11 couper entrées dans délais 7-12
ouvrir entrées dans délais 1-6 12
p
del 10 = 13 * ♪ 3 à ♩ = 79.2 = 3283 ms
boucler, répéter toutes les 3283 ms → (decres. petit à petit jusqu'à la fin)
del 12 = 34 * ♪ 3 à ♩ = 79.2 = 8586 ms
boucler, répéter toutes les 8586 ms → (decres. petit à petit jusqu'à la fin)


84
♩.= ♩.= 79.2
p
13 couper feedback sur les boucles de mesure 75
del 1 = 3 * ♪ 3 à ♩ = 79.2 = 758 ms
del 2 = 5 * ♪ 3 à ♩ = 79.2 = 1263 ms
p (positionnement approximatif) →
del 3 = 8 * ♪ 3 à ♩ = 79.2 = 2020 ms
p (positionnement approximatif) →
del 4 = 13 * ♪ 3 à ♩ = 79.2 = 3283 ms
p (positionnement approximatif) →
del 5 = 21 * ♪ 3 à ♩ = 79.2 = 5303 ms
p
del 6 = 34 * ♪ 3 à ♩ = 79.2 = 8586 ms
p (positionnement approximatif) →


90
ppp
14
fade out entrées dans les délais
ppp
ppp
ppp
ppp
ppp
ppp