Vers un système unifié d'interaction et de synchronisation en composition électroacoustique et mixte : partitions électroniques centralisées

José Miguel Fernández

Sorbonne Université

École doctorale Informatique, Télécommunications et Électronique

(Paris)

IRCAM / Représentations Musicales

Vers un système unifié d'interaction et de synchronisation en composition électroacoustique et mixte : partitions électroniques centralisées

Par José Miguel Fernández

Thèse de doctorat de Composition Musicale

Dirigée par Jean-Louis Giavitto et Pierre Couprie

Présentée et soutenue publiquement le 17 septembre 2021

Devant un jury composé de :

Jean-Louis GIAVITTO	Directeur de thèse
Pierre COUPRIE	Co-directeur de thèse
Alain BONARDI	Rapporteur
Jean-François TRUBERT	Rapporteur
Philippe MANOURY	Examinateur
Yann ORLAREY	Examinateur
Georgia SPIROPOULOS	Examinateur

À mes parents

Remerciements

Aux membres du jury de cette thèse, aux rapporteurs et en particulier à mon directeur et co-directeur de thèse, Jean-Louis Giavitto et Pierre Couprie qui m'ont aidé tout au long de la rédaction de cette thèse avec leurs conseils avisés et leurs disponibilités.

À tous les membres de l'IRCAM qui m'ont accompagné pendant ces années de travail et de recherche avec leurs amitiés, leurs idées et conseils sur différents sujets liés à l'informatique musicale.

À tous les compositeurs avec lesquels j'ai eu la chance de travailler et qui ont partagé leurs pensées musicales et leurs amitiés, en particulier avec Lara Morciano avec qui on a expérimenté différentes idées et formes musicales dans la musique mixte avec l'utilisation de la librairie AntesCollider.

À mes amis percussionnistes Thierry Miroglio et Philippe Spiesser pour leur disponibilités infinis vers l'exploration.

À mes amis vidéastes Raphaël Foulon et Thomas Koppël pour chercher des rapports entre le son et l'image dans une ambiance décontractée sans limites.

À l'association Flashback de Perpignan et son directeur Alexander Vert ainsi qu'à la HEM de Genève pour permettre de créer et expérimenter des nouvelles pièces audiovisuelles en relation avec la captation gestuelle.

À la direction artistique de l'IRCAM qui m'a permis de développer des projets ambitieux et innovants.

À l'équipe de développement du logiciel Antescofo, Arshia, José et Philippe et en particulier Jean-Louis pour le développement du langage Antescofo dont le travail de cette thèse est basé ainsi que pour sa disponibilité permanente et conseils à toutes mes questions pendant tous ces années.

À Alfonsina et Enora pour leurs soutiens et conseils et à mes amis de la Préfecture à Virieu Le Grand où j'ai commencé à écrire cette thèse pendant un confinement.

Sommaire

Résumé

Avec l'avènement de l'informatique et son intégration dans le monde de la composition musicale, de nouveaux champs de recherche compositionnelle et sonore se sont ouverts. Mais si l'on voit depuis des années une pléthore de générateurs de son et de nouvelles techniques de synthèse, il y a peu de propositions d'outils qui adressent le contrôle et la construction formelle musicale électronique à plusieurs niveaux et qui permettent une intégration fine de l'écriture et des processus électroniques interactifs en temps réel dans la composition.

Le travail initié dans cette thèse vise à développer, dans le contexte des musiques interactives mixtes, électroacoustiques et audiovisuelles en temps réel, une notion de partition électronique centralisée permettant au sein d'un même environnement la définition, la composition et le contrôle général de tous les processus électroniques, leurs interactions et leurs synchronisations avec les événements musicaux, gestuels et visuels.

L'enjeu artistique est l'écriture et la réalisation d'interactions mettant en rapport la liberté interprétative de l'artiste sur scène et les processus sonores en temps réel à partir de dispositifs de captation efficaces et de mécanismes de synchronisation. La centralisation et la coordination de ces interactions au sein d'une même partition ont pour objectif une intégration fine des processus électroniques dynamiques et génératifs avec différents médias temporels. Le développement des outils informatiques correspondant doit permettre, lors de l'interprétation d'une œuvre musicale ou scénique, de réaliser les relations temporelles complexes exprimées dans la partition, en contrôlant en temps réel les flux d'événements interconnectant les performeurs, la partie électronique, le public, les équipements permettant de gérer les dispositifs scéniques et les systèmes de production, de diffusion et de transformation du son.

En s'appuyant sur des nouveaux langages plus expressifs pour l'écriture de l'électronique comme Antescofo et des systèmes performants de synthèse et traitement du signal comme SuperCollider, ce travail s'est concrétisé par le développement d'une librairie dédiée : AntesCollider. Cette librairie permet d'expérimenter des nouvelles approches de l'écriture de l'électronique à travers l'organisation et la composition de structures sonores, multitemporelles, multi-échelles et de l'interaction. En tirant partie des notions informatiques d'agents, de processus et d'algorithmes temps réel, ces structures sonores peuvent se combiner dynamiquement et polyphoniquement en relation avec des événements externes (captation gestuelle, interconnexion avec d'autres médias), ouvrant vers de nouveaux paradigmes compositionnels et renouvelant la liberté et la plasticité de la création musicale.

0. Introduction

0.1. De la notation de l'électronique et de l'interaction à la partition centralisée

Le travail de cette thèse porte sur la notation de l'électronique et de l'interaction, dans le contexte de la musique mixte et de la musique électroacoustique (purement électronique). Il défend une notion, *la partition centralisée*, qui vise à combiner la notation instrumentale et la notation de l'électronique, au sein d'un même document. Et comme le domaine musical s'est élargi aujourd'hui à d'autres médias et d'autres pratiques « mixtes », il s'agit aussi d'inclure dans ce même document la gestion d'autres flux temporisés (lumière, vidéo, mécatronique de scène) et d'intégrer d'autres types de performances (gestes).

L'enjeu artistique est l'écriture et la réalisation d'interactions mettant en rapport la liberté interprétative de l'artiste sur scène et les processus sonores en temps réel à partir de dispositifs de captation efficaces et de mécanismes de synchronisation. La centralisation et la coordination de ces interactions au sein d'une même partition a pour objectif une intégration fine des processus électroniques dynamiques et génératifs avec différents médias temporels.

Cette intégration ne va pas de soi. La question de la notation de l'électronique est toujours vivement débattue. Cette dernière serait par essence procédurale et impliquerait de spécifier tous les paramètres de la fabrication du son, ce qui s'opposerait à la déclarativité et à l'abstraction auxquelles est parvenue la notation occidentale en se focalisant principalement sur les seules hauteur, durée et intensité des sons à produire. Une abstraction désirable car elle permettrait de mieux adresser les problématiques de formes, de polyphonie, d'articulation du discours musical, etc.

Mon travail sur mes propres compositions, mais aussi mon travail de RIM (réalisateur en informatique musicale) [Zat13] pour d'autres compositeurs, m'ont convaincu que cette nature procédurale de la définition du son à produire « reste neutre » par rapport aux autres problématiques plus macroscopiques de l'écriture musicale. Ces problématiques doivent en effet être traitées à un autre niveau de la partition. Encore faut-il pouvoir intégrer au sein de la même partition l'écriture de toutes les parties musicales, et donc aussi la production de l'électronique. Il faut souligner que les intégrer au sein d'un même document ne vise pas à unifier les notations destinées aux instruments et à l'électronique, mais à permettre leur mise en relation explicite afin de faciliter l'écriture de leur polyphonie et de composer leurs articulations.

Cette notion de partition centralisée, qui n'est pas une partition unifiée, a émergé de mon activité de compositeur mais est largement restée utopique par manque d'outils permettant d'incarner cette notion. Un fait nouveau est apparu avec le développement du système Antescofo qui a renouvelé la notion de suivi de partition en couplant une machine d'écoute à un langage de programmation. L'intégration de la fabrication du

son électronique (qui doit être programmé) à la spécification de la partie instrumentale (qui doit être écoutée) devient théoriquement possible. Théoriquement, car Antescofo ne permet pas de spécifier directement la synthèse et la transformation des sons.

AntesCollider, issu de ce travail de thèse, est un système qui vise à doter Antescofo d'une capacité de synthèse et de transformation du son, afin d'offrir un premier exemple de partition centralisée. Dans les faits, les traitements audionumériques sont délégués à SuperCollider, qui est vu comme un moteur de rendu sonore piloté depuis la partition Antescofo.

Cet outil permet de spécifier au sein d'un même document, puis de réaliser lors de l'interprétation d'une œuvre musicale ou scénique, les relations temporelles complexes exprimées dans la partition, en contrôlant en temps réel les flux d'événements interconnectant les performeurs, la partie électronique, le public, les équipements permettant de gérer les dispositifs scéniques et les systèmes de production, de diffusion et de transformation du son.

Notre prototype a été utilisé dans plusieurs compositions qui ont nourri notre réflexion et qui ont permis de valider les développements informatiques. AntesCollider permet d'expérimenter de nouvelles approches de l'écriture de l'électronique à travers l'organisation et la composition de structures sonores, multitemporelles, multi-échelles et de l'interaction. En tirant partie des notions informatiques d'agents, de processus et de réactions, ces structures sonores peuvent se combiner dynamiquement et polyphoniquement en relation avec les autres parties de la partition ou avec des événements externes (captation gestuelle, interconnexion avec d'autres médias) ouvrant vers de nouveaux paradigmes compositionnels et renouvelant la liberté et la plasticité de la création musicale.

La seconde partie de cette thèse est dédiée à la présentation des pièces réalisées avec AntesCollider dans le cadre de ce travail et comment un tel système peut apporter des éléments nouveaux de réponses à des problématiques paradigmatiques de l'activité du compositeur.

L'utilisation d'AntesCollider pour réaliser l'électronique de pièces d'autres compositeurs et l'aboutissement de projets multimédias comme *Las Pintas* (projet audiovisuel) ou impliquant le suivi des gestes du performeur comme *GeKiPe* (projet de suivi de geste en collaboration avec la HEM de Genève) indique clairement que certains mécanismes dédiés dans les langages de programmation – comme le contrôle réactif, les acteurs, la représentation de variations temporelles par des données (plutôt que du contrôle) ou la synchronisation – apportent des réponses concrètes à la notation de l'électronique et de l'interaction, tout en ouvrant un espace de pensée fécond pour la musique en elle-même.

AntesCollider n'est qu'une première étape dans la concrétisation de la notion de partition centralisée. Mais les résultats obtenus m'ont convaincu qu'il faut poursuivre dans cette voie.

0.2. Organisation de ce document

Le **chapitre I** élabore la notion de partition centralisée. Celle-ci prend la forme d'une spécification, qui est exécutable par une machine – un programme – pour la partie qui porte sur l'électronique.

Le **chapitre II** esquisse un panorama des outils informatiques développés pour la composition et la performance de l'électronique. Ces outils se distribuent en trois catégories : les systèmes d'aide à la composition, les systèmes pour le contrôle temps réel de la performance et les systèmes de traitement et de synthèse du son. Une partition centralisée est nécessairement à l'intersection de ces trois catégories. L'analyse des outils existants nous amène à développer notre proposition à partir des systèmes Antescofo et SuperCollider.

Le **chapitre III** est une présentation du système Antescofo. Ce chapitre n'est pas une introduction détaillée, l'objectif étant de montrer comment ce système répond à certaines problématiques compositionnelles discutées précédemment. Il s'agit aussi de donner les éléments qui permettent la compréhension des développements de la librairie AntesCollider et les exemples présentés à l'occasion de la description des pièces réalisées dans le cadre de cette thèse.

Le **chapitre IV** remplit un objectif similaire au chapitre III, mais pour le système SuperCollider. Nous nous focalisons sur l'architecture client/serveur qui nous permettra d'adresser directement le serveur comme moteur de rendu sonore, ainsi que sur les propriétés d'extensibilité du système qui sont une garantie de sa pérennité.

Le **chapitre V** détaille la conception et la réalisation de la librairie AntesCollider que j'ai développée au cours de cette thèse. Cette librairie constitue une première incarnation d'un environnement adressant la notion de partition centralisée.

Ce chapitre clôt la première partie de cette thèse. La seconde partie est dédiée à la présentation de plusieurs créations qui ont été développées dans le cadre de cette thèse. Plutôt que de présenter ces pièces une à une, ce qui aurait été peut-être fastidieux et peu éclairant, j'ai choisi de les présenter à travers plusieurs problématiques compositionnelles qu'elles illustrent plus particulièrement.

Le **chapitre VI** présente ma posture de compositeur et mon exigence de liberté musicale. Cette exigence se traduit par un besoin, toujours insatisfait, d'expressivité.

Le **chapitre VII** aborde la thématique de l'écriture de l'espace, qu'il soit utilisé pour localiser et rendre distinguable des sources sonores, ou au contraire pour réaliser une synthèse spatiale. Le contrôle de l'espace physique est aussi abordé à travers le projet que j'ai mené avec la compagnie de théâtre ultra de Lucerne.

Mon travail sur des pièces purement acousmatiques est traité au **chapitre VIII**, tandis que le **chapitre IX** se concentre sur les pièces de musique mixte.

Enfin, le **chapitre X** aborde la problématique d'une partition centralisée qui intègre une dimension visuelle, à travers la pièce audiovisuelle *Las Pintas* qui a été développée

pour la Satosphère à Montréal, un dôme de projection hémisphérique doublé d'un système de diffusion ambisonique pour la spatialisation du son.

Le dernier chapitre esquisse quelques perspectives ouvertes par ce travail et les travaux futurs que je souhaite entreprendre.

0.3. Contributions

Le travail effectué pendant cette thèse s'est concrétisé par le développement logiciel de la librairie AntesCollider.

Cette librairie a été utilisée pour plusieurs créations musicales formalisées dans le cadre de cette thèse et chacune comporte donc aussi d'importants développements logiciels spécifiques (tant dans le langage Antescofo que des patchs Max) aux côtés d'un travail de composition plus classique. Ces pièces ont donné lieu à plusieurs concerts publics, en France et à l'étranger (Pékin, Moscou, Saint-Pétersbourg, Rome, Montréal, Genève).

Mon travail de création musicale a été soutenu par plusieurs résidences obtenues sur concours.

Les idées et les outils issus de ce travail ont aussi largement été diffusés à travers des séminaires et des cours présentés devant un public de musiciens et d'informaticiens dans le domaine de l'informatique musicale.

Développements logiciels

La librairie AntesCollider est constituée d'une librairie Antescofo et de programmes compagnons en *sclang*. La librairie Antescofo représente 24 239 lignes de code définissant 45 processus, 34 acteurs et 69 fonctions.

J'ai aussi développé antescollider_spatial_interface, un programme de visualisation adapté au contrôle et à la représentation graphique des données issues de modèles physiques que j'utilise pour le pilotage de processus de synthèse. Ce programme représente 2 085 lignes de code.

Ces logiciels sont des logiciels libres. Ils sont actuellement diffusés auprès de quelques compositeurs qui m'en ont fait la demande, mais ils seront accessibles librement avec l'achèvement de mon travail de thèse et après un peu de nettoyage.

Création musicale en relation avec la recherche

La librairie AntesCollider a été validée à travers plusieurs de mes créations réalisées pendant ce travail de thèse :

- ***Hypersphères***, Projet GeKiPe (HEM de Genève) pour un performeur, captation gestuelle et vidéo générée en temps réel, 4 758 lignes de code, durée 10-15 minutes
- ***Homotopy*** pour percussions, captation gestuelle et électronique, 11 391 lignes de code, durée 17 minutes

- ***Crossing Points***, œuvre collective avec Alexander Vert, Thomas Köppel et Philippe Spiesser pour 5 toms, électronique, captation gestuelle et vidéo en temps réel, 4 390 lignes de code, durée environ 30 minutes
- ***Curvatura II***, œuvre électroacoustique en HOA, entièrement écrite dans le langage Antescofo, 17 442 lignes de code, durée 11 minutes
- ***Las Pintas***, audiovisuel en temps réel et en HOA, 10 192 lignes de code, 25 minutes
- ***Stück für die Schwerkraft***, pièce utilisant un dispositif particulier de 400 trappes pour laisser tomber des objets sur scène, pour la compagnie de théâtre Suisse ultra.

Je cite aussi une pièce antérieure mais qui est la première à utiliser le couplage entre Antescofo et le logiciel SuperCollider :

- ***Dispersion de trajectoires***, pour saxophone baryton et électronique, 15 000 lignes de code, durée d'environ 20 minutes.

Ce travail se complète aussi par la réalisation de l'électronique pour la compositrice Lara Morciano avec les pièces :

- ***Estremo d'ombra***, pour flûte, saxophones, trombone, alto, contrebasse et électronique
- ***Philiris***, pour piano, captation gestuelle et électronique
- ***Taygeta***, pour percussions transducteurs et captation gestuelle
- ***Lyphira***, pour piano, transducteurs et captation gestuelle.

Concerts en lien avec la recherche

Les événements suivants sont des performances impliquant Antescofo, SuperCollider et/ou AntesCollider :

- 06/02/18 : Perpignan, concert musiques mixtes
- 27/08/18 : Tokyo, Suntory Hall avec Philippe Manoury
- 29/11/18 : Amiens, reprise de la pièce Amas pour 5 instruments et captations gestuelles
- 14/12/18 : reprise de *Raggi di stringhe* de Lara Morciano à la Cité de la musique
- 22/01/19 : Genève, concert hommage à Eric Daubresse, informatique musicale pour *Nachmusik* d'Emmanuel Nunes
- 22/02/19 : Perpignan, création de *Double jeux* pour violon, vidéo et électronique en temps réel
- 19/06/19 : reprise de *Raggi di stringhe* de Lara Morciano à Dijon

- 18/06/19 : concert au festival Les Musiques du GMEM, informatique musicale pour *Chuchotements burlesques* de Alireza Farhang
- 22/06/19 : concert à ICMC New York, informatique musicale pour *Liphyra* de Lara Morciano
- 15/06/19 : concert hommage à Eric Daubresse au Centre Pompidou, informatique musicale pour *Nachmusik* d'Emmanuel Nunes
- Projet *GeKiPe* HEM de Genève, captation gestuelle pour performances audiovisuelles en temps réel. Concerts :
    - 06/07/18 : Perpignan
    - 03/08/18 : Pékin
    - 23/11/18 : Saint-Pétersbourg
    - 24/11/18 : Moscou
    - 08/02/19 : Genève
    - 05/07/19 : Rome
    - 05/08/19 : Pékin
    - 14/11/19 : Globe du CERN, Genève
- 18/01/2020 : concert *GeKiPe*, Perpignan
- 18-19/01/2020 : *Curvatura II*, concert anniversaire de l'Ensemble Vortex de Genève
- 29/02/2020 : *Homotopy*, pièce pour percussions, capteurs et électronique temps réel, Columbia Global Center Paris
- 04/03/2020 : *Las Pintas*, pièce audiovisuelle avec Raphael Foulon (vidéo), concert IrcamLive, Centre Pompidou
- 08-10/07/2020 : enregistrement de *Crossing Points* pour percussions, captation gestuelle et vidéo en temps réel au théâtre de l'Archipel à Perpignan
- 25/09/2020 : *Crossing Points* pour percussions, captation gestuelle et vidéo en temps réel, festival Facyl de Salamanque, Espagne.

Concerts prévus mais annulés en raison des conditions sanitaires en 2020 : Saragosse (Espagne), Saarbrücken (Allemagne), Strasbourg (France).

Résidence recherche-création

Depuis le début de cette thèse, j'ai obtenu plusieurs résidences en lien avec mon travail :

- du 28/06/18 au 08/07/18 : **Perpignan**, nouvelle création pour violon, électronique et vidéo en temps réel. Suivi audio/vidéo du jeu instrumental
- du 16 au 25/11/18 : Russie, résidence et masterclass au conservatoire Tchaïkovski de **Moscou** et à l'Institut français de **Saint-Pétersbourg** sur Antescofo

- du 23 au 31/01/19 : **Stockholm**, résidence à EMS (Elektronmusikstudion) pour la nouvelle création pour violon avec Diego Tosi, violon et Thomas Penanguer, vidéo
- du 18 au 23/02/19 : **Perpignan**, préparation du concert *Double jeux* pour violon, vidéo et électronique en temps réel
- octobre 2019 : résidence à la SAT (Satosphère) de **Montréal** pour la création de *Las Pintas*, une pièce audiovisuelle avec Raphael Foulon à la programmation vidéo
- 08-11/12/2020 et 15-20/02/2020 : résidence avec la compagnie de théâtre ultra à Lucerne, Suisse, pour le projet Apesanteur : conception, développement et mécatronique d'une machine avec 400 trappes pour faire tomber des objets sur la scène.

Workshops, séminaires, cours et diffusion

- 22/02/18 : participation au séminaire « Geste » organisé par le Collegium Musicae à l'Ircam.
- 13-14/10/18 : Jussieu, participation à la Fête de la science avec Collegium Musicae. Présentation des capteurs au public dans des gants pour générer différents types de synthèses sonores.
- 29/11/18 : présentation d'informatique musicale au Conservatoire d'Amiens.
- 08/02/18 : présentation/workshop aux Journées de la percussion suisse à Genève.
- 08/02/18 : visite du Globe du CERN pour la préparation d'une conférence et d'un concert en novembre 2019 dans le cadre de leur programme art-science.
- 29/05/19 : présentation Antescofo et AntesCollider pour les réalisateurs en informatique musicale à l'Ircam.
- 29/05/19 : présentation de AntesCollider au Forum Ircam.
- 4-5/06/19 : présentation de Antescofo et AntesCollider au CIRM, Centre national de création de Nice.
- 21/06/19 : présentation de AntesCollider à ICMC 2019 New York.
- 04/06/19 : présentation de GeKipe et Antescofo à l'Instituto Cervantes de Rome dans le cadre du festival Artescienza du CRM (Centro Ricerche Musicali).
- 10/07/19 : présentation de AntesCollider aux chercheurs de GRAME, Centre national de création de Lyon.
- 09/09/2019 : workshop international GeKiPe. Co-organisation du workshop avec Philippe Spiesser, Alexandre Vert et Pierre Donat Bouillud, à l'HEM Genève.

https://www.hesge.ch/hem/en/recherche-developpement/gekipe-conference
Ce workshop sur invitation a réuni les participants suivants : Frédéric Bevilacqua (Ircam, Paris), Rémy Campos (HEAD et HEM, Genève), Philippe Dinkel (HEM, Genève), José Miguel Fernández (Ircam Paris), Jean-Louis Giavitto (CNRS Ircam, Paris), Fabrice Marandola (McGill University, Montréal), Thomas Penanguer (Ensemble Flashback), Giovanni Santini (Hong Kong Baptist University), Philippe Spiesser (HEM, Genève), Rebeca Stewart (Imperial College London), Alexander Vert (Ensemble Flashback), Daniel Zea (HEAD, Genève).

- 15/10/2020 : présentation d'un poster aux Rencontres nationales de recherche en musique.
- 11/02/2021 : présentation (avec Jean-Louis Giavitto) : Antescofo et AntesCollider. Ircam Forum à Montréal (webinaire).
- 12/02/2021 : participation à la table ronde « Composition électronique », Ircam Forum à Montréal.

Un cours d'informatique musicale et de composition à Geïdaï University de Tokyo prévu en novembre 2020 a été annulé suite aux conditions sanitaires.

Publications

- J.-M. Fernandez, « AntesCollider », Poster présenté à Recherches en musique, Rencontres nationales sur les recherches en musique organisées par la direction générale de la création artistique – Ministère de la Culture, 15 et 16 octobre 2020.
- J.-M. Fernandez, J.-L. Giavitto, « Une codynamique formelle », *Culture et Recherche*, n° 141, printemps-été 2020. Dossier thématique « Cinéma, audiovisuel, son ».
- J.-M. Fernandez, J.-L. Giavitto, P. Donat-Bouillud, « AntesCollider: control and signal processing in the same score », ICMC-NYCEMF 2019 (International Computer Music Conference joined with the New York City Electroacoustic Music Festival), 16-23 juin 2019, New-York. **ICMA Audience Award for Best Paper Presentation**.

I. Notation de l'électronique et de l'interaction : vers la partition centralisée

La première partie de ce chapitre est dédiée à une remise en perspective d'une problématique souvent soulevée quand on évoque la question de la notation de l'électronique : cette dernière serait par essence procédurale et impliquerait de spécifier tous les paramètres de la fabrication du son, ce qui s'opposerait à la déclarativité et à l'abstraction auxquelles est parvenue la notation occidentale en se focalisant sur les seules hauteur, durée et intensité des sons à produire. Cette problématique est souvent perçue comme un problème, mais ce n'est pas mon ressenti dans ma pratique de compositeur. Le caractère procédural me semble incontournable quand le compositeur veut explorer une matière sonore nouvelle et doit donc nécessairement préciser comment fabriquer ces nouveaux sons, que ce soit à travers des modes de jeu sur les instruments acoustiques ou bien à travers la synthèse audionumérique. Plusieurs éléments tirés de l'histoire de la musique montrent que cette dimension procédurale a toujours été présente.

Mon travail sur mes propres compositions, mais aussi mon travail de RIM (réalisateur en informatique musicale) pour d'autres compositeurs, m'ont convaincu que cette nature procédurale de la définition du son à produire « reste neutre » par rapport aux problématiques de formes, de polyphonie et d'orchestration des différentes sources sonores (instruments, voix, synthèses, effets), à la condition de pouvoir intégrer au sein de la même partition l'écriture de toutes les parties musicales et de leurs relations, y compris pour les parties électroniques.

Ce besoin d'intégrer différentes parties musicales et d'exprimer leurs relations amène à la notion de *partition électronique centralisée*.

Nous évoquerons quelques outils utilisés pour la réalisation de l'électronique mais nous reviendrons plus en détail sur ceux-ci dans le chapitre suivant. Dans le panorama esquissé, nous pointons l'apport du système Antescofo qui a renouvelé la notion de suivi de partition en couplant une machine d'écoute à un langage de programmation. L'intégration de la fabrication du son électronique (qui doit être programmé) à la spécification de la partie instrumentale (qui doit être écoutée par la machine) concrétise cette notion de partition centralisée.

Organisation du chapitre

Dans ce chapitre, je présenterai quelques éléments historiques sur la notation musicale (section 1), non pas pour en faire une histoire raisonnée, mais pour montrer que le souci procédural (*i.e.* comment produire le son) a toujours été présent aux côtés du souci déclaratif (*i.e.* quels sons produire).

Cette problématique est devenue prégnante avec le développement de l'électronique et je présenterai quelques éléments marquants (pièces, compositeurs, outils) qui m'ont particulièrement influencé et qui ont structuré mon questionnement (section 2).

Cette analyse m'amène à défendre l'idée que la notion de programme informatique est une notion recevable quand il s'agit de noter l'électronique et que les langages de programmation sont de facto utilisés pour la spécification de la synthèse et la transformation du son (section 3). La programmation visuelle a souvent été défendue comme facilitant cette notation, que ce soit à travers la programmation à flot de données (*dataflow*) ou bien via la représentation graphique de *timeline* dans les séquenceurs et les DAW (*digital audio workstation*). Nous verrons que même si le caractère visuel de ces formalismes les rend plus accessibles, ils présentent des limitations propres. Les outils correspondants et leurs usages spécifiques seront analysés plus précisément du point de vue de ma pratique personnelle dans le chapitre suivant.

Le suivi de partition, apparu au milieu des année 1980, marque un tournant dans cette histoire et montre avec le système Antescofo (2008) qu'il est possible de joindre utilement partition instrumentale et partition électronique (section 4) et préfigure la notion de partition centralisée.

La dernière section élabore cette notion de partition centralisée, et en particulier le problème de la préservation (section 5). En effet, une notation doit transmettre aux musiciens la musique pour être jouée. Le caractère procédural de la notation de l'électronique, lié à un matériel et à des systèmes logiciels spécifiques, doit faire face à un risque certain d'obsolescence, bien plus marqué que pour les partitions instrumentales. Cette obsolescence s'oppose à l'objectif de transmissibilité associé à la partition. Ce problème excède toutefois largement le cadre de cette thèse.

I.1. Notation procédurale et notation déclarative

La musique, par sa nature, est un art du temps éphémère qui s'apprécie dans l'instant. Mais le besoin de représenter la musique par des symboles comme moyen mnémotechnique et de transmission au-delà de l'instant présent s'est développé dans la plupart des civilisations qui ont développés l'écriture. Les trente-six Chants hourrites, une tablette en argile (qui se trouve au musée du Louvre) trouvée dans la ville syrienne de Ugarit [Nel15], est la plus ancienne notation musicale qu'on connaît. Elle remonte à 1400 av. J.-C.

Cette musique écrite en notation cunéiforme, un hymne au dieu Nikal, est une transcription qui représente une voix et son accompagnement instrumental (lyre ou harpe) ainsi qu'une table contenant l'information de comment accorder l'instrument. Il y a aussi des instructions de comment la musique doit être lue, ce qui n'est pas sans rappeler la notion de programme et de données sur un même support, une pratique que nous pouvons retrouver 3 000 ans plus tard dans les langages de programmation utilisés en informatique musicale [Mag19].

D'une certaine façon, cet exemple montre que le besoin de représenter tous les éléments musicaux sur un même support a toujours existé et constitue un ancêtre de la notion de partition électronique centralisée.

Les éléments musicaux notés relèvent de deux catégories : ceux qui portent sur comment faire le son et ceux qui prescrivent le résultat sonore à entendre. Cette distinction, présente depuis le tout début de la notation, oppose une « notation procédurale » à une « notation déclarative », ce qu'il faut faire versus ce qu'il faut entendre.

Cette distinction entre une notation qui serait de nature déclarative et qui s'opposerait à un style procédural, est une distinction usuelle dans les langages de programmation : un langage déclaratif vise à spécifier ce que le programme doit accomplir en termes de résultat (*i.e.* le *quoi calculer*) plutôt que de décrire quelle séquence d'actions élémentaires il faut précisément accomplir pour arriver à ce résultat (*i.e.* le *comment calculer*). La programmation fonctionnelle, la programmation logique ou encore la programmation par contrainte sont des exemples de modèles de programmation qui sont des formes de programmation déclarative. Ces deux notions ne sont cependant pas étanches : les langages logiques peuvent inclurent des mécanismes procéduraux et les langages impératifs, des mécanismes fonctionnels.

Si l'on regarde une partition comme planification des événements musicaux à réaliser pour concrétiser une musique, alors on peut s'apercevoir que la notion de partition et la notion de programme partagent un certain nombre d'objectifs et de propriétés[1] et l'on comprend mieux que les qualificatifs de déclaratif et de procédural s'appliquent aussi bien aux partitions qu'aux programmes.

La notation occidentale est largement déclarative : une partition de piano décrit les notes à entendre, pas les touches sur lesquelles il faut appuyer[2]. Cette nature déclarative est en partie due à la fixation de l'instrumentarium qui permet de laisser implicite les aspects les plus opérationnels de la production musicale. Ces aspects opérationnels font alors partie de la culture implicite et sont partagés par tous. Elle a aussi été rendue possible en concentrant la description du résultat à produire sur des paramètres relativement faciles à standardiser une fois qu'on accepte de les discrétiser : la hauteur, la durée et l'intensité des sons à produire. Une notion comme le timbre est bien plus difficile à traiter de manière abstraite et bien souvent, la meilleure description existante est celle des conditions de sa production.

Cette évolution vers une notation abstraite, symbolique, déclarative est remise en cause avec l'invention des premiers instruments mécaniques au XVII^{e} siècle, période qui coïncide aussi avec l'apparition des premières calculatrices mécaniques[3]. Le contrôle de ces instruments se faisait souvent par des cartons perforés pour activer/désactiver

[1] Nous reviendrons sur ce point plus loin dans ce chapitre.
[2] Comme indiqué précédemment, ces catégories ne sont pas étanches et on peut trouver des informations opérationnelles dans une partition comme l'indication d'un doigté.
[3] https://fr.wikipedia.org/wiki/Calculatrice_mécanique

des tuyaux d'orgue et jouer des mélodies. Ces cartons jouent à la fois le rôle de programme et de partition : on peut dire qu'ils constituent une première version des partitions électroniques qui se généraliseront ultérieurement avec les ordinateurs.

Au XX^{e} siècle, avec l'avènement de l'électricité, c'est un changement radical qui s'opère au niveau de la lutherie instrumentale avec l'apparition des premiers instruments électroniques analogiques comme le thérémine, le telharmonium ou les Ondes Martenot, ce dernier étant utilisé par plusieurs compositeurs au XX^{e} siècle. Si ces instruments électriques n'appellent pas nécessairement un changement de nature dans la notation musicale, l'irruption de l'ordinateur bouscule le paysage et, dès les toutes premières expérimentations musicales avec l'ordinateur, les notions de partition et de programme se brouillent (citons le premier enregistrement de musique générée par un ordinateur d'Alan Turing[4] et la première musique générée algorithmiquement, *Illiac Suite* de Lejaren Hiller et Leonard Issacson).

I.2. Sur l'écriture musicale symbolique

Des notions linguistiques peuvent éclairer les nouveaux besoins de notation de l'électronique [Lev13]. Cet angle de réflexion a été développé par le compositeur Fabien Lévi, par exemple, dont je reprends ci-dessous plusieurs développements.

On attribue en général deux fonctions aux écritures, les fonctions *graphémologique* et *grammatologique*. La première considère que l'écriture représente, à travers des signes et signaux, la transcription de ce qu'elle exprime, comme l'écriture musicale ou l'enregistrement car ils renvoient par la lecture de leur transcription à l'action de jouer, à une séquence musicale au même titre que l'écriture linguistique renvoie à une langue orale. Alors que la fonction *grammatologique* a un rapport avec le processus matériel : l'écriture, les signes sont des substances autonomes qui interrogent la pensée et le matériau de ce qu'ils expriment [Lev13].

La notion de *grammatologie* [Der97] explique comment les humains étendent leur mémoire et leurs techniques corporelles au moyen de « suppléments » externes et comment ces inscriptions ne sont pas une représentation ou une reproduction neutre et fidèle mais affectent fortement la nature et la pensée de la chose transcrite. Bernard Stiegler développe la notion de *grammatologie* et introduit ce qu'il appelle « *grammatisation* » [Sti10] comme « *le processus par lequel le continuum temporel des comportements humains est transformé en un spatial discret, qui permet de les intégrer dans les outils* ».

La notation musicale est pour moi un bel exemple de ce processus de *grammatisation* symbolique qui transforme notre façon de penser la musique. Et en expérimentant de nouvelles procédures sur le signe, on ouvre de nouvelles perspectives musicales. On peut dire que la notation est une technologie qui permet de dépasser une notation qui ne serait qu'analogique (du son à produire), pour développer un système (plus ou moins

[4] https://www.theguardian.com/science/2016/sep/26/first-recording-computer-generated-music-created-alan-turing-restored-enigma-code

autonome) avec une logique propre, de nature linguistique et indépendante du son. De nombreuses opérations illustrent cette idée, comme la polyphonie et la permutation avec des inversions, des rétrogradations, des canons, etc., des compositeurs de l'Ars Nova[5] au sérialisme intégral, en passant par d'autres opérations comme la modification de la vitesse, le filtrage, la lecture à l'envers ou le micro montage dans les musiques électroacoustiques. Ce sont des opérations qui ont un sens dans le monde de l'écriture et qui trouvent ensuite un sens dans le monde sonore. La notation crée un espace de pensée qui se distingue de la chose pensée, ce qui permet l'abstraction.

La notation musicale symbolique (qui correspond ici à une « notation déclarative ») a permis d'expérimenter de nouvelles formes avec différents styles et techniques musicaux jusqu'à nos jours. Elle est aussi devenue de plus en plus complexe et de plus en plus précise. Dans la musique baroque, par exemple, la partition décrivait une sorte de squelette musical que les instrumentistes devaient remplir avec des ornements. Pendant cette période baroque, la distance entre la chose écrite par le compositeur et celle jouée par l'interprète était plus grande que dans des périodes postérieures où cette distance devint nulle grâce à une notation de plus en plus précise, ne laissant que l'aspect interprétatif aux instrumentistes.

Cette quête de précision aboutit paradoxalement à des pièces où l'écriture est d'une telle complexité que cette précision rythmique se perd à l'interprétation pour laisser place à une improvisation dirigée, par exemple dans l'école de la *nouvelle complexité* (*new complexity*, terme assez vague puisqu'on ne sait pas s'il désigne la complexité de la pensée compositionnelle, celle de la notation, celle de l'interprétation ou celle de la perception [Lev13]). Dans ce courant, l'extrême complexité rythmique de l'écriture instrumentale rend la partition imprécise dans son interprétation du fait que les rythmes exprimés sont impossibles à réaliser par un être humain. Cette imprécision est voulue par les compositeurs et permet de faire émerger de l'interprétation des sonorités qui ne sont finalement pas perceptibles dans la densité de l'écriture. À propos de *Unity Capsule* pour flûte solo de Brian Ferneyhough, le compositeur Fabien Levy écrit :

« *La difficulté de la notation-interprétation lui permet de créer un certain nombre de modes de jeux jugés normalement "indésirables" (sons impurs, soufflés, différentes embouchures, multiphoniques) et d'atteindre, selon sa volonté, une diversité de couleurs proche des phonèmes de la voix. L'interprète est dans un état continuel de "surprise d'exécution".* » [Lev13]

Mais pour le compositeur américain Curtis Roads :

« La densité de ces partitions était cependant le signe d'un affaiblissement du langage musical. Il fallait de plus en plus du symbolisme pour exprimer de moins en moins de l'audible. » [Roa15]

Parallèlement à cette complexification de la notation, on peut constater une autonomisation croissante de l'écriture par rapport à la chose dénotée. On est passé par

[5] Par exemple la célèbre composition de Guillaume de Machaut, *Ma fin est mon commencement*.

plusieurs configuration harmoniques (« modales », « tonales ») pour arriver à une égalité entre les hauteurs dans le sérialisme et à une autonomie généralisée avec le sérialisme intégral. Tout intervalle étant égal à tout autre, et toute valeur de paramètre à une autre, le côté référentiel de la notation s'abolit pour se focaliser sur le jeu de combinatoire formelle propre à la notation pure.

À côté de ces enjeux et de ces développements de la notation symbolique et abstraite, des recherches sur la « notation procédurale » se développent en relation directe avec l'idée d'écriture du son. Par exemple, le compositeur Helmut Lachenmann dans sa « musique concrète instrumentale » essaye de décrire comment jouer les instruments pour produire un son avec une énergie et un timbre déterminés plutôt que de décrire des paramètres musicaux traditionnels précis (hauteurs, rythmes, nuances, etc.) :

« *Cela signifie une musique dans laquelle les événements sonores sont choisis et organisés de telle sorte que la manière dont ils sont générés est au moins aussi importante que les qualités acoustiques qui en résultent d'elles-mêmes. Par conséquent, ces qualités, telles que le timbre, le volume, etc., ne produisent pas de sons pour elles-mêmes, mais décrivent ou dénotent la situation concrète : en écoutant, vous entendez les conditions dans lesquelles un son ou action bruitée est effectuée, vous entendez quels matériaux et énergies sont impliqués et quelle résistance est rencontrée.*[6] »

Dans la notation de Lachenmann, il y a tout de même une grande composante de tradition orale, puisque les signes qu'il utilise dans ses partitions pour générer des sonorités ne représentent pas le résultat sonore désiré, il faut donc décrire à l'instrumentiste comment jouer, oralement ou à travers un enregistrement audio/vidéo. C'est une problématique qu'on retrouve aussi dans les représentations graphiques de l'électronique, surtout quand on travaille avec des sons et des superpositions complexes : on ne peut pas décrire très précisément le résultat sonore avec une représentation graphique symbolique, on peut seulement décrire ses contours, une idée qualitative de son amplitude, etc.

Il faut noter que d'autres expérimentations ont vu le jour au XX^{e} siècle, comme les notations graphiques, ou programmatiques[7]. Ces notations expérimentales qui ont proliféré entre les années 1950 et 1970 ont commencé par la suite à être de moins en moins utilisées parce que difficiles à étudier, chacune présentant des caractéristiques différentes pour chaque pièce, et les instrumentistes n'étant pas entraînés à lire ces partitions graphiques comme pour la notation symbolique sur une portée. Selon le compositeur Luciano Berio :

---

[6] Slought Foundation Online Content (7 April 2008), musique concrète instrumentale: Helmut Lachenmann, in conversation with Gene Coleman, https://slought.org/resources/musique_concrete_instrumentale (retrieved 30 March 2021).

[7] Cette notation programmatique spécifie une série d'instructions à suivre pour le déroulé de la pièce, par exemple pour certaines pièces de John Cage, la Mount Young, etc.

« An ultimate sign of the gap between thought and acoustic end-result came about when the musical score became an aesthetic object to be admired only visually – the eye becoming the substitute for the ear… It may evoke the "beauty" of Bach's manuscripts or the "uglines" of Beethoven's sketches; but that "beauty" and "ugliness" have nothing to do with musical processes and functions. » [Ber06]

I.3. L'émergence d'un espace de pensée acousmatique

Si l'écriture symbolique traditionnelle a permis le développement d'un espace de pensée musicale propre, elle est prisonnière des conditions de son avènement : les hauteurs ont été discrétisées et les rythmes simplifiés par un principe de division qui ne recouvre que le domaine atteignable par des instrumentistes humains et des instruments acoustiques qui ont été fabriqués pour un genre de musique en demi-ton. L'écriture symbolique, par sa traduction en signes abstraits, transforme et simplifie l'information musicale. Avec l'avènement des instruments électroniques, ces limites seront surpassées et il faudra recourir à des systèmes d'écriture numérique.

La musique électroacoustique est sans doute l'une des plus importantes révolutions du XX^{e} siècle dans le domaine de la musique d'exploration. Qu'elle provienne des premières expérimentations de la musique concrète de Pierre Schaeffer ou des expériences avec des synthétiseurs dans le studio de Cologne dans les années 1950, elle a pris un chemin complètement différent des notions d'écriture instrumentale traditionnelle pour créer son univers, son solfège, son langage et des écritures propres fondées sur des techniques spécifiques. Ces techniques, sonorités et morphologies, diffusées par des haut-parleurs et fondées sur le paradigme du son et du déroulement énergétique, se sont installées dans la composition contemporaine à partir des années 1950, et ont même été une inspiration pour beaucoup de compositeurs issus du monde instrumental. György Ligeti n'aurait sans doute pas écrit les œuvres emblématiques *Atmosphère* et *Lontano* sans être passé par l'expérience de la composition électroacoustique :

« *Une chose est sûre : la pratique de la composition électronique nous a enrichis ; de nouvelles catégories musicales en sont nées et de nouveaux mondes de l'imagination sonore et formelle ont été explorés.* » [Lig14]

La musique acousmatique[8] est généralement conçue dans un contexte plus souple au niveau temporel et du son que la musique instrumentale du fait qu'on ne se base pas sur une grille temporelle et de hauteurs prédéterminée. La musique acousmatique présente des éléments d'écriture basés sur des techniques opérationnelles de manipulation et de synthèse du son. Ces opérations de composition, construction et manipulation sonores reposaient au début sur des techniques difficiles à mettre en

[8] Terme repris par Pierre Schaeffer dans son *Traité des objets musicaux* à Jérôme Peignot qui le cite dans un article de critique du premier concert de musique concrète à Paris en 1948. Dans le dictionnaire de Littré, au XIX^{e} siècle, une acousmie est un bruit dont on ignore la cause [Van17].

œuvre, comme découper et coller des bandes magnétiques ou piloter des générateurs sinusoïdaux à la main. La situation a considérablement changé avec l'avènement de l'informatique, et notamment les logiciels de montage (comme ProTools[9]) et les techniques d'enregistrement numérique qui facilitent grandement la création, la manipulation et le montage des sons :

« Le montage est l'essence même de l'écriture sur support (celle des "sons fixés" selon la terminologie de Michel Chion[10]), qui permet les possibilités musicales offertes par l'acte, très déterminé, de couper, coller, interrompre, insérer, changer de plan (au sens cinématographique), faire apparaître ou disparaître un événement sans avertissement et comme par enchantement, celui de nos arrières-grand-mères devant les films de Méliès, créer des êtres sonores chimériques. Il s'agit donc, depuis que la musique électroacoustique existe grâce à la bande magnétique, de "couper" résolument dans un déroulement temporel. » [Van17]

La musique acousmatique et sa façon d'écrire et composer, ainsi que sa perception du son comme une morphologie forme/matière [Sch66] sont un apport fondamental au paradigme du son dans la composition contemporaine :

« *L'écriture et la perception morphologiques sont sans doute une des révolutions musicales du XX^e siècle. Le fait d'écrire directement sur support, de pouvoir réécouter à satiété (le « rimage » de François Bayle [Bay93]) des sons très courts comme des séquences entières, fait prendre conscience qu'un son ne s'entend pas seulement comme fréquence, durée, spectre harmonique, mais aussi comme déroulement temporel d'une enveloppe d'amplitude : attaque, entretien, chute (forme), avec sa vie interne et ses caractères de grains (matière). »* [Van17]

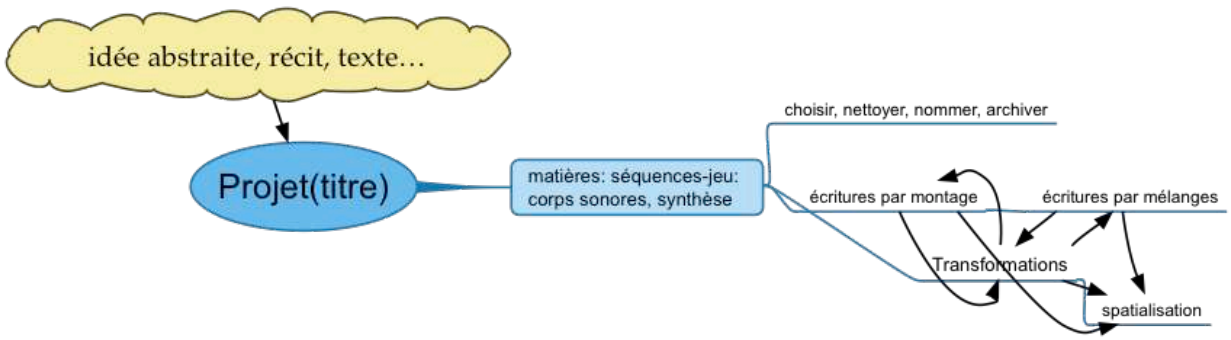


Fig. 1.1 Schéma proposé par la compositrice Annette Vande Gorne pour évoquer les différentes étapes de travail en studio, incluant des étapes d'écriture par montage et mélange [Van17].

L'écriture « analogique » des sons fixés est aussi susceptible d'offrir une logique propre reliée à la logique du support comme la lecture à l'envers, la substitution d'attaque, le micro montage, l'interpolation, l'incrustation, les objets composites, les

[9] https://fr.wikipedia.org/wiki/Pro_Tools
[10] Chion M., 1991, L'art des sons fixés, Fontaine : Metamkine, Nota-bene, Sono-Concept, p. 101.

jeux sur le spectre, les jeux sur le temps et, particulièrement important dans ma démarche, les jeux sur l'espace [Van17]. Ces procédés, par un processus similaire à celui à l'œuvre dans la notation symbolique, deviennent des éléments formels de l'espace de pensée du compositeur au-delà de la manipulation de la matière sonore auxquels ils réfèrent.

On peut observer que l'écriture dans les pièces acousmatiques est souvent (mais pas toujours) plus centrée sur la composition du son lui-même que sur l'écriture rythmique, contrepointiste et par processus à plus grande échelle. Elle a aussi la particularité d'être composée en temps différé et fixée sur un support, ce qui enlève un caractère jusqu'à présent intrinsèque à toutes les musiques : l'interprétation. À ce propos, Pierre Schaeffer écrivait :

« Étais-je ou non à un poste de commandement ? Devait-on régler une fois pour toutes le volume des haut-parleurs, ou fallait-il, selon une vague intuition, répondre par une présence à la présence du public, ne pas le laisser seul en face des tourne-disques, et ajouter une marge d'exécution, si minime fût-elle, à la reproduction automatique de l'enregistrement ? C'est après coup que je me rendis compte de mon audace légitime. Il fallait en effet être présent, et, si peu que ce fût, (apparemment), interpréter. » [Sch52]

Ce manque d'interprétation et de performeurs dans les musiques acousmatiques est surmonté par la pratique de l'interprétation des pièces acousmatiques qui est devenue une discipline à part entière et qui donne un relief supplémentaire à la musique du fait de la projeter dans un orchestre de haut-parleurs (acousmonium) et de jouer en même temps avec le lieu de diffusion ainsi qu'avec le nombre, la nature et la disposition des haut-parleurs. Cette démarche est aussi le résultat d'un fait technique et historique : les premières pièces acousmatiques étaient en mono ou stéréo. C'est seulement à partir des années 1990 que les acousmaticiens ont eu largement accès à des enregistreurs abordables 8 pistes (ADAT ou DA88). À partir de ce moment, on voit deux directions différentes dans l'utilisation de l'espace dans la musique acousmatique : d'un côté, il y a des pièces en format stéréo qui seront interprétées et diffusées dans un acousmonium, et de l'autre des compositions où la spatialisation des sons dans le discours musical est intégrée à l'acte de composition lui-même (je reviendrai sur cette thématique à propos de la *synthèse spatiale*). Par exemple, aujourd'hui, grâce à de nouvelles techniques comme l'ambisonie [Ger73] ou le VBAP [Pul97] (entre autres), il est possible de reproduire et restituer une information spatiale même très élaborée vers différents types de configuration de haut-parleurs.

La musique acousmatique ouvre un nouveau monde de possibilités de compositions et d'articulations sonores qu'il n'était pas possible d'imaginer dans le monde instrumental. Ma pratique de la composition dans ce domaine m'a permis de dégager quelques réflexions, en particulier sur :

- l'orchestration de l'électronique ;
- l'adaptation au lieu de diffusion permise par le temps réel ;

- et la spatialisation non seulement comme un paramètre supplémentaire mais comme un constituant de la morphologie du son et de la forme.

Ces notions seront abordées dans la seconde partie de cette thèse, lors de la présentation des pièces produites au cours de ce travail.

I.4. La partition électronique

« La notation musicale traditionnelle n'a pas été conçue pour décrire la musique dans laquelle le timbre, les textures statistiques, et les qualités spatiales sont l'objectif principal. » [Roa15]

« Une partition de musique instrumentale symbolise les sons par des notes : la notation permet de les évoquer, et connaissant les instruments et les modes de jeu – par exemple un sol grave de violon pizzicato – on peut se donner une idée assez précise de l'effet sonore et recourir à l'écoute intérieure. Mais une telle description n'est pas d'une grande aide pour spécifier un son désiré à un programme d'ordinateur qui doit le calculer. L'ordinateur respecte rigoureusement les ordres des données et réalise le son sur plan exactement comme on lui a demandé : il faut tout lui dire, lui prescrire tous les détails de facture. » [Ris80]

Avec l'avènement de l'informatique musicale, de nouveaux types de partitions ont vu le jour, il s'agit des « partitions électroniques » ou partitions informatiques. Ces partitions, par analogie avec les partitions instrumentales, auront comme but d'écrire les éléments constituants de la partie électronique pour l'enregistrer et la reproduire ultérieurement en temps différé pour la première génération des logiciels de création musicale, et en temps réel à partir de la génération suivante.

La notation de l'électronique est par essence procédurale[11] car, à la différence de la notation instrumentale, les instruments électroniques ne sont pas « stabilisés ». L'ordinateur est « universel » (il peut faire n'importe quel calcul) et il n'est donc pas comparable à un instrument. Mais il peut les émuler tous. Dans le monde des partitions informatiques, la séparation historique entre instruments physiques acoustiques et partitions symboliques sur papier se brouille donc pour donner lieu à un nouveau type de partition où tous les éléments sont interdépendants et existent en tant que programmes avant de générer du son. L'instrument s'adapte à la partition et inversement du fait que la morphologie sonore et musicale est un mix entre le timbre et son déroulement temporel. L'instrument peut se métamorphoser continuellement

[11] Il faut cependant prendre garde à une confusion possible : si l'électronique est notée dans un style procédural, par exemple en spécifiant précisément l'évolution des paramètres d'une synthèse au cours du temps, le langage informatique qui permet de définir cette évolution peut relever du paradigme déclaratif. La notation résultante sera plus concise et plus abstraite que dans un langage impératif, mais ne remet pas en cause la nature opérationnelle de la spécification de l'électronique.

dans le temps et ces changements d'états continus sont décrits dans la partition et, en même temps, la partition est l'instrument qui se déploie dans le temps.

Le programme et les langages de programmation deviennent le moyen de définir les instructions musicales à réaliser. De manière similaire à la partition instrumentale qui instruit le musicien de ce qu'il faut faire, le programme instruit l'ordinateur des calculs à effectuer. Il y a donc une certaine analogie entre la notion de partition instrumentale et la notion de programme. On pourrait dire que le programme est une partition, la « partition électronique »[12] :

Partition	**Programme**
Réalisée par le compositeur	Réalisé par le compositeur ou un RIM
Interprétée par un humain	Interprété par une machine
Instruments acoustiques	Instruments numériques, synthétiseurs
Écriture en notation symbolique standardisée	Écriture dans un langage de programmation ad hoc décrivant les éléments de l'électronique, des programmes (fonction, processus, objets) pour générer de la musique électronique et des actions (événements discrets, contrôles continus, etc.)
Interprétation de la partition	Exécution du programme
Partition d'orchestre	Programme parallèle (partition centralisée)
Notation déclarative (à l'exception de la notation procédurale, comme dans l'écriture de modes de jeu chez Lachenmann)	Notation déclarative et procédurale (Antescofo)
Partition écrite horizontalement (en parallèle)	Code écrit séquentiellement

[12] Cependant, malgré ces convergences, le statut d'une partition et celui d'un programme diffèrent. « *Si partition et programme sont tous deux écrits par un humain pour être ensuite interprétés, la partition musicale est interprétée par des humains et le programme par des machines. En conséquence, les relations attendues entre la musique écrite et ses interprétations divergent de celles voulues entre un programme et son exécution. Guerino Mazzola parle de la partition comme d'un geste surgelé qu'il faut dégeler lors de l'interprétation. Mais ce réchauffement ne peut se comparer à l'exécution d'un programme informatique : c'est un processus ouvert qui demande une création de l'interprète qui échappe aux spécifications de la partition afin, comme l'exprime Theodor Adorno, de passer d'un état solide à l'état liquide de la musique*. » Jean-Louis Giavitto, « Du temps écrit au temps produit en informatique musicale » in *Produire le temps*, Hermann 2014.

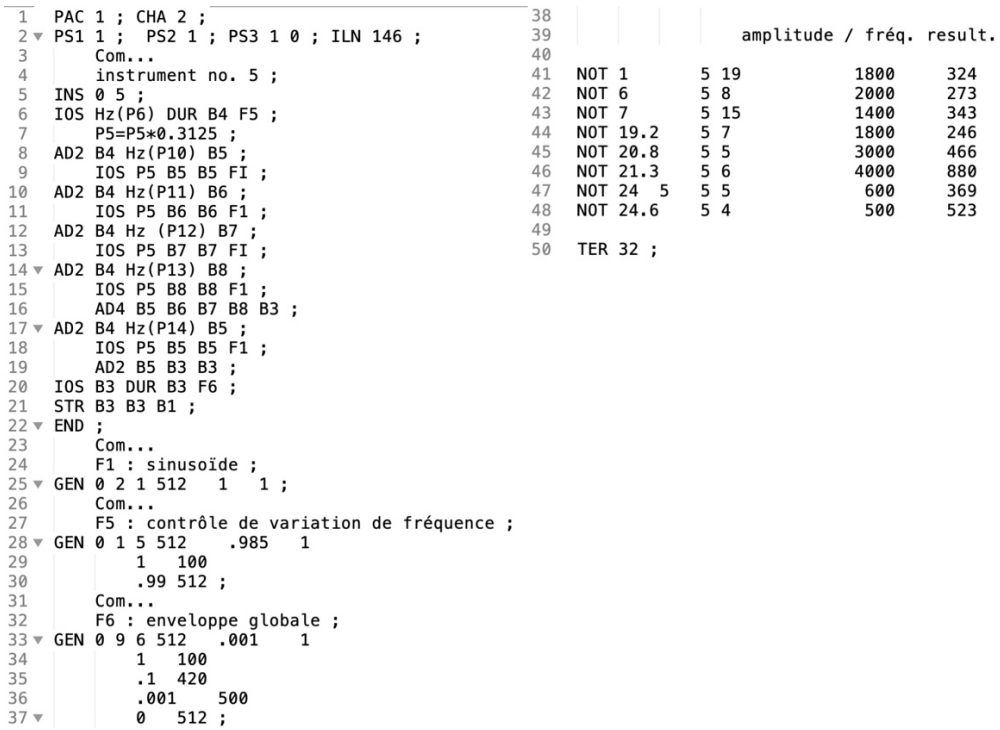
Fig. 1.2 Tableau comparatif entre une partition et un programme.

Le paradigme orchestre/score

Historiquement, le premier son synthétique réalisé par ordinateur dans le programme Music I développé par Max Mathews en 1957 aux Bell Labs[13], est tout d'abord un programme informatique. Dès l'apparition de ces premiers sons numériques, le mariage entre partition et programme informatique donne naissance à un nouveau concept où un programme *dans un langage dédié* sous forme de description de l'instrument et son déploiement temporel est en même temps sa propre partition électronique. C'est aussi le concept qu'a adopté Max Mathews dans la famille des logiciels de synthèse MUSIC-N avec le développement de la notion de *Unit Generators*[14] pour avoir un système de synthèse le plus flexible possible pour la création d'instruments numériques. Ces familles de langages sont déclaratives : l'utilisateur va définir différents événements musicaux dans une partition ou « score » et puis sera le compilateur qui organisera et mettra en œuvre les calculs nécessaires.

[13] https://fr.wikipedia.org/wiki/Laboratoires_Bell

[14] Les *Unit Generators* (UGen) sont des blocs de construction atomiques, souvent prédéfinis, pour générer ou traiter des signaux audio. En plus des entrée/sortie audio, une UGen prend en charge un certain nombre d'entrées de commande qui contrôlent les paramètres associés à l'UGen.



```
PAC 1 ; CHA 2 ;
PS1 1 ;  PS2 1 ; PS3 1 0 ; ILN 146 ;
    Com...
    instrument no. 5 ;
INS 0 5 ;
IOS Hz(P6) DUR B4 F5 ;
    P5=P5*0.3125 ;
AD2 B4 Hz(P10) B5 ;
    IOS P5 B5 B5 FI ;
AD2 B4 Hz(P11) B6 ;
    IOS P5 B6 B6 F1 ;
AD2 B4 Hz (P12) B7 ;
    IOS P5 B7 B7 FI ;
AD2 B4 Hz(P13) B8 ;
    IOS P5 B8 B8 F1 ;
    AD4 B5 B6 B7 B8 B3 ;
AD2 B4 Hz(P14) B5 ;
    IOS P5 B5 B5 F1 ;
    AD2 B5 B3 B3 ;
IOS B3 DUR B3 F6 ;
STR B3 B3 B1 ;
END ;
    Com...
    F1 : sinusoïde ;
GEN 0 2 1 512   1   1 ;
    Com...
    F5 : contrôle de variation de fréquence ;
GEN 0 1 5 512    .985   1
        1   100
        .99 512 ;
    Com...
    F6 : enveloppe globale ;
GEN 0 9 6 512   .001    1
        1   100
        .1  420
        .001    500
        0   512 ;
```

```
                amplitude / fréq. result.

NOT 1       5 19          1800     324
NOT 6       5 8           2000     273
NOT 7       5 15          1400     343
NOT 19.2    5 7           1800     246
NOT 20.8    5 5           3000     466
NOT 21.3    5 6           4000     880
NOT 24  5   5 5            600     369
NOT 24.6    5 4            500     523

TER 32 ;
```

Fig. 1.3 Extrait d'une partition en MUSIC V : à gauche, la description de l'instrument ; à droite, les notes « NOT » avec leur date de départ pour jouer l'instrument (numéro 5). Extrait de l'analyse de la bande magnétique de l'œuvre de Jean-Claude Risset, *Inharmonique*, par Denis Lorrain[15].

En 1968, le programme MUSIC V est le premier système de programmation de musique à être implémenté dans un langage de haut niveau, le FORTRAN[16]. Cela signifiait que MUSIC V pouvait être porté sur n'importe quel système informatique exécutant FORTRAN, ce qui a grandement contribué à la fois à son utilisation généralisée dans la communauté d'informatique musicale et à son développement ultérieur. MUSIC V est le dernier et le plus mature des langages de synthèse de Max Mathews aux Bell Labs et il est probablement le langage de musique informatique le plus influent. Les descendants directs incluent MUSIC 360 (pour l'IBM 360) et MUSIC 11 (pour le PDP-11) de Barry Vercoe et ses collègues du MIT. On peut citer aussi Cmusic de F. Richard Moore qui est plus tardif.

Ces systèmes ajoutaient beaucoup de flexibilité syntaxique et logique, mais au fond restaient fidèles aux principes des langages MUSIC-N : la connexion de Unit Generators et le traitement séparé de la synthèse sonore .orc (orchestre) et de l'organisation musicale .sco (partition). Selon J.-C. Risset :

« *Une partition MUSIC V donne comme une partition habituelle la disposition des sons, mais de plus elle spécifie intégralement la structure des sons "composés" comme des accords.* » [Ris80]

[15] http://articles.ircam.fr/textes/Lorrain80a/
[16] https://fortran-lang.org

MUSIC V reste un modèle pour de nombreux autres langages et environnements de programmation pour la musique jusqu'à nos jours. Ce paradigme orchestre/score ou tout simplement partition selon ses utilisateurs [Ris80], a aussi été adopté postérieurement, par exemple dans des logiciels tels que Csound, Cmusic, Cmix ou CLM [17]. À l'époque des MUSIC-N, ce système d'écriture servait à donner les instructions à l'ordinateur qui après un temps de calcul « *off line* » donnait le résultat de ce rapport entre instruments numériques et partition pour le déploiement temporel des instruments et leurs paramètres.

```
1  (definstrument simp (start_time duration frequency amplitude)
2    (let* ((beg (floor (* start_time *srate*)))
3           (end (+ beg (floor (* duration *srate*))))
4           (j 0))
5      (run
6        (loop for i from beg below end do
7          (outa i (* amplitude (sin (* j 2.0 pi (/ frequency *srate*)))))
8        (incf j)))))
9
10
11 (with-sound () (simp 0 0.25 440.0 0.2))
```

Fig. 1.4 Exemple d'un instrument dans l'environnement de programmation Common Lisp Music (CLM). On définit un instrument *simp* entre la ligne 1-8 avec une fonction sin pour générer un oscillateur pour être joué à la ligne 11 *(with-sound)* avec ses paramètres start_time, duration, frequency et amplitude.

[17] https://ccrma.stanford.edu/software/snd/snd/clm.html

```
<CsoundSynthesizer>
<CsOptions>
; Select audio/midi flags here according to platform
-odac    ;;;realtime audio out
;-iadc    ;;;uncomment -iadc if realtime audio input is needed too
; For Non-realtime ouput leave only the line below:
; -o oscils.wav -W ;;; for file output any platform
</CsOptions>
<CsInstruments>

sr = 44100
ksmps = 32
nchnls = 2
0dbfs  = 1

instr 1

iflg = p4
asig oscils .7, 220, 0, iflg
     outs asig, asig

endin
</CsInstruments>
<CsScore>

i 1 0 2 0
i 1 3 2 2       ;double precision
e
</CsScore>
</CsoundSynthesizer>
```

Fig. 1.5 Exemple d'un instrument simple avec un générateur *oscils* (oscillateur) dans l'environnement de programmation Csound. Comme dans MUSIC V, on définit des paramètres de base comme le taux d'échantillonnage (*sampling rate),* le taux de contrôle (*control rate),* puis l'instrument *instr 1* et à la fin le score pour jouer l'instrument *instr 1* « i 1»[18] avec ses paramètres.

Trois générations de langage

On peut dire que la famille des logiciels MUSIC-N correspond à la première génération des langages pour la synthèse et la composition musicale de l'électronique.

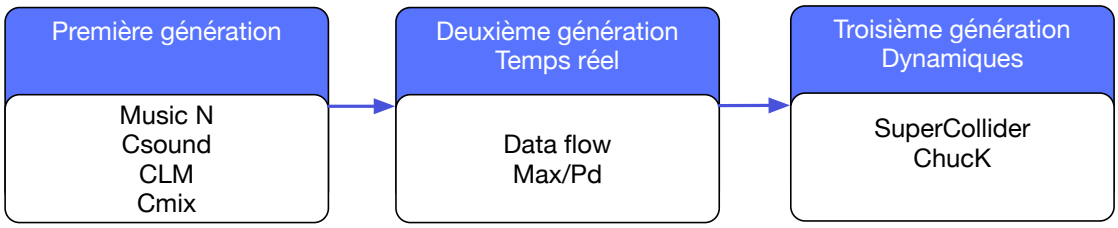


Fig. 1.6 Schéma chronologique de représentation de trois générations de logiciels pour la composition et la synthèse et traitement du son.

Chronologiquement, une deuxième génération de logiciels voit le jour et, à la différence de la famille MUSIC-N qui est un langage textuel, cette nouvelle génération sera basée sur la programmation visuelle (*data flow*). Max et sa librairie audio MSP [Zic98] [Puc02] ainsi que la version open source Pd [Puc96] sont des logiciels de cette deuxième génération et ils partagent avec la première génération l'utilisation et le concept des *Unit Generators*. Les connexions entre les différentes *Unit Generators* se font par des câbles virtuels, un circuit qu'on appelle « *patch* ».

[18] http://www.csounds.com/manual/html/

Dans cette famille de logiciels graphiques *data flow*, il y a une correspondance directe entre le programme graphique et les générateurs de son (*Unit Generators*) sous-jacents. La nouveauté principale de cette famille est qu'ils générèrent le son en temps réel, ce qui présente plusieurs avantages comme le jeu en interaction avec des musiciens pendant l'exécution de la musique ou le fait de ne pas devoir stocker et préenregistrer des fichiers sons et limiter l'utilisation de mémoire. Cette famille de logiciels a été conçue à l'Ircam par Miller Puckette dans un contexte de création et composition musicales, en particulier pour les créations des pièces mixtes de Philippe Manoury, et aussi pour des besoins techniques et pratiques de contrôle. M. Puckette ne voulait pas reproduire l'expérience où pour faire une seule production à l'Ircam, il aurait fallu trois ans et quatre chercheurs :

« "*Si on voulait que la musique numérique temps réel croisse et devienne importante dans le monde, il fallait un système où un compositeur seul pouvait faire une pièce*[19]*", c'était ça le but principal qu'il avait quand il a mis tous ses éléments ensemble dans un Macintosh de façon graphique et utilisable par les non-programmeurs en principe. L'introduction des interfaces graphiques dans cette famille a grandement facilité le contrôle, se rapprochant des instruments acoustiques, on va pouvoir expérimenter et jouer les sons de synthèses à travers le changement des paramètres directement avec la souris de l'ordinateur ou avec différents types d'interfaces de contrôles physiques. »* Pour Philippe Manoury l'interface graphique donnait aussi un accès plus intuitif pour un compositeur que la programmation en texte, il était possible de contrôler le son en temps réel avec des boutons, sliders, etc., ce qui était une grande avancée. *« Il n'y avait plus besoin de formaliser numériquement une structure de timbre, car on pouvait la construire et la faire varier d'une façon analogue à celle d'un musicien qui produit le son par une variation de souffle ou une pression de l'archet.* » [Man97]

Une notion informatique de contrôle commence aussi à émerger avec la possibilité de programmer des interactions et des structures musicales plus complexes, mais la notion d'un contrôle temporel général, fin et dynamique reste toujours difficile à implémenter dans les systèmes *data flow*.

On peut repérer une troisième génération de logiciels pour la synthèse sonore et de composition tels que SuperCollider [MC02] qui sera décrit dans le prochain chapitre, ou ChucK [WC03]. Ils sont aussi conçus pour réaliser des opérations en temps réel, et la distinction entre orchestre et score est moins évidente. Il peut y avoir des cas où l'orchestre et le score ne se différencient pas, par exemple avec les SuperCollider Tweets[20] (code condensé pour générer des petits morceaux de musique électronique).

Cette intégration étroite entre un système de synthèse sonore et un langage plus expressif, généraliste, permet de coupler, d'intégrer et d'expérimenter des idées de

[19] JIM 2020, https://www.youtube.com/watch?v=GKsZ5pDgOmI&t=5039s
[20] https://ccrma.stanford.edu/wiki/SuperCollider_Tweets

synthèse et de composition avec une grande souplesse. De plus, le langage SuperCollider (*sclang*), qui ressemble en partie aux langages de programmation Smalltalk[21] et *C*, est orienté objet et fournit un large éventail de constructions de programmation expressives pour la synthèse sonore et la programmation d'interfaces utilisateur.

Cette intégration entre un système de synthèse puissant et un langage expressif dédié à la musique est à la base de la conception de la librairie pour l'écriture de partitions électroniques centralisées AntesCollider qui sera décrite dans les chapitres suivants.

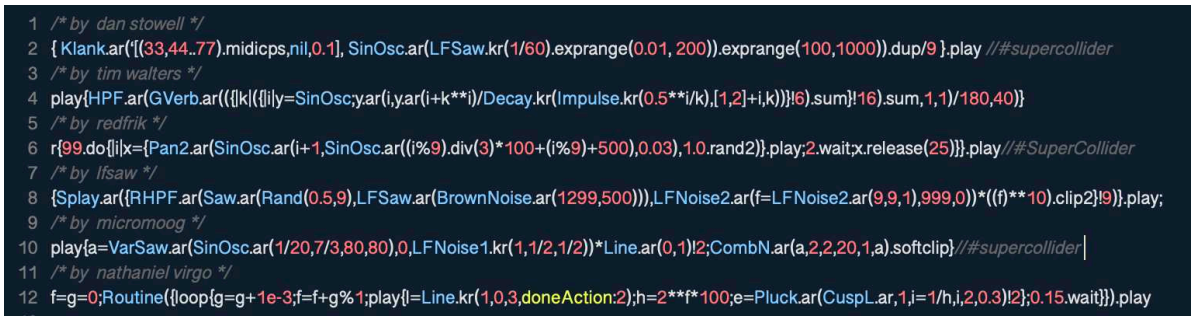


Fig. 1.7 Exemple de SuperCollider Tweets, chaque ligne correspond à une musique ou morphologie sonore pour être partagée sous forme de Tweet : « A popular way to share SuperCollider code is to post it to the social networking site Twitter. This site imposes a restriction on the length of posts, limiting them to 140 characters or less. Creating an interesting sound or piece of music within this constraint has proven to be a popular challenge. »

Un autre exemple d'une intégration entre orchestre et score, dans cette troisième génération, est le langage ChucK pour le *live coding*[22]. La syntaxe de ChucK va permettre de faire des connexions entre les *Unit Generators* ou modules de traitement, ainsi que le changement des paramètres dynamiquement, « à la volée », pendant que le programme s'exécute.

Cette modalité de « programmation à la volée », ou écrire les événements électroniques et les évaluer en temps réel pour créer et contrôler les chaînes de synthèse constitue de toute évidence un renouvellement de la dichotomie historique entre composition et interprétation ainsi que de la modalité de l'écriture de l'électronique. La programmation, l'écriture et l'interprétation de la musique se centrent ici sur l'improvisation avec l'écriture directe à partir d'un code. Le live coding postule que la programmation en code directement sur scène (l'écriture du code est projetée sur un écran sur scène en direct) a aussi un caractère d'interprétation et performance. Du point de vue de la composition et de l'écriture, cette technique peut servir de moyen d'expérimentation ou de recherches sonores et paramétriques qui seront, par la suite, fixé dans un code informatique pour être exécutés ultérieurement pendant le concert.

[21] https://fr.wikipedia.org/wiki/Smalltalk
[22] https://fr.wikipedia.org/wiki/Live_coding

Cette fonctionnalité d'évaluer des bouts de code à la volée tient une place centrale dans mon travail, en facilitant énormément l'expérimentation et l'écoute. Elle a aussi été implémentée dans Antescofo grâce à une commande qui permet d'envoyer des fragments de programme (en OSC) à partir d'éditeurs de texte (Sublime Text[23] [alt + p] ou ATOM[24] [alt-cmd-n]) vers l'interprète qui les évalue dynamiquement. Cette technique est largement utilisée dans la librairie AntesCollider pour faire du *live coding* et expérimenter et tester d'une façon dynamique différentes idées.

Une autre caractéristique technique de ChucK est qu'à la différence d'autres logiciels « classiques », il ne fait pas de distinction entre la fréquence d'échantillonnage pour l'audio et pour le contrôle. Cette technique a aussi été explorée dans un prototype prometteur [DGCNO16] d'intégration entre le langage Antescofo et le programme de synthèse FAUST. Cette évolution rend encore plus artificielle la distinction entre partition et instrument qui devient une différence de degré plutôt que de nature.

```
1  // basic FM synthesis using sinosc
2
3  // modulator to carrier
4  SinOsc m => SinOsc c => dac;
5
6  // carrier frequency
7  220 => c.freq;
8  // modulator frequency
9  20 => m.freq;
10 // index of modulation
11 200 => m.gain;
12
13 // phase modulation is FM synthesis (sync is 2)
14 2 => c.sync;
15
16 // time-loop
17 while( true ) 1::second => now;
```

Fig. 1.8 Exemple d'une synthèse par modulation de fréquence dans l'environnement de programmation (strongly-timed) Just-in-Time ChucK pour le *live coding*. On définit deux oscillateurs (ligne 4) où le premier oscillateur *m* se connecte avec un deuxième oscillateur *c*. Entre les lignes 7 et 11 sont donnés les paramètres des oscillateurs. Les générateurs ainsi que le contrôle sont créés dynamiquement.

Cette faculté de pouvoir créer dynamiquement des processus musicaux à partir de l'interaction avec l'utilisateur confère à l'ordinateur la faculté de devenir aussi un interprète et pas simplement un programme exécuter de manière classique, sans interaction avec l'environnement. On va pouvoir créer des processus, les manipuler, les transformer, les superposer, les faire interagir entre eux pour jouer en direct (pour le *live coding* et l'improvisation) ou pour enregistrer ces manipulations dans le cadre de la composition pour les intégrer à la partition électronique.

Écrire la musique comme un code informatique

Comme déjà exprimé, la création des partitions en code commence en même temps que les premiers systèmes de composition du son numériquement comme dans la

[23] https://www.sublimetext.com
[24] https://atom.io/packages/atom-antescofo

famille de logiciels MUSIC-N. Depuis, plusieurs œuvres ont été réalisées entièrement en code dans des logiciels tels que Csound, CLM ou SuperCollider pour en citer quelques-uns. Un exemple de code-partition est donné à la figure 1.9.

L'écriture de la musique directement en code est un exercice d'abstraction complexe et laborieux qui nécessite un apprentissage spécifique. Il faut d'abord se familiariser avec un langage de programmation, puis le pratiquer et le maîtriser. À la différence de la notation musicale traditionnelle dont la maîtrise repose sur plusieurs années de solfège, de composition, de contrepoint, d'orchestration, etc., la programmation reste peu enseignée dans les cursus musicaux. C'est aussi une expérience compositionnelle qui se distingue de la composition électroacoustique dans un DAW qui passe par une représentation graphique.

```
 1  t   0   120 ; DECLARATION DE TEMPO FIXE : DEUX FOIS PLUS RAPIDE QUE LE TEMPO PAR DEFAUT
 2
 3  ; ins   init    dur ampdb   frq fc1 fc2 reson gain
 4
 5  i 134   0   .1  90  8.09    8000    80  1
 6  i . +   .   <   8.095   <   <   <
 7  i . .   .   .   8.10    .   .   .
 8  i . .   .   .   8.105   .   .   .
 9  i . .   .   .   8.11    .   .   .
10  i . .   .   .   8.115   .   .   .
11  i . .   .   .   9.00    .   .   .
12  i . .   .   .   9.005   .   .   .
13  i . .   .   .   9.01    .   .   .
14  i . .   .   .   9.015   .   .   .
15  i . .   .   80  9.02    9000    60  50
16  s
17  t   0   120 1   30  ; TEMPO VARIABLE : DEUX FOIS MOINS RAPIDE QUE LE TEMPO PAR DEFAUT
18
19  ; ins   init    dur ampdb   frq fc1 fc2 reson gain
20
21  i 134   0   .1  90  8.09    8000    80  1
22  i . +   .   <   8.095   <   <   <
23  i . .   .   .   8.10    .   .   .
24  i . .   .   .   8.105   .   .   .
25  i . .   .   .   8.11    .   .   .
26  i . .   .   .   8.115   .   .   .
27  i . .   .   .   9.00    .   .   .
28  i . .   .   .   9.005   .   .   .
29  i . .   .   .   9.01    .   .   .
30  i . .   .   .   9.015   .   .   .
31  i . .   .   80  9.02    9000    60  50
```

Fig. 1.9 Exemple d'un extrait de partition séquentielle événement par événement dans l'environnement Csound[25]. On peut voir qu'il y a des indications de tempo (**t**) (lignes 1 et 17) ainsi que de sections (**s**) (ligne 16) et d'autres symboles pour faciliter l'écriture comme le *carry* (**.**) qui copie la valeur d'un paramètre propre à une note à la note suivante, le *ramp* (**<**) interpole linéairement entre deux valeurs d'un même paramètre et le (**+**) qui est réservé au *p-field p2*. Il calcule automatiquement le temps de commencement de la note courante, en additionnant le temps de commencement et la durée de la note précédente (*p2 + p3*).

Pour moi, le code informatique et les instructions pour générer des actions sonores, musicales et morphologiques correspond à une partition à part entière, même si ces codes sont génériques et spécifiques à l'informatique et n'ont aucune relation à première vue avec la musique. Il ne s'agit plus de représenter des actions musicales d'une façon symbolique mais d'exprimer des idées musicales, sonores, morphologiques à travers un langage de programmation informatique expressif dédié.

[25] http://www.csounds.com/chapter1/french/

L'écriture directement en code dans des systèmes interactifs tels que le *live coding* ou l'écriture des partitions électroniques comme dans Antescofo est une pratique relativement nouvelle qui, par les possibilités ouvertes et la précision rendue possible, s'écarte de la composition « classique » de l'électroacoustique qui reste très contrainte et normalisée par les DAW.

L'écriture entièrement en code textuel ne répond cependant pas à tous les problèmes et il reste beaucoup de travail de recherche et beaucoup de travail de développement pour disposer d'outils performants et plus facilement utilisables par les compositeurs. Nous reviendrons sur les difficultés spécifiques de l'approche code dans le chapitre consacré aux pièces électroacoustiques en temps réel (lors de la présentation de *Curvatura II* par exemple) mais on peut déjà noter que ce paradigme souffre de ne pas offrir de bons repères au niveau temporel (par exemple si on veut jouer ou placer un son ou processus à un moment déterminé de la pièce, on peut se perdre assez facilement dans les lignes de code !). On présentera aussi un exemple d'approche hybride où les dates de départ des processus sont spécifiées dans une *timeline* « augmentée », une approche utilisée dans la première partie de la pièce audiovisuelle *Las Pintas*.

En tant qu'utilisateur du langage de programmation Antescofo et avec la pratique, j'arrive à anticiper avec beaucoup de précision certains éléments de l'électronique et à imaginer des nouvelles possibilités grâce à la flexibilité et l'expressivité du langage. Cette pratique est assez similaire à celle de la notation classique dans laquelle on va pouvoir imaginer dans une partition d'orchestre par exemple les sonorités des mixtures d'instruments et du discours musical. En même temps, les possibilités étendues d'un système de composition et synthèse avec ordinateur laissent aussi beaucoup d'espace à l'expérimentation et à l'improvisation.

Notation graphique de l'électronique

Les approches visuelles sont souvent présentées comme facilitant la programmation ou, plus généralement, comme simplifiant la mise en œuvre de tâches complexes.

On peut écarter les tentatives de représentations graphiques analogiques du son. Quand on veut représenter les sons graphiquement, il y a toujours le problème que le visuel et le sonore ne sont pas dans une même échelle. Et on ne peut pas voir un son ou écouter une image si on ne passe pas par un *mapping* direct entre eux. On peut par exemple voir des figures de Chladni[26] mais on n'est pas en mesure de savoir précisément quelle hauteur ou avoir une notion du timbre à partir des figures obtenues.

Dans le domaine acousmatique, on peut cependant citer quelques systèmes où le graphique est transcrit directement en son comme l'UPIC [MRM90] de Xenakis ou

[26] https://fr.wikipedia.org/wiki/Figure_de_Chladni

l'utilisation de logiciels dérivés comme HighC[27] ou Methasynth[28] (le système IanniX[29], bien qu'inspiré de l'UPIC, relève du séquenceur multicurseur).

Les musiques acousmatiques ne nécessitent que peu d'interprétation pour être jouées, à l'exception de la spatialisation dans le cadre d'un acousmonium. Ce dernier n'a pas besoin de savoir comment faire les sons qui seront joués (ils sont fixés) mais d'avoir une notion claire de la forme et des moments spécifiques d'articulation qui sont nécessaires au regard de la diffusion. Le montage dans un DAW remplit ce besoin et constitue une partition graphique. Mais cette représentation reste pauvre puisqu'elle donne seulement l'information de départ et de fin du playback d'un fichier audio.

D'autres types de descriptions comme le sonogramme[30] sont plus informatives quant au contenu sonore, en représentant le son au niveau spectral (représentation de la pression acoustique en fonction de la fréquence et du temps). Mais regarder un sonogramme ne garantit en aucun cas une écoute interne d'un son comme avec la notation symbolique traditionnelle. Des systèmes tels que iAnalyse ou l'acousmographe utilisent cette technique plutôt dans un but d'analyse que de composition ou d'écriture.

Dans les musiques mixtes, la représentation graphique de l'électronique peut aider le compositeur à élaborer sa partition en explicitant les rapports entre les mondes instrumental et électronique. Il faut pour cela que la représentation visuelle vienne enrichir la partition classique. Expliciter ces relations est aussi utile pour les interprètes ainsi que pour l'analyse de l'œuvre. Dans des musiques mixtes où la partie électronique repose sur des hauteurs, des rythmes ou des modulations déterminés comme dans la musique instrumentale (par exemple des harmoniser, frequency shifter, modulation en anneau, etc.), l'écriture peut être assez claire et déterministe comme dans la notation traditionnelle. Ces représentations, compatibles avec la partition traditionnelle, contraignent souvent à décrire l'électronique en termes de déclenchement, ce qui est une contrainte. Et dans le cas où il faut décrire précisément la fabrication du son, par exemple avec des sons inharmoniques, des spatialisations complexes, ou devant intégrer des données de captations gestuelles ou d'autres types d'analyse en temps réel, ou encore faisant appel à des interactions procédurales, la notation devient moins évidente ou impossible à intégrer dans une partition classique.

Les notations graphiques de l'électronique se sont donc largement focalisées sur la programmation visuelle de celle-ci, en relation avec des outils informatiques et indépendamment de la notation instrumentale classique. Les outils de la seconde génération (*e.g.* Max) ont proposé des approches visuelles qui ont été présentées comme un grand progrès par rapport aux outils textuels de la première génération (*e.g.* Music-N).

---

[27] http://highc.org/history.html
[28] https://uisoftware.com/metasynth/
[29] https://www.iannix.org/fr/
[30] https://fr.wikipedia.org/wiki/Sonagramme

On peut classer les approches visuelles existantes en deux grandes catégories : celles qui relèvent de la *timeline*, et les autres qui correspondent à des systèmes de programmation *data flow* (comme Max ou Pd). Parmi les systèmes basés sur une *timeline*, il y en a ceux qui sont « explicites » comme les séquenceurs DAW et ceux qui sont plus « sophistiqués » comme Ossia score, Iannix, InScore ou la maquette/séquenceur dans Open Music. Antescofo avec son interface Ascograph fait aussi partie des « sophistiqués ». Cette classification est illustrée à la figure 1.10.

Avec les *timelines* « sophistiquées », on peut faire des actions temporelles qui ne sont pas possibles dans des *timelines* explicites, par exemple sauter d'un endroit à un autre ou faire des choix de déclenchement en fonction des conditions, c'est-à-dire avec un degré plus ou moins grand de programmabilité et d'interactivité. Une autre caractéristique de ces *timelines* « sophistiquées », qui sera utilisée dans mes productions, est la multitemporalité, c'est-à-dire pouvoir jouer des phrases musicales à différents tempi, en synchronisation ou pas.

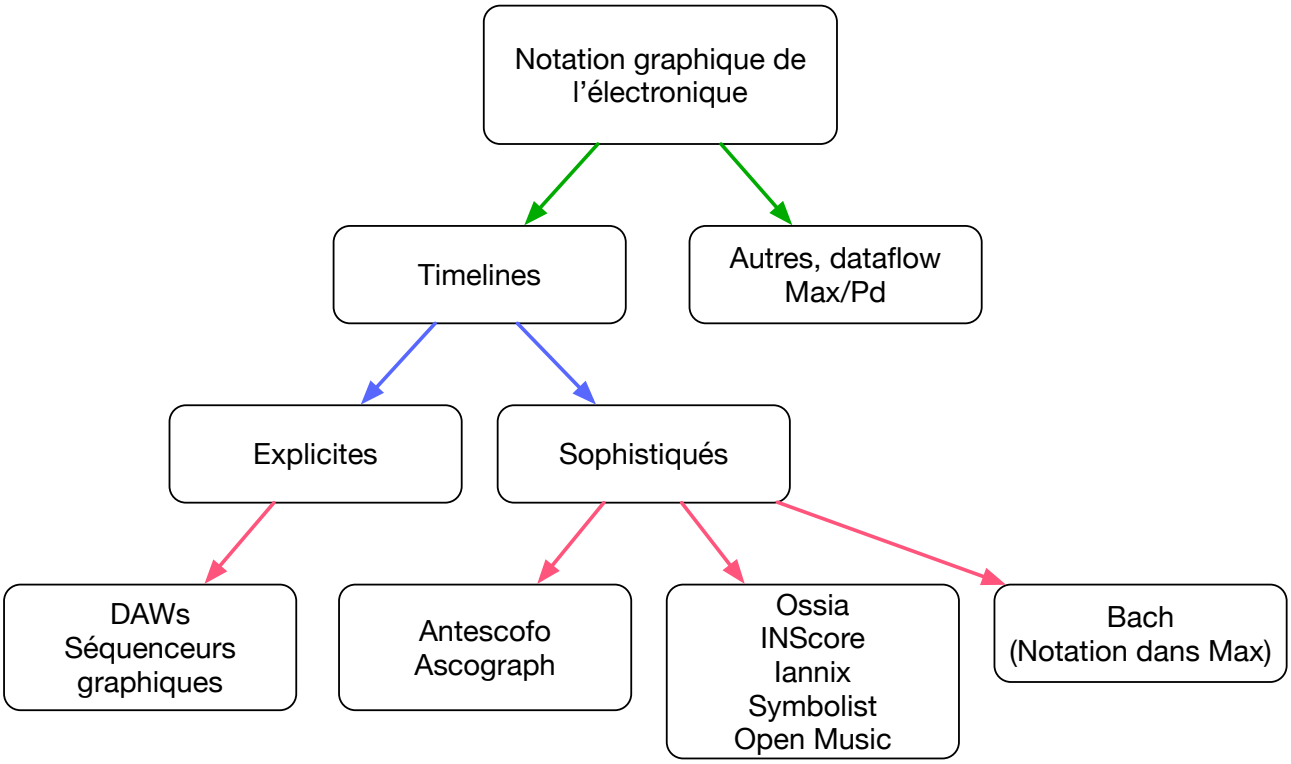


Fig. 1.10 Graphique représentant les familles de systèmes graphiques pour l'écriture de l'électronique.

Limitations de la composition avec des objets graphiques

La représentation graphique est aujourd'hui extrêmement dominante, la plupart des logiciels pour faire de la musique et du design sonore possèdent des interfaces graphiques pour les manipuler et une timeline explicite pour les organiser temporellement. Bien entendu, ce n'est pas un hasard et l'interface graphique et gestuelle dans l'informatique en général est l'une des façons la plus ergonomique et directe pour communiquer à haut niveau avec l'ordinateur. Dans la composition électroacoustique, ce paradigme est représenté par l'approche graphique de la *timeline*

dans les logiciels de montage de type DAW. Dans cette approche, on pourrait dire qu'on compose avec les yeux, la souris et les rapports de distance et d'alignement. Les fichiers sont présentés comme des boîtes contenant des formes d'ondes qu'on déplace dans le temps linéairement, ce qui conditionne forcément l'acte de composition et le résultat musical.

Cependant, en regardant par exemple la partition électronique de *Curvatura II*, pièce électroacoustique en temps réel composée directement en code sans aucun recours à une représentation graphique[31], la question se pose de la pertinence d'utiliser des interfaces ou symboles pour la composition de ce genre de pièces.

Vouloir à tout prix avoir des représentations graphiques pour composer, manipuler des objets visuels qui représentent des sonorités ou des morphologies sonores avec des symboles et des figures ne serait-il pas finalement une limitation pour l'imagination sonore qui elle peut aller bien au-delà des relations représentables graphiquement ?

L'expérimentation de nouvelles façons de composer sans une *timeline* explicite (et mono-curseur) comme dans l'exemple de *Curvatura II* est très importante dans ma démarche. Ce type de composition présente une liberté qui va au-delà des conditionnements de la « composition avec les yeux ».

Il ne faut cependant pas minimiser les difficultés de l'approche textuelle. Elle est difficile à maîtriser, il faut apprendre la programmation, et acquérir des routines qui permettent de se concentrer sur un niveau d'abstraction sans représentations graphiques. Cette pratique n'est pas répandue, elle nécessite un effort et un apprentissage supplémentaire par rapport aux logiciels graphiques où, avec seulement deux mouvements de souris et un click, on peut déjà produire du son. Elle présente aussi des difficultés pratiques certaines, comme quand on veut superposer des couches temporelles indépendantes et polyphoniques dans un langage textuel qui lui est écrit et conçu séquentiellement. Et gérer une partition avec des milliers de lignes de code n'est pas toujours facile…

L'écriture de l'électronique reste un domaine d'étude très actif, et on peut s'attendre à de nouvelles propositions. Depuis 2015, il existe par exemple une conférence annuelle, TENOR[32], qui est consacrée à la recherche des technologies numériques pour la notation musicale en général. Dans le cadre de cette conférence, plusieurs recherches sont consacrées à la notation de l'électronique. Aujourd'hui, il y a de nouvelles expérimentations pour la notation de l'électronique, par exemple dans des systèmes d'aide à la composition tels que Open Music [Ago98], le séquenceur dans OM-sharp [BrBiCaSc17], Bach [AG12] dans l'environnement Max, ou avec des propositions

[31] Cette pièce sera présentée dans la seconde partie de ce document.
[32] https://www.tenor-conference.org/

comme l'objet *struct* (*data Structures*) dans Pd, et de nouveaux logiciels comme l'Acousmoscribe [DiS13], Symbolist [GoB18], InScore [FOL12] ou Ossia Score[33].

Visualisation des partitions électroniques

Il y a une autre façon d'utiliser les représentations graphiques, peu utilisée, mais qu'on peut voir à l'œuvre dans un logiciel dédié à l'analyse musicologique comme iAnalyse[34]. L'idée n'est pas de représenter la prescription des actions musicales à réaliser, mais le résultat de cette prescription, la musique qui a été jouée.

On évite ainsi le problème de représenter opérationnellement l'électronique à réaliser, au profit de la représentation *a posteriori*, « post-mortem », du rendu musical. La représentation graphique peut alors être générée au moment de l'exécution, comme un sous-produit de la réalisation musicale.

Ce rendu graphique du résultat de la partition électronique en code peut servir à des fins d'amélioration ou ajustements de certains paramètres dans un aller-retour lors de la composition entre idées musicales et expérimentation, ou encore après la composition de l'œuvre à des fins d'étude et d'analyse.

Pour ce qui concerne des paramètres et événements déterministes et statiques de l'électronique, quelques pistes et possibilités de développement ont été élaborées comme le projet AscoGraph [CCG14], et d'autres sont envisageables, comme avec l'intégration des partitions électroniques réalisées dans Antescofo dans le logiciel d'analyse musical iAnalyse. Dans le développement de AscoGraph, des tentatives d'exprimer des éléments dynamiques de la partition électronique ont été théorisées [BuCoPo16], notamment avec l'utilisation d'arborescences et différents types de visualisation. Malheureusement, le projet AscoGraph n'a pas été poursuivi.

[33] https://ossia.io
[34] Pierre Couprie, http://logiciels.pierrecouprie.fr

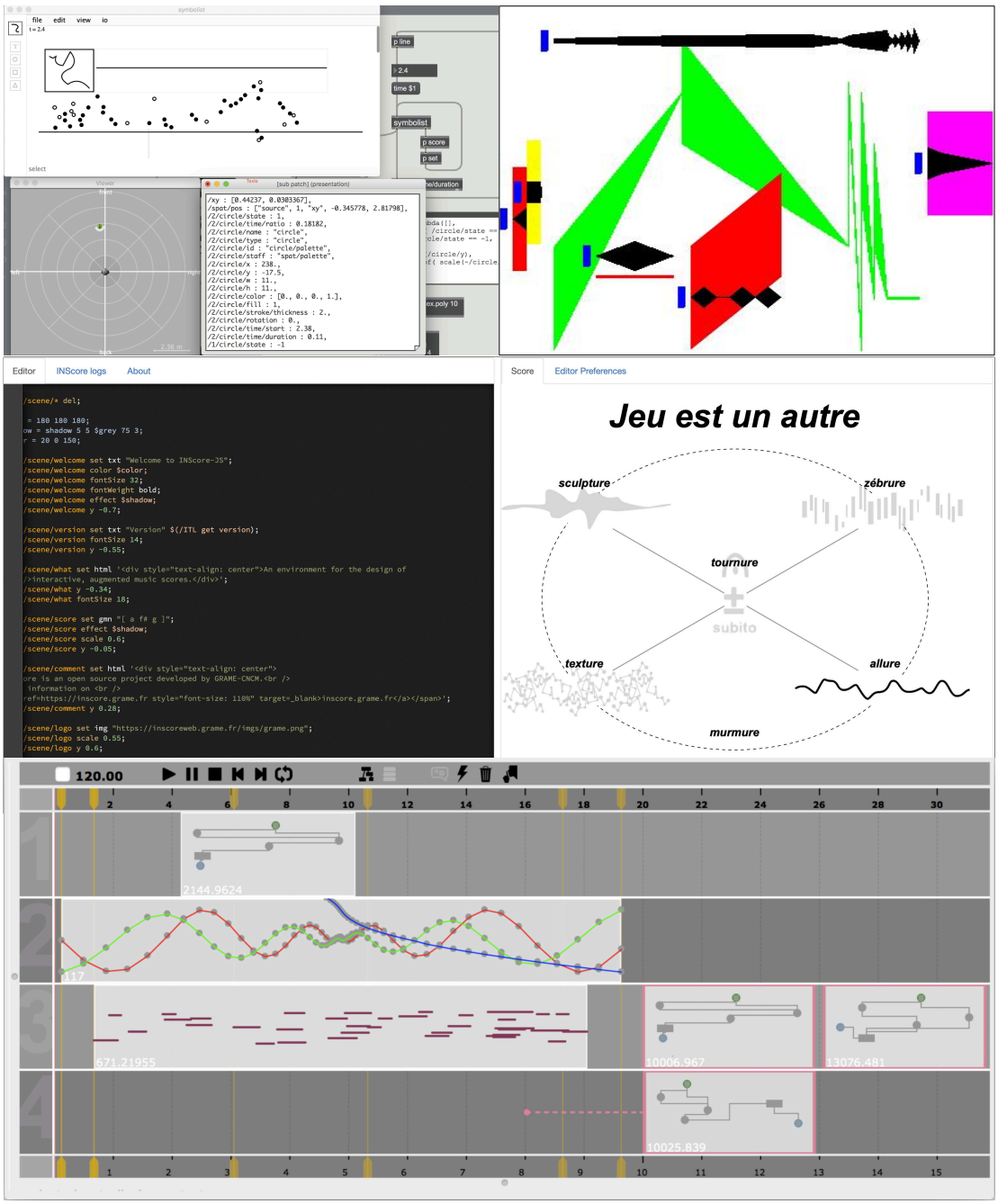


Fig. 1.11 En haut à droite, une capture d'écran du logiciel de notation symbolique Symbolist ; à gauche, l'objet Pd struct ; au milieu, INScore et en bas, le séquenceur de OM-sharp. Ils permettent d'associer une représentation graphique à des événements sonores et musicaux. Dans le cas de Symbolist et INScore et leur intégration du protocole OSC, il est possible d'associer n'importe quel type d'événement à un objet graphique.

I.5. Musiques mixtes et suivi de partition

La musique mixte est l'un des genres musicaux qui m'a le plus intéressé, offrant plein d'idées compositionnelles, de palettes sonores hybrides, de nouvelles formes d'interprétation et d'interaction, et d'avancées technologiques novatrices grâce au mariage et à l'interaction entre le jeu instrumental en direct avec l'électronique aujourd'hui joué par l'ordinateur.

Cette mixture entre sons instrumentaux et sons électroniques et leur synchronisation a commencée avec des rapports plutôt fixes comme ce fut le cas des premières pièces avec bande de Schaeffer, Varèse ou Maderna. Puis ce genre n'a cessé d'évoluer et de se transformer avec les premières expériences des musiques de Cage et de Stockhausen où l'électronique est réalisée en direct, jusqu'à l'arrivée de l'informatique en temps réel avec des ordinateurs assez rapides pour calculer des sons et des manipulations sonores dans le temps de la performance. De cette union entre instrumentiste humain et ordinateur, de nouvelles musiques pleines d'inspiration et de nouvelles idées ont commencé à apparaître depuis le milieu des années 1970. Cette émergence a été initialement possible par le développement de ressources matérielles dédiées : à l'Ircam avec la création des premiers ordinateurs pour la synthèse en temps réel à partir de la 4A, 4B, 4C pour aboutir à la 4X[35] en 1981. Au même moment et plus dans la démarche des musiques acousmatiques et électroniques au GRM, il y a eu le développement de SYTER, un autre système audionumérique temps réel. Citant aussi à la même époque mais dans un autre registre l'UPIC de Xenakis développé au CEMAMu (Centre d'études de mathématique et automatique musicales) pour la génération de synthèse du son à partir de dessins.

Ces développements n'ont cessé de se renouveler et de s'enrichir avec les nouveaux apports de la technologie aussi bien au niveau logiciel que matériel.

Suivi de partition

La musique mixte pose de manière frontale le problème de la coordination de l'électronique aussi bien au moment de la composition que de l'interprétation. Dans ce contexte, le *suivi de partition*, processus consistant à aligner automatiquement une exécution musicale en direct avec une partition électronique, se présente comme une solution à la problématique de synchronisation entre le monde instrumental et l'électronique. Il va permettre de faire jouer la partie électronique en synchronisation avec la partie instrumentale et va se comporter comme un chef d'orchestre, surtout dans le cas des pièces avec beaucoup d'interactions et de couches d'électronique. Cette interaction et cette synchronisation automatique peuvent permettre de jouer l'électronique d'une façon plus fine, précise, que ne pourrait le réaliser un humain qui déclencherait les événements électroniques « à la main » (par exemple dans le cas où il faut réaliser un traitement note à note).

Le suivi de partition introduit en 1984 avec les articles fondateurs de Roger Dannenberg et Barry Vercoe [Dan84, Ver84] est toujours un axe de recherche très actif dans le domaine de la musique mixte interactive. À l'Ircam, cette recherche a débuté en 1986, d'abord avec le suivi d'*instruments midifiés*[36], notamment avec le processeur numérique en temps réel 4X. Parmi les premières œuvres réalisées à l'Ircam et jouées sur des instruments MIDI, on peut citer *Jupiter* et *Pluton*, fruit d'une collaboration

[35] http://articles.ircam.fr/textes/DiGiugno81a/

[36] Il s'agit de rajouter à un instrument acoustique la possibilité d'envoyer ou de recevoir des messages MIDI en lui rajoutant des composantes électroniques de détection des notes, de vitesse, d'amplitude, etc.

historique entre le compositeur Philippe Manoury et le mathématicien Miller Puckette qui a donné naissance à Max, l'un des logiciels les plus répandus aujourd'hui dans le domaine de l'interaction musicale [Puc91]. La technique a rapidement évolué afin de permettre la détection des hauteurs à partir de l'analyse du flux audio des instruments acoustiques avec le système ISWP[37] (une station d'informatique musicale, précurseuse des logiciels de deuxième génération où le contrôle et le traitement du signal sont incorporés dans un même environnement graphique). L'intégration de l'analyse audio pour estimer les hauteurs d'instruments acoustiques par une détection FFT (Fast Fourrier Transform) de la f0 (fréquence fondamentale) a largement simplifié l'utilisation à n'importe quel instrument acoustique monophonique ou polyphonique sans nécessité de le midifier. La partition de l'instrumentiste à suivre étant connue, l'incorporation de modèles probabilistes (chaînes de Markov cachées) a grandement amélioré l'estimation et l'alignement entre le jeu instrumental en direct et la position dans la partition électronique.

Antescofo, la nouvelle génération de suiveurs de partitions, développé vers 2007 à l'Ircam par Arshia Cont, est un système qui couple une machine d'écoute pour réaliser le suivi et un langage de programmation dédié permettant de décrire les actions électroniques. Il inclut aussi un aspect temporel fort, chose qui n'était pas présente dans les anciens systèmes avec une estimation du tempo courant d'exécution par l'instrumentiste.

Ces deux points, le langage de programmation et l'inférence du tempo du musicien, constituent une avancée fondamentale dans le domaine. La prise en compte du tempo va permettre non seulement de déclencher des événements discrets de l'électronique, comme dans les premières générations de suiveurs, mais aussi d'avoir des synchronisations fines au niveau de la durée (et non seulement des événements) et des contrôles continus, par exemple pour synchroniser des bandes (technique de time-stretching), de la spatialisation, des paramètres de la synthèse, etc. De plus, l'estimation du tempo peut être utilisée comme une nouvelle donnée de captation de l'interprétation qui peut servir à manipuler différents éléments de l'électronique en temps réel.

L'intégration du langage de programmation (qui sera décrit dans le chapitre III) augmente les possibilités d'écriture de l'électronique ainsi que la capacité à coordonner plusieurs événements en parallèle comme dans une partition pour orchestre.

Partitions électroniques pour le temps réel et l'interactivité

Les premières cue-list [38], ou partitions électroniques pour décrire les actions électroniques en temps réel pour contrôler le synthétiseur 4X à l'Ircam, sont composées de deux fichiers, un pour le suivi de notes MIDI et l'autre pour les actions à réaliser pour la note MIDI détectée. La première œuvre à utiliser un suiveur de partition et les potentialités de l'interactivité entre un instrument acoustique et l'ordinateur avec des

[37] https://en.wikipedia.org/wiki/ISPW
[38] Liste d'événements à réaliser à un moment déterminé et en ordre.

sons de synthèse est *Jupiter* pour flûte et électronique de Philippe Manoury, œuvre créée en 1987. Dans *Jupiter*, l'ordinateur va recevoir l'information de la hauteur et de l'intensité de la flûte, ce qui va permettre l'utilisation du suivi de partition ainsi que l'alimentation des algorithmes de production de l'électronique pour créer l'interaction en temps réel. *Jupiter* ouvre le cycle de trois pièces *Sonus ex machina* qui explorent des possibilités nouvelles d'interaction entre instruments et ordinateur et posent les bases de la musique interactive mixte en temps réel, avec notamment l'article « Les partitions virtuelles » où Philippe Manoury théorise ses idées sur l'interaction : « *La partition virtuelle est donc une partition dont on connaît, a priori, la nature des éléments qui vont être traités mais dont on ignore les valeurs exactes qui vont définir ces éléments.* » [Man97]

Les deux parties de la figure 1.12 représentent seulement le suivi et les actions, les processus et le patch sont décrits dans d'autres fichiers et exécutés dans un autre ordinateur. À l'époque, il y avait un nombre limité des traitements et synthèses fixes, comme les harmoniser, frequency shifter, délais, sampling, la réverbération infinie et un synthétiseur additif.

L'intégration du contrôle et du signal dans un même environnement graphique dans la deuxième génération de systèmes temps réel (ISWP) a donné lieu à de nouvelles possibilités de contrôle beaucoup plus fines, interactives et relationnelles entre le son acoustique et l'électronique. Un exemple musical d'utilisation de ce nouveau système est la pièce *En echo* de Philippe Manoury où l'analyse des formants de la voix (maxima d'énergie dans le spectre de la voix) va permettre de contrôler la synthèse en temps réel.

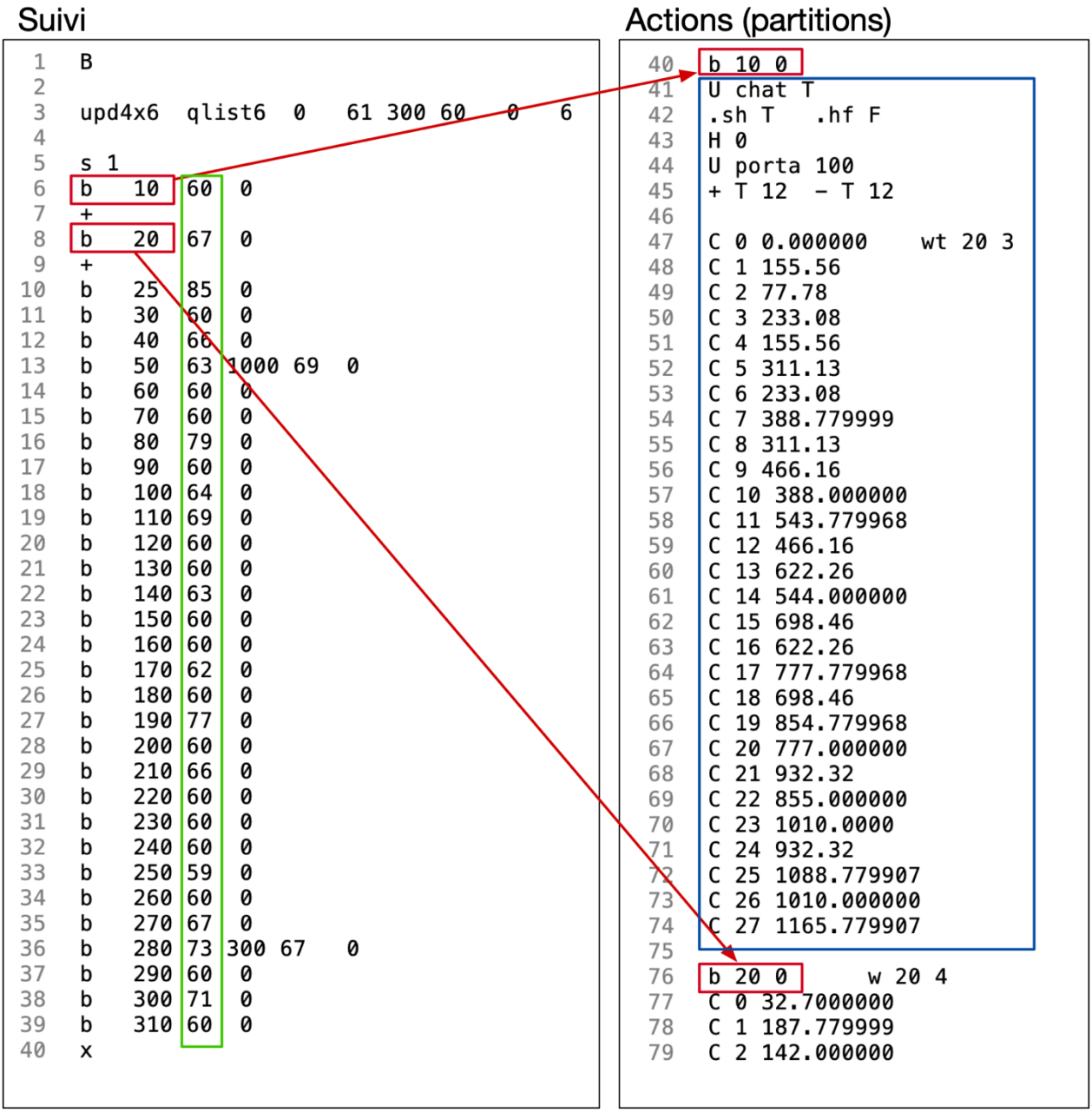


Fig. 1.12 Exemple de la partition pour l'ordinateur 4X de la section 6 de la pièce *Jupiter* (1987) de Philippe Manoury pour flûte et électronique en temps réel. À droite, la partie qui décrit l'ordre d'apparition des notes MIDI (en vert) et sa correspondante action (en rouge), note 60 déclenche « b 10 ». À gauche, l'étiquette (label) qui correspond aux actions à déclencher (en bleu) pour cet événement. Dans les parties actions, il y aussi la spécification des connexions, par exemple à la ligne 42 `.sh T` correspond au synthétiseur qui rentre dans l'harmonisateur et c'est T (true), `.hf` l'harmonisateur rentre dans le frequency shifter `F` (false). Entre les lignes 47 et 74, il y a les fréquences pour alimenter un synthétiseur additif joué en temps réel, processus appelé « le chapo » par le compositeur. Ce traitement tisse un lien entre les hauteurs jouées à la flûte et la distribution de l'énergie dans le spectre de la synthèse additive comme un filtre formantique à travers les hauteurs MIDI de la flûte. « L'idée initiale consistait en un prélèvement de l'enveloppe spectrale de la flûte, afin de contrôler les amplitudes d'un groupe d'oscillateurs » [Man03].

Cette intégration du contrôle et du signal dans le même environnement a permis, à partir de ce moment, un grand nombre d'expérimentations et de mapping du monde du signal vers le contrôle, et vice versa. La partition électronique, même sans suivi, a aussi adopté ce nouveau système avec par exemple l'introduction de l'objet Max *qlist* qui

reprend la partie action des programmes de la 4X. Ces objets sont inclus dans les *patchs de concert*[39]. L'objectif est de définir une série de messages discrets et de déclencheurs de sons ou de processus dans une liste d'ordonnancement temporel qui conditionne chaque action à l'écoulement d'un délai (principe qui sera repris dans Antescofo). Ces objets peuvent être déclenchés depuis l'extérieur à partir d'un suivi de partition, issu de l'analyse de la hauteur d'un instrument, d'une pédale de déclenchement, d'un clavier d'ordinateur ou d'un autre contrôleur externe. Une fois déclenchés, ils exécutent leurs actions en suivant l'ordonnancement prévu.

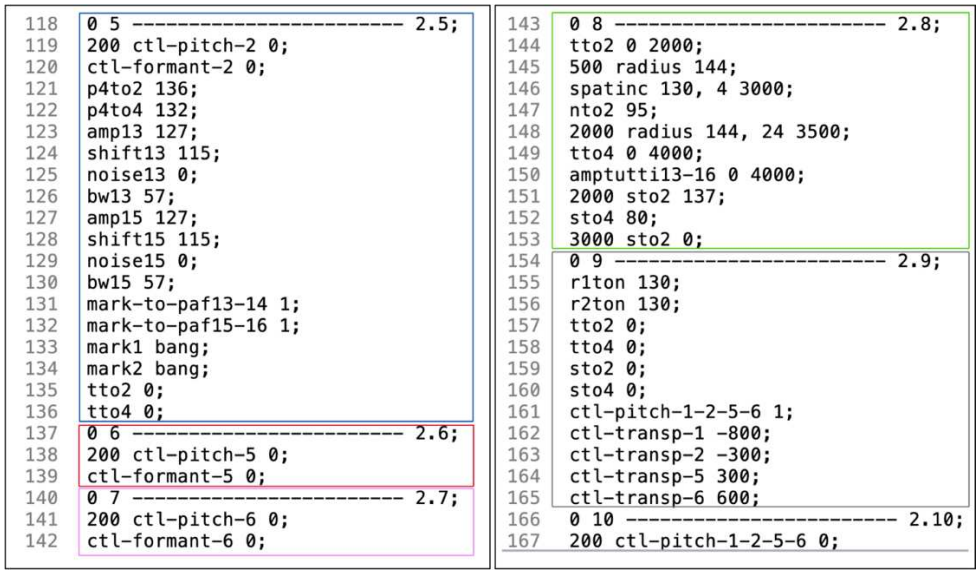


Fig. **1.13** Exemple d'écriture de l'électronique dans l'objet *qlist* dans l'environnement Max pour la pièce *En écho* (1994) de Philippe Manoury. Chaque événement correspond à une liste d'actions à réaliser à un certain moment de la performance et déclenchés par un suivi de partition intégré dans le système. On peut voir aussi l'utilisation des délais en millisecondes avant les messages (ex. : lignes 119-145-148…).

Les actions regroupées dans un objet qlist peuvent aussi être éclatées des « boîtes de messages ». Comme pour l'objet *qlist*, il s'agit d'envoyer des paramètres pour des traitements et synthèses comme pour déclencher des processus et aussi en intégrant des délais. Ce système présente l'inconvénient que les données sont difficiles d'accès, il faut aller chercher les paramètres dans des sous-patchs et puis les modifier dans des boîtes séparées, ce qui ne permet pas d'avoir une vision d'ensemble de la partition. Les messages comme dans les objets *qlist* restent discrets et ponctuels.

[39] Un patch de concert est un programme réalisé dans Max spécifiquement pour un concert, avec une architecture qui comprend des processus de contrôle, des modules de traitement, synthèse et analyse audio et un système de qlist ou séquences pour piloter l'ensemble en synchronisation avec la partie instrumentale.

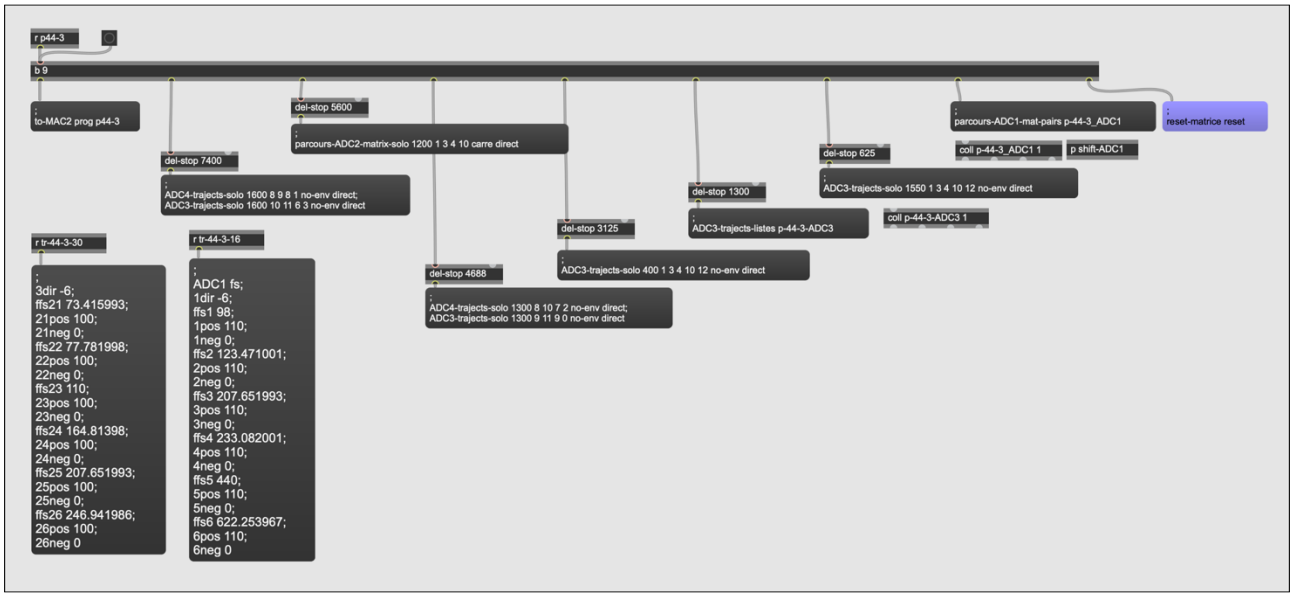


Fig. 1.14 Exemple d'un événement dans une « boîte de messages » Max de la pièce *Lichtung III* (2007) pour ensemble et électronique de Emmanuel Nunes. Il s'agit de décrire les actions de l'électronique dans le style Max avec l'utilisation des délais et des informations de spatialisation dans des objets *coll* (dictionnaires). Dans ce cas, les événements sont déclenchés par un clavier MIDI qui joue sur scène avec les autres instruments.

Vers la partition électronique centralisée

Avec l'arrivé du suivi de partition Antescofo, l'idée d'intégrer tous les éléments de l'électronique au sein d'une même partition se concrétise dans la syntaxe même du langage de programmation du système.

En effet, à la différence de ses prédécesseurs, la partie du programme dévolue au suivi de partition[40] et la partie dévolue aux actions électroniques sont présentes au sein d'un même fichier texte. Leur interdépendance est explicite : chaque action est rattachée à un événement musical (note, accord, trill, etc.) qui sera détecté par la machine d'écoute (cf. figure 1.15). Cela n'empêche pas certaines actions d'avoir une durée propre et de se dérouler en parallèle avec le discours instrumental. La partition Antescofo entremêle ainsi (une forme simplifiée de) la partition instrumentale et la partition électronique.

La partie instrumentale peut être issue d'une partition classique éditée dans un logiciel de gravure musicale tel que Finale, Sibelius ou Notability Pro. La procédure consiste à exporter cette partition en format *Music XML* puis à la traduire dans le format Antescofo. Ce format contient moins d'informations que celui d'une partition instrumentale classique et n'adresse qu'un seul instrument (l'instrument à suivre). Mais il démontre la possibilité et l'intérêt d'inclure la spécification instrumentale et la spécification électronique au sein d'un même document.

[40] Il faut souligner que même si le suivi n'est pas utilisé, le compositeur peut spécifier des événements abstraits qui peuvent servir à structurer la partition. Il est par exemple possible de démarrer l'exécution d'une partition Antescofo à partir d'un événement abstrait quelconque.

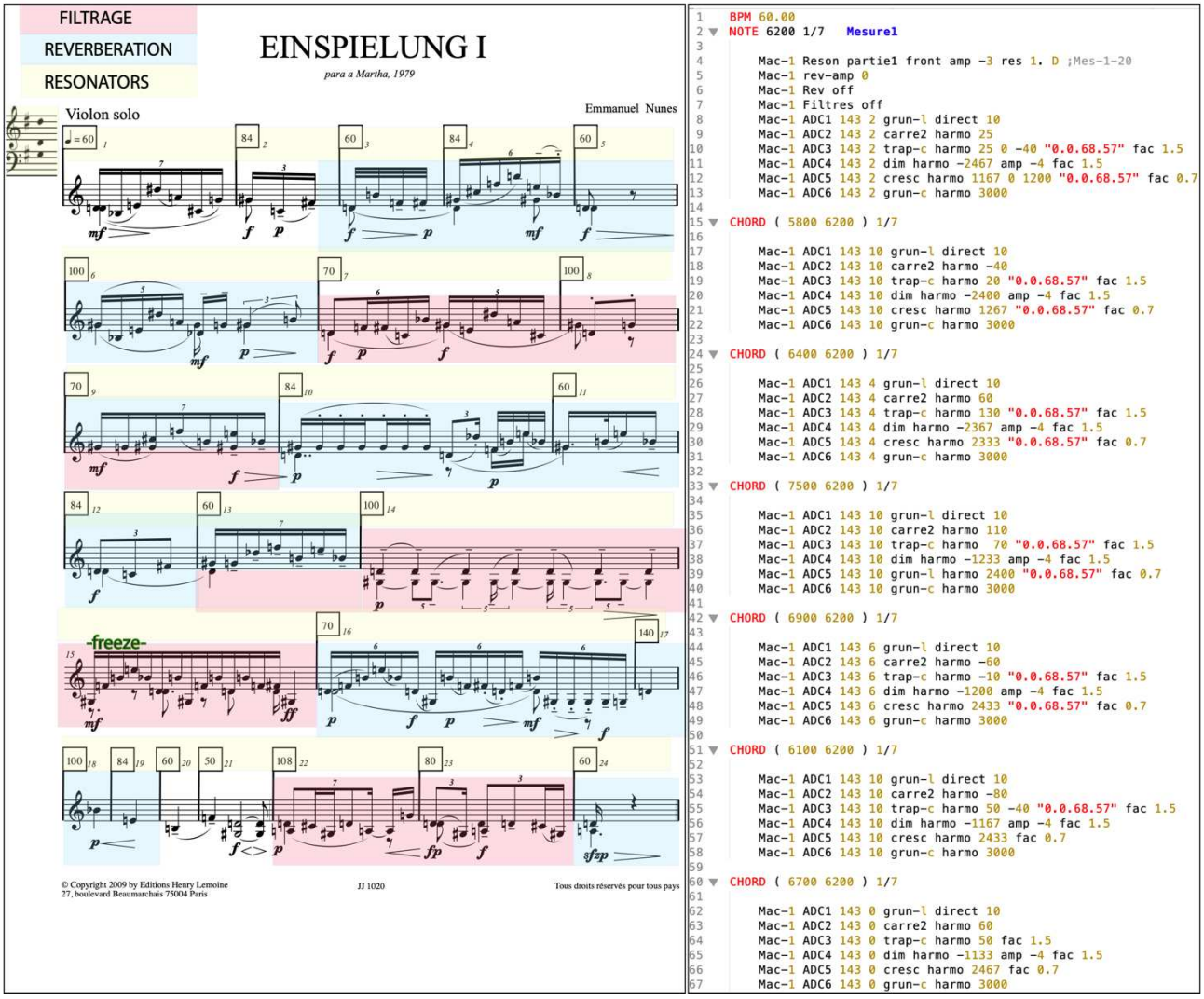


Fig. 1.15 Exemple de l'écriture de l'électronique dans *Einspielung I* d'Emmanuel Nunes pour violon et électronique (version avec électronique temps réel de 2011). À gauche, la partition pour violon (en couleur, les parties traitées avec filtrage, réverbération et résonateurs). À droite, la partition électronique de la première mesure en septolet, chaque note déclenche une série d'actions (traitement note à note) pour la spatialisation avec l'utilisation d'enveloppes sur le son du violon qui sont traitées par six chaînes de traitement indépendantes (ce qu'Emmanuel Nunes appelle ADC).

La pièce *Einspielung I* d'Emmanuel Nunes est un bon exemple d'évolution des techniques d'écriture de l'électronique dans un système de suivi de partition en temps réel. En 1994, quand Emmanuel Nunes et Eric Daubresse (réalisateur en informatique musicale) ont travaillé pour faire la partie électronique de la pièce dans le système ISPW, la carte *DSP*[41] de traitement du son installée dans la NeXT[42] s'est arrêtée de fonctionner juste avant la générale du concert et le projet est resté en attente pendant 17 ans. Quand Nunes a repris la partie électronique de la pièce en 2011, il avait à sa disposition de nouveaux moyens techniques pour réaliser les idées théoriques qu'il ne pouvait pas réaliser lors de la première tentative en 1994. Les nouvelles possibilités

[41] *Digital Signal Processing*, processeur spécialisé dans le traitement du signal.
[42] Ordinateurs des années 1990 utilisés à l'Ircam pour la ISPW, développés par l'entreprise américaine NeXT Computer, Inc.

techniques offertes par Antescofo lui ont aussi permis de repenser complètement l'électronique grâce à un environnement d'écriture de l'électronique plus souple, avec un suiveur de partition plus robuste et prenant en compte la dimension temporelle de l'interaction.

Au niveau suivi de partition, la partition électronique Antescofo comporte différents événements musicaux issus du monde de l'écriture symbolique comme des notes, des accords, des trilles ou des glissandi qui sont reconnus par la machine d'écoute. Pour l'instant, une des limitations du suivi de partition pour la musique contemporaine est le fait que la machine d'écoute ne prend pas en compte des sons plus complexes comme les modes de jeux particuliers à chaque instrument, ni des sons avec une grande composante de bruit, ce qui réduit son utilisation et donc conditionne une certaine écriture instrumentale.

Avec notamment les travaux de José Echeveste sur les stratégies de synchronisation [Ech15] et de Philippe Cuvillier [Cuv16] sur l'intégration des paramètres temporels dans le modèle probabiliste de suivi, la synchronisation entre les parties instrumentales solistes et l'écriture de la partie électronique s'est grandement améliorée et enrichie.

Dès ses premières versions, Antescofo permet de regrouper des séquences des actions dans une structure de groupe (*group*) représentant une phrase musicale dotée de sa stratégie de synchronisation. La notion de *loop* permettant d'itérer une séquence d'actions était aussi présente dès le début, avec la possibilité de définir des macros incluant des expressions rudimentaires. Voici un exemple d'une définition de fonction et de macro dans la première version de Antescofo pour calculer en temps réel des temporalités à partir du jeu et une conversion de notes MIDI vers des fréquences en hertz :

```
@MACRO_DEF midi2hz(X)
{
    expr{440.0 * exp((X-69) * log(2) / 12 )}
}
```

Cette fonctionnalité est très importante parce qu'elle préfigure l'idée d'un langage de programmation incorporé à la partition, permettant l'intégration de processus complexes dans la partition elle-même et en relation avec celle-ci. Cette intégration permettra par la suite une grande flexibilité, et une grande expressivité. Elle ouvre aussi vers une indépendance de la partition au système hôte dans lequel on va programmer les patchs ou traitements et synthèses : le contrôle dans tous ses aspects sera centralisé dans la partition.

Les primitives de calcul d'Antescofo se sont rapidement enrichies et ont évolué vers un langage de programmation complet à partir de 2011 avec l'arrivée d'un nouveau chercheur sur le projet, Jean-Louis Giavitto. Le langage Antescofo introduit des variables, des fonctions, des structures de données et de contrôle et d'autres éléments qui seront décrits dans le chapitre dédié au langage Antescofo ainsi que des nouvelles fonctionnalités.

Si le langage d'Antescofo se présente comme un langage généraliste, il comporte plusieurs mécanismes spécifiquement dédiés aux besoins musicaux. Les *curves* qui servent à interpoler et échantillonner des variations continues de paramètres sont un exemple de constructions dédiées. La première œuvre à utiliser les *curves* dans Antescofo fut *Iki-no-michi* (*Les Voies du souffle*) pour saxophones et électronique en temps réel de Ichiro Nodaira, commande de l'Ircam et créée en 2012 par Claude Delangle.

Cette construction illustre tout l'intérêt de réunir au sein d'une même partition l'instrumental et l'électronique : elle permet en effet des contrôles continus avec une synchronisation en relation directe avec le jeu instrumental. Cette fonctionnalité est très importante puisqu'elle donnera naissance à une nouvelle technique qui dépasse la notion d'événements discrets pour aller vers le continu. Ainsi, les contrôles continus ne seront plus fixes et indépendants de l'interprétation mais il s'adapteront à l'agogique du jeu instrumental où ils pourront être déformés, étirés ou rétrécis par un contrôle d'un tempo externe.

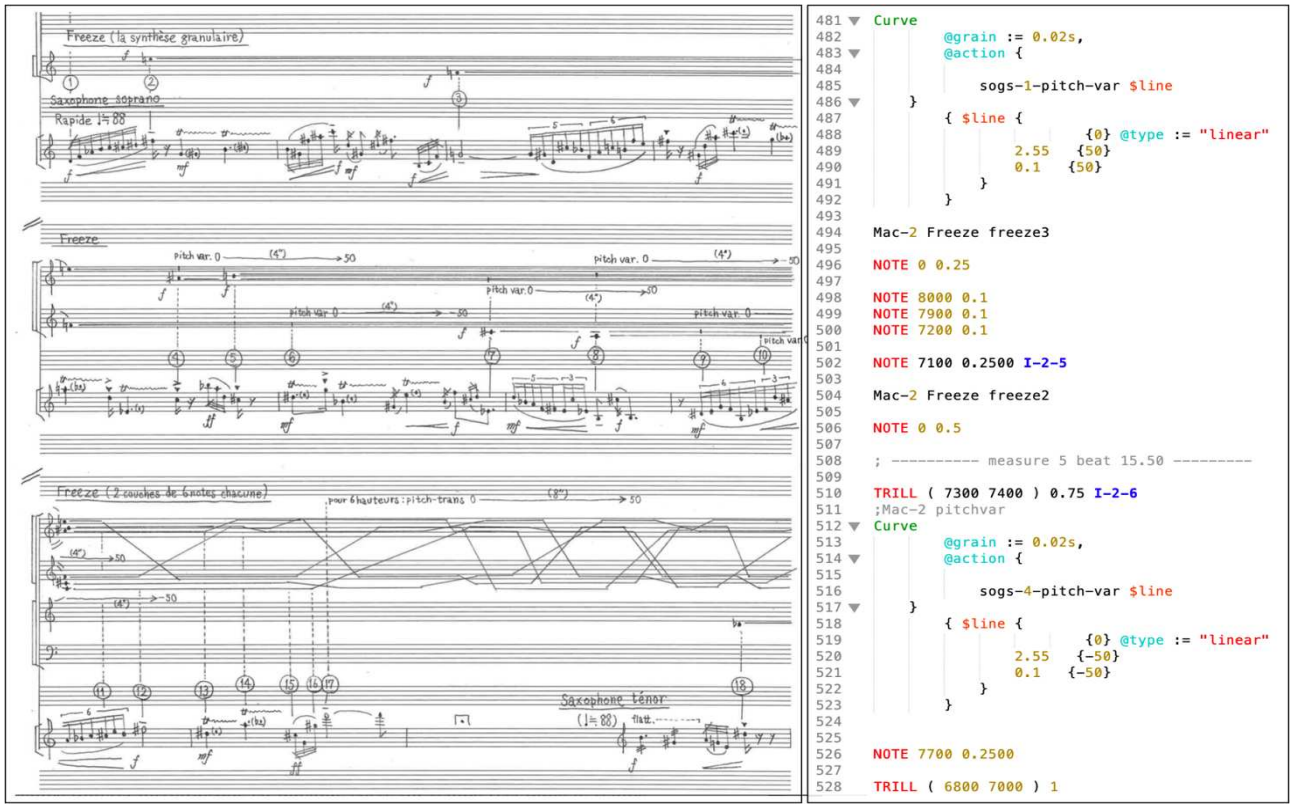


Fig. 1.16 Exemple de l'écriture de l'électronique dans *Iki-no-michi* pour saxophones et électronique en temps réel. À gauche, la partition pour saxophone avec des annotations de la partie électronique. À droite, la partition électronique avec les notes (NOTE) correspondant à la partie symbolique de la partition. Cet exemple montre pour la première fois la capacité du programme à générer des interpolations dans le temps en relation directe au tempo du jeu de l'instrumentiste. Dans ce cas, il s'agit de faire des « freezes » (gels) de notes à un moment déterminé puis commencer à faire des glissandi de ces hauteurs polyphoniquement.

Dans l'exemple suivant, issu de ma pièce *Dispersion de trajectoires* pour saxophone baryton et électronique (créée au festival *Traiettorie* à Parme, Italie, par Claude

Delangle en 2014), l'intégration des processus en temps réel dans le langage est poussée à un niveau d'interaction supérieur. Le son du saxophone est analysé en temps réel par différents types des descripteurs audio [Pee04] pour piloter la synthèse et traitement en temps réel. *Dispersion de trajectoires* est la première pièce à intégrer le suivi de partition et langage Antescofo à un système de synthèse dynamique dans SuperCollider.

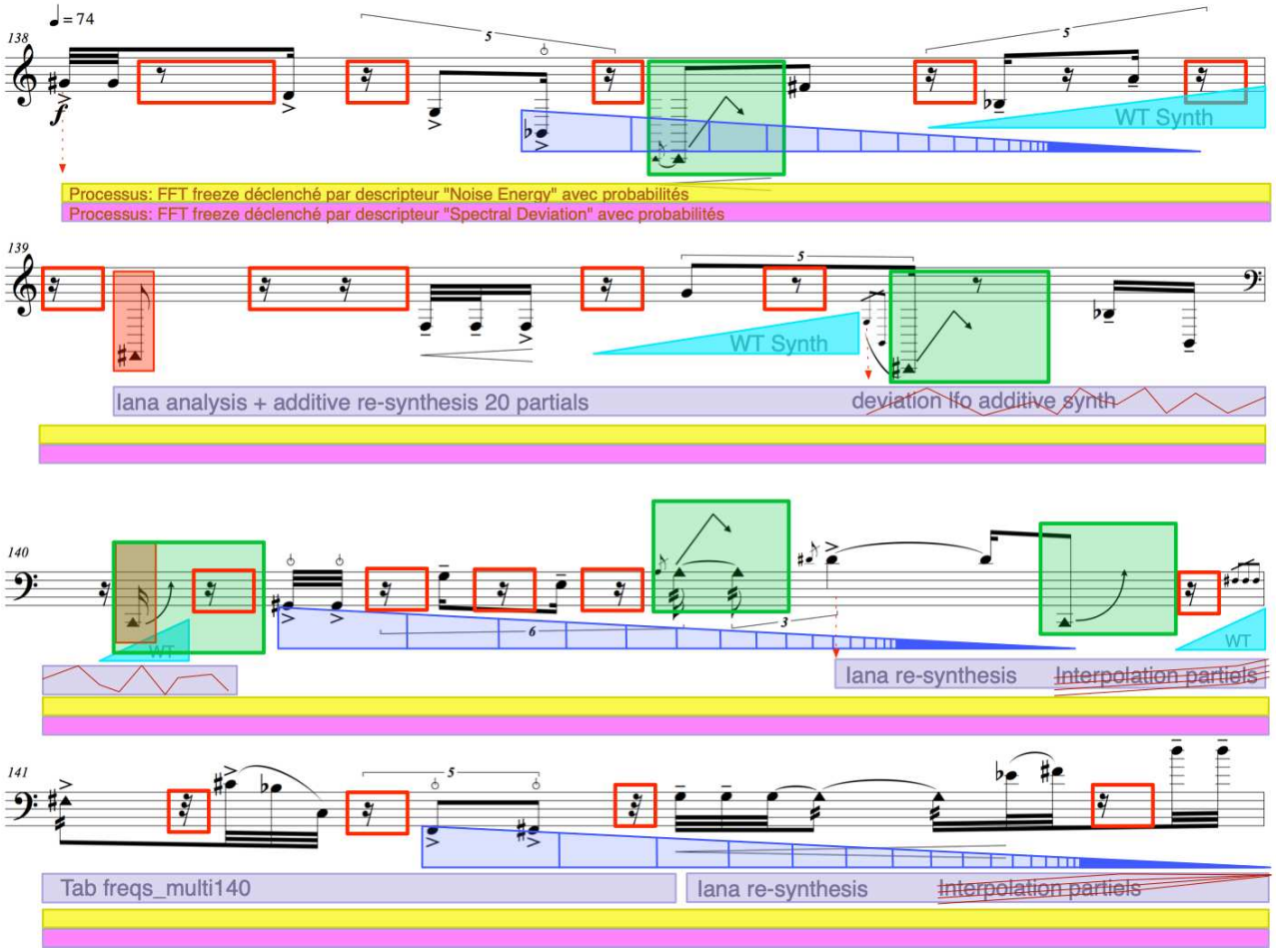


Fig. 1.17a Extrait de la partition de *Dispersion de trajectoires* (entre les mesures 138 et 141) avec l'indication de l'interaction. Les figures en couleur représentent les interactions dans la partition des différents éléments et processus de l'électronique. Cette représentation reste très qualitative. Si elle est utile à l'instrumentiste en lui donnant une certaine idée de l'électronique, elle ne permet par exemple pas la réalisation de celle-ci.

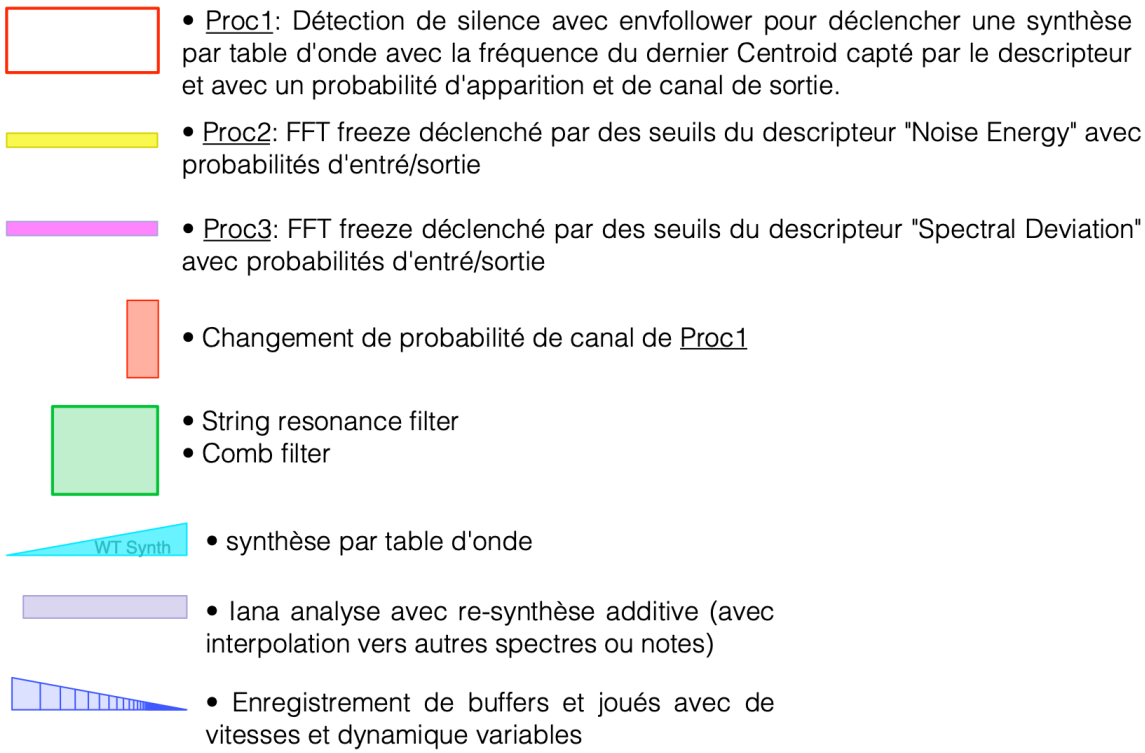


Fig. 1.17b Description de chaque élément et de l'interaction avec le type d'analyse, de synthèse, de traitement et de processus déclenché par le suivi de partition. Cette description qualitative de haut niveau donne lieu à une spécification complète et exécutable dans la partition électronique.

Ce couplage entre contrôle et synthèse développe encore plus avant le concept de partition centralisée et donnera lieu par la suite au développement de la librairie pour la composition électroacoustique AntesCollider. Cette approche de programmation directement dans la partition des tous les éléments d'interactivité, de traitements et de synthèses stimule la créativité compositionnelle et agrandit la palette sonore. Grâce à ce système, on peut écrire la partie électronique comme « une orchestration de l'interactivité » avec plusieurs couches qui interagissent entre elles et avec le jeu instrumental en direct.

Avec ce système, on peut arriver à un niveau de finesse de l'écriture de l'électronique chirurgical qui permet par exemple d'enregistrer des petits buffers audio de l'instrument « note à note » pour alimenter une synthèse granulaire et contrôler les rythmes et enveloppes de chaque grain indépendamment et les modifier dynamiquement dans le temps.

```
7496  BPM 74 @modulate
7497  NOTE G#4 1/8 mesure138
7498
7499  antescofo::suivi 1
7500  descriptors-on-off 1
7501  // active freeze avec NoiseEnergie
7502  $freeze_noisenergie := ::NoiseEnegie_trig_spec_freeze("freeze_noisenergie", 0.15, 0.1, 0.1, [0.5,0.5,0.5, 1, 1, 1, 1, 0.5], 3, 0.8, 0.6)
7503  // active freeze avec SpectralDeviation
7504  $freeze_specdev := ::SpecDeviation_trig_spec_freeze("freeze_specdeviation", 0.4, 0.1, 0.1, [0.5,0.5, 1, 1, 0.2, 0.2, 1, 1], 3, 0.95, 0.6)
7505
7506  NOTE G#4 1/8
7507  // track_name, seuil, fade_in, fade_out, prob_hp, amp, prob
7508  $trig_wt1 := ::env_foll_trig_wtable("wavetab1", 0.1, 0.1, 0.2, [1,1,0.5, 0.3, 0.2, 0.2, 0.3, 0.5], -20, 1)
7509
7510  NOTE 0 1/2
7511  NOTE D4 1/4
7512  NOTE 0 1/5
7513  NOTE G3 2/5
7514
7515  group chop_mes138 // prepare la prochaine note
7516  {
7517      1/5 crea_track8 chop_mes138 0 0.1 -1 AudioInput input $saxIn #-> TCortaOnset amp 2 gran_dur @dur2sec(2/5) trig_time 10 amp 0
7518
7519      Curve trig_time  @Grain := 0.01s,  @Action := crea_track8 chop_mes138 set 01_TCortaOnset trig_time $y
7520              {
7521              $y
7522                  {
7523                      {10}
7524                  2 {60}
7525                  }
7526              }
7527
7528      Curve trig_time  @Grain := 0.01s,  @Action := crea_track8 chop_mes138 set 01_TCortaOnset amp $amp
7529              {
7530              $amp
7531                  {
7532                      {0} //@type "exp_out"
7533                  5 {-50}
7534                  }
7535              }
7536
7537      2/5 crea_track8 chop_mes138 set 01_TCortaOnset ingain 0.
7538      5 crea_track8 chop_mes138 off 1
7539  }
7540
7541  NOTE 0 0 ;rajout
7542
7543  NOTE G#2 1/5
7544
7545  1/5 crea_track8 chop_mes138 set 01_TCorta t_trig 1
7546
7547  NOTE 0 1/5
7548  NOTE C2 0
7549
7550  group combs_mes138
7551  {
7552      crea_track8 comb_mes138 0 0.01 0 AudioInput input $saxIn #-> TCombC amp -5.75 delaytime 0.02899 decaytime -6.17
7553      crea_track8 streson_mes138 1 0.01 0 AudioInput input $saxIn #-> TStreson amp 0 reson 0.90600001811981 delayTime 0.026000000536442
7554
7555      1 crea_track8 comb_mes138 set 00_AudioInput amp -120
7556      crea_track8 streson_mes138 set 00_AudioInput amp -120
7557  }
```

Fig. 1.18 Extrait de la partition électronique de *Dispersion de trajectoires* de l'exemple précédent (mesure 138). On peut voir l'utilisation des variables ($) et l'instanciation des processus (lignes 7502, 7504 et 7508) avec différents paramètres pour l'interaction avec des descripteurs audio ainsi que d'autres traitements. On peut aussi voir la création des chaînes de traitements dynamiquement (lignes 7517, 7552 et 7553). La variable `$saxIn` représente le bus d'entrée audio provenant des microphones du saxophone dans SuperCollider.

```
1928    // NoiseEnegie_trig_spec_freeze(track seuil fade_in fade_out hp_prob amp)
1929    @proc_def ::NoiseEnergie_trig_spec_freeze($track, $seuil, $fade_in, $fade_out, $hp_prob, $amp, $prob_on, $prob_off)
1930    @abort{
1931            fftfreezetrig $track off 1 // libere la synth
1932    }
1933    {
1934        @local $val, $last_val, $poly, $inc, $fade_in, $fade_out, $actif, $noterest, $noterest2
1935
1936        $actif := 0
1937        $val := -1
1938        $last_val := -2
1939        $inc := 0
1940
1941        fftfreezetrig $track (@vprob([$x | $x in 8], $hp_prob)) $fade_in $amp AudioInput input $saxIn #-> FftFreeze bufft1
1942        fftfreezetrig $track amp -120
1943
1944        0.2s $actif := 1 // activer le whenever 0.2s apres instantier le proc/synth a cause du init freeze spectral
1945            whenever($actif && ($NoiseEnergy == $NoiseEnergy))
1946            {
1947                $val := ($NoiseEnergy >= $seuil ? 0 : 1)
1948
1949                if ($val != $last_val) { //filtre repetitions change
1950                    $last_val := $val
1951
1952                    $noterest := ((@rand(1.) < $prob_on)? 1: 0)
1953                    $noterest2 := ((@rand(1.) < $prob_off)? 1: 0)
1954
1955                    if (($val == 1)&&($noterest == 1)) {
1956                        0.1 fftfreezetrig $track pos ((@vprob([$x | $x in 8], $hp_prob) * 2)/8) // pan probabilistique avec formule PanAZ SC
1957                        fftfreezetrig $track amp 0
1958                        fftfreezetrig $track set 01_FftFreeze trig 1
1959                    }else{
1960                            if ($noterest2 == 1) {
1961                            0.5 fftfreezetrig $track set 01_FftFreeze trig 0
1962                            fftfreezetrig $track amp -120
1963                        }
1964                    }
1965                }
1966        }
1967    }
```

Fig. 1.19 Exemple du processus `NoiseEnergie_trig_spec_freeze` pour faire des gels (*freezes*). La structure de contrôle `whenever` (ligne 1945) reçoit la variable `$NoiseEnergie` qui provient directement de l'analyse du descripteur audio NoiseEnergie de l'objet Max *IrcamDescritptors*[43]. En fonction d'un seuil, le processus déclenche les freezes FFT (ligne 1958) avec l'envoi du son gelé vers une position spatiale (parmi 8 haut-parleurs) avec une fonction probabiliste (ligne 1956).

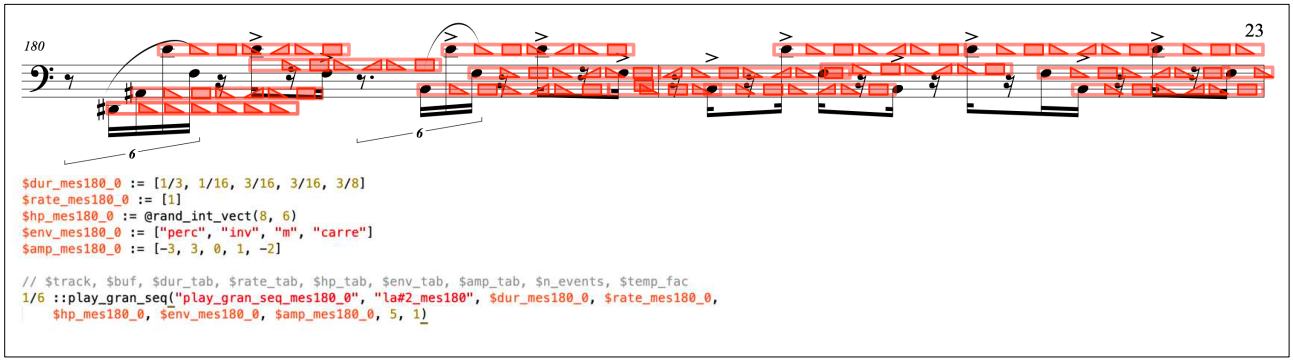


Fig. 1.20 Extrait de la mesure 180 de la partition de *Dispersion de trajectoires*. Le système enregistre un segment de chaque note pour alimenter plusieurs générateurs de synthèse granulaire instanciés dynamiquement. Ceci permet d'avoir un système polyphonique à *n* voix qui se superposent chacune avec un développement rythmique et des types d'enveloppes et de spatialisation différents. En bas, un extrait du code de la partition pour générer la synthèse granulaire avec différentes possibilités, souvent géré par des probabilités. Le processus `::play_gran_seq` utilise des tableaux de rythmes, transposition, spatialisation, type d'enveloppe et amplitude pour générer la synthèse granulaire.

[43] https://forum.ircam.fr/projects/detail/max-sound-box/

I.6. La partition électronique centralisée

Les exemples précédents illustrent la notion de partition électronique centralisée. C'est une partition pour l'exécution puisque l'électronique est définie à travers le programme qui la réalise. Mais pour peu que le langage de programmation impliqué soit un langage offrant un haut niveau d'expressivité, c'est aussi un espace de pensée qui permet l'écriture précises et complète des processus temporels, des traitements, des synthèses et une intégration des différents éléments nécessaires aux musiques interactives.

On peut voir la partition centralisée comme une partition d'orchestre dans laquelle chaque pupitre correspondrait à la gestion des différents programmes en parallèle, comme la synthèse, la captation gestuelle, la vidéo, les lumières, etc., avec la synchronisation et l'interaction entre ces programmes (cf. figure 1.21).

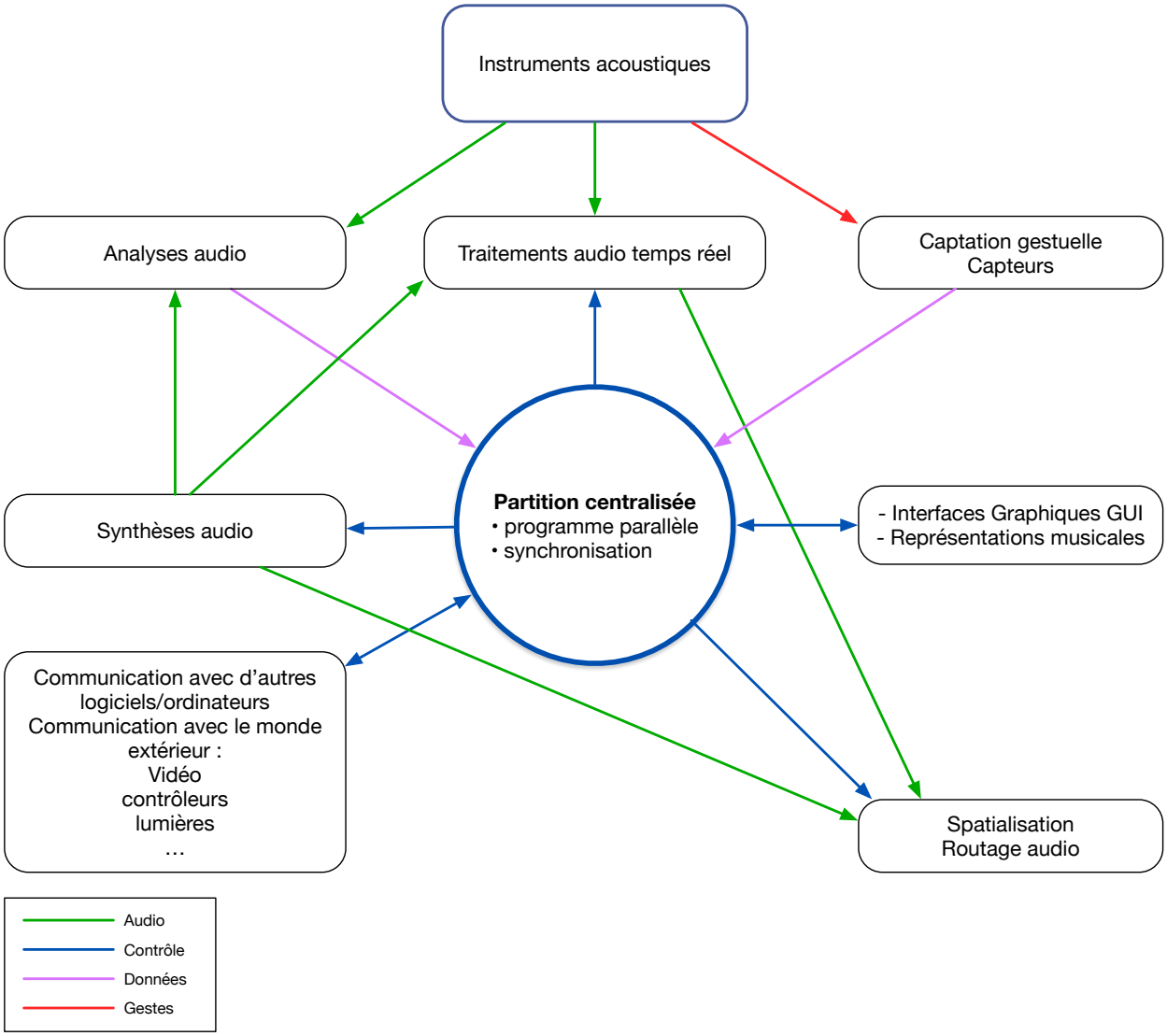


Fig. 1.21 Représentation des différents éléments qui constituent aujourd'hui les musiques interactives et la place de la partition centralisée qui va coordonner tous ces éléments au sein d'une même partition. Elle peut s'adresser à n'importe quel média temporel et aussi les faire interagir entre elles à travers des communications et traitements de différentes données dans une *timeline* « sophistiquée » en utilisant en même temps les avantages d'un langage de programmation expressif.

Dans le cadre de ce travail, nous proposons de réunir tous ces programmes dans une seule et unique partition décrite dans la partition Antescofo. De cette façon, lors de la performance, Antescofo va gérer chaque pupitre (et même créer et jouer les instruments dans le cas de la synthèse grâce à AntesCollider) et va les coordonner comme un chef d'orchestre. La partition centralisée est donc à la fois le support d'écriture, la partition d'orchestre et le chef d'orchestre.

Les possibilités offertes par cette approche de l'écriture de l'électronique sont fascinantes et s'adressent à différents genres musicaux :

- *musiques mixtes* : des pièces mixtes avec une nécessité de précision dans la synchronisation et une orchestration des interactions ;
- *musiques acousmatiques* : pour leur écriture et la composition du son, leurs transformations dans le temps et la superposition des couches temporelles de leur morphologie et leur construction à tous les niveaux, de la micro à la macro forme ;
- *œuvres multimédias* : que ce soit dans le domaine de l'opéra, du théâtre ou de la danse, un système d'écriture avec un langage de programmation temporelle dans une même partition permet de synchroniser différents médias, coordonner les différents éléments entre eux et les faire interagir.

Structure d'une partition électronique centralisée

La partition électronique dans le langage Antescofo s'organise en différentes parties qui suivent un ordre fixe. La partition débute par les librairies de fonctions et de processus préprogrammés par les utilisateurs qui seront utilisées dans le reste de la partition. Puis il y aura une étape d'initialisation des paramètres et de l'environnement qu'on va utiliser, par exemple Max ou AntesCollider. Enfin, il y a la partition proprement dite, qui entremêle événements musicaux et écriture de l'électronique. C'est dans cette partie qu'on compose l'électronique et les interactions avec les différents médias.

Une courbe d'apprentissage variable

La nécessité d'apprendre un langage de programmation est souvent invoquée comme une des difficultés majeures qui s'oppose à la généralisation des partitions électroniques. Cette remarque s'applique tout autant à la partition centralisée qui est un aboutissement de la partition électronique.

J'ai eu l'opportunité de travailler avec plusieurs compositeurs. Bien que la maîtrise nécessaire de la programmation prenne un temps certain, voire des années, les compositeurs peu habitués peuvent aussi se familiariser assez vite avec la syntaxe des partitions électroniques à un niveau plus adapté. Il ne s'agit pas de rentrer dans la programmation de processus complexes mais d'utiliser une syntaxe de base, avec des librairies prédéfinies, pour écrire une partition selon des principes et processus prédéterminés. Le compositeur peut alors écrire sa partition en réutilisant des structures

déjà programmées et en les modifiant graduellement pour répondre à de nouvelles nécessités musicales.

Un des objectifs de la librairie de composition AntesCollider (qui sera présentée dans le chapitre V) est justement de donner aux compositeurs plusieurs possibilités de création et de structuration des formes musicales, paramètres, synthèses, etc. pour qu'ils puissent écrire l'électronique de façon plus simple et fluide sans rentrer dans une programmation de plus bas niveau.

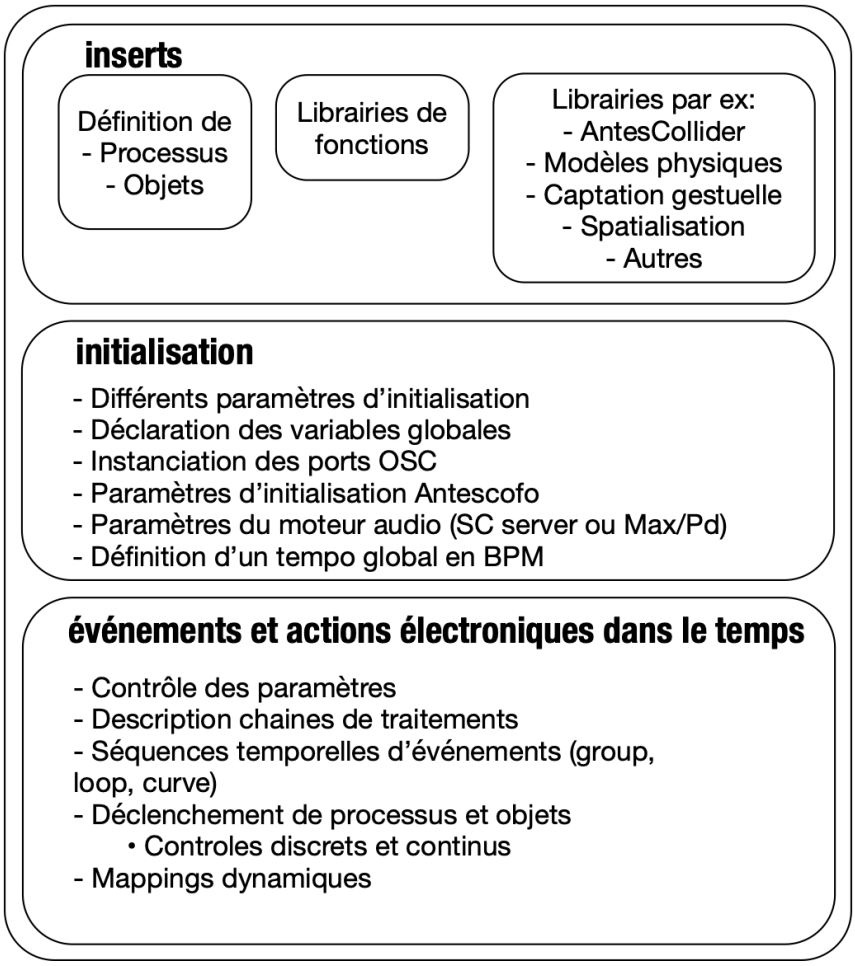


Fig. 1.22 Structure en trois parties d'une partition électronique dans le langage Antescofo. La première partie *inserts* (@insert), correspond à toutes les fonctions, processus et librairies qui peuvent être inclus dans la partition, comme la librairie AntesCollider, des modèles physiques ou la *Trajectory Score Library*[44] pour la spatialisation. La partie *initialisation* concerne toutes les variables et paramètres qui vont être calculés avant le départ de la musique pour préparer et instancier les différents composants de l'électronique. La troisième partie *événements et actions électroniques dans le temps* est la partition proprement dite, elle décrit les différentes actions à effectuer à des moments précis du déroulement de la pièce musicale.

[44] https://forum.ircam.fr/projects/detail/trajectory-score-library/

I.7. Préservation de l'électronique

Pour conclure ce chapitre, il faut évoquer un problème potentiel qui peut empêcher les partitions électroniques et les partitions centralisées de jouer pleinement le rôle de notation de l'électronique : leur obsolescence.

La partition, avec sa notation fixée par la tradition, a comme fonction de transmettre aux musiciens la musique pour être jouée et en même temps de pouvoir l'enregistrer pour la reproduire dans le futur, y compris après le décès du compositeur. Ce problème de transmission se prolonge en un problème d'archivage, de conservation et de pérennité.

Dans le monde de la musique électroacoustique en temps réel, cette problématique est toujours présente et s'aggrave du fait qu'à la différence de la notation symbolique occidentale qui a eu un développement relativement lent sur un support matériel stable comme le papier et visant un ensemble d'instruments prédéfinis, la technologie numérique en revanche évolue en permanence. Les évolutions du matériel et du logiciel impliquent par exemple de mettre à jour ou de redévelopper des effets ou des synthèses pour pouvoir rejouer les pièces.

La question de la pérennité des supports numériques est un des grands défis de nos sociétés contemporaines et les solutions son bien loin d'être satisfaisantes. Même avec les techniques actuelles très sophistiquées, nous sommes encore loin d'avoir des supports aussi pérennes que la table en argile des 36 Chants hourrites et ses 3 400 ans ! L'espoir de supports spécifiques de longue durée n'a jusqu'à présent pas été réalisé. Des supports comme le CD ou les disques durs s'abîment inévitablement avec le temps et, à la différence de la représentation analogique de l'information sur papier ou sur argile, l'information numérique est fragile et s'altère beaucoup plus facilement. Et indépendamment de la pérennité de l'encodage physique, la course technologique est la cause d'une obsolescence à court terme. Par exemple, des supports spécifiques pour le son comme le DAT (bandes magnétiques) ou pour le stockage de données comme les disques Syquest, ne sont plus fabriqués une vingtaine d'années après leur apparition, et on ne retrouve plus aujourd'hui de dispositif de lecture.

Apple aura utilisé en 30 ans 4 architectures de processeurs différents, ce qui signifie à chaque fois un travail supplémentaire de recompilation pour les développeurs. Les logiciels dont le développeur « est passé à autre chose » ne fonctionneront tout simplement plus dans les nouvelles architectures. Et l'obsolescence des outils informatiques aujourd'hui n'est pas seulement due au matériel, mais aussi aux évolutions des systèmes d'exploitation.

Ce problème n'est pas récent. L'exploitation du répertoire à l'Ircam passe pour chaque reprise par une étape de *portage*. Cette étape vise à assurer les mises à jour nécessaires et même le redéveloppement des librairies externes qui ne sont plus disponibles. Cela peut se faire plus simplement si les logiciels sont des logiciels en « open source ». Mais au-delà de la mise à jour, la phase de portage est aussi nécessaire pour adapter la pièce

à la salle de concert (par exemple pour la réverbération), aux dispositifs de captation (micro) et de diffusion (par exemple multicanale).

Nos sociétés contemporaines de consommation, du déchet et mercantiles poussent à l'obsolescence, un paradoxe auquel doivent faire face les institutions vouées à un travail de conservation du patrimoine et de la culture. Cette contradiction nous amène à interroger une préconception implicite de la musique savante occidentale : est-ce que les musiques qu'on fait aujourd'hui sont destinées à être rejouées dans le futur ou doivent-elles être considérées comme des objets éphémères liés à un moment et à un lieu, comme c'est le cas dans d'autres arts (la mise en scène dans le théâtre par exemple) ou dans d'autres cultures ?

Bien entendu, nous n'avons pas assez de recul pour prévoir ce qui arrivera aux créations musicales actuelles. On peut déjà constater qu'il y a un plus grand pourcentage de pièces mixtes avec informatique qui ont été jouées une ou deux fois puis sont passées à l'oubli que de pièces qui font répertoire. Bien sûr, toute création n'est pas destinée à rentrer dans le répertoire, mais cette tendance est plus marquée pour les pièces mixtes avec des dispositifs complexes, ce qui montre l'impact de l'obsolescence technologique.

Ce phénomène va de pair avec le fait que les compositeurs ont aussi besoin d'avoir des commandes et tout un système de diffusion (les salles de concerts, les festivals, les orchestres) qui privilégie les créations plutôt que les reprises. Si on ajoute les contraintes de temps (et d'argent) de plus en plus marquées qui amènent à négliger la documentation des dispositifs électroniques impliqués, et l'obsolescence du matériel utilisé (ordinateurs, capteurs, caméras, interfaces de contrôle, contrôleurs, dispositif de diffusion multicanale…), on voit bien que la situation n'est pas près de s'améliorer, au moins dans le temps où j'écris cette thèse ni dans un futur proche…

La mise à jour des pièces avec électronique est gérée pour l'instant par les institutions et les pièces sont mises à jour si l'œuvre en question est reprise. Le temps fait son travail de filtrage. Quand les outils ou librairies ne sont pas disponibles et que le compositeur n'est plus parmi nous, c'est tout un travail de restauration à faire pour essayer de recréer les mêmes sonorités que la version d'origine, en s'appuyant sur des enregistrements quand ils existent. Il faut donc avoir une infrastructure et consacrer des ressources pour conserver et actualiser ces œuvres en permanence en s'adaptant aux nouvelles technologies de production sonores.

Il faut tout de même relativiser le problème : la difficulté évoquée existe aussi pour la musique classique. Sa perpétuation semble poser moins de problème mais implique tout un écosystème, des instrumentistes, des orchestres, des conservatoires, des professeurs, des lieux de concerts, des luthiers, etc., pour pouvoir se perpétuer.

Parmi des réponses envisageables qui diminuent le coût de la pérennité des pièces mixtes ou avec dispositifs interactifs en temps réel, il y a :

- la standardisation et la rétrocompatibilité logicielle et matérielle ;

- des architectures logicielles qui permettent de pérenniser le programme en exécutant des applications dans différents environnements (comme *Docker* ou comme les systèmes de virtualisation) ;
- le développement de l'open source et de l'open hardware.

Une initiative comme la plateforme Software Heritage [45] montre bien que ces problèmes ne sont pas spécifiques à la musique et les solutions apportées bénéficieront aussi au domaine musical. Si la disponibilité des sources ne garantit pas une pérennisation des logiciels, elle y contribue grandement en permettant théoriquement, même dans un futur lointain, de recompiler les logiciels nécessaires.

Les logiciels open source permettent aussi un partage de connaissances et, dans le cas de l'informatique musicale, des librairies de traitements, de synthèses, de représentation des données musicales, etc. La librairie AntesCollider développée au cours de cette thèse, est une librairie open source. Elle est basée sur le logiciel open source SuperCollider, moteur de synthèse sonore, et Antescofo, langage de programmation actuellement en accès gratuit mais dont la partie langage est destinée à devenir open source.

Une posture plus radicale est aussi possible à l'instar d'autres cultures où la musique relève d'une tradition orale. La transmission se concentrerait pendant les répétitions, l'exécution et le concert. Le concert est conçu comme un moment créateur d'émotions et de nouvelles idées et la pièce n'est pas destinée à être conservée comme un tableau, une sculpture ou un objet matériel. La musique par essence est un art du temps, du présent, éphémère et immatériel.

[45] Software Heritage est une initiative qui vise à archiver tous les programmes open source et à les garder exécutables : https://www.softwareheritage.org/

II. Outils pour la composition de musiques électroacoustiques et mixtes

> « *La technologie à elle seule ne suffit pas à susciter des outils de création, non plus que l'intuition du musicien. Il faut engranger tout un savoir, un savoir-faire et des idées : sur les possibilités technologiques ; les modes de* synthèse, *leurs caractéristiques psycho-acoustiques, leurs implications sur les mises de jeu ; les situations musicales qu'on veut mettre en œuvre, et en particulier, les programmes d'aide à la réalisation et à la composition qu'elles supposent, la place à donner au "temps réel", aux commandes gestuelles, et au temps différé, au recul critique, aux représentations hors temps.* » [Ris85]

II.1. Introduction

Dans ce chapitre, je vais exposer ma conception des outils informatiques pour la composition de musiques mixtes, interactives et électroacoustiques à partir de ma pratique compositionnelle et de RIM. L'exploration des outils et instruments électroniques/informatiques m'a questionné sur son rapport avec la créativité et la nécessité d'avoir des systèmes flexibles qui permettent non seulement d'aider le processus de création mais aussi de le stimuler avec de nouvelles possibilités. Cette exploration m'a permis aussi de me confronter à différents environnements de travail conçus pour des tâches spécifiques et pour la création de différents types de sons et de musiques.

On peut distinguer différentes catégories d'environnements informatiques pour la création musicale et la composition. Il y a des logiciels *ad hoc* de type séquenceur comme les DAW qui ont des limitations évidentes du fait qu'on ne va pas pouvoir changer leur comportement interne. On dira qu'ils sont « fermés ». D'un autre côté, il existe des environnements beaucoup plus « ouverts » comme ceux proposant une programmation visuelle (de type data flow) tels que Max/Pd qui permettent de construire des événements musicaux en temps réel. Dans une autre catégorie encore, il y a les logiciels d'aide à la composition (CAO) tels que OpenMusic qui vont permettre de modéliser et organiser des pensées compositionnelle à un niveau plus abstrait et symbolique mais qui se limitent à des opérations hors temps.

Un système comme Antescofo échappe à la dichotomie temps réel/hors temps et propose de ce fait une approche nouvelle. Il permet en effet une unification du contrôle et de l'écriture générale de l'électronique à un haut niveau expressif, permettant de spécifier des événements compositionnels et de les générer en temps réel en tenant compte de la temporalité de la performance.

Le graphique ci-dessous offre une classification des outils offerts par l'informatique musicale à partir de trois pôles : la composition, la performance et le traitement et la manipulation du son. Historiquement, ces trois classes ont été distinguées car elles

correspondent à des besoins en ressources de calculs et à des contraintes de temps réel très différents. Avec l'augmentation de la puissance de calcul (en vitesse mais aussi en mémoire), les systèmes récents tentent souvent de se positionner à l'intersection de ces catégories traditionnelles, ouvrant ainsi de nouvelles dimensions créatives et facilitant l'expression des idées musicales.

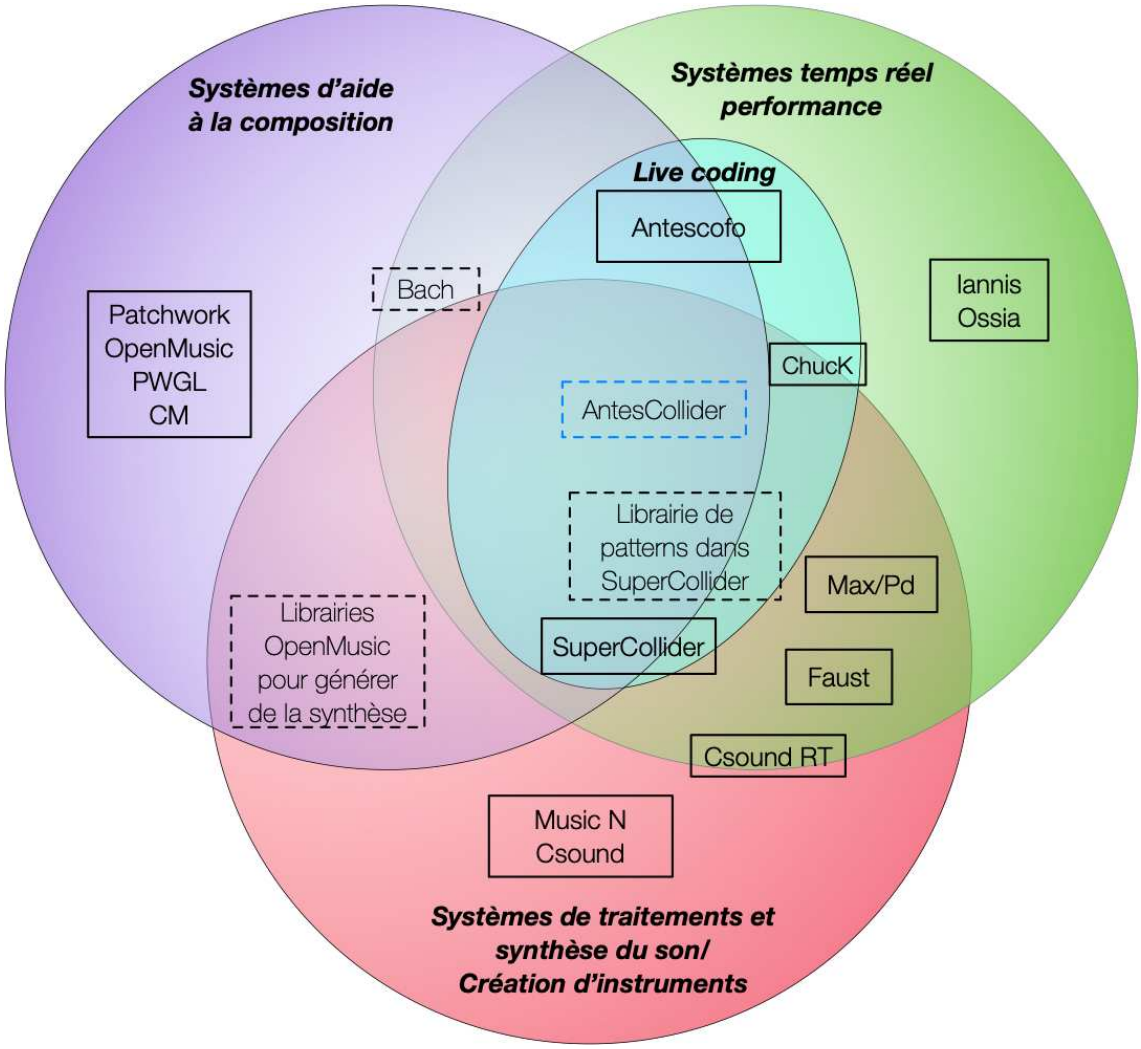


Fig. 2.1 Une classification des logiciels musicaux en systèmes de composition, de performance (temps réel) et de synthèse et traitements. La liste des logiciels cités n'est pas exhaustive mais elle donne une bonne idée du paysage et permet de positionner la librairie AntesCollider dans le panorama des logiciels et librairies pour la performance et création sonore aujourd'hui. Les contours en lignes pointillées représentent des librairies programmées dans des langages dédiés à la musique.

Le langage Antescofo permet une écriture fine et un contrôle de l'électronique en synchronisation avec des performeurs. Il se situe donc à l'intersection des pôles

composition et performance. Mais il ne possède pas de système natif de traitement et de synthèse sonore[46].

Mon travail avec différents environnements, la nécessité qui en a émergé de composer avec des systèmes plus flexibles permettant une écriture fine, souple et dynamique de l'électronique et le besoin d'adresser ces systèmes à partir d'une partition unique permettant de concrétiser ma pensée musicale, m'ont amené au développement de la librairie AntesCollider. Cette librairie qui tire parti du langage de programmation Antescofo et du moteur de synthèse SuperCollider va permettre la création de structures compositionnelles de haut niveau et la construction dynamique de chaînes de traitements du son et de synthèses sophistiquées. Utilisée comme système d'aide à la composition en temps réel, cette librairie permet aussi d'écrire l'électronique d'une façon centralisée dans une partition augmentée où l'on pourra décrire tous les éléments de l'électronique d'une façon précise au même titre qu'une partition instrumentale traditionnelle, en profitant des possibilités, des potentialités et de la puissance de calcul que les langages informatiques offrent aujourd'hui.

AntesCollider se situe donc à l'intersection des trois pôles que nous avons pointés, en complétant Antescofo par des fonctionnalités adressant la synthèse du son.

Organisation du chapitre

La suite de ce chapitre offre un panorama des outils informatiques musicaux du point de vue de ma pratique. Il détaille les motivations qui m'ont poussé à développer la librairie AntesCollider. La section suivante présente la co-évolution qui existe de fait entre la technologie et la pensée musicale. Ces interactions, ancrées dans la facture instrumentale et la production du son, ont dépassé les problématiques instrumentales en chamboulant l'écriture et les pratiques compositionnelles, par exemple en dépassant la notion de *timeline* et en construisant la composition comme le résultat d'un programme.

L'analyse présentée dans ce chapitre se poursuivra aux chapitres suivants par une introduction au langage Antescofo (chapitre III) et à SuperCollider (chapitre IV). Nous aurons alors tous les éléments nécessaires pour exposer en détail la librairie AntesCollider au chapitre V.

II.2. De l'instrument à l'outil informatique : entre exploration, création, programmation et composition

La technologie a toujours été liée au processus de fabrication d'instruments, et l'instrument de son côté a influencé la création et la pensée musicales. Les œuvres musicales pour piano de la période romantique, avec leur virtuosité et leurs sonorités,

[46] Voir cependant la thèse de Pierre Donat-Bouillud, « Models, Analysis and Execution of Audio Graphs in Interactive Multimedia Systems », soutenue en décembre 2019 et qui porte sur l'intégration des traitements audio dans Antescofo. Les propositions de cette thèse ne sont cependant pas suffisamment matures pour avoir été intégrées dans le langage.

n'auraient sans doute pas vu le jour sans l'invention et l'évolution du piano à leur époque. La musique des compositeurs romantiques comme Beethoven ou Chopin est écrite avec et pour le piano, instrument très évolué et raffiné au niveau technologique. Cette musique est conçue pour être jouée par un pianiste expert qui a étudié son instrument pendant des années grâce à une pratique et une étude traditionnelle de sa technique. L'instrument, lui, continue à exister et à être fabriqué grâce à sa popularité, ce qui lui a permis de devenir un instrument stable et toujours utilisé dans différents genres musicaux jusqu'à nos jours. On voit bien que l'outil ou l'instrument est ainsi indispensable et inséparable de la musique qu'il produit.

Aujourd'hui, les nouveaux outils informatiques ont complètement révolutionné le rapport au son et à la composition tous styles confondus. À partir des premières expériences audionumériques, de la famille de logiciels Music-N aux Bells Laboratories en passant par les logiciels d'édition comme ProTools (première version en 1989), les compositeurs en quête de nouvelles formes et de nouvelles sonorités se sont très rapidement appropriés et ont intégré ces outils dans leur espace de travail et de pensée compositionnelle. Dans le même temps, ces outils n'ont cessé d'évoluer et sont passés très rapidement d'un paradigme analogique (instruments acoustiques, bande et synthétiseurs électroniques analogiques) au paradigme numérique (les DAW, stations audionumériques et logiciels de synthèse sonore et de composition assistée par ordinateur, CAO).

Parmi ces systèmes, ceux qui sont conçus pour que l'utilisateur puisse bénéficier d'une plus grande flexibilité, modularité et programmabilité permettent aujourd'hui à de nombreux compositeurs et réalisateurs en informatique musicale de programmer leurs propres environnements et instruments audionumériques. *Cette programmabilité est fondamentale* puisqu'elle permet la rétroaction entre l'espace de pensée compositionnelle et la lutherie numérique (création d'instruments, des interactions, des morphologies…).

Le nombre d'outils commerciaux ou de recherche proposés aujourd'hui rend impossible de tous les explorer et de les maîtriser dans le temps d'une vie humaine. Ce point est intéressant du fait que pour composer ou faire de la musique aujourd'hui, il faut tout d'abord avoir une idée forte de ce que l'on veut faire mais aussi de ce que l'on peut faire. Peu importe le style musical, nous sommes aujourd'hui tous confrontés aux outils numériques et il faut savoir quels seront parmi tous ces écosystèmes de logiciels musicaux ceux qui seront les plus adaptés à la pensée musicale qu'on veut exprimer.

II.3. De l'outil informatique fermé à l'outil programmable

Avec l'avènement de l'ordinateur dans le monde musical de la deuxième partie du XX^{e} siècle, nous avons vu émerger différentes musiques qui se sont adaptées aux outils dont elles proviennent.

Ceci n'a pas conduit les outils à se stabiliser, mais au contraire ils n'ont cessé d'évoluer en ayant recours aux techniques informatiques issues des recherches les plus actuelles. Par exemple, le Machine Learning et l'intelligence artificielle (IA) contribuent depuis

déjà des années à la création musicale, notamment dans des centres de recherche comme l'Ircam [47] ou ses équivalents au niveau international. Les grandes multinationales du numérique s'intéressent aussi à ces nouvelles méthodes permettant de générer du son et faire de la musique, comme le projet Magenta[48] de Google. Dans ce domaine, on trouve des applications avec un grand potentiel dans le domaine de la cocréation et d'autres relevant de la production musicale de masse. Ces nouveaux assistants visent principalement le *casual musician*[49] en simplifiant la production musicale dans un domaine limité mais significatif. Ils favorisent le workflow créatif en offrant une certaine automatisation, au prix cependant d'une grande spécialisation, par exemple en restreignant de manière importante le domaine d'application. Ces outils risquent d'amener le son et la musique vers une standardisation : l'assistant va proposer des mix jugés « bons » (c'est par exemple le cas des plug-ins qui reposent sur le Machine Learning comme iZotope[50] et autres) en fonction de standards qui, à leur tour, sont renforcés par les outils technologiques eux-mêmes. Ces systèmes sont donc particulièrement fermés et, dans la plupart de ces nouvelles techniques, la notion de composition et de génération de structures temporelles à plus grande échelle est une dimension qui manque cruellement.

Les stations audionumériques et les logiciels de composition musicale par *drag & drop*[51] sur une *timeline* fixe sont très répandus de par leur simplicité d'utilisation et offrent un modèle plus ouvert. Avec ces systèmes, il est possible de faire de la musique en quelques minutes sans aucun besoin d'avoir étudié la musique et le solfège et sans aucune notion compositionnelle. Ceci n'est pas une critique en soit, chacun fera la musique qui lui plaît et avec les outils qu'il voudra, mais personnellement, c'est une voie qui m'intéresse peu. Ces logiciels souffrent de sérieuses contraintes même s'ils possèdent un grand nombre de possibilités au niveau timbrique avec des centaines, voire des milliers, de sonorités et de préréglages (*presets*) tous prêts à l'emploi. Au niveau temporel, ils ont été conçus initialement pour faire des genres particuliers de musiques, comme celles basées sur le *groove*/*loops*. Et par conséquent, bien que les musiques réalisées avec ces outils soient vastes et variées, elles restent toujours dans un cadre déterminé, souvent liées à une « sonorité » déterminée, dans un contexte fixé, et réglées par une horloge fixe et un curseur temporel global. Il est très difficile d'exprimer des relations temporelles complexes, par exemple au niveau rythmique, ou des changements de paramètres basés sur d'autres concepts que la linéarité et des grilles prédéfinies (que ces grilles soient temporelles ou bien qu'elles adressent

[47] On peut citer le renouvellement des transformations vocales apportées par l'apprentissage profond, ou de nouveaux outils créatifs come ceux développés dans le projet ACIDS : https://acids.ircam.fr
[48] https://magenta.tensorflow.org
[49] Le terme *casual creator* a été introduit par Katherine Compton dans sa thèse « Casual Creators: Defining a Genre of Autotelic Creativity Support Systems » (2019).
[50] https://www.izotope.com/en/learn/speed-up-your-workflow-with-assistive-audio-technology.html
[51] Le glisser-déposer en français, dans une interface graphique.

d'autres paramètres comme la hauteur). Par exemple, l'expression de la polytemporalité et de la synchronisation avec un performer sont inaccessibles dans ces environnements. On va créer et composer dans un cadre prédéfini, limité et structuré (même s'il est vrai que ces limites sont repoussées avec des systèmes de plus en plus élaborés tels que Ableton Live[52] et l'intégration de Max for Live[53] par exemple).

Si l'on souhaite sortir de ce cadre prédéfini, ces logiciels ne sont tout simplement pas adaptés et il faut chercher d'autres outils, sans doute plus complexes à utiliser mais plus ouverts et flexibles. Pour « sortir du cadre » et disposer de plus de flexibilité, il faut nécessairement adresser plus directement la machine, c'est-à-dire rentrer dans la programmation.

Il y a tout de même une alternative pour les compositeurs qui souhaitent « sortir du cadre » et qui n'ont pas ou ne veulent pas rentrer dans la programmation : c'est de recourir à un expert en informatique musicale afin de lui déléguer la réalisation de leurs parties électroniques (ou même des parties instrumentales dans le cas de la CAO ou de l'aide à l'orchestration). Dans cette approche, souvent pratiquée par les institutions de création musicale partout dans le monde, le réalisateur en informatique musicale[54] (RIM [Zat13], « *computer music designer* » ou « *computer music performer* » en anglais) dispensera son expertise pour orienter le compositeur et lui apporter des idées et solutions techniques, par exemple des traitements, logiciels, systèmes de programmation, etc. Il ne faut cependant pas oblitérer le fait que la pensée musicale se nourrit aussi de la maîtrise de l'outil informatique, tout comme la composition instrumentale « la plus classique » intègre les contraintes spécifiques de chaque instrument. Ainsi, le compositeur qui souhaite utiliser une technologie spécifique doit « s'approprier » cette technologie (ou au moins ces résultats sonores) pour créer ses pièces.

En tant que compositeur et réalisateur en informatique musicale, et dans la continuité des musiques contemporaines et exploratoires, je me suis intéressé non seulement à l'interaction et au dialogue entre le monde instrumental et électronique mais aussi à comment et avec quels outils réaliser ces instruments et ces interactions. Comment avoir des systèmes performants, plus flexibles pour pouvoir exprimer des idées musicales novatrices en relation directe avec la pensée musicale ? Bien que nous soyons encore bien loin de l'utopie de composer directement avec la pensée, peut-on se rapprocher de systèmes qui donnent plus de liberté et de possibilités au niveau de la manipulation sonore et musicale et qui permettent d'incarner plus directement l'intention musicale dans la tête du compositeur ? Comment créer des outils et instruments qui puissent en même temps stimuler l'espace de pensée et de créativité musicale à travers l'exploration des possibilités offertes par les nouvelles technologies

[52] https://www.ableton.com
[53] https://www.ableton.com/fr/live/max-for-live/
[54] Laura Zattra, « Les origines du nom de RIM (Réalisateur en informatique musicale) », Journées d'informatique musicale (JIM 2013), Saint-Denis, 2013, p. 113-120.

disponibles ? Pourquoi ne pas chercher aussi d'autres façons de faire avec d'autres outils et paradigmes ?

Ce travail d'exploration entamé depuis des années converge vers l'utilisation des logiciels et langages de programmations spécifiques pour la composition, la synthèse et traitement du son qui se sont à la fois stabilisés sur certains concepts qui ont prouvé leur efficacité (par exemple la notion de patch) et étendus depuis leur création pour offrir et explorer de nouvelles possibilités et fonctionnalités. On peut citer par exemple Max/Pd et SuperCollider qui ont émergé comme des outils incontournables.

Pour la composition et construction des structures temporelles de plus haut niveau, mon cheminement personnel m'a rapproché dans un premier temps de systèmes d'aide à la composition tels que OpenMusic et, plus tard, avec la nécessité d'une intégration dans un système temps réel, du langage Antescofo. Ces recherches ont été réalisées avec la volonté d'un renouvellement des rapports à l'outil et pour tirer parti des possibilités inédites offertes par les développements informatiques actuels dans la création des musiques interactives, de leur interaction et de leur synchronisation avec d'autres médias temporels.

Le système auquel j'ai abouti dépasse la notion de boîte à outils technique pour devenir un instrument à part entière dans mon imaginaire et mon espace de pensée, apportant une plus grande liberté et souplesse à ma musique. L'ordinateur et le programme informatique deviennent ici un instrument d'aide à la création et à l'expérimentation avec cette rétroaction qui permet aussi des trouver des idées qui n'auraient pas pu voir le jour sans cet aller-retour constant entre exploration, création, programmation et composition. Comme l'écrit Yann Orlarey :

> *« L'ordinateur rentre dans cette boucle, entre le cogitare, la circulation des idées, la production d'idées nouvelles, et l'intellegere le fait de choisir et de reconnaître les idées intéressantes… Le langage de programmation est extrêmement important dans cette description-là. Il y a une qualité, une ergonomie cognitive qui fait que c'est plus ou moins facile de penser les problèmes. Il nous donne une grille de lecture du monde d'une certaine manière, comme la notation musicale. »* [Pot09]

II.4. Les systèmes de programmation musicale temps réel et leurs limites

Aujourd'hui, il y a pléthore d'instruments, de systèmes de programmation et d'exemples de synthèses et traitements de sons. Ces systèmes mettent en œuvre des technologies de plus en plus élaborées et en pleine expansion comme le Machine Learning et l'IA. Cependant, on peut dire qu'il y a beaucoup de sons mais pas assez de musique. Plus précisément, il y a peu de systèmes qui permettent de structurer et manipuler les sons dans un discours plus vaste comme c'est le cas d'une composition avec une écriture fine de l'électronique et de l'interaction.

Les systèmes qui adressent ce problème relèvent de deux catégories :

- d'un côté, il y a des systèmes d'aide à la composition comme OpenMusic [Ago98]. Très performants et utilisés depuis des années par beaucoup des compositeurs, ces systèmes restent dans le domaine de la génération du matériel en temps différé et ne prennent pas en compte l'aspect performatif ;
- de l'autre, il y a des systèmes temps réel comme Max ou Pd qui sont très performants au niveau interaction temps réel mais qui n'ont pas une véritable notion de structure et d'organisation temporelle.

Nos projets musicaux nous ont amenés à nous focaliser sur la seconde catégorie des systèmes de programmation pour la musique. En effet, les systèmes de CAO sont utilisés « hors temps » et ne permettent pas de concevoir, de réaliser et de contrôler en temps réel des interactions musicales, lors de la performance.

Des outils largement utilisés comme Pd [Puc96] ou Max et son extension audio MSP [Zic98] ont démontré leur flexibilité et leur adaptabilité à un grand nombre de projets interactifs à travers une notion intuitive de « *patchs* » qui sont des programmes à flot de données (*dataflow*) rappelant la manière de configurer les premiers synthétiseurs analogiques à l'aide de connexions. Cette façon de programmer, ou de « *patcher* » dans le vocabulaire de la communauté, a largement contribué à la popularité de ces logiciels par son modèle visuel de programmation, sa simplicité, et la rapidité du prototypage. Ces qualités ont permis de faciliter l'apprentissage de ces environnements, surtout pour les étudiants en composition et plus généralement les musiciens.

Ces systèmes ont démontré leur efficacité et leur pérennité : Max est un logiciel qui existe depuis plus de 30 ans, qui continue à être maintenu et utilisé par une grande communauté artistique internationale non seulement dans le domaine de la musique et du son mais aussi dans le domaine des arts plastiques, de la vidéo, de la scénographie, de la danse, etc.

Ces logiciels ont cependant leurs limites quand il s'agit de la gestion du temps, de la polyphonie, des ressources de calcul. Ils manquent en particulier de dynamisme et de flexibilité dans la construction et le contrôles des processus et des chaînes de traitements. Des solutions existent mais elles sont ad hoc, peu naturelles et pas toujours efficaces (on peut penser à l'objet *poly~* de Max par exemple). Ces systèmes à flots de données, construits sur l'analogie du circuit électrique, sont en effet des systèmes statiques : quand on programme un patch, il est fixe et on ne va pas le changer, le remplacer ou le modifier lors de la performance ou au cours de son exécution.

Un autre défaut est une gestion temporelle globale et locale des *patchs* qui reste trop élémentaire. Par exemple, l'objet *transport*[55] pour le contrôle d'un tempo global et introduit dans Max 5 ne permet pas l'expression d'une multitemporalité et n'est pas adapté à des musiques avec une composante rythmique complexe.

[55] https://docs.cycling74.com/max5/refpages/max-ref/transport.html

Le modèle à flot de données sous-jacent à Max et à PD se focalise sur les traitements sans représentation explicite de leur occurrence temporelle comme cela serait le cas avec une représentation de type « *timeline* ». Il n'existe donc pas de notion explicite de partition électronique dans ces systèmes. La notion de *cue-list* a cependant été introduite : il s'agit d'une série d'instructions discrètes envoyées à des dates données par des « boîtes de messages » ou des *presets* (l'objet *pattstorage* introduit à partir de Max 6 a été une grande avancée). Ces *cue-list* ne permettent pas d'exprimer une itération ou bien une exécution conditionnelle. Le calcul des dates reste aussi trop limité.

Des systèmes de *timeline* plus sophistiqués ont été introduits ultérieurement dans Max comme les objets externes rs.delos[56] ou note~ (Université de musique de Bâle), tous les deux abandonnés depuis, ou plus récemment l'objet bach.roll et bach.score dans la librairie Bach [AG12]. Ce dernier système intègre des fonctionnalités de CAO (aide à la composition) en temps réel dans Max, en important des structures de données et des traitements spécifique de la notation symbolique. Ce système reste cependant mal adapté pour gérer des systèmes plus complexes de musiques interactives génératives avec beaucoup de traitements et d'interactions en temps réel.

Il faut mentionner aussi des alternatives externes de contrôle comme les séquenceurs Vezér[57], Ossia[58], Iannix[59] ou encore des alternatives intégrées plus sophistiquées de contrôle, synthèse et composition comme Usine[60]. Ces systèmes plus ou moins souples et sophistiqués offrent de meilleures possibilités de contrôle, d'interaction et de bonnes performances mais ils ne permettent pas une écriture fine des événements électroniques pour les musiques mixtes. Par exemple, ils ne supportent tous qu'une notion de *timeline* mono-curseur (à l'exception de Iannix qui permet un contrôle multicurseur mais qui reste statique, les *timelines* étant fixées à l'avance).

II.5. Dépasser le modèle visuel

Les systèmes cités reposent le plus souvent sur des langages de programmation visuelle. Les langages visuels ont souvent été proposés car ils sont censés simplifier la programmation et l'accès aux ressources informatiques [GrPe96] [LaSi87], par exemple en limitant les erreurs syntaxiques et en visualisant explicitement le cheminement et les transformations des données. Ils contribuent aussi à un mode de travail plus interactif et incrémental entre l'utilisateur et les programmes. Si ces propriétés ont pu être validées, notamment dans le domaine de la composition assistée par ordinateur, où cette interactivité améliore considérablement le potentiel créatif des outils informatiques [AgAsBr06] [Ass98], ainsi que dans les applications de traitement

[56] http://arts.lu/roby/index.php/site/maxmsp/rs_delos
[57] https://imimot.com/vezer/
[58] https://ossia.io
[59] https://www.iannix.org/fr/whatisiannix/
[60] https://www.brainmodular.com

du signal, elles sont discutables dans d'autres contextes [Whi97-1] [Whi97-2] [GrPe92].

En particulier, la programmation visuelle peine à rendre compte simplement des structures itératives, rend difficile l'expression de traitements algorithmiques complexes ou la gestion fine de processus dynamiques (récursion, concurrence, interruptibilité, arrêt). Ces difficultés sont bien connues [AgGhGi19] et ont amené par exemple à intégrer récemment un langage textuel, Bell [GIAG19], au sein de bach. Cependant, si bach étendu par Bell est très performant pour générer des structures et du matériel symbolique en temps réel ainsi que pour une écriture de l'électronique, il n'est pas conçu pour gérer des tâches dynamiques complexes avec un nombre élevé de traitements, de synthèses et d'interactions en parallèle comme c'est le cas d'une performance interactive entre musiciens, électronique et autres médias temporels.

III. Antescofo, un langage de programmation pour écrire le temps musical

Ce chapitre propose une présentation générale du système Antescofo. Ce n'est pas un manuel utilisateur ou un manuel de référence : l'objectif est de montrer comment il répond à certaines problématiques compositionnelles discutées précédemment. Il s'agit aussi de donner les éléments qui permettent la compréhension des développements de la librairie AntesCollider et les exemples présentés à l'occasion de la description des œuvres.

Les capacités d'écoute du système sont peu utilisées dans le travail présenté ici et nous nous concentrons sur la partie langage. Nous utilisons d'ailleurs le terme Antescofo pour désigner indifféremment le système ou bien son langage de programmation.

Ce chapitre débute par une description succincte de mon travail avec cet outil. Il se poursuit (section 2) par une présentation du langage et du contexte qui a donné lieu à ce projet. Nous situons le langage dans la problématique du suivi de partition, des systèmes temps réel et des langages de programmation.

Nous abordons ensuite plusieurs notions spécifiques à Antescofo qui présentent un intérêt particulier dans le cadre des applications à la musique. Nous proposons de tirer parti de la notion de processus et d'acteur pour structurer les partitions électroniques (section 3).

Antescofo est un langage qui offre un modèle du temps adapté aux processus musicaux. Ce modèle du temps permet la simultanéité en tirant parti de notions développées dans le domaine des langages temps réel synchrones (section 4).

Antescofo est un langage réactif et la section 5 introduit une structure spécifique, le *whenever*, qui permet de réagir à des événements arbitraires. C'est aussi un langage temporisé dans lequel le temps s'écoule. La section 6 introduit la notion de *curve* différentielle qui permet de traiter des modèles physiques dans un temps continu.

Les sections 7 et 8 présentent respectivement quelques mécanismes d'exécution utile pour préparer une performance et quelques fonctions de la bibliothèque Antescofo qui sont particulièrement utilisées dans mes pièces.

Pour finir, je présenterai quelques réflexions sur la possibilité d'une représentation graphique des programmes. En effet, Antescofo a fait le choix d'une représentation textuelle pour pallier plusieurs difficultés inhérentes aux approches visuelles. Mais les langages visuels ont des avantages qu'on ne peut négliger en termes de lisibilité et de facilité d'utilisation. J'esquisse une piste possible pour tirer parti des deux représentations.

III.1 Introduction

C'est à partir du système Antescofo et de son langage de programmation que nous avons développé notre notion de partition centralisée. Nous avons utilisé Antescofo dans plusieurs créations à l'Ircam à partir de 2010 avec la réalisation de la partie informatique de *Einspielung I* d'Emmanuel Nunes pour violon et électronique (Diego Tosi, violon). Dans cette œuvre, on a utilisé pour la première fois le suivi de partition avec des traitements note à note pour piloter le logiciel Max. Par la suite, nous avons réalisé la partie électronique de *Raggi di stringhe* de Lara Morciano (Hae-Sun Kang au violon) où on a utilisé le suivi de partition et le calcul du tempo en temps réel avec des déformations temporelles (accelerando et rallentando) pour moduler des traitements électroniques. Cette pièce riche et vertueuse aussi bien au niveau électronique qu'instrumental a été lauréate du prix Giga Hertz Awards du ZKM/Experimentalstudio en Allemagne.

Antescofo est un système en évolution continuelle. Plusieurs de mes pièces ou de mes réalisations électroniques ont motivé certains développements du langage ou bien ont permis de valider de nouvelles constructions. Ainsi, la réalisation de *Iki-no-michi* pour saxophones de Ichiro Nodaira, a été la première à utiliser les « *curves* » et les variables dans le langage Antescofo. Nous avons intégré ces *curves* pour faire des glissandi et des gels (*freezes*) des notes des saxophones pour créer des couches qui se superposent au jeu instrumental. Le langage a évolué rapidement vers un système complet de programmation avec l'arrivée de Jean-Louis Giavitto (en 2011), le travail de thèse de José Echeveste (2011-2015) et la création du projet Mutant à l'Ircam.

Après ces premières expériences, une résidence en recherche artistique à l'Ircam en 2016 m'a permis de me familiariser encore plus avec Antescofo et de participer à l'extension de la librairie, en particulier pour la manipulation de tableaux (la librairie native Antescofo inclut ainsi les mêmes fonctions sur les tableaux qu'on peut trouver dans SuperCollider). Cette résidence m'a permis aussi d'expérimenter les premiers couplages à SuperCollider. Dans cette approche initiale, le couplage visait le langage de SuperCollider : *sclang*. On verra au chapitre V que la version actuelle de AntesCollider vise directement le pilotage du serveur scsynth et son développement coïncide avec le début de ce travail de thèse.

III.2. Le langage Antescofo

Antescofo est né à partir du travail entre le compositeur Marco Stroppa et le chercheur Arshia Cont pour la création de la pièce *Off silence* pour saxophone alto et électronique en temps réel (œuvre créée à Shizuoka par Claude Delangle). Il se présente sous la forme d'un objet Max ou Pd.

Contexte

Antescofo se présente comme un système de *suivi de partition*. On définit traditionnellement le suivi de partition comme l'alignement automatique d'un flux audio correspondant à un ou plusieurs musiciens sur une partition symbolique.

Dannenberg et Vercoe ont été dans les années 1980 les premiers à proposer un système de suivi de partition [Dan84a, Ver84]. Les entrées sont symboliques (protocole MIDI) avant d'être étendues par la suite à des entrées audio.

À la différence des systèmes précédents de suivi de partition, Arshia Cont propose pour la première fois (en 2007) de coupler une machine d'écoute avec un langage permettant de décrire les séquences d'actions dans le temps de la partition. L'originalité de ce couplage est de permettre l'écriture du déroulement temporel de processus électroniques, en coordination avec des instrumentistes, sans pour autant figer le temps de ces processus.

En effet, le système prend en compte la dimension temporelle du suivi, c'est-à-dire qu'il infère le tempo auquel l'instrumentiste joue à partir des événements musicaux suivis, et utilise ce tempo dans le séquencement des actions électroniques. Ceci permet d'avoir une électronique « adaptative ou élastique » qui va se synchroniser à l'interprète d'une façon dynamique. À la différence d'autres systèmes comme les *cue-list* ou les boîtes de messages Max, la partie électronique ne sera plus seulement un accompagnement rigide de l'interprète avec des événements discrets à la temporalité fixée *a priori*, mais pourra interagir d'une manière plus vivante, réactive et expressive grâce au langage.

En suivant la partie du musicien, le système est capable d'exécuter les commandes de l'électronique en s'adaptant aux variations de l'interprétation. La musique électronique peut désormais être interprétée et le compositeur peut écrire les actions électroniques dans un temps musical partagé et synchronisé avec celui d'un musicien, d'un performeur, avec d'autres médias ou comme un méta-séquenceur sophistiqué, aussi bien pendant le processus de création que pendant le concert, la performance ou l'installation sonore.

Le système est suffisamment mature pour avoir donné lieu à la création d'une *startup* qui commercialise l'application Metronaut[61] pour l'accompagnement automatique des instruments traditionnels. La partie langage continue d'être développée par Jean-Louis Giavitto et les objets Max et PureData restent disponibles librement pour les applications artistiques.

Le reste de cette section décrit les caractéristiques du langage de programmation de plusieurs points de vue. Notre travail n'implique pas la machine d'écoute, ce qui explique notre focalisation sur la partie langage. Afin de mieux situer les apports spécifiques du langage, nous esquisserons les liens entre Antescofo et les systèmes temps réel, le parallélisme, la programmation impérative et les *DSL* (*domain specific language*).

[61] https://metronautapp.com/fr

Antescofo et les systèmes temps réel

Antescofo est un système temps réel qui couple un système de suivi de partition et un langage de programmation dédié :

- Le langage est utilisé pour l'écriture de pièces musicales impliquant des musiciens humains sur scène et des processus informatiques en temps réel.
- Une machine d'écoute permet de détecter les événements musicaux dans un flux audio. Ces événements sont communiqués au système temps réel qui assure l'exécution et la synchronisation des actions électroniques avec la performance instrumentale malgré les erreurs d'écoute ou d'exécution.
- Ce système est un *système réactif* conçu pour l'interaction : on parle de système événementiel (ou *event-driven*) qui réagit à l'occurrence d'événements (internes ou externes au système).
- C'est aussi un *système temporisé* (ou *time-driven*) avec un (ou plusieurs) tempo et une synchronisation continue : le système est cadencé par l'écoulement de délai arbitraire et par des activités périodiques.
- Le système Antescofo peut être vu comme un « système cyber-physique avec humain dans la boucle[62] » puisqu'il interagit avec le monde extérieur comme le musicien ou d'autres médias qui composent ensemble un réseau d'interactions et de synchronisations complexes, hétérogènes et multitemporelles.
- Le langage Antescofo est par ailleurs un langage de programmation généraliste, qui pourrait servir à programmer et contrôler d'autres événements ou médias temporels que la musique.

Le langage de programmation d'Antescofo a la particularité d'inclure à la fois la description des événements musicaux à détecter dans l'entrée audio (ce qui correspond à la partition de l'interprète humain que la machine d'écoute va suivre) et la spécification des actions électroniques.

Le langage traite les événements et l'écoulement du temps, ce qui permet de réaliser des séquencements sophistiqués dans une performance et permet au compositeur de manipuler et transformer les différents temps impliqués par les actions électroniques d'une façon souple et expressive.

Cette expressivité étend et renouvelle en permanence l'espace de pensée compositionnelle. L'électronique et son écriture deviennent une entité à part entière et son caractère n'est plus le résultat d'une pensée nettement instrumentale sujette à un suivi et un accompagnement mais peut se penser en termes d'interactions variables entre des processus autonomes. On verra d'ailleurs dans des exemples qui seront

[62] Cette notion a été popularisée sous le nom anglais de « *human in the loop cyber-physical system* » dans la communauté de contrôle et de l'automatique, voir par exemple Nunes D., Silva J. S. et Boavida F. (2018), *A Practical Introduction to Human-in-the-loop Cyber-physical Systems*, Wiley.

présentés plus tard, des réalisations purement électroacoustiques qui n'impliquent aucun suivi.

Un langage parallèle

Une des principales caractéristiques du langage Antescofo, par rapport aux langages classiques de programmation où tout est en séquence, est son parallélisme implicite : il permet en effet d'exécuter plusieurs processus en parallèle comme dans une partition d'orchestre.

Ce parallélisme va permettre non seulement d'exécuter plusieurs processus mais aussi de les synchroniser entre eux, ce qui est fondamental non seulement pour la synchronisation des événements de l'électronique dans des pièces musicales interactives mais aussi pour synchroniser plusieurs médias temporels comme la vidéo, les lumières, des robots… Cette synchronisation se réalise par exemple avec le suivi de partition, sa machine d'écoute et ses différentes stratégies de synchronisation (voir la thèse de José M. Echeveste [Ech15]) ainsi que du modèle probabiliste de suivi temporel dans la machine d'écoute (thèse de Philippe Cuvillier [Cuv16]).

Mais Antescofo a aussi été utilisé pour se synchroniser avec des données issues de captations gestuelles, par exemple le suivi gestuel d'un chef [Lem19] ou avec des processus extérieurs arbitraires grâce à un mécanisme générique, les variables temporelles [63], qui utilisent le même algorithme que la machine d'écoute pour l'inférence d'un tempo et sur lesquelles on peut se synchroniser comme on se synchronise sur le musicien.

Un langage impératif généraliste

Antescofo est un langage de programmation *impératif* avec par exemple une notion de variable tout à fait similaire aux variables en C, python ou Java, mais le langage est non typé (les vérifications de type se font au vol, à l'exécution). Le style impératif permet de développer des programmes efficaces.

Mais Antescofo présente aussi de forts *traits déclaratifs*, c'est-à-dire permettant de décrire ce qui est produit plutôt que la manière de le produire. Parmi les traits déclaratifs, il y la spécification des événements musicaux à suivre, définissant une *timeline* explicite. Plusieurs structures de contrôle de haut niveau sont aussi de nature déclarative, comme les courbes ou bien le *whenever* qui définit une règle qui déclenche un processus à partir d'une condition logique arbitraire. Antescofo offre aussi des fonctions de processus qui sont des valeurs de premier ordre : on peut les passer en arguments pour paramétrer d'autres fonctions ou processus (fonction d'ordre supérieur) ce qui permet un style de programmation fonctionnel.

Comme il est classique dans le monde fonctionnel, les données sont allouées dynamiquement. La gestion de la mémoire est implicite : le programmeur n'a pas à

[63] Voir dans la documentation la section consacrée aux @tempovar.

gérer les allocations ni les libérations mémoire. Mais Antescofo utilise des compteurs de référence pour éviter les interruptions dues à un ramasse-miettes (*garbage collector)* qui serait trop intrusif dans un contexte temps réel.

Dans un contexte musical, Antescofo adresse plus précisément la création des formes et le contrôle temporel fin des événements électroniques. Il n'inclut pas par lui-même un système de synthèses et de traitements audio car le langage de programmation est dédié à la création des structures temporelles génératives et interactives. Cependant, des intégrations de moteurs de rendu audio dans le langage Antescofo ont déjà vu le jour, notamment le prototype intégrant le langage de synthèse et traitement FAUST dans Antescofo [DGCNO16]. La librairie AntesCollider [FEGIDO19], qui sera décrite en détail dans le chapitre V, est un autre exemple de couplage à un moteur audio.

Antescofo comme langage dédié

Antescofo est un DSL (*domain specific language*) dédié à un domaine d'application précis avec l'introduction dans le langage de mécanismes spécifiques pour la musique et la gestion temporelle. Mais ces mécanismes sont traités comme des mécanismes généralistes uniformes dans le langage.

Par exemple, les NIM[64] (*New Interpolated Map*), qui permettent de représenter des fonctions définies par morceaux entre des *breakpoints*, sont des valeurs du langage au même titre que les tableaux où les chaînes de caractères. Les courbes sont des structures de contrôle permettant d'échantillonner des NIM ou bien des fonctions définies par un système différentiel[65] (ODEs) et elles ont le même statut qu'une boucle. On va ainsi pouvoir traiter les NIM comme n'importe quelle autre valeur du programme, ce qui permet de les manipuler facilement, les mettre dans des tableaux ou des dictionnaires, les passer en argument, les créer dynamiquement, les jouer et les rejouer à un moment donné… comme tous les autres types de valeurs manipulées dans le langage.

Cette capacité à représenter des processus et des traitements complexes par des données est très importante puisqu'elle permet de traiter ces différents processus comme des éléments abstraits dans l'espace de pensée à l'instar de la notation symbolique traditionnelle (grammatologie). On pourra ainsi les composer, les manipuler et les faire interagir dans le temps avec les structures de contrôle et de données propres au langage Antescofo. Par exemple, dans la pièce *Hypersphères*, les données de captation gestuelle (plusieurs flux en parallèle) sont enregistrées dans des NIM, ce qui correspond à une transformation d'une action dans le temps vers une valeur en mémoire. Cette valeur en mémoire de la captation gestuelle peut être par la suite traitée algébriquement, par exemple pour la simplifier, la renverser, la compresser ou la dilater et l'utiliser pour contrôler n'importe quel paramètre de l'électronique.

[64] http://antescofo-doc.ircam.fr/Reference/data-nim/
[65] http://antescofo-doc.ircam.fr/Reference/compound_ode/#differential-curve-experimental

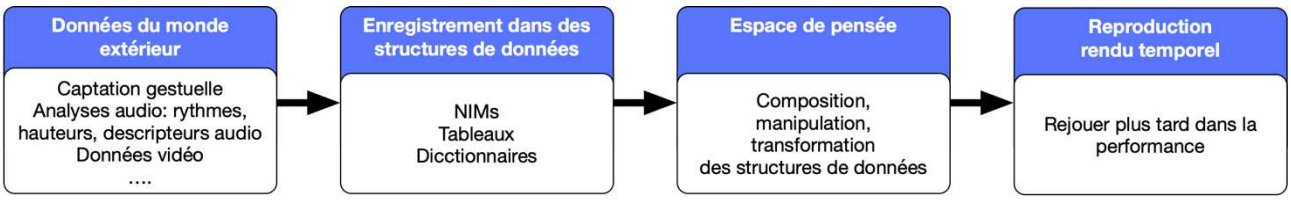


Fig. 3.1 Schéma de la transformation d'événements temporels en données stockées dans la mémoire de l'ordinateur. Les données sont enregistrées dans des structures de données qui par la suite sont incorporées dans l'espace de pensée et manipulées pour reproduire des nouveaux événements musicaux qui seront rejoués après dans le temps musical de la performance.

Contrairement à un programme classique qui s'exécute du début à la fin, un programme Antescofo permet de déclencher et activer des parties du programme non seulement de façon séquentielle mais aussi en sautant d'un endroit à un autre de la partition. Cette possibilité d'exécuter un programme à partir d'un instant arbitraire est fondamentale, par exemple pendant les répétitions des pièces mixtes ou pendant la composition d'une pièce purement électronique. Cette fonctionnalité repose sur plusieurs types de fonctions de *transport*[66] qui vont permettre de naviguer dans la partition en déclenchant ou pas ce qui a précédé.

III.3. Structuration des actions : les processus et les acteurs

La structuration dans l'écriture de l'électronique est primordiale. À la différence de l'écriture instrumentale qui s'adresse à un humain avec un instrument acoustique réalisé a priori par un luthier, l'électronique s'adresse à un ordinateur générique qui a besoin d'avoir explicitement tous les niveaux d'information, de la programmation des instruments, synthèses et traitements sonores jusqu'à la macro-forme, nécessaires pour générer des événements musicaux. En conséquence, la quantité d'information nécessaire pour décrire des événements musicaux dans une partition électronique est largement plus volumineuse que pour une partition instrumentale traditionnelle, et nécessite impérativement une structuration pour faciliter cette description.

L'approche adoptée par Antescofo est de se reposer sur les constructions informatiques utilisées pour structurer le programme pour structurer les actions musicales, comme jouer des rythmes avec un son synthétique ou interagir avec un performeur ou d'autres médias en temps réel.

Des structures de contrôles informatiques telles que les *processus*, les boucles (*loops*), les règles de déclenchement (*whenever*) ou les blocs d'actions (*group*) ont une correspondance directe avec des actions musicales et offrent les structures de base qui vont nous permettre de construire toutes les actions électroniques et de les manipuler d'une façon abstraite.

[66] http://antescofo-doc.ircam.fr/Reference/tempo_transport/#transport

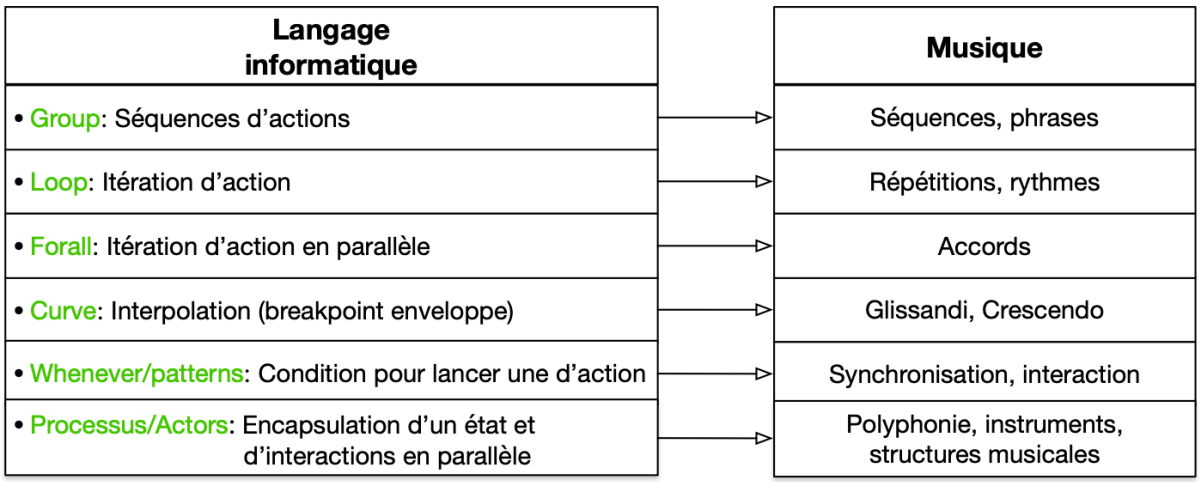


Fig. 3.2 Des structures de contrôle du langage avec des correspondances musicales.

Processus

La notion de processus introduite dans le langage Antescofo va permettre d'abstraire et de paramétrer une séquence d'actions et d'instancier cette séquence dynamiquement, en parallèle avec les autres traitements, autant de fois que nécessaire. Les processus sont polyphoniques, par exemple un processus de génération rythmique peut être déclenché plusieurs fois et se superposer polyphoniquement avec des arguments différents pour changer le résultat. On a ainsi plusieurs instances du même processus qui s'exécutent « en parallèle ». Ce parallélisme est « gratuit » dans le sens où il ne requiert aucune intervention particulière de la part du programmeur. Par défaut, les calculs s'effectuent en parallèle, pas en séquence.

En général, un processus s'exécute à partir d'un instant donné, repéré dans la partition à partir d'un événement détecté par la machine d'écoute. Son déclenchement peut aussi être causé par l'utilisateur ou par un événement extérieur comme le déclenchement à partir de l'analyse d'un flux de captation gestuelle. Chaque processus peut être arrêté arbitrairement (avec la commande `abort`) à n'importe quel moment dans l'exécution du programme et de la performance.

```
1   @proc_def ::my_rand_group()
2   {
3       group seq
4       {
5           // sampler_process <nom_fichierr_audio> <amplitude> <canal_sortie>
6           sampler_process "Slap1.aif" 0 (@rand_int(4)+1)
7           (1/4+@rand(0.5)) sampler_process "Slap2.aif" 0 (@rand_int(4)+1)
8           (1/8+@rand(0.5)) sampler_process "Slap3.aif" 0 (@rand_int(4)+1)
9           (1/16+@rand(0.5)) sampler_process "Slap5.aif" 0 (@rand_int(4)+1)
10      }
11  }
12
13
14  group proc_seq
15  {
16      ::my_rand_group()
17      1 ::my_rand_group()
18      1.2 ::my_rand_group()
19      0.7 ::my_rand_group()
20  }
```

Fig. 3.3 Exemple d'une définition de processus @proc_def sans arguments. Dans ce cas, il s'agit d'un processus qui contient une séquence de 4 événements pour déclencher un échantillon avec 3 rythmes avec une déviation aléatoire de 0.5 temps et un canal de sortie sélectionné aléatoire entre 0 et 4. Les lignes 14-20 spécifient un *group* qui regroupe les actions permettant de jouer une séquence de 4 instances du processus qui se tuilent suivant des délais spécifiés devant l'appel du processus.

```
1   @proc_def ::spat_circ($source, $dir, $offset, $tpo)
2   {
3       @local $g := 0.01, $lab := 0
4
5       loop 1 @tempo := $tpo
6       {
7           abort $lab
8           $lab := {curve @tempo := $tpo
9               @grain := $g
10              @action :=  {
11                      spat source $source az (($x + $offset)*$dir)
12                      }
13              { $x {   { 0  }
14                1      { 360} } }
15          }
16      }
17  }
18
19  $circ_spat1 := ::spat_circ(1, 1, 0, 30)
20  $circ_spat2 := ::spat_circ(2, -1, 90, 45)
21  $circ_spat3 := ::spat_circ(3, 1, 135, 72)
22  $circ_spat4 := ::spat_circ(4, -1, 270, 90)
23
24  10 abort $circ_spat1
25  abort $circ_spat2
26  abort $circ_spat3
27  abort $circ_spat4
```

Fig. 3.4 Exemple d'une définition de processus permettant de bouger des sources de l'Ircam Spat circulairement autour d'un hot spot. Les lignes 19-22 spécifient 4 instances du processus avec différents arguments pour contrôler 4 sources avec des directions, angle initial et vitesses différentes. Assigner la valeur retournée par le lancement d'un processus à la variable `$circ_spat1-4` permet de tuer une instance précise du processus, dans ce cas 10 temps après sa création.

Acteurs

La notion d'objet est aujourd'hui largement répandue dans les langages de programmation. Ce concept est utilisé pour faciliter l'organisation du code en rassemblant plusieurs paramètres dans un état et en rendant explicites les interactions possibles avec cet état à travers la notion de *méthode*.

Une notion connexe et moins populaire est le concept d'*acteur*. Le modèle de programmation de l'acteur a été développé au début des années 1970 avec le travail de Carl Hewitt et des langages comme Act[67]. De nos jours, plusieurs langages de programmation actuels incluent cette notion d'acteur, par exemple le langage de programmation *Ptolemy*[68] et des langages comme *Scala* ou *Erlang* qui offrent des « objets parallèles ». Alors que les objets se concentrent sur la réutilisation du code avec des mécanismes tels que l'héritage, le sous-typage de méthodes, le masquage d'état, etc., les acteurs se concentrent sur la gestion des activités concurrentes d'entités autonomes.

Antescofo permet de définir des acteurs spécifiant des entités qui encapsulent un état et fournissent des interactions simultanées, parallèles et temporisées avec cet état. Antescofo utilise aussi le terme d'objet pour désigner ces entités, même s'il n'y a pas de notion comme l'héritage.

Une définition d'acteur peut être instanciée et des méthodes peuvent être appelées sur cette instance. Il y a plusieurs sortes de méthodes :

- la première correspond à des fonctions et réalise des calculs instantanés ;
- le second type de méthodes correspond à des processus dont l'exécution peut durer dans le temps. Ces processus peuvent être vus comme des *timelines* indépendantes contrôlées par l'acteur sur lequel on a lancé la méthode ;
- une méthode peut aussi correspondre à un traitement exécuté simultanément sur toutes les instances d'un acteur donné (par exemple, à des fins de synchronisation) ;
- certaines méthodes ne sont pas déclenchées par un appel explicite mais sont activées de manière reflexe lorsqu'une expression logique devient vraie ;

[67] C. Hewitt, "Viewing control structures as patterns of passing messages", *Artificial intelligence*, vol. 8, n° 3, 1977, p. 323-364.
[68] J. T. Buck, S. Ha, E. A. Lee and D. G. Messerschmitt, "Ptolemy: A framework for simulating and prototyping heterogeneous systems", 1994.

- cette notion d'activité reflexe est étendue pour exécuter une méthode à la création ou à la destruction d'une instance[69].

Toutes les exécutions de méthodes sont des calculs comme les autres et peuvent être synchronisées avec le musicien ou sur une variable, elles peuvent être exécutés sur un tempo donné, etc. La concurrence entre les méthodes, les processus et tous les autres calculs est gérée implicitement et efficacement par le système d'exécution (*run-time*) d'Antescofo[70].

```
@obj_def objet(<arguments>)
{
    @local //variables locales à l'objet
    @init
    {
        // définit une séquence d'actions qui seront lancées lors de l'instanciation d'objet
    }

    @fun_def // méthode(fonction)
    {
        // corps de la fonction
    }

    @proc_def // méthode(processus), s'exécute en parallèle
    {
        // corps du pocessus avec des actions temporelles,
        // plusieurs instances de la même méthode peuvent être simultanément actives pour le même objet.
    }
    @broadcast // fonction qui s'exécute simultanément sur toutes les instances de l'objet
    {
        // action à réaliser
    }
    @whenever // crée un démon qui déclenche une séquence d'actions lorsqu'une expression logique est vraie
    {
        // action à réaliser
    }
    @react //réaction(expression).
    {
        // action à réaliser
    }
    @abort
    {
        // définit une séquence d'actions qui seront lancées lorsque l'objet est tué
    }
}
```

Fig. 3.5 Structure d'un acteur dans le langage Antescofo.

La notion d'acteur en Antescofo ne correspond pas à une nouvelle entité primitive dans le langage. Un acteur est construit au-dessus de la notion de processus. En effet, une instance de processus peut être utilisée comme une sorte d'entité autonome encapsulant certaines données. En fait, un processus en cours d'exécution peut être vu comme un objet ou comme un acteur :

[69] Cela permet par exemple de réaliser très simplement un gestionnaire d'instances, mécanisme très utile quand par exemple une instance correspond à une source sonore et qu'il faut allouer et libérer des ressources extérieures au langage dans le SPAT.

[70] J.-L. Giavitto, J.-M. Echeveste, A. Cont et P. Cuvillier, "Time, Timelines and Temporal Scopes in the Antescofo DSL v1. 0", in *International Computer Music Conference (ICMC)*, 2017.

- Une instance de processus est similaire à l'instance d'une classe : le processus est la classe et l'appel à un processus correspond à l'instanciation d'une classe.
- L'intervalle de temps entre l'appel du processus et la fin du processus correspond à la durée de vie de l'objet.
- Un processus en cours d'exécution est dénoté par une valeur de type *exe*[71] et l'exe de l'instance correspond à une référence à l'instance de l'acteur.
- L'état de l'objet correspond aux valeurs des variables locales de l'instance de processus.
- Les interactions avec l'objet peuvent être réalisées en affectant ses variables locales.
- Les messages correspondent à des méthodes, *i.e.* des fonctions ou des sous-processus qui accèdent aux variables locales du processus père. Ces méthodes peuvent mettre à jour l'état du processus en cours d'exécution, prendre des décisions locales, créer d'autres processus, envoyer des messages ou déterminer comment répondre au prochain message reçu.

Même si les acteurs ne sont pas implémentés pas un nouveau type d'entité primitive (leur implémentation repose sur les processus), ils disposent de constructions syntaxiques dédiées dans le langage, afin de faciliter leur définition (construction *@obj_def*) et leur utilisation

III.4. Parallélisme implicite et hypothèse synchrone

Le langage Antescofo offre des structures de contrôle dédiées que nous allons présenter brièvement.

Nous avons plusieurs fois mentionné que les calculs s'effectuent implicitement en parallèle. Le parallélisme en Antescofo est une notion qui diffère du parallélisme usuel dans un langage impératif. Les différences proviennent du cadre synchrone du langage.

Classiquement, le temps se définit à travers deux notions, les instants et les durées, qui s'organisent à travers trois relations : la simultanéité, la succession et la permanence (le fait de durer). Les langages de programmation séquentielle ne traitent généralement que des instants et de leur succession : la durée réelle d'un calcul n'a pas d'importance, et on ne gère que la succession. Ce modèle du temps est semblable au modèle MIDI où la durée d'une note est implicitement représentée par l'intervalle de temps entre l'activation de cette note (*on*) et sa désactivation (*off*). On ne peut pas dire par exemple qu'un accord commence à un moment donné, car le démarrage de l'émission des notes des accords sont des événements séquentiels distincts, la simultanéité est dans ce cas émulé. Cependant, au niveau conceptuel, cela signifie que la simultanéité ne peut pas

[71] Cet *exe* peut être utilisé pour tuer l'instance du processus, accéder à ses données locales, tester si le processus est toujours en cours d'exécution, etc. C'est une valeur comme une autre qu'on peut stocker dans un tableau ou bien passer en argument.

être directement exprimée dans le langage, ce qui rendra plus difficile la spécification de certains comportements temporels.

Pour exprimer la simultanéité, on doit imaginer idéalement que les calculs élémentaires s'effectuent instantanément (avec une durée nulle). Ce modèle idéal est désigné sous le terme de « modèle synchrone » ou bien d'hypothèse synchrone[72] car il permet aux événements d'être considérés comme atomiques et de parler formellement de simultanéité. Les conséquences de l'hypothèse synchrone ont été étudiées dans le développement des langages synchrones pour les systèmes embarqués temps réel tels que Esterel [BG92], Lustre [HCRP91] ou Lucid Synchrone [Pou06].

Le langage d'Antescofo reprend le modèle synchrone qui permet de déclencher des événements simultanément. Mais comme pour tous les langages synchrones, les calculs simultanés sont exécutés séquentiellement dans un ordre déterministe qui dépend de l'apparition de l'événement dans la partition. Bien que la notion d'une simultanéité ordonnée séquentiellement peut sembler paradoxale, on peut définir des modèles du temps logiquement cohérent qui formalisent ces notions[73]. Imposer un ordre séquentiel d'exécution à des calculs (de durée nulle) simultanés est fondamental, par exemple pour l'initialisation correcte d'actions audio (par exemple, on ne va pas changer les paramètres d'un synthétiseur s'il n'est pas encore instancié dans la chaîne audio).

Le parallélisme en Antescofo correspond à la simultanéité des calculs. Quatre opérateurs, appelés *opérateurs de continuation*, permettent de contrôler la simultanéité ou la succession de deux calculs *a* et *b* :

- une opération implicite permet de lancer deux calculs simultanément. Cet opérateur se note par la simple juxtaposition des calculs : *a b* ;
- le délai permet de retarder le lancement de la seconde action par rapport au lancement de la première d'un délai *d*. Il se note en intercalant le délai *d* entre les deux calculs : *a d b* ;
- la continuation simple permet de lancer *b* après la terminaison du calcul *a* et se note : *a* ==> *b* ;
- les calculs pouvant déclencher des sous-calculs de durée variable, récursivement, il existe un dernier opérateur de continuation composée permettant de lancer un calcul *b* après la terminaison de *a* et de tous les sous-calculs induits par *a.* Cette opération se note *a* +=> *b* et correspond à l'opérateur de succession dans les langages séquentiels classiques.

La figure suivante illustre ces opérateurs :

---

[72] N. Halbwachs, *Synchronous programming of reactive systems*, Springer Science & Business Media, 2013, vol. 215.

[73] O. Maler, Z. Manna et A. Pnueli, "From timed to hybrid systems", in Workshop/School/Symposium of the REX Project (Research and Education in Concurrent Systems), Springer, Berlin, Heidelberg, 1991. p. 447-484.

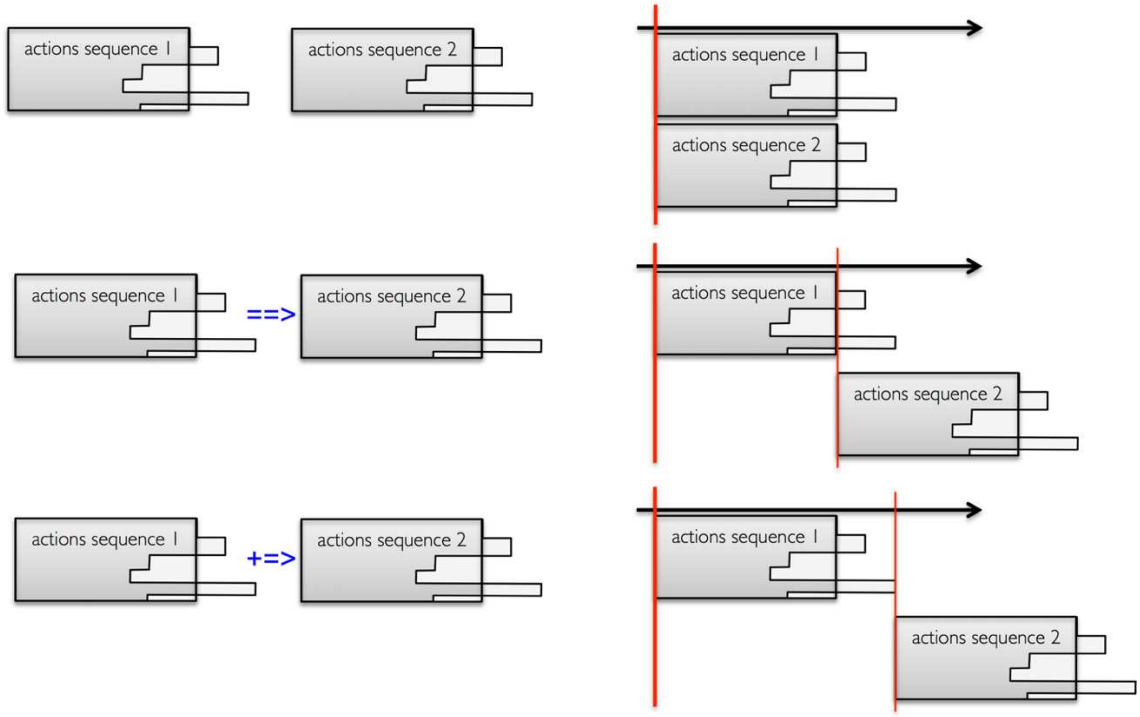


Fig. 3.6 Illustration imagée des opérateurs de simultanéité, de continuation simple et de continuation.

III.5. Le *whenever*

La structure de contrôle *whenever* permet de déclencher des actions et des processus à chaque fois qu'une condition logique est vérifiée.

La vérification de la condition n'est pas statique mais dynamique et peut se produire à n'importe quel moment du déroulement de la pièce. Une fois lancé, le *whenever* reste actif jusqu'à ce qu'il soit stoppé explicitement (par une commande *abort*), sinon il reste actif jusqu'à la fin du programme. Quand il est actif, il déclenche un processus (le corps du *whenever*) chaque fois que sa condition est vérifiée.

Cette structure est fondamentale dans le contexte des musiques interactives puisque c'est elle qui va permettre d'interagir avec le monde extérieur qui met à jour un ensemble de variables. Cette mise à jour, qui se produit parallèlement à l'exécution du programme, est surveillée par un *whenever* pendant le temps de sa durée de vie. Comme d'autres structures dans Antescofo, elle est polyphonique, c'est-à-dire que le corps du *whenever* sera déclenché autant de fois que la condition logique est vérifiée et s'exécutera en parallèle avec les autres activités du programme.

```
1  $wt_seq_dur1 := [1, 1/8, 1/8, 1/2, 1/16, 3/16, 3/16, 3/8, 1/8, 1/4, 1/8, 1/16, 1/16, 3/16, 1/4, 1/16, 3/16, 1/4, 3/16, 1/4]
2  $wt_freqs1 := [123.73, 125.54, 132.31, 136.25, 137.03, 138.27, 139.08, 142.37, 144.03]
3  $wt_map := MAP{}
4  $inc_kick_R_wt := 0 // initialisation incrementation
5  whenever kick_wt_R ($R_kick==$R_kick)
6  {
7      group
8      {
9          @local $nom, $pan, $dur_seq, $wt_map
10         $pan := @rand(5.)
11         $nom := ("kick_wt_R"+$inc_kick_R_wt)
12         $dur_seq := @rand_range(2, 5)
13         $wt_map.insert($nom, ::wavetable_seq3($wt_seq_dur1, $wt_freqs1*@rand_range(0.5, 3), @rand_range_int(60, 250), $pan)
14         $dur_seq abort $wt_map($nom)
15     }
16     $inc_kick_R_wt := $inc_kick_R_wt +1
17 }
```

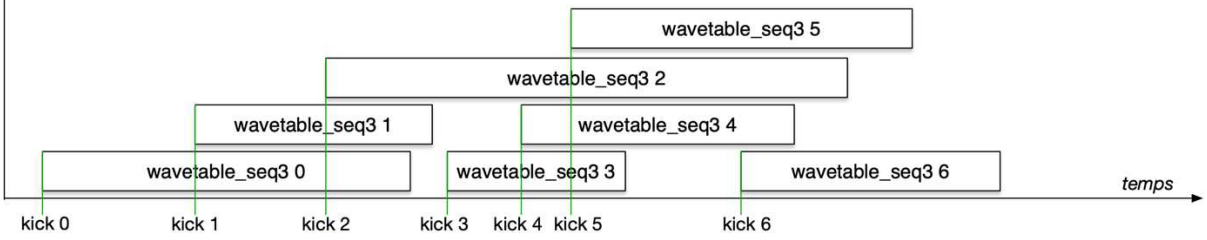


Fig. 3.7 Exemple d'utilisation d'un *whenever* pour déclencher une séquence rythmique à partir d'un *kick* (mouvement brusque) de la main droite capté par un accéléromètre.

La figure ci-dessus illustre l'utilisation d'un *whenever* pour déclencher une séquence rythmique à partir d'un *kick* (mouvement brusque) de la main droite capté par un accéléromètre. Le kick est détecté par un traitement effectué en amont d'Antescofo, au niveau du capteur de mouvements. Chaque occurrence d'un kick de la main droite est notifié à Antescofo par un envoi de message Max qui met à jour la variable `$R_kick`. De manière interne au programme, la condition logique `$R_kick == $R_kick` est vérifiée dynamiquement, ce qui déclenche le processus `::wavetable_seq3` qui joue une séquence rythmique d'une synthèse par table d'onde avec des hauteurs faites à partir des tableaux des lignes 1 (rythmes) et 2 (fréquences) et d'autres paramètres aléatoires pour chaque instance. La durée de chaque processus est variable et arrêtée par le *abort* entre 2 et 5 temps (ligne 12) après son activation. Le graphique en bas représente la polyphonie intrinsèque en fonction de l'occurrences des kicks.

Tout se passe comme si un *whenever* réévaluait constamment sa condition logique afin de décider s'il faut déclencher l'évaluation de son corps. Cependant, l'implémentation des *whenever* est optimisée et l'évaluation de la condition est paresseuse : elle ne se fait que si sa valeur est susceptible de changer, *i.e.* quand l'une des variables qui apparaît dans la condition est mise à jour (que ce soit lors d'un calcul interne ou bien de manière externe par un message).

III.6. Courbes différentielles

Nous ne présenterons pas la notion de courbes échantillonnées (*breakpoint functions*) car cette notion est classique dans les langages musicaux. Cette structure de contrôle permet d'échantillonner une fonction définie par morceaux. Cette fonction peut être définie conjointement à la structure de contrôle ou bien être une NIM, c'est-à-dire une valeur représentant une fonction définie par morceaux. La seule spécificité des *curves* Antescofo est de permettre de parcourir la fonction échantillonnée dans un temps

absolu (temps physique) ou relatif (*i.e.* synchronisé avec un autre processus ou avec le musicien).

La notion de *curve* a été étendue en Antescofo pour permettre de résoudre des équations différentielles dont la variable est le temps. La solution de l'équation différentielle est échantillonnée (comme pour une courbe définie par une NIM) et on peut aussi définir des points remarquables (appelés *zero crossing*). Ces points sont définis par le passage à zéro d'une quantité arbitraire qui dépend du temps, de la fonction solution et de sa dérivée. En un *zero crossing*, il est possible de changer la valeur de la fonction et/ou d'équation différentielle. Cette fonctionnalité permet dans les faits de définir des fonctions par morceaux, réalisant la notion de systèmes différentiels hybrides largement utilisée en automatique pour le contrôle de dispositifs physiques.

En Antescofo, les courbes différentielles permettent notamment le calcul optimisé de modèles physiques de contrôle. Attention, cette construction n'est pas adaptée à la définition d'oscillateurs pour la synthèse sonore : comme les autres processus Antescofo, les échelles de temps mises en jeu sont la milliseconde et au-delà.

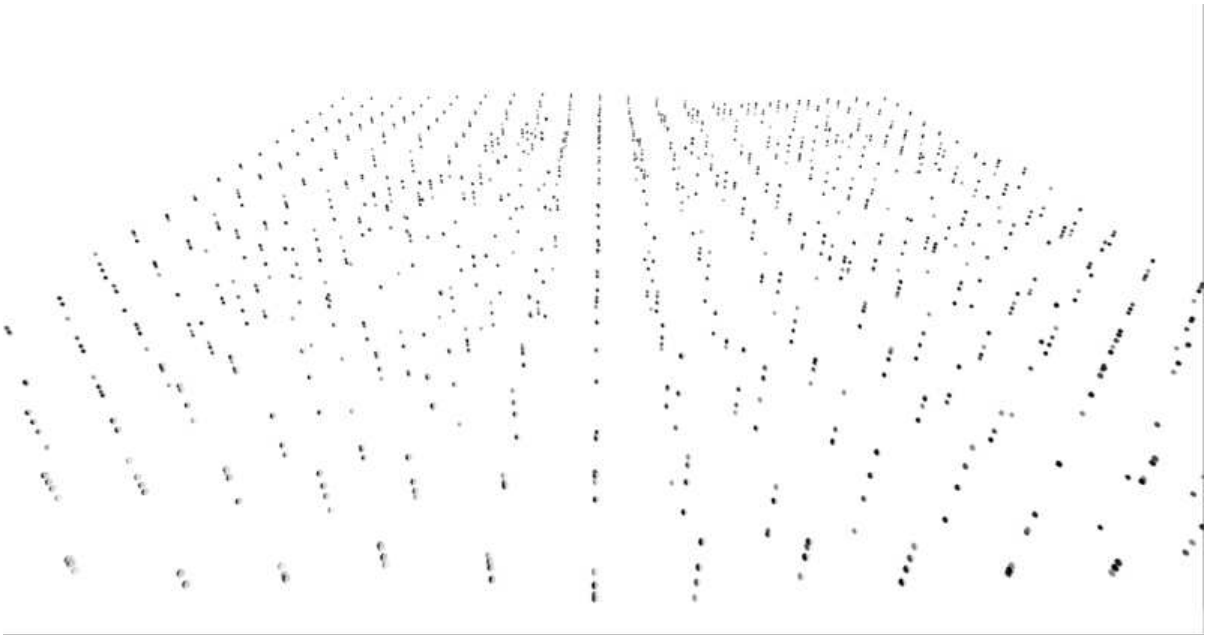


Fig. 3.8 Visualisation d'un système masse-ressort avec une grille de 400 masses. Le système programmé avec le solveur d'équations différentielles intégré dans le langage Antescofo utilise environ 4 % de CPU dans un processeur Intel Core i9 à 2,9 GHz. À titre indicatif, un système masse-ressort de 49 particules programmé directement dans le langage Antescofo utilise environ 30 % de CPU.

III.7. Mécanismes d'exécution dédiés

Le langage Antescofo propose plusieurs mécanismes d'exécution dédiés particulièrement pertinents dans un contexte d'applications musicales temps réel.

Propagation de constantes

Afin de minimiser les calculs à l'exécution, conséquent agressive de propagation des constantes. Toutes les expressions que l'on peut détecter de façon évidente au moment

du chargement du programme comme étant constantes, sont précalculées et propagées comme constantes dans les expressions englobantes.

Évaluation au chargement

Lors du chargement, il est possible d'exécuter avec la construction `@eval_when_load` des calculs qui ne seront évalués qu'une seule fois au moment du changement du programme. Cette construction permet d'initialiser à l'avance des chaînes audio ou un set-up complexe.

Évaluation à la volée

L'interprète Antescofo réagit aux messages de Max ou de PD, mais aussi à des messages OSC. Un canal OSC est spécifiquement réservé à l'évaluation à la volée d'expressions textuelles. Cela permet d'offrir sous des éditeurs de texte tels que Sublime ou Atom une fonction permettant d'évaluer depuis ces logiciels des parties du programme en cours d'écriture. Cette fonctionnalité est particulièrement cruciale pour évaluer des parties de l'électronique pendant le processus de composition, tester des parties du code, par exemple pour des processus génératifs ou interactifs, ou encore faire du *live coding*.

Fonctions de transport

Spécifiques au langage Antescofo, les fonctions de transport[74] permettent d'activer ou désactiver la machine d'écoute, de jouer et de naviguer dans la partition électronique avec différentes modalités.

Pour citer les plus importantes, la commande « *start* » permet de lancer l'exécution d'un programme, « *stop* » pour l'arrêter, « *startfromlabel* » active la machine d'écoute à partir d'un endroit spécifique dans la partition, « *next_label* » force la détection de l'événement suivant et déclenche les actions qui sont entre l'événement courant et l'événement suivant, « *visit_label* » permet de se positionner à un endroit sans déclencher aucune action précédente, etc.

Ces fonctions sont fondamentales pour jouer et se déplacer dans la partition, par exemple pour commencer à des moments spécifiques de la partition, notamment pendant le processus de composition ou pendant les répétitions des pièces mixtes.

Les fonctionnalités visées sont généralement pensées comme des commandes « extérieures » à un langage de programmation et sont généralement réduites au lancement d'un programme et éventuellement à son arrêt. Pour Antescofo, ces commandes font partie du langage et peuvent se lancer par calcul dans le langage lui-même.

[74] http://antescofo-doc.ircam.fr/Reference/tempo_transport/#transport

Compilation à la volée

Les programmes Antescofo sont des programmes interprétés. Cependant, il est possible de compiler au vol (un sous-ensemble) des expressions (notamment les expressions arithmétiques et mathématiques). Ce mécanisme permet de produire une version équivalente mais plus efficace afin d'optimiser les ressources CPU de l'ordinateur lors de l'évaluation de ces expressions. Pour ce faire, le code Antescofo est traduit dans un code C++ qui est généré, compilé et lié dynamiquement au programme Antescofo. Lors de l'évaluation de l'expression, c'est le code compilé qui est exécuté au lieu de faire appel à l'interprète.

Ce mécanisme est utilisé pour les *curves* différentielles et dans ce cas, la compilation au vol se fait au chargement du programme, ce qui évite de perturber l'exécution de la pièce pendant son exécution. Il est possible de demander explicitement la compilation d'une fonction, et cette compilation se fait alors au moment de cette demande (compilation « *just in time* »).

Cette fonctionnalité est très prometteuse puisqu'elle pourrait dans le futur s'étendre à plus d'expression, ce qui optimiserait les ressources pour faire plus de calculs et libérerait des ressources pour d'autres traitements (audio, vidéo…) dans un même ordinateur.

```
@fun_def @scale($x, $in_low, $in_high, $out_low, $out_high, $power)
{
    if(($x-$in_low)/($in_high-$in_low) == 0)
    {
        $out_low
    }
    else
    {
        if((($x-$in_low)/($in_high-$in_low)) > 0)
        {
            ($out_low + ($out_high-$out_low) * @pow((($x-$in_low)/($in_high-$in_low)), $power))
        }
        else
        {
            ( $out_low + ($out_high-$out_low) * -(@pow(((-$x+$in_low)/($in_high-$in_low)), $power)))
        }
    }
}

$ret := @compilation(MAP{
    @scale -> [["double", "double", "double", "double", "double", "double"], "double"],
})
```

Fig. 3.9 Exemple de compilation de la fonction utilisateur *scale* (l'équivalent de l'objet scale dans Max). La fonction `@compilation` va faire la compilation et générer une nouvelle fonction `@scale_compiled`. L'argument donné à `@compilation` est un dictionnaire qui décrit la signature des fonctions à compiler. La signature d'une fonction est la spécification du type de ses arguments et du type de la valeur retournée.

III.8. Bibliothèque ad hoc

Le langage Antescofo propose une bibliothèque de plus de 300 fonctions permettant de manipuler les types de données de base (symboles, booléens, entiers, flottants, chaînes de caractère, vecteur, dictionnaire, NIM, fonctions, processus, acteurs, *exe*). Parmi ces fonctions, ont peux noter un ensemble de fonctions spécialisées particulièrement utiles dans les applications musicales.

KNN

K-nearest neighbors (KNN) est une famille d'algorithmes permettant de déterminer rapidement le voisinage d'un point dans un nuage de points. La bibliothèque Antescofo est basée sur *nanoflann*[75]. Nous utiliserons ces fonctions entre autres pour le système de synthèse concatenative spatiale en 3D.

Réification d'un score Antescofo

Il est possible d'accéder via un ensemble de fonctions à la description des événements musicaux à reconnaître dans une partition Antescofo. Il est par exemple possible de récupérer sous la forme d'une NIM, les changements de tempo dans la partition. Autre exemple, la fonction @bach_score permet de transcrire une partition Antescofo dans un format compatible avec l'objet bach.roll[76] dans Max.

Formats d'entrée/sortie

Plusieurs fonctions permettent de représenter des données dans des formats divers. Il est ainsi possible de définir une valeur Antescofo dans le format Json et inversement, de transformer une valeur Json en une valeur Antescofo. Les valeurs Antescofo sont implicitement traduites en valeur OSC lors de l'envoi d'un tel message, et inversement, les arguments d'un message OSC sont traduits automatiquement en valeur Antescofo. La fonction `@nim2vezer` permet d'exporter des NIMs (*breakpoints*) vers le séquenceur OSC Vezér[77] basé sur une *timeline* graphique.

III.9. Représentations graphiques de la partition

Antescofo propose un langage textuel, contrairement à des langages dédiés à la musique comme Max ou OpenMusic. Ces derniers sont des langages à flots de données (*dataflow*), ce qui rend difficile l'explicitation de *timelines* parallèles clairement identifiées comme dans la notation musicale symbolique traditionnelle. Antescofo améliore cet état de fait en proposant une notion explicite de *timeline* à travers les notions de groupe et de processus. Cependant, ces *timelines* étant dynamiques et pouvant être le résultat d'un calcul (par exemple en lançant un processus), il reste difficile d'identifier le code produisant un effet spécifique dans des partitions électroniques comptant des milliers de lignes de code, surtout s'il y a beaucoup de polyphonie et des couches multitemporelles. Cette difficulté impacte grandement la compréhension, la lisibilité et la composition musicale.

Des solutions peuvent être apportées pour pallier ce problème, comme Ascograph, un logiciel permettant de visualiser (et d'éditer) les événements électroniques dans une partition Antescofo sous la forme d'une *timeline* très semblable à celle qui existe dans la plupart des séquenceurs DAW. Cet outil est resté à un stade de maquette et n'est

[75] https://github.com/jlblancoc/nanoflann
[76] https://www.bachproject.net
[77] https://imimot.com/vezer/

actuellement plus développé pour des raisons techniques (les bibliothèques utilisées ne sont plus maintenues).

Des approches alternatives doivent être développées. Nous avons mentionné la fonction *@bach_score* qui permet de construire un piano roll et de l'animer pendant l'exécution du programme Antescofo. Cette approche reste cependant limitée et ne permet pas d'interagir avec la *timeline*.

Une direction de recherche

Nous suggérons ici une piste possible, déjà présente d'une façon rudimentaire dans Ascograph et qui consiste en l'association de *timelines* graphiques à des parties du code.

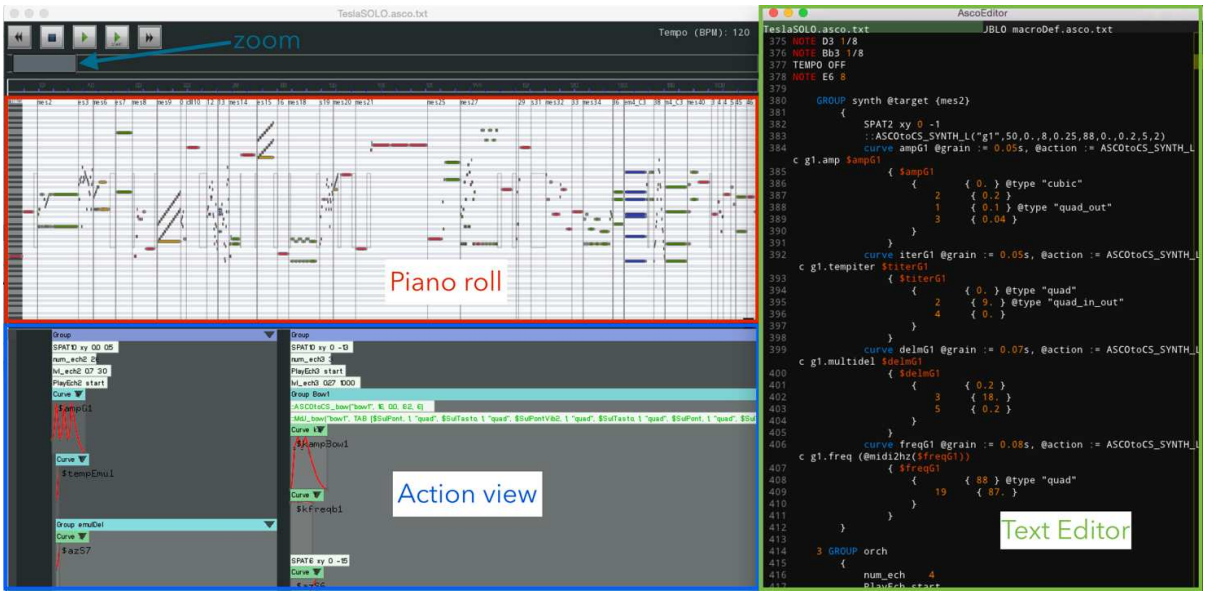


Fig. 3.10 Interface du logiciel Ascograph avec ses 3 représentations. En haut à gauche, une représentation de type piano roll de la partition instrumentale en notes MIDI, à gauche l'éditeur de texte de la partition instrumentale et de l'électronique et en bas, la représentation graphique des actions électroniques.

Ascograph adopte cette approche mais uniquement pour la spécification des événements musicaux à suivre. Plus généralement, on peut séparer des parties de l'électronique qui fonctionnent en parallèle, par analogie avec les instruments dans une partition d'orchestre, et avoir accès à une représentation de chaque partie aussi bien dans sa représentation graphique qu'en code à un moment déterminé de la partition et pour toutes les parties de l'électronique.

L'interchangeabilité entre le code textuel et la visualisation graphique (l'un étant le miroir de l'autre) serait alors fondamental pour passer d'une représentation à l'autre et pouvoir bénéficier des avantages de chacune. D'une part, le graphique permet de se retrouver facilement dans le déroulement temporel de la musique (*timeline*), et permet de modifier graphiquement certains paramètres (comme les *breakpoints* des courbes, des valeurs de paramètres, etc.). D'autre part, le code textuel permet de spécifier de

manière concise des actions électroniques fines, génératives et expressives grâce au langage de programmation.

Ces deux représentations (*timeline* visuelle et code textuel) seraient utiles à la fois pour le travail de composition, de réalisation et de performance du compositeur et/ou du RIM. On peut aussi imaginer différents rendus graphiques, par exemple un rendu plus général et simplifié représentant le résultat acoustique plutôt que les procédures pour le produire, à destination des chefs d'orchestre, des musiciens, et des performeurs (acteurs, danseurs…). D'autres types de rendu graphique peuvent aussi être imaginés à d'autres fins, par exemple dans des logiciels d'analyse musicale tels que iAnalyse, développé par Pierre Couprie. Un premier test de faisabilité a permis de valider l'importation d'une partition Antescofo dans iAnalyse. Ce travail pourrait aider l'analyse et la documentation, par des musicologues, des pièces mixtes et interactives.

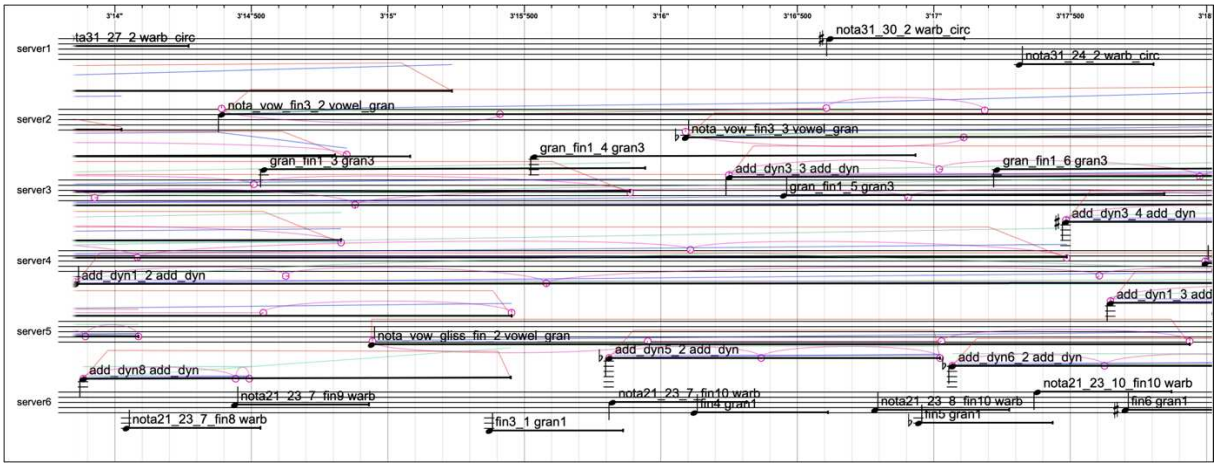


Fig. 3.11 Exemple d'écriture de l'électronique, extrait du début de *Las Pintas,* dans la librairie de notation symbolique dans Max *bach*. Bien que cette écriture présente plusieurs avantages, grâce notamment aux métadonnées qui peuvent être associées à chaque note comme par les différents paramètres de l'électronique et leur évolution, et malgré sa construction dynamique, elle n'est pas adaptée à l'interaction avec le code textuel du langage Antescofo qui peut lancer des processus dynamiquement.

IV. SuperCollider

Ce chapitre donne un bref aperçu du système SuperCollider (SC) et de son architecture client/serveur. Nous supposerons que le lecteur connaît les principes de la programmation audio et que la notion de graphe audio lui est familière. Nous n'entrerons pas dans les détails de la programmation du serveur de SuperCollider. L'objectif de ce chapitre est de détailler les éléments qui nous ont fait choisir ce système pour un couplage à Antescofo.

IV.1. SuperCollider dans mon travail

SuperCollider [MC02] a été conçu par James McCartney, au moment où la puissance de calcul des ordinateurs personnels (type Macintosh PowerPC) a permis de faire du traitement en virgule flottante en temps réel [Wil11]. La première version de SuperCollider date de 1996 et la version 3 qui est devenue par la suite open source date de 2001. Cette dernière a été portée sur différentes plateformes (MacOS, Windows et Linux).

Dans le domaine du traitement du signal et de la synthèse sonore, le logiciel est devenu mon outil privilégié, tout d'abord pour sa capacité à créer d'une façon concise et puissante une grande variété de types de sons de synthèse et de traitements et aussi pour sa qualité au niveau du rendu audio (qualité reconnue par beaucoup d'utilisateurs dans le monde). Une autre caractéristique fondamentale est que SuperCollider a été conçu à la base pour le traitement audio en temps réel, l'interaction, la polyphonie et le multicanal, ce qui lui confère une souplesse et un dynamisme qu'on trouve difficilement dans d'autres environnements de synthèse et de manipulation sonore.

Dans la communauté de l'informatique musicale, SC est un logiciel relativement moins répandu que des systèmes comme Max ou PD. Cela est peut-être dû à sa programmation textuelle moins intuitive que des systèmes graphiques de type *dataflow*.

La première fois que j'ai utilisé SC dans sa version 2, j'ai été impressionné par la qualité du rendu audio et par son langage de programmation avec une syntaxe concise : une seule ligne de code suffit souvent à définir un son très riche. J'ai continué à utiliser SC pendant des années sous l'angle d'un synthétiseur générant du matériau difficilement réalisable dans d'autres environnements. Par la suite, j'ai commencé à programmer un environnement modulaire de composition dans le langage SC sclang mais qui adressait uniquement la partie traitement du signal et pas le contrôle temporel.

IV.2. Un exemple introductif

À l'instar de nombreux autres systèmes de synthèse, SC est utilisé pour définir des *graphes audio* où les nœuds représentent des UGEN (sommets sans prédécesseurs) ou bien des transformations audio (entre les entrées et les sorties du nœud).

La ligne de code suivante est un exemple simple de synthèse avec deux oscillateurs.

```
{ [SinOsc.ar(440, 0, 0.2), SinOsc.ar(442, 0, 0.2)] }.play;
```

Ce fragment de code définit une fonction (entre `{}`) qui appelle la méthode `.play`. Cette méthode indique qu'il faut envoyer le résultat de la fonction dans le serveur scsynth pour un rendu sonore. La fonction spécifie deux oscillateurs : ce sont des UGENs (*Unit generators*), des objets qui produisent un signal audio. Du fait qu'ils sont entre [ ] la sortie de la fonction est une sortie multicanale : chaque élément de la liste devient un canal audio. L'exemple va donc jouer un oscillateur à 440 Hz avec une amplitude de 0.2 dans le canal 1 (gauche) et en même temps un second oscillateur à 442 Hz avec une amplitude de 0.2 dans le canal 2 (droite).

La forme textuelle présente de nombreux avantages sur la forme visuelle qui s'est imposée dans de nombreux outils. À titre de comparaison, entre un langage textuel (SuperCollider) et un *data flow* (Max), voici un exemple de la synthèse appelée « *bubbles* » réalisée par James McCartney. Elle génère un son de synthèse qui module dans le temps et qui est assez riche. La figure suivante est la fonction qui permet de définir cette synthèse et de la jouer.

```
1 // analog bubbles
2 {
3    f = LFSaw.kr(0.4, 0, 24, LFSaw.kr([8,7.23], 0, 3, 80)).midicps; // glissando function
4    CombN.ar(SinOsc.ar(f, 0, 0.04), 0.2, 0.2, 4) // echoing sine wave
5 }.play
```

Fig. 4.1 Code représentant l'exemple « *bubbles* », synthèse réalisée en SuperCollider par James McCartney, avec utilisation de l'expansion multicanale. L'expression `LFSaw.kr([8, 7.23])` avec un tableau de deux éléments permet d'une façon simple de créer différentes instances du générateur LFSaw en multicanal, dans ce cas en stéréo.

La figure suivante est la même synthèse, avec le même résultat sonore, défini dans Max de façon visuelle. On peut apercevoir que la représentation graphique, même si plus intuitive et facile à comprendre à première vue, reste beaucoup moins concise. Avec la pratique, il n'y a plus de différence et le langage textuel est finalement beaucoup plus lisible.

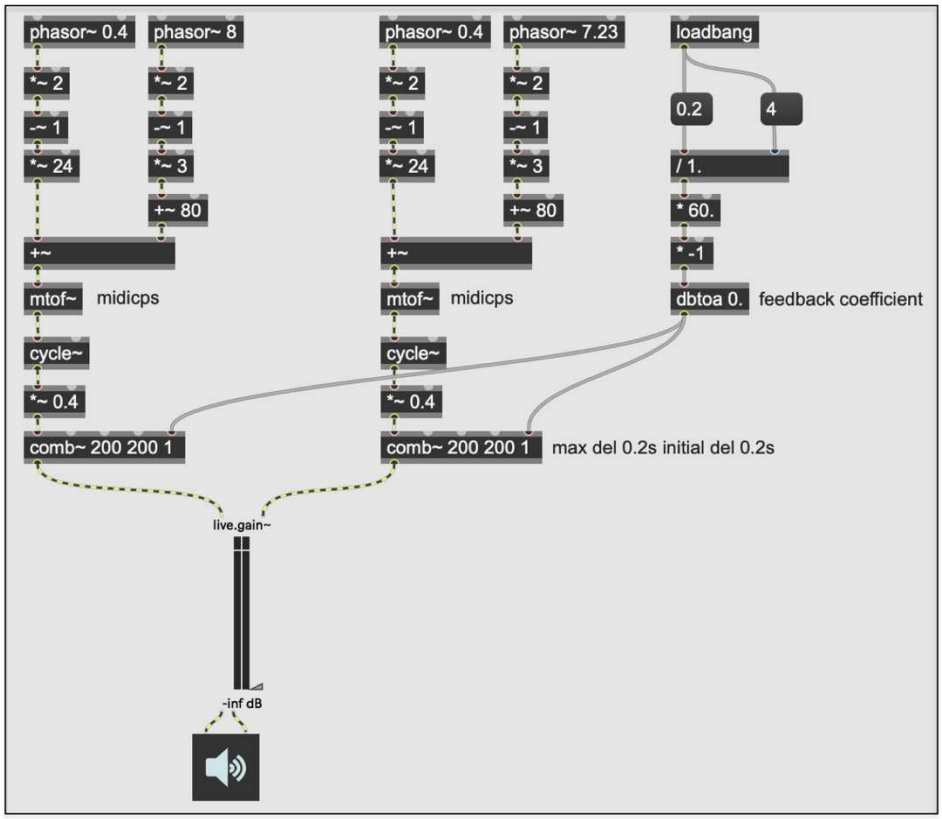


Fig. 4.2 La même synthèse « *bubbles* » réalisée dans l'environnement Max.

IV.3. Une architecture client/serveur

SuperCollider est fondé sur le paradigme serveur/client où le client est un IDE (environnement de développement) pour le langage de programmation sclang, et le serveur scsynth un moteur de rendu audio. Le logiciel est conçu et structuré pour le temps réel et isole ces traitements dans le seul serveur.

L'exemple précédent est écrit en sclang et est compilé en une suite d'instructions pour une machine abstraite audio temps réel qui est implémentée par le serveur scsynth. Mais le serveur est agnostique : sclang est un client particulier, mais n'importe quel programme peut adresser directement le serveur. C'est ce qui sera fait pour AntesCollider qui permet d'utiliser Antescofo comme un client de scsynth.

Une autre propriété importante inhérente au paradigme client/serveur adopté par le système sclang/scsynth est que le contrôle et les interfaces graphiques sont complètement séparés du moteur audio. Le serveur audio scsynth est dédié à l'analyse, à la synthèse et au traitement audio en temps réel (mais aussi différé, cf. plus bas). Le contrôle des processus de traitement du signal est assuré par le client (habituellement un programme dans le langage sclang, mais comme nous venons de l'indiquer, n'importe quel programme qui interagit avec scsynth).

Cette séparation client/serveur permet aux interfaces graphiques de ne pas interférer avec les parties de traitement du signal. De plus, par son caractère dynamique, seuls les traitements graphiques et les traitements audio réellement nécessaires seront actifs. Ceci permet une grande optimisation des ressources.

L'autre avantage de cette architecture est de compartimenter les traitements, propriétés importantes en cas de crash d'un élément du système. Cette architecture simplifie aussi la réalisation de dispositifs redondants permettant d'activer des instances de rechange (*spare)* du client ou du serveur, prenant automatiquement le relais pendant l'exécution de la pièce en cas de problème. Le système offre donc un grand gain en ressources et un système plus robuste.

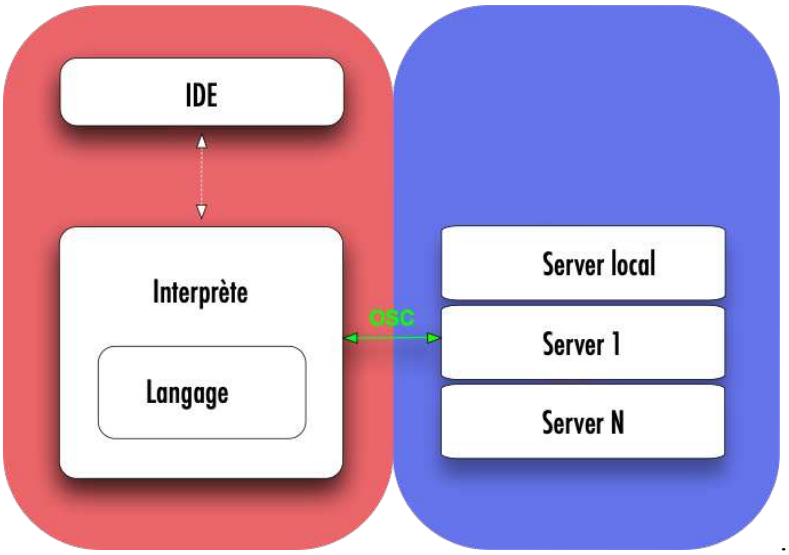


Fig. 4.3 Architecture serveur/client du logiciel SuperCollider. À gauche, la partie client avec un IDE, l'interprète et le langage pour contrôler les serveurs scsynth ainsi que pour générer des interfaces graphiques dynamiquement. À droite, une représentation des serveurs audio, on peut instancier plusieurs en parallèle.

IV.4. La structure dynamique d'un graphe audio dans SC

SuperCollider est un descendant direct des langages de synthèse audio Music-N avec le concept de *Unit Generators* (UGens) (générateurs de signaux élémentaires) dont les calculs sont implémentés dans le serveur de synthèse scsynth. Les UGens permettent un contrôle en temps réel de tous les paramètres de la synthèse. Leurs combinaisons et la transformation de leur sortie permettent de créer efficacement un système modulaire de chaînes audio dynamiques.

Les interconnexions des UGens sont organisées en arborescences (*Nodes*) dans des définitions de synthèse (*SynthDef*). Cette architecture permet une grande flexibilité, modularité et restructuration. En effet, on peut dynamiquement et en temps réel instancier, placer, déplacer et détruire les *Nodes* à n'importe quel endroit de l'arborescence : « *Les graphs audio (construction des chaînes de traitement audio) peuvent être regroupés en arbres arbitraires dont l'ordre d'exécution et d'évaluation peut être modifié dynamiquement*. » [MC02]

```
SynthDef(\SimpleSine, {|freq = 440, out| Out.ar(out, SinOsc.ar(freq, 0, 0.2)) }).add;

x = Synth(\SimpleSine);
y = Synth(\SimpleSine, [\freq, 660]);

x.free; y.free;
```

Fig. 4.4 Code représentant un exemple de base d'une définition de synthèse SynthDef dans SuperCollider avec une UGen *SinOsc* pour générer une forme d'onde sinusoïdale avec une fréquence par défaut de 440 Hz. Les variables *x* et *y* vont instancier dynamiquement deux synthèses différentes, *x* avec la fréquence par défaut et *y* une fréquence de 660. x.free; y.free; permettent de libérer les deux synthèses et de les enlever de la chaîne de DSP.

La librairie JIT (*Just-in-Time*) intégrée dans SuperCollider permet de modifier dynamiquement non seulement les paramètres des définitions de synthèses *SynthDef* précompilés, mais aussi leur structure interne (*graph*). On peut ainsi, comme déjà évoqué, n'activer que les traitements nécessaires à un moment donné. Cependant, cette dynamicité permet d'aller plus loin et ouvre la voie à de nouveaux types d'interactions. Par exemple, la librairie *JIT* (*Just-in-Time*) permet de remplacer un générateur sinusoïdal *SinOSC*[78] par un générateur de bruit *WhiteNoise*[79] dans la chaîne de traitement. Cette capacité ouvre de nouvelles possibilités pour le *live-coding* et l'expérimentation dans le domaine de la synthèse sonore :

> « *It is impossible to write a program while it runs. A program describes and determines a process, so changing the description implies a new outset, a new process. Yet it is possible to structure a program in such a way that parts of it can be interchanged dynamically, and its textual form can be arranged to allow rewriting those parts while the whole process continues. Thus, instead of first designing an application that has fixed interaction points (parameters), the program text itself becomes the main interface.* » [Wil11]

IV.5. Le choix de scsynth comme moteur audio

Ce dynamisme est un avantage considérable par rapport aux systèmes graphiques de type *dataflow* comme Max/Pd et une des principales motivations pour la création de la librairie AntesCollider.

Parmi d'autres avantages, il y a la polyphonie qui fait partie intégrante de l'architecture : on peut lancer plusieurs instances d'un même processus de synthèse et les contrôler ou les arrêter de manière globale ou locale. Cette fonctionnalité est aussi très importante dans la librairie AntesCollider du fait que le langage Antescofo (comme

[78] https://doc.sccode.org/Classes/SinOsc.html
[79] https://doc.sccode.org/Classes/WhiteNoise.html

vu précédemment) permet aussi l'instanciation d'un même processus polyphoniquement et à la volée.

Enfin, la gestion du multicanal ou « *mutichannel expansion* »[80] (chaque élément d'un tableau représente simplement la sortie du canal audio correspondant) est aussi une fonctionnalité fondamentale du système. Elle permet de travailler sur des chaînes audio de n'importe quel nombre de canaux d'une façon simple et globale (à titre de comparaison, le multicanal dans Max a été introduit seulement en 2018 dans sa version 8).

En résumé, le choix de SC pour prendre en charge les traitements audio a été motivé par les caractéristiques suivantes :

- moteur audio précis et efficace de haute qualité ;
- fréquence d'échantillonnage (192 k +) et taille de bloc entièrement réglables ;
- chaîne de signal audio en arithmétique flottante 32 bits ;
- l'échantillonnage de buffers (*sampling buffers*) utilise des flottants en 64 bits ;
- modulation du taux de contrôle rapide et fluide ;
- un riche ensemble de 250 générateurs (UGens) prédéfinis dans la distribution de base ;
- des centaines d'autres UGens développés par la communauté de développeurs ;
- la prise en charge de n'importe quel nombre de canaux d'entrée et de sortie, caractéristique critique pour la gestion des grandes configurations multicanales et des systèmes de spatialisation comme le HOA ou la WFS ;
- le logiciel est libre et en open source.

Il faut aussi citer une fonctionnalité qui n'est pas encore intégrée dans la librairie AntesCollider mais qui est particulièrement intéressante : le rendu hors ligne (*Non-Real-Time*, NRT). Ce mode d'exécution du serveur permet de faire des traitements audio qui ne sont pas possibles en temps réel (pour des raisons de puissance de calcul) ou pour générer du matériau audio en masse (*batch processing*[81]).

Un choix largement partagé

Par sa performance, son optimisation des ressources, son architecture de gestion des *Nodes* et sa notion de graphe audio dynamique et versatile, le serveur SuperCollider scsynth est utilisé par différents systèmes en tant que serveur et moteur de rendu audio,

[80] https://doc.sccode.org/Guides/Multichannel-Expansion.html
[81] Traitements par lots.

notamment pour des systèmes de *live coding* tels que Tidal Cycles[82], Sonic Pi[83] et a été interfacé avec d'autres langages de programmation comme Scheme (rsc3[84]), Common Lisp (cl-collider[85]), Haskell (hsc3[86], Vivid[87]), Python (FoxDot[88]), Java (JCollider[89]), Scala (ScalaCollider[90]), Lua (Lua2SC[91]), JavaScript (supercollider.js[92]), entre autres[93]. SuperCollider a aussi été intégré dans des systèmes embarqués, notamment dans Raspberry Pi[94] et BeagleBone Black[95], ou dans des DMI intégrés comme Organnelle[96], Bela[97] ou Norns[98]. Ces langages pilotent le serveur en temps réel.

IV.6. Extensibilité du serveur *scsynth*

Bien que SuperCollider comprenne une vaste librairie de UGens pour faire toutes sortes de synthèses, traitements et analyses, il est possible d'étendre les possibilités existantes en programmant de nouvelles UGens directement dans un langage *C++* [Wil11]. Une vaste collection existe déjà, les *sc3-plugins*[99] et font partie intégrante du logiciel.

Une autre manière de spécifier des modules de traitement et de synthèse est de les définir à partir du langage FAUST[100] développé au GRAME. Il est en effet possible de traduire le code des traitements *DSP* spécifiés en FAUST vers différentes plateformes incluant les SuperCollider UGens. Il est ainsi possible d'étendre la librairie de SC. Et la librairie AntesCollider utilise ainsi plusieurs modules issus de FAUST, notamment pour la spatialisation avec les librairies SC-HOA [GRLE17] basée sur la librairie Ambitools [LEGA15] et *Ambisonic Decoder Toolbox*[101] pour faire des traitements dans le domaine de l'ambisonie.

---

[82] https://tidalcycles.org/index.php/Welcome
[83] https://sonic-pi.net
[84] http://rohandrape.net/?t=rsc3
[85] https://github.com/byulparan/cl-collider
[86] http://rohandrape.net/?t=hsc3http://rohandrape.net/?t=hsc3
[87] https://www.vivid-synth.com
[88] https://foxdot.org
[89] https://www.sciss.de/jcollider/
[90] http://github.com/Sciss/ScalaCollider
[91] https://github.com/sonoro1234/Lua2SC
[92] https://github.com/crucialfelix/supercolliderjs/
[93] https://supercollider.github.io/community/systems-interfacing-with-sc
[94] https://www.raspberrypi.org
[95] https://beagleboard.org/
[96] https://www.critterandguitari.com/organelle#whats-new
[97] https://blog.bela.io/2017/10/29/bela-and-supercollider-live-coding-sensors/
[98] https://monome.org/docs/norns/
[99] https://github.com/supercollider/sc3-plugins
[100] https://faust.grame.fr/index.html
[101] https://bitbucket.org/ambidecodertoolbox/adt/src/master/

La possibilité d'intégrer des plugins VST[102] dans scsynth (VSTPlugin UGen[103]) ouvre la porte à tout un univers de plugins issus de différents instituts de recherche (par exemple les célèbres IEM *Plug-in Suite*[104] ou SPARTA[105] pour étendre les possibilités dans le domaine ambisonique) ainsi qu'à des plugins du commerce comme des instruments virtuels ou des traitements, notamment pour le *mastering*.

Grace à cette possibilité, la librairie AntesCollider intègre aussi les plugins VST et FAUST qui deviennent des modules banalisés qui peuvent être intégrés, instanciés dynamiquement et contrôlés algorithmiquement dans le temps par le langage Antescofo.

Cette extensibilité, son architecture client/serveur et son statut de logiciel libre (*open source*) confère à SuperCollider scsynth la possibilité d'intégrer simplement de nouveaux systèmes de synthèse dans différents domaines, et aussi d'être intégré comme moteur de synthèse dans des systèmes indépendants. Ces propriétés assurent la pertinence et la pérennité du système à travers le temps.

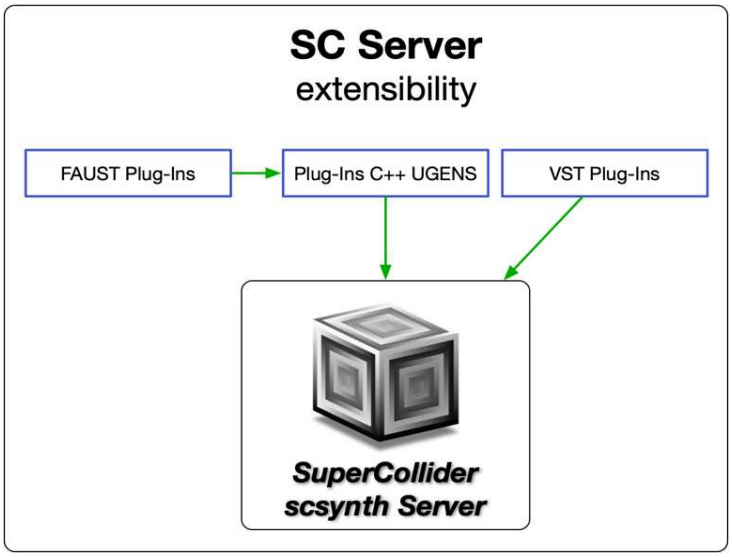


Fig. 4.5 Schéma représentant l'extensibilité des modules de traitement et synthèse de scsynth grâce à la possibilité de programmer des UGens SuperCollider dans le langage C++, FAUST ce qui permet la création des définitions de synthèses SynthDef ainsi que l'intégration des plugins VST dans la chaîne de traitements.

[102] VST (Virtual Studio Technologie) est un format ouvert de plug-ins créé par la société allemande Steinberg en 1996.
[103] https://git.iem.at/pd/vstplugin/-/releases
[104] https://plugins.iem.at/
[105] http://research.spa.aalto.fi/projects/sparta_vsts/

V. La librairie AntesCollider : un système dynamique de composition et d'écriture de l'électronique en temps réel

Dans ce chapitre, nous détaillons la conception et la réalisation de la librairie AntesCollider que j'ai développée au cours de cette thèse. Cette librairie est une première étape dans le développement d'un environnement adressant la notion de partition centralisée.

Nous récapitulons tout d'abord les éléments qui nous ont amenés à développer cette librairie et les objectifs attendus (section 1). Les principales caractéristiques de la librairie sont présentées (section 2) avant d'entrer dans les détails techniques.

La communication entre Antescofo et scsynth est détaillée (section 3) avant de présenter la structure des graphes audio qui seront créés et gérés par AntesCollider (section 4). Ces graphes sont construits puis contrôlés automatiquement à travers des objets Antescofo et nous détaillons les plus importants (section 5). La section suivante détaille un exemple d'utilisation de la librairie, de bout en bout (section 6). Les arguments permettant de paramétrer la création des objets AntesCollider sont listés à la section 7 (un listing plus exhaustif est donné en annexe de ce document, cf. section 8). Nous présentons ensuite la création automatique des interfaces utilisateurs permettant de commander dynamiquement les traitements audio (section 9).

Nous avons aussi implémenté un programme indépendant permettant de visualiser dynamiquement des données issues des modèles physiques que nous avons développés sous Antescofo. Ces modèles physiques servent à piloter les synthèses sous AntesCollider et il est important de pouvoir visualiser les calculs en temps réel (section 10).

Les œuvres développées avec AntesCollider, qui ont permis de valider ce travail, sont recensées à la section 11. Le chapitre se termine par quelques retours critiques et l'évocations des développements futurs qui pourraient y répondre (section 12).

V.1 De la partition centralisée à AntesCollider

Motivations

Aujourd'hui, il existe toute une panoplie d'instruments électroniques, d'objets de synthèses et de traitements accessibles depuis un DAW ou encore des environnements tels que Max/Pd/FAUST (entre autres). Ces systèmes peuvent s'étendre grâce à des plugins (VST ou autres formats) et adressent la performance en *live*. Ces systèmes sont adaptés à un grand nombre des genres musicaux mais pour l'exploration qui m'intéresse, il leur manque un chef d'orchestre qui va permettre de coordonner d'une façon souple le jeu et l'interaction entre eux ainsi qu'avec le monde extérieur (instrumentiste/performeur, interfaces de contrôle, capteurs, etc.).

On a vu aussi que les solutions de gestion et composition temporelles à différentes échelles et pendant une performance en temps réel, restent encore trop limitées, surtout si on les compare à la richesse et à la diversité des traitements du son et des synthétiseurs qu'ils soient du commerce, libres ou issus de centres de recherche.

Ces considérations m'ont amené à développer la librairie AntesCollider [FEGIDO19]. Programmée dans le langage Antescofo, AntesCollider est le résultat d'études, de recherches et de pratiques sur les outils informatiques existants en relation avec la création contemporaine dans le but d'étendre les possibilités musicales et de composition de l'électronique. Elle s'inspire à la base aussi bien des logiciels commerciaux (tels que les DAW) que des logiciels et langages de programmation plus spécifiques du travail de composition tels que Max ou OpenMusic.

AntesCollider vise le développement d'un environnement de travail centralisé d'écriture de l'électronique. Les utilisateurs ciblés sont les compositeurs, les designers sonores et les RIM qui s'intéressent à la composition, à la création sonore, à l'interactivité et au temps réel. La souplesse, l'expressivité et la variété des possibilités offertes par cette librairie devraient aider à augmenter l'espace de pensée compositionnelle et induire de nouvelles approches pour concevoir la musique électronique et l'interactivité musicale.

Objectifs

AntesCollider est une librairie qui essaye d'aller vers une plus grande flexibilité en intégrant un langage de programmation du temps, Antescofo, avec un moteur de synthèse audio performant, souple et dynamique, le serveur SuperCollider scsynth :

- l'intégration à un langage de programmation dédié textuel permet de s'affranchir des systèmes fermés où tout est déjà préprogrammé, et où les structures du calcul sont fixes et statiques ;
- l'intégration à un serveur scsynth permet de spécifier finement les processus de synthèses et de transformations sonores au sein d'une même partition centralisée Antescofo.

Un langage textuel comme Antescofo, pleinement intégré à Max, permet de combler le manque de coordination temporelle auquel nous nous sommes malheureusement habitués avec les outils existants. Il apporte aussi des structures de données modernes (fonction de premier ordre, acteur, dictionnaire) avec un ensemble très riche de fonctions dédiées permettant de les manipuler. Les possibilités offertes par le langage facilitent la manipulation et enrichissent l'espace de pensée du compositeur avec de nouvelles interactions, et permettent l'écriture de structures temporelles et compositionnelles en relation avec le jeu en direct du performeur.

V.2. Principales caractéristiques de la librairie AntesCollider

La librairie AntesCollider exploite l'architecture client/serveur du logiciel SuperCollider en pilotant un ou plusieurs serveurs scsynth à partir du langage

Antescofo. Ceci va permettre d'intégrer les avantages d'un langage permettant un contrôle synchrone et expressif à un moteur de synthèse dynamique et performant, et offrir ainsi un environnement d'écriture de l'électronique centralisée et adapté à la composition des musiques interactives en temps réel.

Cette intégration présente plusieurs avantages, surtout au niveau de la souplesse et de l'optimisation des ressources de calcul par rapport à des systèmes plus standard comme les stations audionumériques ou les logiciels plus versatiles de type *dataflow* tels que Max/Pd.

Elle permet de créer des processus polyphoniques de contrôle grâce aux processus et objets Antescofo et des processus audio par l'intermédiaire des *SynthDefs* SuperCollider. En effet, on va pouvoir associer des processus Antescofo à des processus audio dans scsynth et les activer autant que nécessaire de façon naturelle puisque cette caractéristique polyphonique est propre aux deux systèmes.

Ce dynamisme est le principal avantage de la librairie, il va permettre de créer et détruire dynamiquement n'importe quelle configuration musicale électronique quels que soient les processus impliqués, processus audio comme processus de contrôle. Le système permet ainsi de passer d'une configuration à une autre, arbitrairement différentes, avec une grande fluidité et de façon discontinue ou bien continue par interpolation des paramètres, par des fondus enchaînés ou encore par des processus algorithmiques quelconques.

La composition, l'ordonnancement et la synchronisation d'événements musicaux sont spécifiés, gérés et contrôlés grâce aux fonctionnalités de suivi de partition et au séquenceur sophistiqué intégré dans Antescofo. Ces fonctionnalités sont de bien plus haut niveau que celles offertes par le langage sclang dans l'environnement SuperCollider. En particulier, grâce à son contrôle de plus haut niveau, la librairie AntesCollider permet de simplifier significativement la gestion des modules de traitements et les connexions entre modules à l'intérieur du serveur *scsynth.* Voici quelques avantages de la librairie AntesCollider sur l'utilisation du client sclang :

- La création et la connexion de chaînes de traitements, le routage vers différentes destinations (par exemple vers des pistes auxiliaires) et la création de pistes pour la spatialisation (HOA, VBAP…) se font directement dans la librairie. L'utilisateur n'a pas à gérer les « bus » de connexion audio, la création des groupes et *Nodes* et leurs ordres, ce qui dans sclang devient assez complexe si on n'a pas un système qui les organise et les gère.
- Bien que sclang dispose d'un système de « *pattern* » rythmique assez complet, performant et sophistiqué, ainsi que des « *routines* » pour gérer des événements temporels, il présente plusieurs contraintes structurelles. Il n'existe pas de notion globale de contrôle et la gestion de différentes temporalités devient rapidement complexe. Dans le langage Antescofo, la gestion temporelle est transparente, complètement programmable et modifiable grâce aux structures de données et

de contrôle (par exemple, chaque processus peut disposer de son tempo local, indépendamment du tempo global du musicien suivi).

- Le modèle de programmation offert par Antescofo offre un séquenceur sophistiqué qui permet de spécifier une partition électronique, avec par exemple la multitemporalité et les différentes fonctions de transport.
- L'écriture du temps dans le langage Antescofo est exprimée aussi bien en temps relatif à un tempo dynamique (qui peut changer au cours du temps) qu'en temps absolu (en secondes ou millisecondes). Elle permet une modulation de temporalité souple et en relation avec l'environnement (par exemple l'instrumentiste dans le cas du suivi de partition). Dans sclang, bien qu'il y ait des extensions et des librairies externes (comme Quarks[106]) pour définir et organiser des structures temporelles et de tempi, ils ne sont pas complètement intégrés à un système de partition électronique centralisée et à un suivi de partition comme dans Antescofo et la librairie AntesCollider.

V.3. Communications entre le serveur scsynth et Antescofo

Le langage Antescofo intègre le protocole de communication Open Sound Control OSC [WRFR97] pour communiquer en entrée et sortie vers d'autres environnements de programmation et logiciels musicaux ainsi que pour interagir avec des applications vidéo, lumières, microcontrôleurs, etc. La gestion des messages OSC est réalisée dans Antescofo à travers trois primitives :

- *Oscsend :* définit un canal de sortie OSC utilisé pour envoyer des messages à un ensemble de récepteurs OSC spécifiés par une adresse IP, un numéro de port et, éventuellement, un en-tête prédéfini de message. Les messages sont ensuite envoyés sur ce canal avec la même syntaxe qu'un message Max.
- *Oscrecv :* définit un canal d'entrée OSC utilisé pour traiter les messages OSC entrant dans un ensemble de ports spécifiés. En option, un en-tête de message peut être spécifié pour limiter les messages traités à ceux avec cet en-tête. Antescofo réagit à la réception de ce message en déclenchant un processus ou bien en assignant les arguments du message à des variables (qui peuvent être surveillées par un *whenever* si besoin).
- *osc_client :* définit un canal de communication bidirectionnel (entrée et sortie). Cette primitive ne spécifie pas de préfixe de message : lorsqu'un message est envoyé, le premier argument du message est utilisé comme en-tête et le reste des paramètres sont les arguments du message OSC. C'est ce mode de communication qui est adapté à la communication avec scsynth.

[106] https://github.com/supercollider-quarks/quarks

Les deux commandes *oscsend* et *oscrecv* établissent uniquement un canal unidirectionnel avec un processus externe, comme c'est le cas dans la plupart des implémentations OSC, par exemple les objets *udpsend* et *udpreceive* dans Max.

Pour communiquer avec scsynth, Antescofo utilise la commande *osc_client* : le serveur utilise les métadonnées UDP du message envoyé par Antescofo pour répondre[107] sans avoir besoin de lui spécifier à l'avance un port d'écoute qui serait utilisé par Antescofo. Une fois lancé, le nom du canal peut être utilisé pour envoyer des messages OSC à un récepteur, le serveur scsynth, tandis que le rappel est activé pour gérer les messages entrants du serveur. Le programme Antescofo agit comme un client dans une relation client/serveur bidirectionnelle.

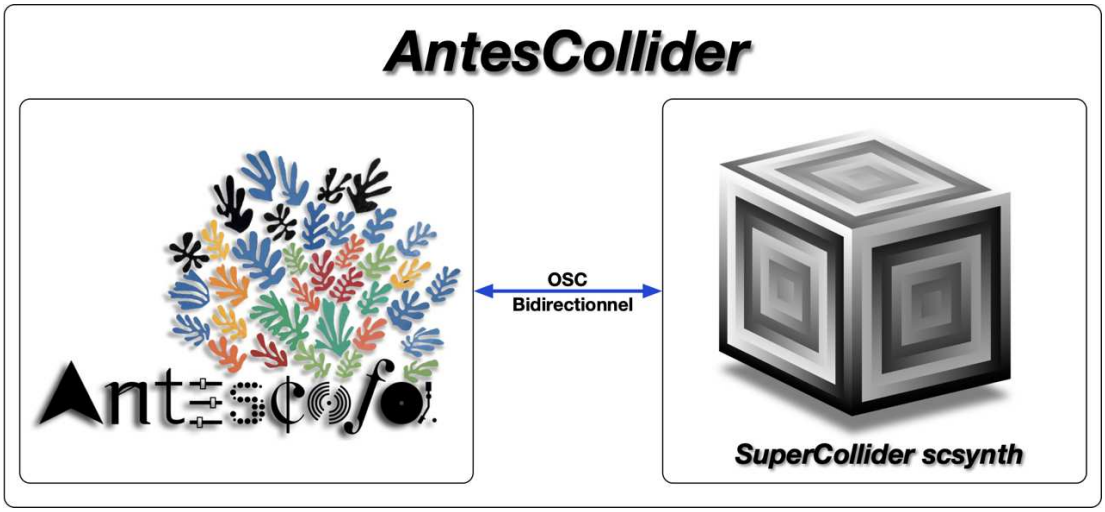


Fig. 5.1 : Architecture client-serveur de la librairie AntesCollider.

Cependant, le serveur répond aux requêtes du client mais ne s'engage pas dans une interaction non initiée par le client [FEGIDO19].

Ce pont de communication bidirectionnel OSC entre le client Antescofo et le serveur scsynth est la base du fonctionnement de la librairie AntesCollider et il va permettre tous les échanges d'ordres et des données entre les deux logiciels.

[107] W. R. Stevens, B. Fenner et A. M. Rudoff, *UNIX Network Programming: The Sockets Networking API*, Addison-Wesley Professional, 2004, vol. 1.

```
1   // Lance une instance de SuperCollider scsynth
2   // et demande d'écouter les demandes du client dans le port 57110
3   _ := @system("scsynth -u 57110 &")
4
5   // ouvre un canal de communication OSC en mode client vers SuperColider
6   // la variable $server reçoit la réponse du serveur scsynth
7   osc_client scServer localhost:57110 * $server
8
9   // vérifie la variable $server pour réagir à la communication du serveur
10  whenever ( $server )
11  {
12      print "ANSWER: " $server
13  }
14
15  // initier la connexion avec scsynth
16  // en envoyant la commande "/notify"
17  scServer "/notify" 1  // le serveur répondra "/done /notify"
```

Fig. 5.2 Exemple de création d'une communication bidirectionnelle élémentaire entre le langage Antescofo et le serveur scsynth avec la commande `osc_client`. Quand Antescofo envoie via *osc_client* le message « `/notify 1` », le serveur répond « `/done /notify` ». Cette réponse sera notifiée au programme Antescofo en assignant à la variable `$server` un tableau contenant l'en-tête et les arguments de la réponse. Une construction `whenever` est utilisée pour réagir à cette assignation et imprimera la valeur de la variable.

V.4. Structuration des traitements scsynth dans la librairie AntesCollider

Comme expliqué précédemment dans la partie consacrée au logiciel SuperCollider, l'architecture d'interconnexions entre les modules de traitements, synthèses et d'analyses audio du serveur scsynth est fondée sur la notion d'arborescence. Cette arborescence se structure en *Nodes* (nœuds). Pour scsynth, un *Node* peut être aussi bien une définition de synthèse *SynthDef* qu'un groupe *group* (un nœud de groupes sur le serveur, qui est une collection d'autres nœuds organisés sous forme de liste liée). Les nœuds sont dénotés par un identifiant unique. Les *Nodes* au sein d'un *group* peuvent être contrôlés ensemble et peuvent être à la fois des *SynthDef* (synthés, traitements et analyses) ou d'autres groupes imbriqués. Les *groups* sont donc utiles pour l'organisation et le contrôle d'un certain nombre de *Nodes* en parallèle et, lorsqu'ils sont utilisés comme cibles, ils permettent de contrôler l'ordre d'exécution (l'ordre dans la chaîne DSP).

AntesCollider utilise l'organisation et l'imbrication de *Nodes* en groupes et groupes de synthèse, ce qui permet un contrôle des paramètres des *Nodes* soit individuellement soit en groupe à différents niveaux. Par exemple, on va pouvoir modifier l'amplitude du niveau de sortie du serveur (volume master), d'un groupe de pistes (*tracks*) de synthèses, des *tracks* ou des modules de synthèses isolés.

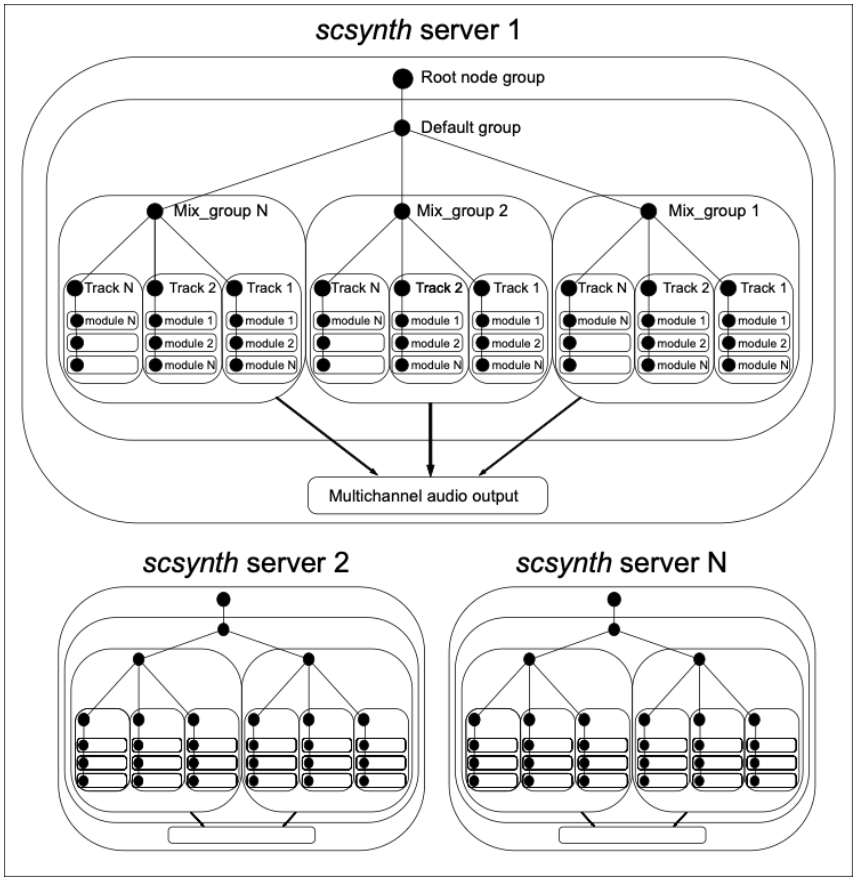


Fig. 5.3 Structure en arborescence et imbriquée en groupes dans une architecture multiserveur *scsynth*. Chaque serveur peut contenir plusieurs *Nodes* qui sont créés dynamiquement. Le *Node* racine avec un index 0 se crée au lancement du serveur et ne peut pas être détruit. Il contient le groupe « *Default* » qui à son tour contient l'ensemble des *Mix_group*. Ces groupes contiennent les *tracks* qui sont eux-mêmes des groupes qui contiennent les modules de synthèses et traitements.

Cette arborescence et les connexions audio multicanales entre les modules (avec des bus d'interconnexions audio) peut devenir complexe si on instancie plusieurs chaînes de traitements en parallèle, ce qui est souvent le cas dans les musiques interactives. La libraire AntesCollider prend en charge la construction et l'organisation des arborescences grâce aux structures de contrôle et données du langage Antescofo.

Le modèle de l'organisation des arborescences dans AntesCollider est basé sur le paradigme de pistes (*tracks*) des tables de mixage audio classique ou des stations audionumériques. Cette architecture permet d'organiser et structurer les *Nodes* des groupes et de traitement/synthèse d'une façon intuitive et facile à utiliser.

V.5. Objets de la librairie AntesCollider

Pour instancier et contrôler les *Nodes* des serveurs scsynth, la librairie AntesCollider implémente trois types d'acteurs qui sont créés, contrôlés et détruits dynamiquement. Les quatre figures suivantes illustrent ces constructions.

Sc_server

sc_server permet de créer un serveur scsynth et de contrôler tous ses paramètres. On peut créer autant de serveurs qu'on veut, même si cette fonctionnalité a été pensée pour utiliser au mieux le nombre de cœurs du processeur de l'ordinateur et profiter au maximum des capacités de calcul parallèle de la CPU.

Le serveur scsynth est séquentiel (*monothread*). L'expérience montre que si l'on dispose de *n* cœurs de calcul, il est utile de créer jusqu'à 2*n* serveurs. La répartition des instances de serveurs dans les processeurs est assurée par le système d'exploitation. Une version multithread de SuperCollider appelée SuperNova [Ble10] existe et permettrait d'utiliser un seul serveur qui accéderait à toutes les ressources de calcul, mais elle n'a pas encore été intégrée à AntesCollider.

Mix_group

mix_group implémente un container de groupes de *tracks* (les tracks sont définis ci-dessous). Ce groupe peut gérer n'importe quel nombre de canaux (par défaut 8) et peut aussi avoir un format de spatialisation spécifique de type VBAP [Pul97], DBAP [LBH09] ou HOA *(High Oder Ambisonic)* [Mal99]. Un *mix_group* en format HOA peut être créé avec un ordre de 1 à 5.

Les *tracks* instanciés dans un *mix_group* héritent de son format. Par exemple, dans *mix_group_HOA* d'ordre 5, tous les *tracks* qui seront créés en son sein auront le format HOA d'ordre 5 avec des bus audio de 36 canaux (codification en ordre 5).

Les différents *mix_group* peuvent utiliser différents formats : on peut donc mixer de la stéréophonie avec de l'ambisonie en HOA ou en VBAP en même temps dans un ou plusieurs serveurs scsynth.

Crea_track

crea_track gère un groupe qui contient des modules de synthèse et de traitements. Il peut y avoir plusieurs *tracks* dans un *mix_group* qui héritent du format du *mix_group* dans lequel ils sont créés.

Crea_aux

crea_aux *(tracks* auxiliaires*)* similaire aux *tracks* précédents mais avec la possibilité d'avoir en entrée d'autres *tracks*, analogue aux pistes auxiliaires des tables de mixage ou des logiciels de type DAW pour faciliter l'organisation et la distribution des *tracks*. Un cas d'utilisation typique est celui des faire aller plusieurs *tracks* vers un *track* auxiliaire avec un module de réverbération pour utiliser la même réverbération pour tous les *tracks* qui vont vers elle, ce qui permet un contrôle général avec une seule réverbération et optimise l'usage des ressources de l'ordinateur.

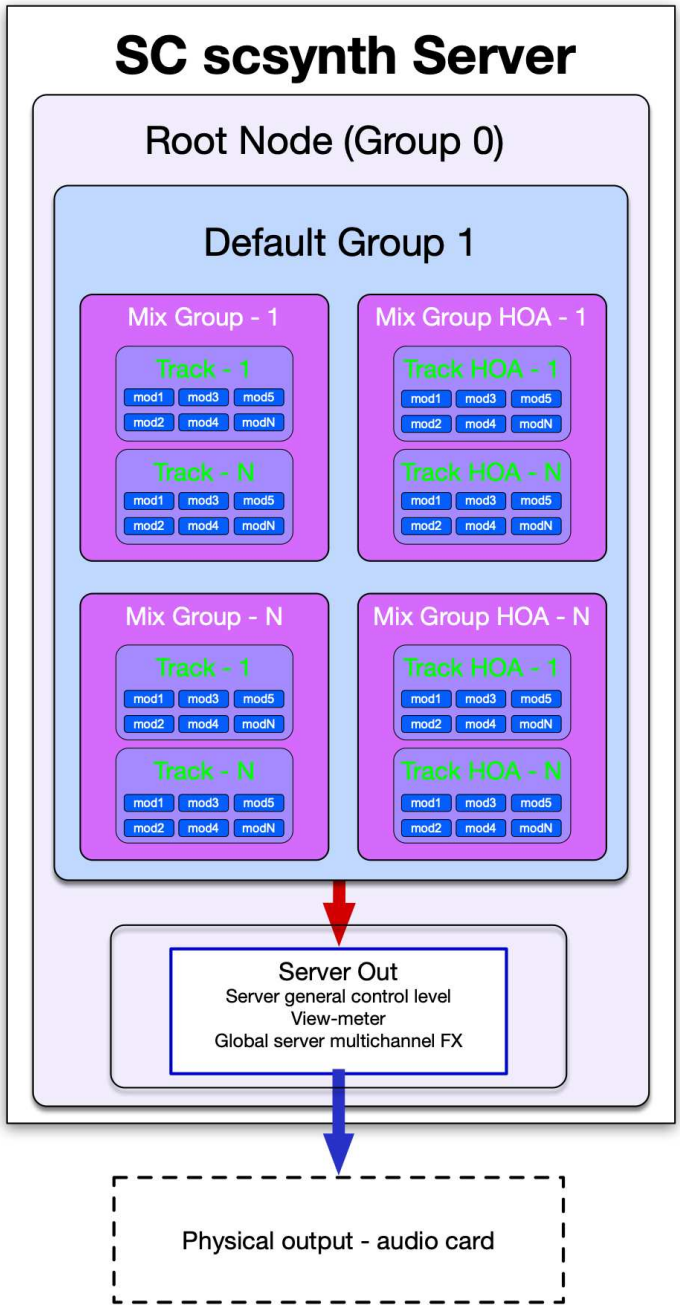


Fig. 5.4 Schéma de représentation dans la librairie AntesCollider de l'architecture imbriquée des graphes audio *scsynth*. Le « *Root Node* » (groupe racine avec index 0) contient le « *Default Group* » (index 1) qui a son tour contient les « *mix_groups* » multicanaux et HOA de la librairie. Cette architecture permet toutes les connexions audio entre les groupes et modules dans un serveur. Les sorties générales de tous les groupes vont vers un *Node* (Server Out) qui fonctionne comme un « master » de volume ou pour rajouter des traitements multicanaux génériques pour faire du *mastering* (compresseurs, réverbérations, équalisations, etc.).

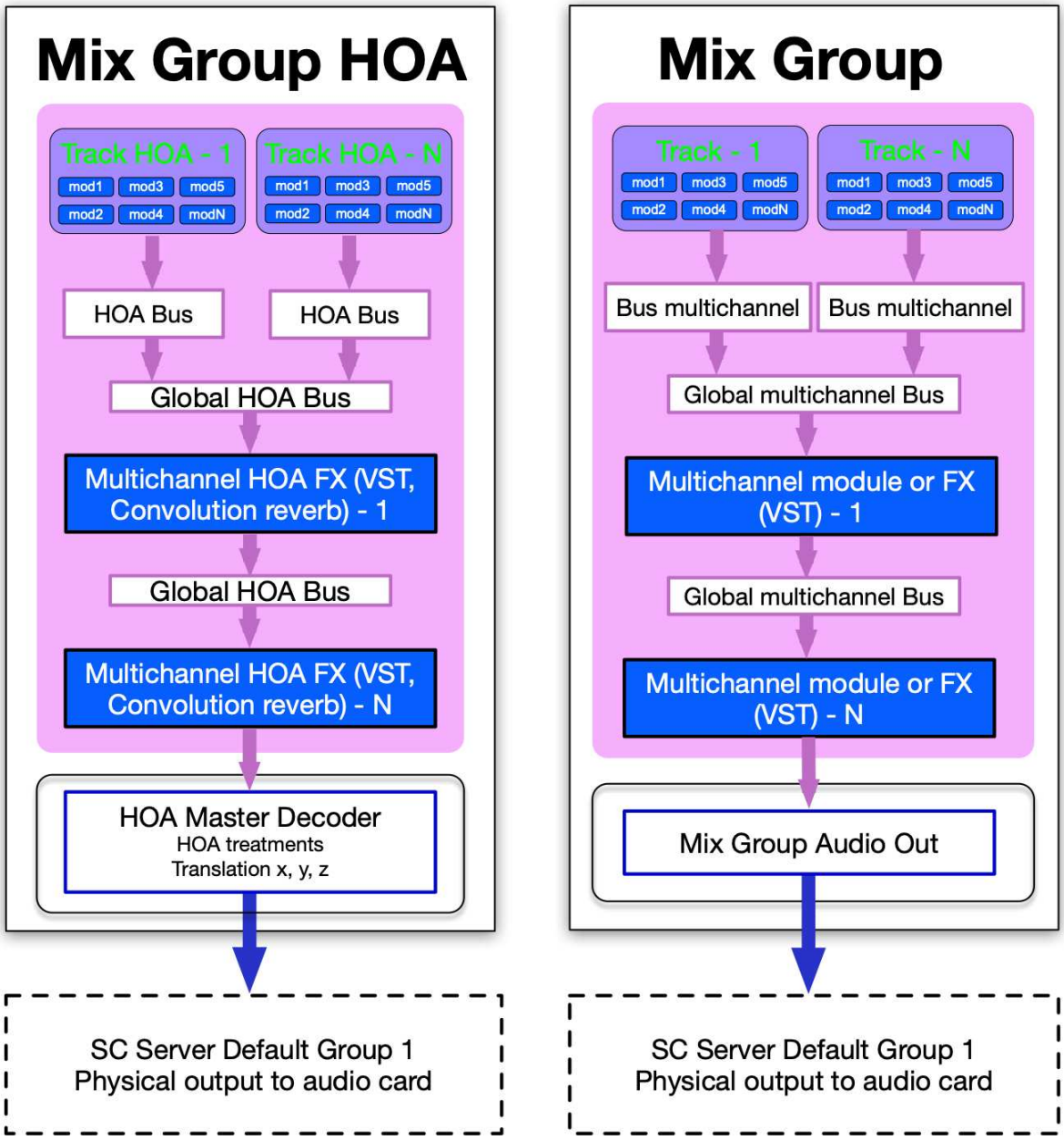


Fig. 5.5 Schéma de représentation d'un *mix_group* multicanal et HOA dans la librairie AntesCollider. La librairie prend en charge la création dynamique des *tracks* à l'intérieur des *mix_groups* ainsi que les bus de communication audio. Les *tracks* sont mixés dans un bus global qui peut être utilisé pour appliquer des traitements par *mix_group* à travers des traitements intégrés dans SuperCollider scsynth ou des VST plugins multicanaux. L'utilisation d'un *mix_group* permet un contrôle global et individuel des paramètres des *tracks* et l'organisation des traitements et des synthèses par groupes : on pourra par exemple contrôler le niveau général de chaque *mix_group,* le mettre en pause ou le tuer, ce qui arrêtera tous les *tracks* qu'il contient et supprimera les processus de la chaîne de traitement audio. La partie *decoder,* à la sortie du *track* (*HOA Master Decoder*) va permettre de faire des traitements spatiaux sur tout le *mix_group_HOA* comme des translations, rotations, etc.

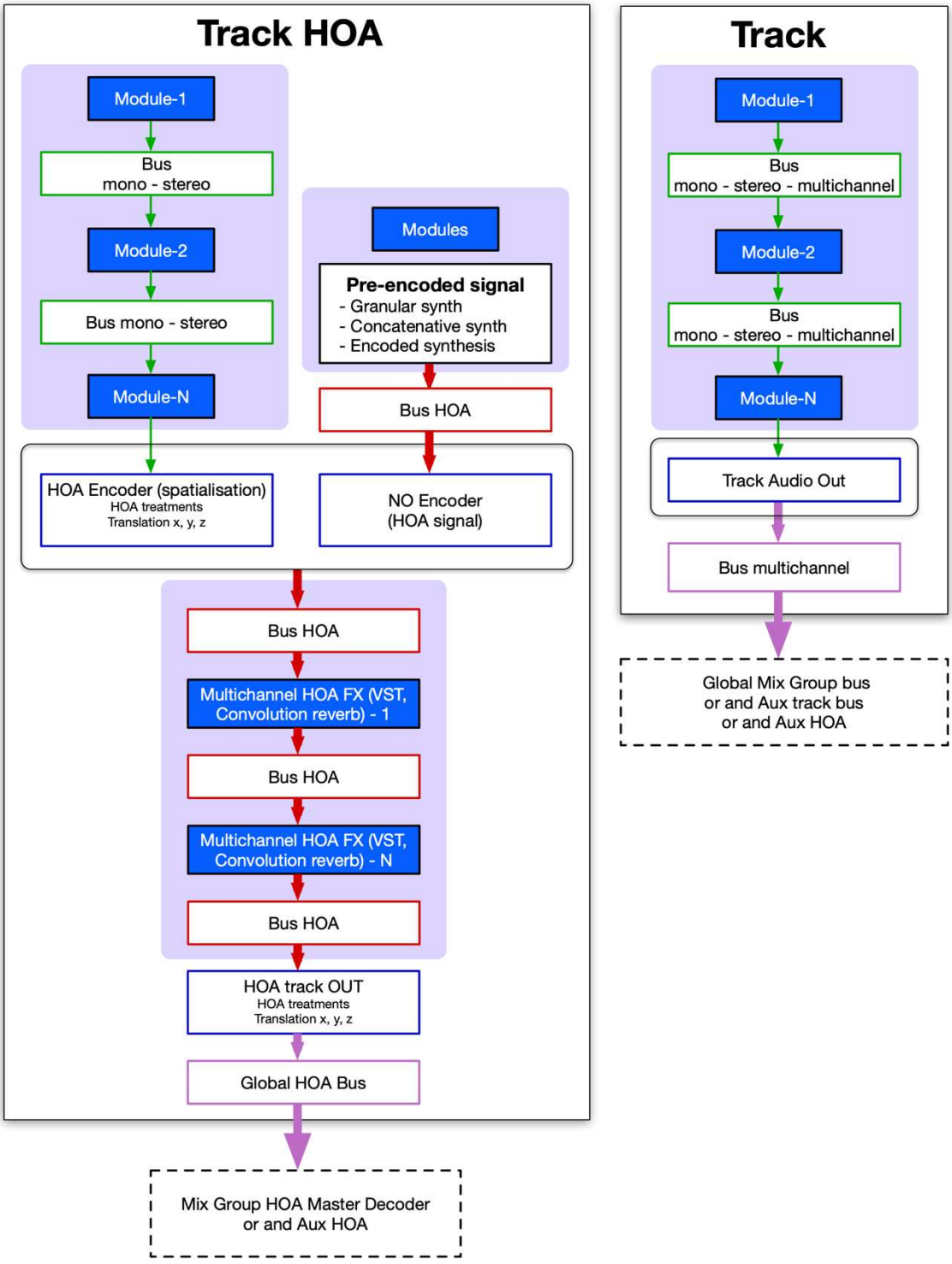


Fig. 5.6 Schéma de représentation d'un *track* multicanal et HOA dans la librairie AntesCollider. La librairie prend en charge la création dynamique des modules à l'intérieur des *tracks* ainsi que les bus de communication audio. Les modules sont instanciés à la volée et leur ordre dans la chaîne audio peut être changé dynamiquement. La librairie permet aussi d'appliquer des processus de contrôle sur chaque module ou groupe de modules en temps réel. Par exemple, on pourra appliquer des modulations de paramètres à toutes les instances d'un même module, ce qui sera le cas par exemple pour la sonification des modèles physiques. Les *tracks* HOA vont encoder des enchaînements de modules pour spatialiser leur sortie et les traiter dans le domaine ambisonique. Ils peuvent aussi traiter des signaux déjà encodés comme pour la synthèse granulaire et concatenative spatiale, entre autres.

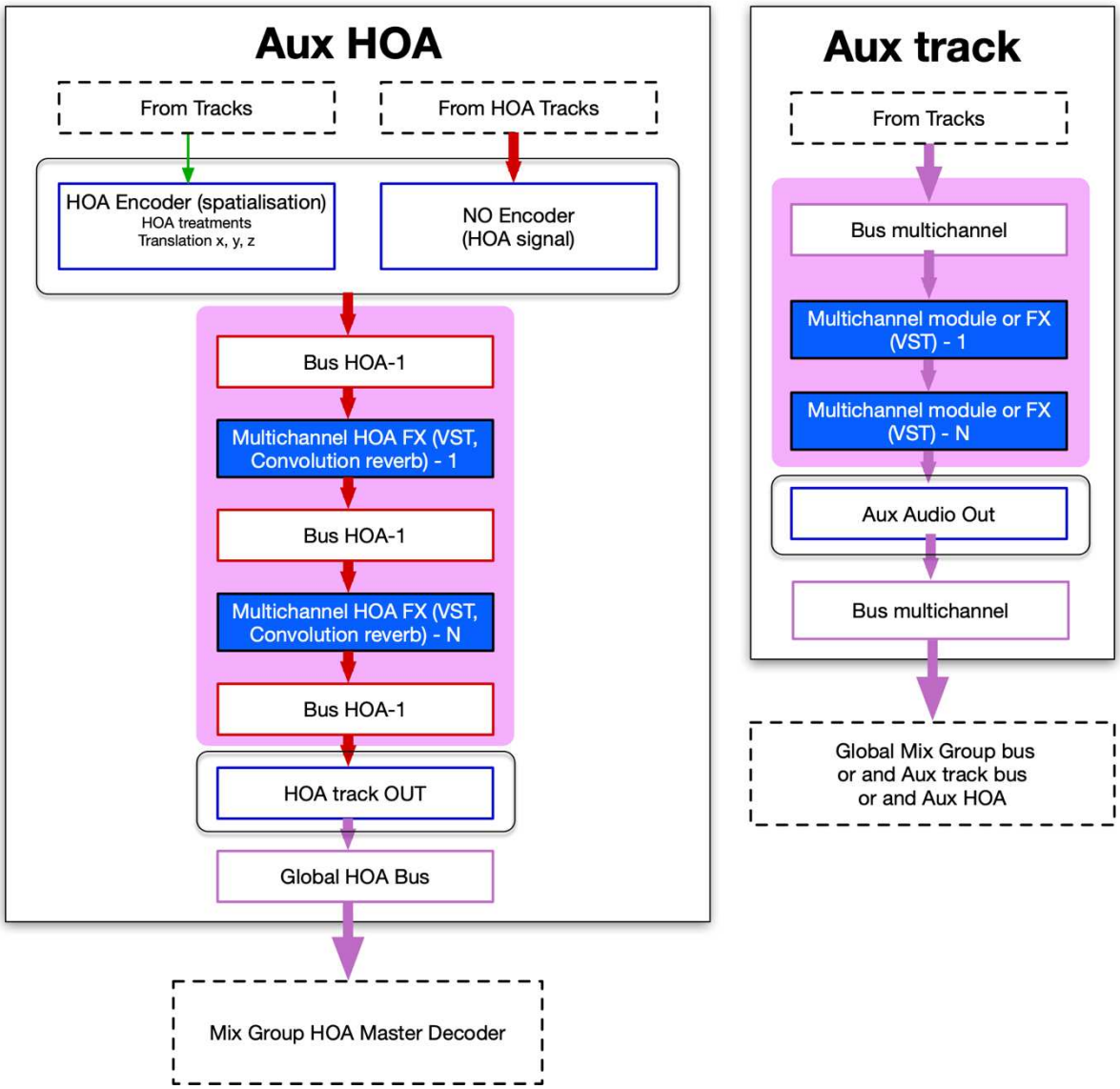


Fig. 5.7 Schéma de représentation d'un *track* auxiliaire multicanal et HOA dans la librairie AntesCollider. Comme pour les *tracks* normaux, ils permettent d'instancier des modules de traitements à la volée et d'avoir aussi en entrée d'autres *tracks*. Les *tracks* auxiliaires HOA peuvent avoir en entrée soit un *track* HOA, soit un *track* multicanal. Dans ce dernier cas, il sera encodé en format HOA pour être traité par la suite dans la chaîne HOA avec un ordre défini dans le `mix_group_HOA`. Les *tracks* auxiliaires sont très utiles aussi pour transformer des *tracks* mono ou multicanaux en *tracks* HOA, notamment pour passer de l'un à l'autre, par exemple une chaîne de synthèse qui passe d'un système standard de diffusion à un système HOA par fondu-enchaîné.

V.6. Exemple d'utilisation

Nous présentons dans cette section un exemple d'utilisation de la librairie AntesCollider, de bout en bout.

Dans une partition électronique utilisant la librairie AntesCollider, il est tout d'abord nécessaire d'inclure les fichiers de définition de la librairie à travers la primitive `@insert` : `extra_functions4_compiled.asco.txt` qui correspond aux fonctions utilisateurs compilées et `AntesCollider_lib1.asco.txt` aux objets de la librairie AntesCollider.

```
1   @insert "inserts/extra_functions4_compiled.asco.txt"
2   @insert "inserts/AntesCollider_lib1.asco.txt"
```

La première action consiste à instancier le ou les serveurs SuperCollider scsynth avec l'objet `obj::sc_server`. Cette commande va lancer les serveurs scsynth en tâche de fond avec des arguments par défaut ou spécifiés par l'utilisateur. Par exemple, dans les lignes suivantes, on va instancier trois serveurs et définir pour chacun la carte son `Digiface Dante (23938075)` comme matériel audio d'entrée et sortie.

```
13  obj::sc_server("server1", 57110, device = "Digiface Dante (23938075)")
14  obj::sc_server("server2", 57111, device = "Digiface Dante (23938075)")
15  obj::sc_server("server3", 57112, device = "Digiface Dante (23938075)")
```

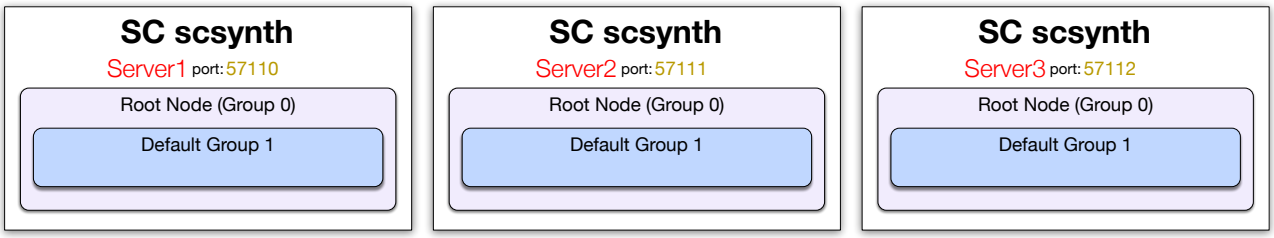


Fig. 5.8 Représentation graphique des trois serveurs scsynth instanciés avec les identifiants « Server1 », « Server2 », « Server3 », avec les ports OSC respectifs 57110, 57111, 57112. Avec l'instanciation d'un serveur, sont automatiquement créés le *Node* « *Root* » Group avec index 0 (qui contiendra tous les autres *Nodes* créés) et le *Node* « *Default* » Group avec index 1 (où toutes les synthèses, les traitements et les analyses seront créés).

Une fois les serveurs lancés, on peut commencer à instancier des groupes, des synthèses, des traitements ou des analyses.

Dans les lignes suivantes, on instancie trois groupes de mixage ambisonique dans les trois serveurs existants avec un objet `mix_group_HOA`. Le premier argument est le nom du groupe, puis le serveur dans lequel ce groupe va être créé, puis le décodeur ambisonique à utiliser (dans ce cas, le décodeur binaural IEM[108]) et puis l'ordre ambisonique à utiliser (ordre 5 dans cet exemple). Toutes les synthèses créées dans ces

[108] https://plugins.iem.at/docs/plugindescriptions/#binauraldecoder

groupes héritent de l'ordre ambisonique, dans ce cas des bus audio de 36 canaux (ordre 5) sont créés.

```
22   obj::mix_group_HOA("group_hoa1", "server1", "iem_binaural", 5)
23   obj::mix_group_HOA("group_hoa2", "server2", "iem_binaural", 5)
24   obj::mix_group_HOA("group_hoa3", "server3", "iem_binaural", 5)
```

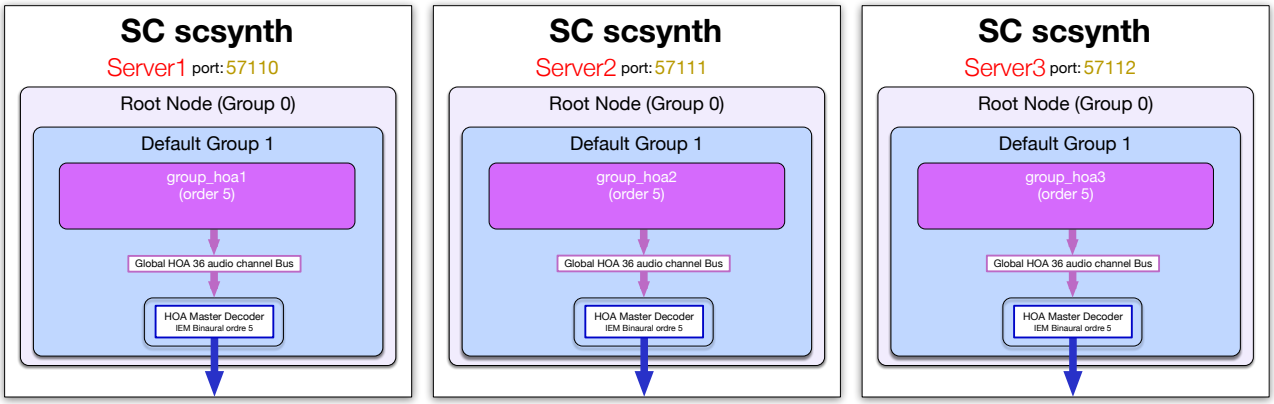


Fig. 5.9 Représentation graphique du code de l'instanciation des `mix_group_HOA` « group_hoa1 », « group_hoa2 » et « group_hoa3 » avec des décodeurs binauraux IEM d'ordre 5 dans les serveurs *scsynth*.

On peut à présent créer des synthèses, des traitements ou des analyses dans les groupes de mixage de chaque serveur. On peut créer plusieurs groupes de mixage par serveur et ils peuvent aussi avoir d'autres formats que le HOA, comme mono, stéréo, multicanal, VBAP…

Dans les lignes suivantes, on crée différentes synthèses dans chaque groupe de mixage :

```
29  // Tracks group_hoa1
30  obj::crea_track_HOA("track_hoa1", "group_hoa1", amp = 0, doppler = 0)
31  obj::crea_track_HOA("track_hoa2", "group_hoa1", amp = 0, doppler = 0)
32
33  // Tracks group_hoa2
34  obj::crea_track_HOA("sc_track3", "group_hoa2", fade_in = 1, amp = 0,
35      [["BassSynth1", "freq", 35.9, "fmrange", 1.5, "fmfreq", 7.85, "lpf", 214., "hpf", 51.2, "rq", 0.35, "amp", -10],
36       ["TRingMod1", "modfreq", 164.4, "ampmod", 0, "lpfilt", 3822.5, "hpfilt", 50, "amp", -2.6],
37       ["TAdCVerb", "revTime", 0.3]])
38
39  // Tracks group_hoa3
40  obj::crea_track_HOA("track_orbit1", "group_hoa3", [["Impulse_Pluck", "freq", 548, "freq_imp", 10., "del", 0.006, "amp", -3]])
41  obj::crea_track_HOA("track_orbit2", "group_hoa3", [["Impulse_Pluck", "freq", 600, "freq_imp", 20, "del", 0.0005, "amp", -3]])
42  obj::crea_track_HOA("track_orbit3", "group_hoa3", [["Impulse_Pluck", "freq", 300, "freq_imp", 20, "del", 0.0005, "amp", -3]])
43  obj::crea_track_HOA("track_orbit4", "group_hoa3", [["Impulse_Pluck", "freq", 400, "freq_imp", 30, "del", 0.0002, "amp", 0]])
44
45  // Insère des modules synthèses (SynthDef) dynamiquement pour le groupe "group_hoa1"
46  $tracks("track_hoa1").mod_add(["TestSynth3", "freq",32, "amp", -3])
47  $tracks("track_hoa1").mod_add(["TDust", "freq", 10, "amp", -1])
48  $tracks("track_hoa2").mod_add(["TAddic_20_8_mod_mono", preset, "psycho2"], fade_in = 5)
49  // Rajoute des VST plugins
50  $tracks("track_hoa2").mod_add(["MultiEQ"])
51  // Insère des modules à des endroits spécifiques de la chaine
52  $tracks("track_hoa2").mod_add(["TFlanger", "flangefreq", 1.822], "before", "MultiEQ")
```

L'objet `crea_track_HOA` instancie des pistes HOA. Par exemple, les lignes 30 et 31 instancient deux *tracks* vides dans le groupe de mixage « group_hoa1 » qui est dans le serveur `Server1`. De la ligne 34 à 37, on crée un *track* avec une chaîne de synthèse qui commence par `"BassSynth1"` qui va vers une modulation en anneau (TRingMod1) qui à son tour va vers une réverbération (TAdCVerb) et ses paramètres d'initialisation. Les lignes 40 à 43 définissent 4 *tracks* avec les synthèses `"Impulse_Pluck"` avec différents paramètres. Les lignes 46 à 48 rajoutent dynamiquement les synthèses

`"TestSynth3"` et `"TDust"` dans le *track* `"track_hoa1"`, la synthèse `"TAddic_20_8_mod_mono"` et un plugin VST `"MultiEQ"` dans le *track* `"track_hoa2"`. Ces modules peuvent être placés à n'importe quelle position de la chaîne du *track*, par exemple à la ligne 52, on va insérer le module `"TFlanger"` avant le plugin VST `"MultiEQ"`. La figure suivante illustre le graphe audio obtenu.

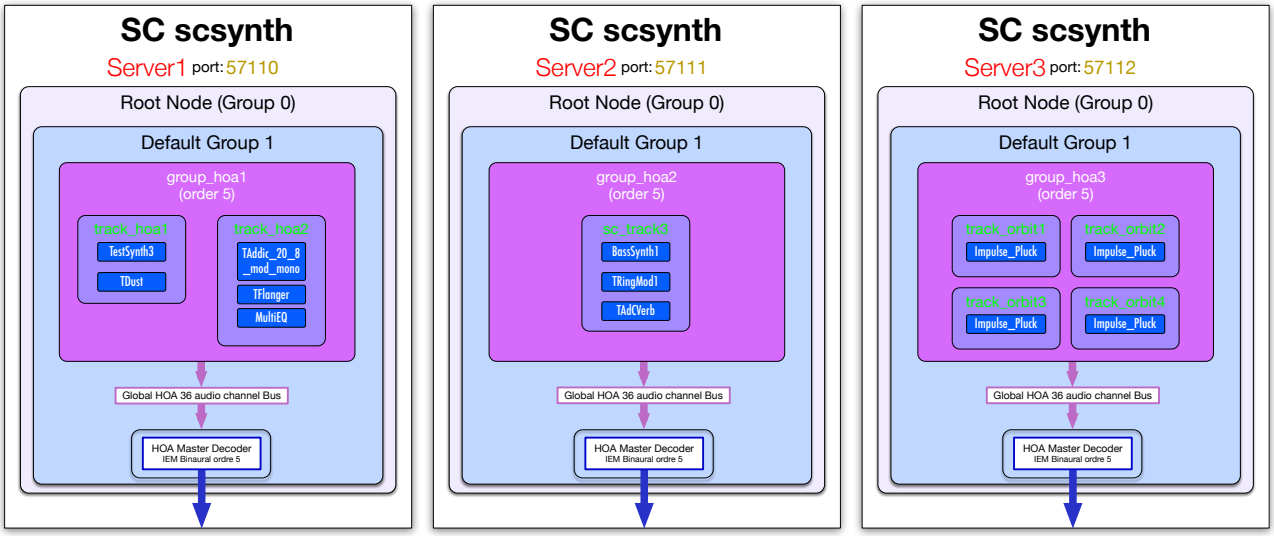


Fig. 5.10 Représentation graphique du code de l'instanciation des *tracks* HOA `"track_hoa"`, `"track_hoa2"`, `"sc_track3"` et `"track_orbit1-4"`.

Une fois les modules instanciés dans les différents groupes de mixage, il est possible de les contrôler avec différentes méthodes de l'objet `crea_track_HOA`. Dans le listing ci-dessous, la méthode `.set` permet de changer ponctuellement les paramètres des modules. La méthode `.bpf_param` utilise une NIM (*breakpoint functions*) pour faire varier continûment un paramètre. Des méthodes plus spécialisées utilisant des variations périodiques de types de LFO (comme `.rand_lfo` ou `.brown_lfo`) existent aussi. Il est aussi possible de moduler les paramètres avec d'autres algorithmes, par exemple des modèles physiques, à partir du suivi d'un instrument, ou encore à partir de données issues d'une captation gestuelle.

```
54  // set parameter (statiques)
55  $tracks("track_hoa1").set("TestSynth3", ["freq", 40])
56  $tracks("track_hoa1").set("TestSynth3", ["amp", -3])
57  $tracks("track_hoa1").set("TestSynth3", ["fc", 2000])
58
59  // set modulation (dynamiques)
60  $tracks("track_hoa1").bpf_param("TestSynth3", "fc", [2000, 5, "exp", 20, 3, 3000, 10, "exp_out", 100])
61  $tracks("track_hoa1").rand_lfo("TDust", "freq", 1, 50, 10, "linear", 60)
62  $tracks("track_hoa2").brown_lfo("TAddic_20_8_mod_mono", "ampmod", 0, 30, 2, 0, "linear", 120)
63  $tracks("track_hoa2").brown_lfo("TAddic_20_8_mod_mono", "transp", 1.305, 1.315, 0.1, 1.31, "linear", 60)
64  $tracks("sc_track3").rand_lfo("BassSynth1", "fmrange", 1, 10, 1.5, "linear", 120)
65  $tracks("sc_track3").rand_lfo("BassSynth1", "freq", 35.3, 37.2, 35.9, "linear", 120)
```

Au niveau de la spatialisation des *tracks*, il y a plusieurs méthodes correspondant à différents types d'algorithmes de trajectoires spatiales (voir en annexe les méthodes des objets). Dans les lignes de code suivantes on a quelques exemples :

```
67  // Trajectoire spatiales algorithmiques
68  $tracks("track_hoa1").hoa_pol_rota(10, 0.6, 0, 0, 360, 0, 1)
69  $tracks("track_hoa2").traj_lib_lissajou3D(loop_num = 20)
70  $tracks("sc_track3").ambi_rand_lfo_sphere2(0.5, -120, -20, 0, 360, 0, 0, "linear", 29)
71
72  $tracks("track_orbit1").orbit_traj(0, 0.06)
73  $tracks("track_orbit2").orbit_traj(0, 0.04)
74  $tracks("track_orbit3").orbit_traj(0, 0.033)
75  $tracks("track_orbit4").orbit_traj(0, 0.01)
```

La ligne 68 spécifie une rotation horizontale pour spatialiser le *track* `"track_hoa1"`. La ligne 69 utilise la librairie Trajectory Score Library pour le track `"track_hoa2"` et la ligne 70 génère une trajectoire aléatoire autour d'une sphère pour le track `"sc_track3"`. Les lignes 72 à 75 définissent quatre trajectoires orbitales avec différentes vitesses pour spatialiser les *tracks* `"track_orbitX"`.

Chaque *track* va pouvoir être visualisé à travers une interface (GUI). On va pouvoir appeler les interfaces avec la méthode `.gui`. Dans le cas des plugins VST, la méthode `.gui` va ouvrir l'interface native du VST. Pour les modules réalisés à partir de définition des synthèses (SynthDef), les interfaces seront générées dynamiquement par le biais du langage SuperCollider sclang dans l'environnement QT[109]. Les lignes suivantes permettent d'ouvrir l'interface du track `"track_hoa1"` et du plugin VST `"MultiEQ"` du `"track_hoa2"`.

```
77  // GUI
78  $tracks("track_hoa1").gui() // ouvre l'interface graphique du track dans SC sclang
79  $tracks("track_hoa2").gui("MultiEQ") // ouvre l'éditeur du plugin VST
```

V.7. Arguments des objets de la librairie

Les arguments fournis à l'instanciation de chaque objet définissent sa structure et son comportement. Un objet offre aussi des méthodes pour changer son comportement une fois qu'il est instancié. Nous détaillons ci-dessous les arguments d'instanciation des trois principaux types d'objets fournis par la librairie AntesCollider.

La création (et les méthodes) impliquent beaucoup d'arguments. Il faut noter que l'instanciation d'un objet et l'appel à ses méthodes peut se faire avec des arguments nommés : l'avantage est qu'alors on peut donner les arguments « dans le désordre ». Par ailleurs, la possibilité de définir des valeurs par défaut permet de ne pas donner explicitement tous les arguments (si les valeurs par défaut son pertinentes). Ces deux possibilités simplifient grandement la création et la manipulation de ces objets.

Les méthodes des objets AntesCollider sont présentés *in extenso* dans l'annexe de cette thèse. Nous ne mentionnons ici que les arguments à la création.

[109] https://www.qt.io

sc_server

Les principaux arguments du serveur scsynth dans la librairie AntesCollider reprennent les arguments d'initialisation (paramètres de la ligne de commande) correspondante du serveur scsynth.

Arguments	Defaut AntesCollider	Description
$server_name		Nom du Server SC scsynth dans le dictionnaire (MAP) $Servers dans AntesCollider
$port		Un numéro du port OSC de communication entre Antescofo (le client) et le serveur scsynth. Numéro compris entre 0 et 65 535
$audio_bus	20 000	Nombre maximum de bus audio
$inputs	24	Nombre d'entrées physiques (dépend de l'interface audio utilisée)
$outputs	24	Nombre de sorties physiques (dépend de l'interface audio utilisée)
$sr	48 000	Le taux d'échantillonnage (*sampling rate*) préféré. Si elle n'est pas définie, l'application serveur tentera de définir le taux d'échantillonnage de l'interface audio
$device	"Built-in Microph"	Nom de l'interface audio à utiliser par le serveur SC scsynth. Si scsynth ne trouve pas l'interface définie, il utilisera celle par défaut du système d'exploitation
$block_size	64	Nombre d'échantillons à calculer dans une période de contrôle
$num_buffers	8 192	Nombre maximum de buffers (tampons d'échantillons) audio à utiliser
$max_num_synth_defs	4 096	Nombre maximum de synth_defs à charger lors du lancement du serveur scsynth.
$max_Nodes	16 384	Nombre maximum de *Nodes* (groupes et Synth_Def) de l'arborescence
$memSize	20 97 152	Le nombre de kilo-octets de mémoire en temps réel alloués au serveur. Cette mémoire est utilisée pour allouer les synthèses et toute mémoire que les UGens eux-mêmes allouent

Arguments	Defaut AntesCollider	Description
$numWireBufs	256	Le nombre maximum de tampons (buffers) qui sont alloués pour interconnecter les UGens (à ne pas confondre avec les buffers audio ou tampons d'échantillons globaux). Cela fixe la limite de la complexité des SynthDefs qui peuvent être chargés au moment de l'exécution.
$max_logins	8	Un nombre entier indiquant le nombre maximum de clients qui peuvent recevoir simultanément des notifications du serveur.

Arguments des mix_group

Arguments	Default	Description
$group_name		Nom du groupe multicanal dans le dictionnaire (MAP) $Groups dans AntesCollider
$server_name		Nom du server scsynth dans lequel le groupe va être instancié
$num_channels	8	Nombre des canaux audio du groupe
$out_offset	0	Décalage de tous les bus audio de sortie
$pan	false	Type de système de panoramique à utiliser (Panning, VBAP, DBAP)
$destination	<undef>	Permet d'envoyer les bus de sortie audio du groupe multicanal vers un groupe *mix_group_HOA* pour passer de l'un à l'autre (par exemple du stéréo vers HOA et/ou vice versa)
$pan_init	-1	Initialisation de la position spatiale « panning » du groupe

Arguments des mix_group_HOA

Arguments	Default	Description
$group_name		Nom du groupe multicanal dans le dictionnaire (MAP) $Groups dans AntesCollider
$server_name		Nom du server scsynth dans lequel le groupe va être instancié

Arguments	Default	Description
$decoder		Nom du décoder à utiliser, par exemple « studio1 » qui correspond à la configuration du Studio 1 de l'Ircam
$order		Ordre Ambisonic à utiliser de 1 à 5 pour la librairie SC-HOA qui est utilisée dans AntesCollider
$lev_name	"hrir_christophe_lebedev50"	Nom du filtre HRIR pour un rendu ambisonique pour écouter sur casque

Argument des crea_track

Arguments	Default	Description
$track_name		Nom du *track* dans le dictionnaire (MAP) $tracks dans AntesCollider
$mix_group		Nom du mix_group dans lequel le *track* sera créé
$synth_collection	[ ]	Tableau de synthèses et traitements dans l'ordre d'apparition, le premier sera la plus haut dans la chaîne audio
$fade_in	0.01	Durée en temps relatif de la rampe d'entrée fade in
$amp	0	Amplitude générale du *track* en dB
$num_inputs	1	Nombre de canaux en entrée

Argument des crea_track_HOA

Arguments	Default	Description
$track_name		Nom du *track* dans le dictionnaire (MAP) $tracks dans AntesCollider
$mix_group		Nom du *mix_group* dans lequel le *track* sera créé
$synth_collection		Tableau de synthèses et traitements dans l'ordre d'apparition, le premier sera la plus haut dans la chaîne audio

Arguments	Default	Description
$fade_in		Durée en temps relatif de la rampe d'entrée fade in
$amp	0	Amplitude générale du *track* en dB
$spk_radius	1.07	Rayon de la disposition des enceintes du système de reproduction
$doppler	1	Activation (1) ou désactivation (0) de l'effet Doppler du *track*
$encoder	true	Nom du mix_group dans lequel le *track* sera créé
$in_channels	"mono"	Tableau de synthèses et traitements dans l'ordre d'apparition, le premier sera la plus haut dans la chaîne audio
$xyz_init_pos	[0, 0, 0]	Durée en temps relatif de la rampe d'entrée fade in

V.8. Les autres ressources de la librairie AntesCollider

Fonctions utilisateurs

Comme exposé dans le chapitre sur le langage Antescofo, il est possible de créer des fonctions utilisateurs externes au langage par les primitives `@fun_def`. Plusieurs fonctions font partie de la librairie AntesCollider. Ces fonctions ont un intérêt indépendamment de la librairie et peuvent aussi être utilisées dans d'autres contextes.

Les fonctions de la librairie AntesCollider sont présentées dans l'annexe de cette thèse.

Processus

Plusieurs processus ont été développés pour accompagner et compléter la librairie AntesCollider.

Les processus de la librairie AntesCollider sont présentés dans l'annexe de cette thèse.

Objets

Outre les trois objets principaux déjà présentés, AntesCollider en propose d'autres pour répondre à des besoins variés, notamment pour la synthèse granulaire spatiale.

Les objets de la librairie AntesCollider sont présentés dans l'annexe de cette thèse.

V.9. La visualisation graphique et le contrôle interactif des chaînes de traitements audio

Les interfaces graphiques (GUI) sont aujourd'hui indispensables dans la plupart des logiciels musicaux, elles permettent à l'utilisateur de représenter et de contrôler facilement et ergonomiquement les éléments de l'électronique. Elles permettent aussi de naviguer simplement et efficacement dans les paramètres de l'électronique pour intégrer ensuite les résultats de ces expérimentations à l'écriture de l'électronique.

La librairie AntesCollider crée et gère ces interfaces graphiques mais délègue leur réalisation au langage SuperCollider sclang (qui sert ici de couche graphique). L'idée, en accord avec le caractère dynamique des traitements, est de créer les interfaces graphiques à la volée, quand elles sont nécessaires. La création dynamique de l'interface se fera surtout pendant le processus de composition et d'expérimentation dans le but d'explorer les paramètres des traitements et pour les enregistrer en tant que préréglages (*presets*) ou encore, comme dans les DAW, pour créer des mouvements continus à travers des fonctions (*breakpoint fonctions* ou NIMs dans le langage Antescofo). Cette approche dynamique permet d'optimiser les ressources du fait que les interfaces ne seront créées que si nécessaire et ne seront pas incluses implicitement dans le programme comme c'est souvent le cas dans des patchs de concert réalisés avec Max ou Pd.

Les interfaces graphiques créées et contrôlées depuis AntesCollider reposent sur le schéma MVC (Model-View-Controller) :

- Le modèle (*Model*) correspond aux constructions de la librairie et à leur miroir dans le serveur scsynth. C'est par exemple un nombre flottant utilisé pour représenter une fréquence.
- La vue (*View*) se réfère à l'objet graphique (GUI) qui représente le modèle pour l'utilisateur.
- Le Controller permet de lier une ou plusieurs vues à un modèle. Il permet d'informer le modèle d'un changement via l'interface graphique et inversement, de notifier les interfaces graphiques d'une mise à jour du modèle.

Le schéma MVC permet d'avoir plusieurs vues et contrôles attachés à un seul modèle, ce qui permet de partager les visualisations des mêmes objets depuis plusieurs instances et différentes vues des interfaces graphiques. Par exemple, si un utilisateur utilise un *slider* pour modifier un paramètre spécifique d'une synthèse, le modèle va être mis à jour et les contrôleurs du paramètre informeront toutes les instances graphiques de cette synthèse du changement, ce qui permettra de refléter les changements dans toutes les interfaces graphiques qui réfèrent à ce paramètre. Cette fonctionnalité est très importante puisqu'on peut avoir dans la librairie plusieurs façons de visualiser l'état des paramètres des traitements.

La figure ci-dessous est une copie d'écran d'une interface créée dynamiquement pour contrôler un module de synthèse additive.

Le code sclang qui permet de créer cette interface est donné après la figure. Il illustre la définition d'une interface graphique dans le langage sclang pour contrôler graphiquement la synthèse *SynthDef* `"TAddic_20_8_mod"` (un module de synthèse additive avec 20 x 3 partiels avec modulation d'amplitude par partiel). L'objet `TAddic_20_8_mod` hérite de la classe `Tmodule2` pour la création automatique des interfaces et leur intégration au système. La méthode `metadata` de cet objet (lignes 44 à 79) décrit tous les éléments graphiques comme les sliders, les multisliders, les boutons…, ainsi que leur mapping (définie par l'objet `ControlSpec`) pour déterminer les limites, les pas, la valeur initiale, le type de variations (`CurveWarp`, linéaire, exponentiel…) et l'unité (dB, Hz…).

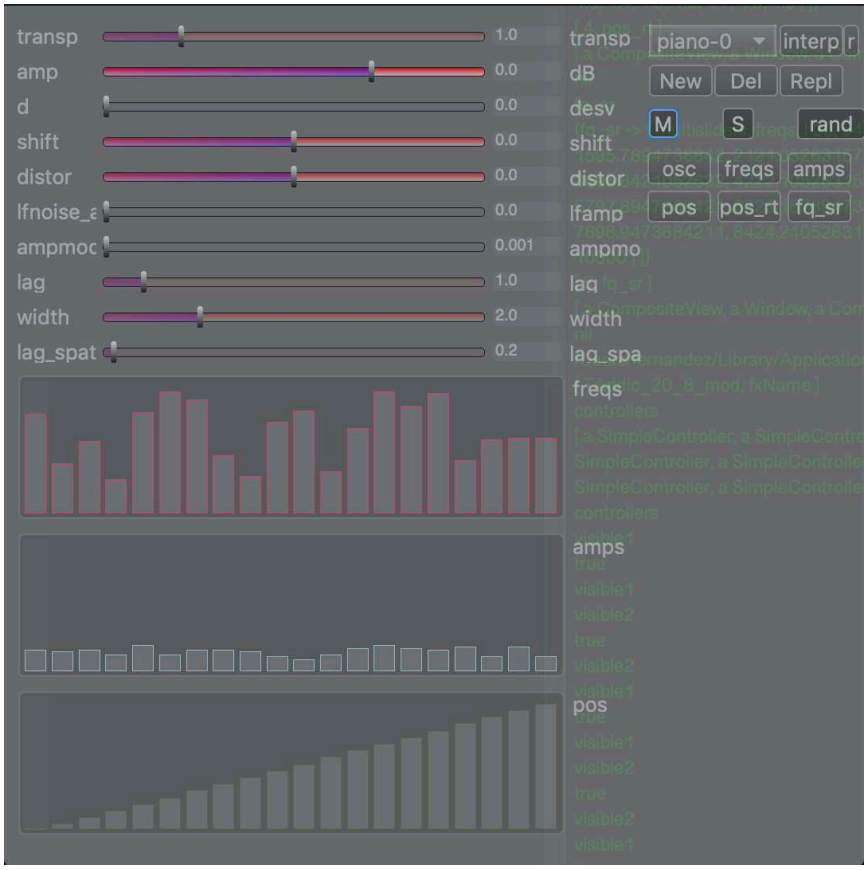


Fig. 5.11 Interface créée automatiquement et dynamiquement à partir de la description d'interface de la méthode metadata. Toutes les interfaces héritent d'un même objet qui va permettre de les organiser spatialement et d'avoir les possibilités de créer des presets, de faire des interpolations de presets, de moduler les paramètres de façon aléatoire, de mettre le module en solo ou mute…

```
TAddic_20_8_mod : Tmodule2 {

   classvar n = 20; //numero de componentes
   // : TEffectModule
   *def
   {
      ^SynthDef(\TAddic_20_8_mod, {|out = -1, outbus = -2, transp = 1.0, amp = 0, d = 0, shift = 0, distor = 0, lfnoise_amp = 0, lag = 1, ampmod = 0, gate = 1, free = 1, matrix_ramp = 0.01, width = 2, orientation = 0, lag_spat = 0.2|

         var freqs, amps, pos, sig, fqs, conca, len, lista, distarray, offset, envgate, envpause;

         envgate = EnvGen.kr(Env.asr(matrix_ramp, 1.0, matrix_ramp, \welch ), free, doneAction:2);
         envpause = EnvGen.kr(Env.asr(matrix_ramp, 1.0, matrix_ramp, \welch ), gate, doneAction:1);
         //offset = Line.kr(0, -0.02, 60);
         distarray = Array.series(20, 1);
         freqs = \freqs.kr(Array.fill(20, { [(40 + 75.rand) * (0.50.midiratio),(40 + 75.rand)].choose.midicps}));
         amps = \amps.kr(Array.fill(20, { 0.02 + 0.07.rand }));
         pos = \pos.kr((Array.series(20, 0)*(2/20)));
         sig = Mix.fill(20*3, { |i|
            var ampc;
            ampc = AmpCompA.kr(freqs.wrapAt(i));

            PanAz.ar(8, SinOsc.ar(Lag.kr((freqs.wrapAt(i)*transp) + (Rand(-2.0, 2.0) * d  + shift) * (distarray.wrapAt(i) ** distor), (0.5 + 2.0.rand)*lag) , 0, Lag.kr(amps.wrapAt(i) , ((0.5 + 2.0.rand)*lag)) * SinOsc.ar(ampmod, 0, 0.5, 1) * max(0, LFDNoise3.kr(lfnoise_amp,1.0))), Lag.kr(pos.wrapAt(i), lag_spat), 1, width, orientation) * ampc;
         });
         fqs = (freqs*transp+shift) * (distarray ** distor);
         conca = [fqs, amps].flop;
         len = conca.size*2;
         lista = conca.reshape(len);
         SendReply.kr(Impulse.kr(30), '/mi-list', Lag.kr(lista, lag));
         sig = 0.25 * sig * envgate * envpause;
         Out.ar(out, sig*amp.dbamp.lag(1));
      }).load;
   }
   // TAddic_20_8_mod.def
   *metadata
   {
    ^(metadata: (
         synthdefname: "TAddic_20_8_mod",
         type: \gen,
         main: \transp,
         sliders: [
            \transp -> ControlSpec(0.001, 5, \lin, 0.001, 1, \transp),
            \amp -> ControlSpec(0.ampdb, 2.ampdb, \db, 0, 0, \dB),
            \d -> ControlSpec(0, 200, \lin, 0.1, 0, \desv),
            \shift -> ControlSpec(-1200, 1200, \lin, 0, 0, \shift),
            \distor -> ControlSpec(-2, 2, \lin, 0.001, 0, \distor),
            \lfnoise_amp -> ControlSpec(0, 20, \lin, 0.001, 0, \lfamp),
            \ampmod -> ControlSpec(0, 20, \lin, 0.001, 0.001, \ampmod),
            \lag -> ControlSpec(0.001, 10, \lin, 0.001, 1, \lag),
            \width -> ControlSpec(0, 8, \lin, 0.001, 2, \width),
            \lag_spat -> ControlSpec(0.001, 10, \lin, 0.001, 0.2, \lag_spat)
         ],
         multisliders: [
            \freqs -> [ControlSpec(20, 10000, \exp, 0.01, 20, \freqs), n],
            \amps -> [ControlSpec(0, 2, \amp, 0.01, 0.02), n],
            \pos -> [ControlSpec(0.0001, 2, \lin, 0, 0.3, \vol), n]
         ],
         \buttons: [ //[<\que controla>, <\parametro>, <action (funcion, listas, elementos,etc)>]
            \osc -> [\osc, n],
            \freqs -> [\multisliders, \freqs, {
               var freq_min = 40, freq_max = 75; //en midi quart de ton
               [(freq_min + freq_max.rand) * (0.50.midiratio),(freq_min + freq_max.rand)].choose.midicps}, n],
            \amps -> [\multisliders, \amps,{ rrand(0.0, 2.0) }, n],
            \pos -> [\multisliders, \pos, { rrand(0.0, 2.0) }, n],
            \pos_rt -> [\multisliders, \pos, Array.series(n, 0)*(2/n)],
            \fq_sr -> [\multisliders, \freqs, Array.series(n, 1).linlin(1, n, 20, 10000)]
         ]
      ))
   }
}
```

Fig. 5.12 Exemple de la description d'une interface graphique dans le langage SuperCollider (cf. texte).

V.10. Le visualisateur *antescollider_spatial_interface*

En complément à la librairie AntesCollider, j'ai aussi développé au cours de ce travail de thèse une interface graphique pour visualiser les mouvements des sons dans l'espace générés algorithmiquement à partir du langage Antescofo ou pour visualiser les résultats des modèles physiques que j'utilise pour le contrôle de différentes synthèses sous AntesCollider.

Cette interface appelée antescollider_spatial_interface permet la visualisation des mouvements des sources ou des masses sonores comme la synthèse granulaire et concatenative spatiale. Le logiciel, programmé en C++ utilise la couche openFrameworks pour la création d'images et de vidéos en temps réel. Les données sont transmises par Antescofo à l'interface en utilisant le protocole OSC.

Ce visualisateur permet aussi quelques interactions utilisateurs permettant de contrôler dynamiquement depuis l'interface les processus de synthèses spatiales en envoyant des messages OSC ad hoc directement au langage Antescofo.

Ce visualisateur propose différents modes de visualisation des données reçues :

- **trajectories** : pour visualiser des trajectoires réalisées algorithmiquement dans AntesCollider ;
- **grains** : pour visualiser différentes couches de synthèses granulaires. Chaque couche peut avoir une couleur déterminée. Seulement la position et la durée du grain sont affichées. Dans le futur, on pourrait aussi imaginer une visualisation avec d'autres caractéristiques du son comme le centroid, la loudness, etc. ainsi que d'autres interactions pour l'expérimentation et le jeu en live ;
- **tmolecules** : pour visualiser des modèles physiques particulaires dans l'espace (jusqu'à plusieurs centaines de particules) ;
- **list3D** : pour visualiser une liste construite avec des position en x, y et z. La taille de la liste peut varier dynamiquement.
- **concat3D** : pour visualiser et contrôler des têtes de lectures spatiales dans un système de synthèse concatenative en 3D.

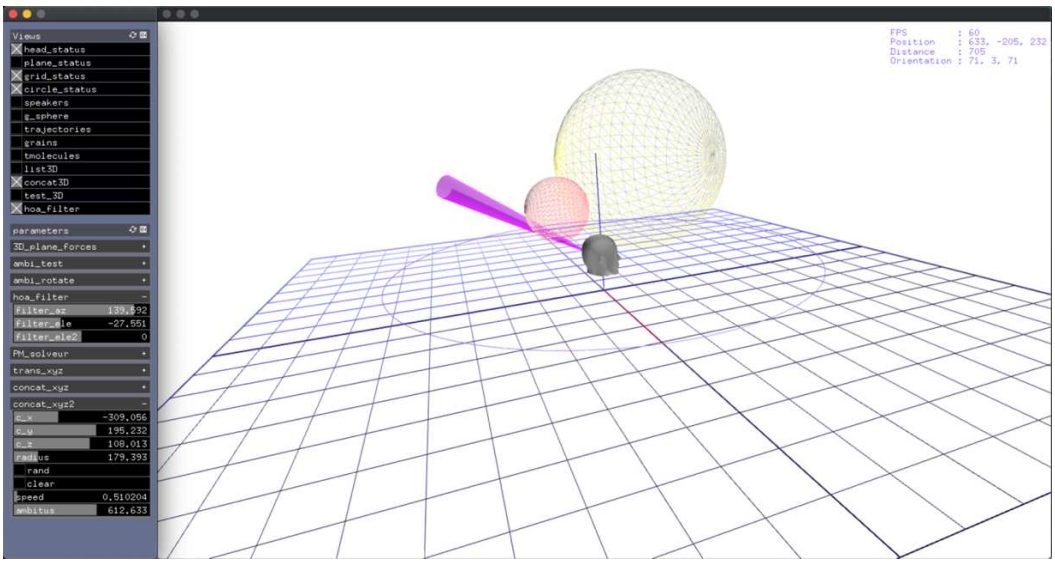


Fig. 5.13 Capture d'écran du logiciel *antescollider_spatial_interface* pour la visualisation en 3D des mouvements spatiaux.

V.11. Œuvres réalisées avec AntesCollider

La librairie AntesCollider a été validée à travers plusieurs de mes créations réalisées pendant ce travail de thèse et par la réalisation de l'électronique de pièces écrites par un autre compositeur :

- ***Dispersion de trajectoires***, pour saxophone baryton et électronique.
- ***Estremo d'ombra***, pour flûte, saxophones, trombone, alto, contrebasse et électronique (composition de Lara Morciano).
- ***Hypersphères***, Projet GeKiPe (HEM de Genève) pour un performeur, captation gestuelle et vidéo générée en temps réel.
- ***Homotopy***, pour percussions, captation gestuelle et électronique.
- ***Philiris***, pour piano, captation gestuelle et électronique (composition de Lara Morciano).
- ***Taygeta***, pour percussions transducteurs et captation gestuelle (composition de Lara Morciano).
- ***La curvatura del cristal encantado***, œuvre électroacoustique 16 pistes.
- ***Lyphira***, pour piano, transducteurs et captation gestuelle (composition de Lara Morciano).
- ***Curvatura II***, œuvre électroacoustique en HOA, entièrement écrite dans le langage Antescofo.
- ***Las Pintas***, audiovisuel en temps réel et en HOA.

Certaines de ces pièces seront détaillées dans les chapitres suivants.

J'ai aussi présenté AntesCollider à ICMC 2019 dans l'article « AntesCollider: control and signal processing in the same score » [FEGIDO19] qui a obtenu le **ICMA Audience Award for Best Paper Presentation**.

V.12. Futurs développements

Développements à court terme

À court terme, le développement de la librairie AntesCollider vise à intégrer complètement tous les éléments du traitement du signal dans la partition électronique.

Concrètement, cela implique de permettre la spécification des définitions de synthèse SynthDef pour la création des synthétiseurs, traitements et analyses audio directement dans Antescofo. Pour l'instant, les définitions de synthèse se font dans le langage SuperCollider sclang et non pas directement dans la partition électronique Antescofo. En effet, les définitions des synthèses peuvent être envoyées directement depuis le client vers les serveurs SuperCollider via OSC en format binaire.

Une autre fonctionnalité utile (qui s'appuierait sur la précédente) serait d'intégrer les possibilités de la librairie JIT pour créer et modifier des graphes audio dynamiquement en *live coding*. On pourrait ainsi explorer et prototyper facilement différents types de synthèses sans devoir les précompiler à l'avance.

Une autre intégration sera aussi d'exploiter les possibilités de rendu *off-line* (*Non-Real-Time*) de scsynth pour faire de la synthèse avec précision à l'échantillon (*sample-accurate*). Ce type de rendu est utilisé aussi quand le processeur de l'ordinateur n'arrive pas à calculer le son en temps réel, comme des rendus complexes de spatialisation d'ordre HOA élevé avec des réverbérations par convolutions multicanales ou pour la création de synthèses audio algorithmiquement par échantillons (par exemple à partir de modèles physiques complexes, d'algorithmes stochastiques, de méthodes d'apprentissages ou de résolutions de contraintes, etc.). La flexibilité au niveau temporel du langage Antescofo devrait permettre de réaliser des séquences complexes avec une grande qualité sonore. Cette intégration pourrait aussi être utilisée pour faire de la CAO avec du traitement par lots (*batch processing*) pour la génération des matériaux électroacoustiques algorithmiquement ou à partir des descriptions de plus haut niveau comme des NIM, etc.

Au niveau de l'édition d'une partition Antescofo, l'intégration de fonctionnalités comme l'auto-complétion des fonctions, processus et objets qui listerait au vol les arguments de ces entités, serait très utile. Ces fonctionnalités permettraient de faciliter l'apprentissage du langage. Elles sont nécessaires pour faciliter l'utilisation des grandes librairies qui proposent de nombreuses entités.

Développements à moyen terme

Le visualisateur antescollider_spatial_interface continuera à être développé en intégrant de nouvelles fonctionnalités. Ce visualisateur devrait évoluer pour devenir un

véritable serveur vidéo/visuel permettant la visualisation dynamique des données (comme c'est le cas actuellement) mais aussi la création des interfaces graphiques dynamiques (dont la réalisation graphique est prise en charge aujourd'hui par sclang).

Concernant Antescofo et son langage, il y a plusieurs développements possibles qui seraient particulièrement utiles comme l'intégration d'analyses audio et gestuelles directement dans le langage aussi bien en temps réel que différé (*off-line*).

Il serait commode d'intégrer différents types d'analyse et de calcul de descripteurs. Ces descripteurs pourraient être directement associés à différents types de traitements grâce aux dictionnaires, sans devoir passer par des logiciels ou des librairies externes comme c'est le cas aujourd'hui. L'intégration de ces analyses permettrait par exemple de piloter des synthèses par descripteurs comme la synthèse concatenative ou de construire des recompositions morphologiques à partir d'analyses en temps réel ou différé avec des technique de *machine learning* par exemple.

Au niveau du suivi de partition, une fonctionnalité importante serait d'étendre la machine d'écoute afin de permettre la reconnaissance de sons autres que harmoniques (notes). La perspective de faire du suivi de modes de jeux particuliers (pour les instruments acoustiques) ou d'autres sons non harmoniques (par exemple des sons environnementaux) est particulièrement attractive pour les compositeurs mais pose toujours des problèmes complexes de recherche. Une approche polyphonique du suivi (avec un suivi en parallèle de plusieurs instruments), bien que très difficile à mettre en place, peut être une piste intéressante à développer aussi dans le futur. L'intégration du suivi de geste (ou autres types de données) dans Antescofo est aussi un champ à explorer, cette technique a déjà été expérimentée en interaction avec le *Gesture Follower* [BZS+10] de l'équipe ISMM[110] de l'Ircam dans l'environnement Max en couplage avec Antescofo [Lem19], technique qui sera utilisée dans une pièce pour ensemble et électronique qui sera créée en 2022.

Au niveau graphique, après le travail sur Ascograph qui a été abandonné et l'intégration de l'éditeur Vezér pour l'édition des courbes, différentes pistes sont en cours d'exploration comme une visualisation ou rendu visuel de l'écriture de la partition électronique dans un navigateur ou dans des environnements comme *bach*. Un des avantages de Ascograph était qu'on pouvait modifier la partition électronique directement via l'interface graphique, notamment les courbes. Cette fonctionnalité devrait être étendue à tous les paramètres. La multidimensionnalité que peuvent exhiber des partitions électroniques complexes (par exemple *Curvatura II*), avec des superpositions temporelles et procédurales, présente un grand défi pour la visualisation et l'écriture de l'électronique.

[110] https://www.ircam.fr/recherche/equipes-recherche/ismm/

Un travail d'intégration des partitions Antescofo avec le logiciel iAnalyse[111] développé par Pierre Couprie pour l'analyse des pièces interactives en temps réel dans un but musicologique et compositionnel est aussi une perspective très prometteuse.

Développements à plus long terme

Durant ce travail de thèse et en relation avec la création des outils présentés ici pour la composition et l'interaction en temps réel, j'ai plusieurs fois réfléchi à un environnement « idéal » (tout au moins pour moi) en tenant compte des possibilités de l'informatique musicale d'aujourd'hui, de la modularité et du caractère dynamique nécessaire au travail d'expérimentation et au travail d'écriture, dans le contexte de la partition électronique centralisée.

La librairie AntesCollider qui permet de créer un pont entre Antescofo et SuperCollider est un premier pas vers un tel environnement. C'est aussi un point de départ pour prototyper et expérimenter de nouveaux outils dynamiques de création, porteurs de nouvelles idées techniques et musicales.

Une dimension importante de cet environnement idéal est l'extensibilité et l'adaptabilité à différents projets de recherche en composition et interprétation de l'électronique. La possibilité d'étendre l'environnement pour intégrer des techniques émergentes comme celles issues de l'intelligence artificielle ou pour incorporer de nouvelles techniques de composition, de synthèse, de traitement et de spatialisation sonore, est une caractéristique qui permettrait de pérenniser les développements en assurant la pertinence de l'approche au cours du temps. Comme Max, qui s'appuie sur la programmation visuelle par patch, cet environnement idéal devrait pouvoir s'adapter à différents utilisateurs, chercheurs, développeurs, compositeurs/musiciens et artistes, et à différents niveaux d'expertise, des débutants jusqu'aux utilisateurs experts.

Une approche possible pour ce faire, est d'associer une représentation visuelle des programmes à la représentation textuelle, avec la possibilité de basculer d'une représentation à une autre. C'est l'approche adoptée pour les objets Max gen~ et *CodeBox*[112]. Cette double représentation devrait permettre l'expression graphique des chaînes de traitements et graphes audio, représentation plus naturelle et intuitive pour les utilisateurs non experts, et simultanément la représentation textuelle qui permet de spécifier des synthèses plus élaborées difficilement représentables manuellement dans des systèmes de type *block-diagram* (par exemple des filtres récursifs).

On pourrait ainsi réaliser un système de partitions centralisées où même les éléments de très bas niveau comme les *Unit Generators* (unités de génération, analyse et traitement audio souvent écrites en C/C++) ou des librairies entières (par exemple une librairie HOA pour la spatialisation) pourraient être inclus en tant qu'objets compilés. Ceux-ci pourraient être représentés aussi bien en code et/ou patch que dans un langage de bas niveau tel que C/C++. Tous ces éléments devraient aussi être disponibles en tant

[111] Pierre Couprie, http://logiciels.pierrecouprie.fr
[112] https://cycling74.com/tutorials/gen~-for-beginners-part-5-the-codebox-operator

que logiciel libre (open source) puisque les codes sources devraient aussi faire partie de cette partition globale en tant que inserts exportables et compilés pour n'importe quel système d'exploitation.

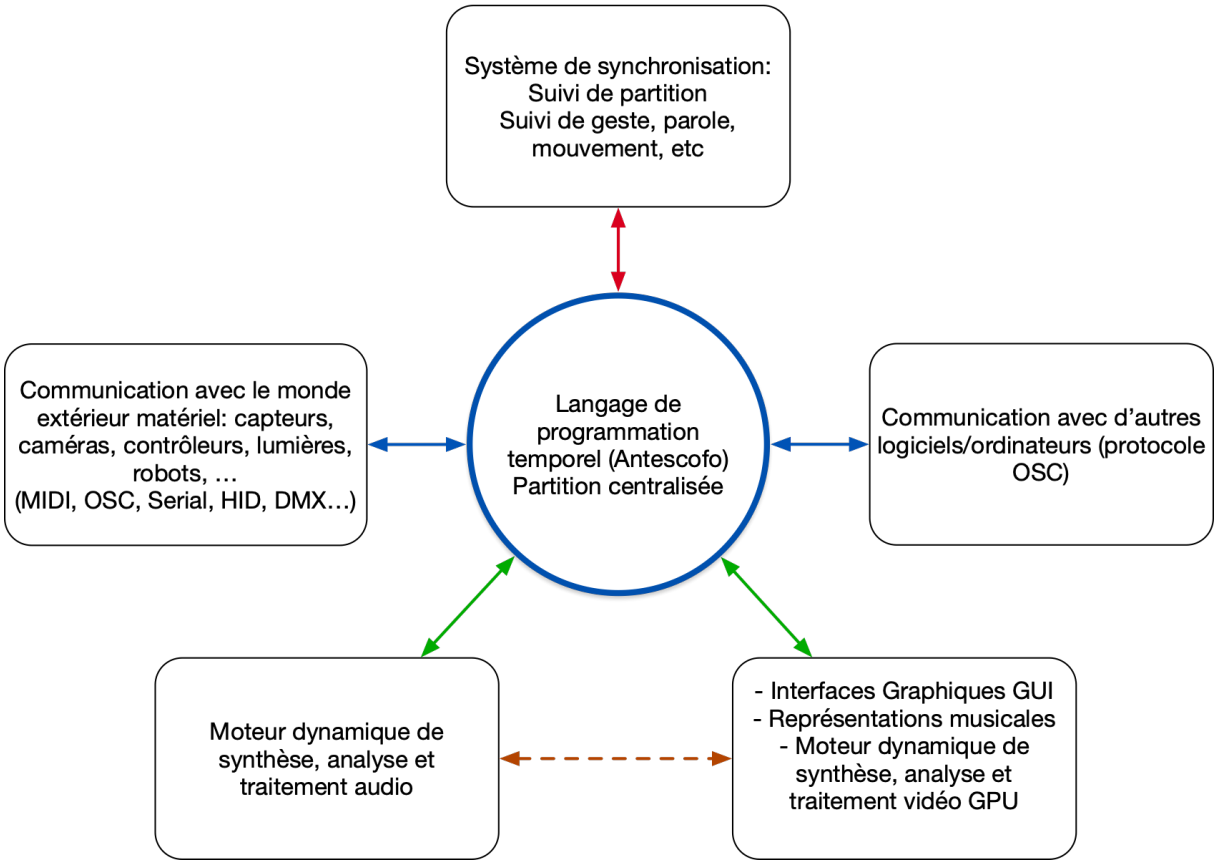


Fig. 5.14 Diagramme d'un système modulaire intégré. Au centre, la partition électronique centralisée qui contrôle et communique avec l'ensemble des éléments.

VI. Paradigmes compositionnels

> *« Il ne faut pas oublier que la division de l'octave en douze demi-tons est purement arbitraire. Il n'y a aucune raison de continuer à tolérer cette restriction. Tout comme le peintre peut créer des couleurs et varier leur intensité, le musicien peut également obtenir des vibrations sonores qui ne correspondent pas nécessairement aux tons et aux demi-tons traditionnels, mais qui varient, en fin de compte, d'un son à l'autre. Nous ne pourrons vraiment explorer l'art du son (c'est-à-dire la musique) que si nous avons des moyens d'expression entièrement nouveaux. Oublions immédiatement le piano et les restrictions mécaniques arbitraires qu'il nous impose et ne voyons pas cela comme une proposition iconoclaste. »* [Var83]

Ce chapitre a un statut un peu particulier. Il ne vise pas à présenter un travail effectué pendant cette thèse ou bien à défendre et à argumenter une idée en lien avec mon travail. Son but est de présenter quelques éléments biographiques, évoquer certaines positions artistiques qui me sont propres, et mettre en avant certaines problématiques musicales qui me préoccupent plus particulièrement. L'objectif est de mettre en contexte les créations présentées dans les chapitres suivants.

VI.I. Quelques éléments biographiques

Ma pensée musicale s'est forgée au fil des années par des études classiques qui ont commencé par le piano puis, à l'université du Chili, se sont complétées par des études générales (contrepoint, harmonie…) et par la composition.

C'est ensuite au LIMP (Laboratoire de recherche et production musicale), à Buenos Aires en Argentine, que j'ai commencé à approfondir la programmation informatique, notamment avec le langage LISP.

Après ces riches expériences, je suis venu en France étudier la composition instrumentale et mixte au CNSMD de Lyon, dans le département de composition anciennement appelé SONVS, où la programmation faisait partie intégrante du cursus, notamment dans le cadre des cours de Denis Lorrain et de Robert Pascal.

À la fin des études au CNSMD, j'ai participé au cursus de composition de l'Ircam. Il s'agit d'une année intense sur l'apprentissage de différents logiciels et techniques développés à l'Ircam, aboutissant à une pièce pour un instrument soliste et électronique en temps réel en fin du cursus.

La fabrication du son

Pendant toutes ces années de formation, mon intérêt pour les nouvelles technologies, en particulier la programmation, s'est développé progressivement. Cet intérêt s'est accompagné d'une réflexion sur la création des formes et techniques compositionnelles permettant une plus grande souplesse entre la pensée musicale et sa concrétisation sonore. Le son, comme pour beaucoup de compositeurs de la seconde moitié du XX^{e} siècle, est devenu primordial et sa création, à travers l'expérimentation technologique et son écriture fine, est aussi devenue un acte quotidien et naturel au même titre que l'écriture de notes sur une partition.

Le monde de la synthèse sonore par ordinateur s'est ouvert à moi assez tôt et j'ai tout de suite commencé à expérimenter le mélange des sons de synthèse et des sons instrumentaux pour faire émerger une musique unique et une sonorité globale. Dès mes premières compositions mixtes, mon travail s'est focalisé sur la recherche de mélanges composites avec la composition simultanée des deux parties – l'instrumental et l'électronique – même lorsque cela implique la confrontation d'éléments très hétérogènes. Dans ce cas, aucune des deux parties ne doit avoir de rôle prépondérant : chacune a ses propres moyens et va contribuer à l'expérience musicale. De même, la virtuosité de chaque partie va être comparable dans la limite des possibilités techniques et perceptives. Chacune jouera avec son potentiel. Par exemple, un instrument ne pourra pas jouer des *ribatutti* aussi vite qu'un ordinateur, et un ordinateur ne pourra pas jouer avec les subtilités et les richesses interprétatives d'un instrumentiste professionnel (du moins avec les technologies actuelles).

Avec l'intégration des nouvelles technologies, l'espace sonore dans tous ses paramètres s'ouvre à une granularité beaucoup plus fine des rythmes et des hauteurs. De plus, l'électronique permet de composer l'espace avec une souplesse, une flexibilité et des possibilités inatteignables avec les instruments acoustiques. Les sons de synthèse permettent la création de sons inouïs, par exemple sans une temporalité définie, ils peuvent jouer sans s'arrêter, se métamorphoser, suivre des trajectoires spatiales, créer des masses sonores, être étirés ou raccourcis, etc.

L'électronique, et plus particulièrement l'informatique musicale, peut être considérée comme un instrument élargi et multitâche qui va permettre de créer et transformer des sons ainsi qu'interagir avec le monde externe – instruments, médias, environnement, etc. – à travers des capteurs – microphones, capteurs gestuels, capteurs physiques, physiologiques, etc. Par ses possibilités étendues, l'électronique des musiques mixtes vient renouveler l'écriture et apporter de nouvelles idées musicales en élargissant l'univers des timbres et des morphologies sonores. La composition avec ces nouveaux instruments doit tenir compte de toutes les interactions possibles et trouver les meilleures stratégies pour les formaliser et les implémenter, aussi bien dans le cadre de la musique écrite que dans celui des musiques improvisées ou d'autres formes comme les installations sonores.

Par ailleurs, le monde instrumental acoustique porte l'aspect interprétatif, vivant et interactif du moment présent, en apportant la richesse des sonorités et une projection

du son bien différente et bien plus subtile que celle offerte par des haut-parleurs. De même, les techniques instrumentales ont évolué et l'introduction de nouveaux modes de jeux a étendu les sonorités des instruments acoustiques.

Les deux mondes interagissent pour faire émerger une musique où la relation entre l'humain et la machine est forte et interdépendante.

L'impact des courants musicaux

Différents courants musicaux se sont développés au cours du XX^{e} siècle et on a assisté à une multitude d'expérimentations aussi bien au niveau formel (par exemple avec l'introduction du hasard, de formes ouvertes ou encore de la génération algorithmique) qu'au niveau sonore. Les compositeurs d'aujourd'hui sont confrontés à toutes ces nouvelles techniques tout en étant liés à la pensée musicale élaborée durant l'histoire de la musique occidentale. Dans mon cas, je me sers de toutes les idées qui pourront m'aider à créer ma musique, à lui donner sa liberté et son expressivité.

Dans cette histoire de la pensée musicale, la musique spectrale m'a particulièrement influencé en contribuant fortement à la libération des techniques et par son approche de différents processus combinatoires ou aléatoires. Comme indiqué par Grisey, la musique se nourrit de toute la palette des sons, depuis des superpositions temporelles extrêmement complexes jusqu'à des situations d'une simplicité absolue comme une sinusoïde ou un silence dans une transformation continue. Et c'est un des avantages de notre époque que de pouvoir faire face à cette richesse sonore avec une grande liberté pour créer sa musique : les règles et contraintes au niveau compositionnel sont données par le compositeur lui-même et chacun peut s'exprimer et expérimenter comme il le souhaite, avec les moyens qu'il a à disposition. De plus, nous avons des outils de plus en plus performants et versatiles qui nous permettent de donner à la musique une grande variété et une grande latitude dans les approches, nous assurant des conditions d'une liberté encore jamais atteinte dans l'histoire de la musique.

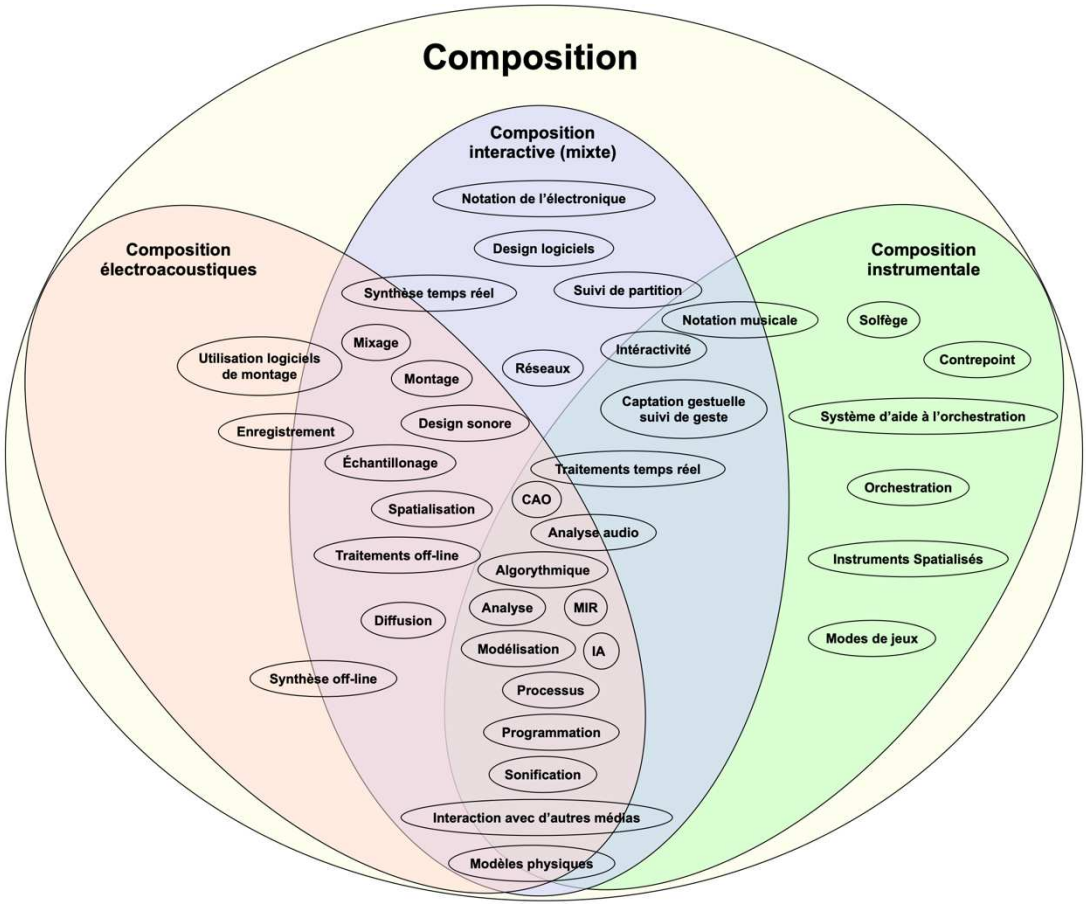


Fig. 6.1 Schéma de représentation de différents éléments techniques dont dispose un compositeur aujourd'hui pour créer la musique et le son. Ce schéma de composition délimite trois catégories : la composition instrumentale incluant ses traditions, la composition électroacoustique et la composition des musiques interactives ou mixtes.

VI.2. Une liberté issue de la diversité et de la plasticité musicale

« Je rêve les instruments obéissant à la pensée et qui, avec l'apport d'une floraison de timbres insoupçonnés, se prêtent aux combinaisons qu'il me plaira de leur imposer et se plient à l'exigence de mon rythme intérieur », The Liberation of Sound, Edgar Varèse [Var66].

« Une inépuisable diversité » comme déterminant de ma poïétique

Comme je l'ai déjà évoqué, je pense que nous avons la chance incroyable à notre époque de pouvoir accéder à un nombre presque illimité de musiques. Il suffit de parcourir les catalogues des plateformes dédiées à l'écoute musicale ou de visiter différents sites de radios sur Internet pour être sidéré par la quantité et la diversité des styles proposés. Ils ne représentent pourtant qu'une faible partie de toutes les musiques existantes. Bien entendu, à l'échelle humaine, nous ne disposons pas du temps nécessaire pour écouter toutes ces musiques : par exemple, Spotify[113] dispose d'un catalogue de plus de 60 millions de titres. Nous avons donc le choix d'aller explorer

[113] https://www.spotify.com

différentes musiques et styles musicaux selon nos envies ou nos recherches personnelles, sans jamais les épuiser. Et de fait, nous pouvons découvrir et écouter comment la musique s'exprime de différentes façons, pour différents types de personnes et à différentes occasions, que ce soit pour un événement religieux ou festif, pour un ascenseur, pour de jeunes occidentaux qui font la fête ou pour un rituel sacré d'une tribu en Amazonie, etc. La musique fait partie intégrante de l'être humain et « s'adapte » à son auditeur (et vice versa). Elle prend des formes variées et multiples en fonction du créateur ou de l'auditeur, elle deviendra pour chacun « sa musique ». On peut dire que « la perception de la musique est cognitive et culturelle » [Lev13].

Dans ma démarche compositionnelle, cette diversité de formes, de contenus et d'attendus constitue ma poïétique fondamentale. Ma liberté musicale est le produit de cette plasticité, et plus particulièrement de l'articulation de cette plasticité et du devenir du son et de « sa transformation continuelle d'énergie » [Gri08]. Elle me permet de tenter de dépasser les catégories culturelles et structurelles préétablies même si je suis bien conscient que nous sommes des êtres forgés par notre réalité culturelle, environnementale et notre espace de croyances.

Ma démarche compositionnelle ne s'appuie pas sur l'utilisation de musiques existantes, mais sur la possibilité d'accéder à cette diversité de sonorités instrumentales, mécaniques, électriques, numériques, algorithmiques, chaotiques ou statiques. Ma recherche musicale et créatrice s'en inspire et se l'approprie. Je peux alors laisser la musique s'exprimer et voyager librement dans mon imaginaire, mon écoute intérieure, sans « trop » de contraintes. Il devient théoriquement possible d'utiliser toute la palette des sons et combinaisons au niveau des timbres, des structures, des morphologies et des espaces qui existent et qui sont mis à ma disposition par les nouveaux outils technologiques.

Un déploiement entre écoute et possibilités techniques

S'instaure ainsi *un double jeu entre d'une part la perception et l'écoute, et d'autre part les possibilités techniques actuelles et tout l'héritage musical* : un double jeu qui est au service de ma liberté musicale et qui est à la base de ma conception musicale. Cette liberté se déploie au niveau du timbre, mais aussi au niveau formel où elle permet de jouer avec la perception et l'agencement de l'énergie en fonction du parcours de la musique et du son[114]. La composition musicale, pour moi, doit exister sans contraintes théoriques formelles et sans se laisser subordonner par des structures et des formalisations au niveau de la macro-forme. Plus précisément, mon processus compositionnel ne peut pas s'ancrer dans une théorie formulée a priori. Il repose sur *des idées élaborées et expérimentées dans un aller-retour avec le son*, son écoute et

[114] Par exemple, dans la pièce *Las Pintas*, l'idée est littéralement de voyager entre différents univers sonores, pouvoir se déplacer au niveau spatial et énergétique et de composer les articulations entre les univers avec différentes stratégies grâce à l'écriture de l'électronique.

l'appréhension des relations spatiales, temporelles et dynamiques entre éléments sonores.

Mais cette attention première portée à la matière sonore ne m'empêche pas d'intégrer les possibilités formelles apportées par l'utilisation de différents types d'algorithmes ou de structures induites par des procédés génératifs déterministes ou stochastiques. En effet, aujourd'hui, toutes ces techniques font partie intégrante de la palette des couleurs sonores du compositeur. L'interaction entre ces techniques et leurs articulations font partie de mes préoccupations quand elles s'inscrivent dans une intentionnalité et une nécessité musicale induites par le son lui-même. Par exemple, pour créer des masses ou des textures sonores, j'utiliserai des processus de génération aléatoires puisque les faire à la main serait maladroit, prendrait trop de temps et le résultat risquerait de ne pas être aussi caractéristique que celui réalisé par un programme informatique génératif.

L'espace de la représentation

La musique peut aussi se diffuser aujourd'hui de différentes manières. Elle s'est ouverte à un nouvel univers qui est celui de l'espace et du lieu physique de représentation. À cet égard, des pièces comme *Curvatura II* et *Las Pintas* sont des exemples de compositions électroacoustiques qui peuvent et doivent s'adapter au lieu de représentation. Il est possible de changer des paramètres de base comme le décodeur ambisonique, de nuancer les réverbérations pour prendre en compte l'acoustique de la salle et de moduler des éléments et des processus pendant la performance car ils sont générés en temps réel à partir d'une partition électronique centralisée (AntesCollider). Ainsi, la pièce pour ensemble spatialisé et électronique dont la création est prévue à l'automne 2022 pour la réouverture de l'espace de projection de l'Ircam avec son système de multidiffusion Ambisonic/WFS, pourra être rejouée dans d'autres lieux en s'adaptant à leur acoustique, au système de diffusion et à la configuration des haut-parleurs. La prise en compte de l'environnement et de l'espace physique de représentation avec ses qualités acoustiques spécifiques permet de jouer très finement sur le timbre et le son de l'œuvre (un bon exemple est offert par certaines pièces de A. Di Schipio [DiSh03]). Cette dimension spatiale peut aussi être utilisée et développée pour des installations sonores ou des pièces en interaction avec le public.

VI.3. Composition des processus sonores et de leurs interactions

L'interaction des processus sonores

La composition des pièces interactives mixtes ou uniquement électroacoustiques fondées sur des processus temps réel présente aujourd'hui plusieurs enjeux. Un des premiers concerne l'écriture et la construction des différentes interactions sonores. Par là je veux désigner principalement l'interaction des différentes couches ou sources sonores entre elles (plutôt qu'une interaction avec l'environnement). L'agencement des éléments en interaction est une tâche complexe et nécessite un environnement qui peut soutenir toutes ces interconnexions et interactions :

« *L'émergence d'une approche articulée autour du concept d'interaction généralisée (interne à l'œuvre) nous permet aujourd'hui d'envisager à la fois l'existence de passages possibles entre des dimensions disjointes du temps et la nature des non-linéarités qui découlent de leur interaction. Le problème, pour un compositeur intéressé dans l'extension d'une syntaxe à toutes les dimensions temporelles, c'est de trouver les moyens d'articuler cette complexité.* » [Vag95]

Dans ce type de musiques, différentes couches interagissent. Elles impliquent plusieurs échelles, plusieurs temporalités, et se superposent, chacune avec leurs propres modes opératoires. Le degré d'interaction entre tous ces éléments et toutes ces temporalités est très riche. L'exploration et la maîtrise de ces relations nécessitent des outils particulièrement expressifs pour mettre en œuvre et contrôler les interactions souhaitées ainsi que de nombreuses expérimentations et tests d'écoute.

Dans ce contexte, la notion de partition électronique centralisée et des librairies comme AntesCollider sont fondamentales puisqu'elles vont permettre justement d'organiser et faire interagir les différents éléments dans un même environnement, une partition électronique avec un langage expressif et flexible.

Voici un exemple permettant d'illustrer notre propos : un instrumentiste joue des rythmes en superposition et en interaction avec différents sons de synthèse granulaire qui fonctionnent, eux, à des vitesses beaucoup plus rapides. Ces synthèses granulaires peuvent en même temps se déplacer en masse dans l'espace avec une autre temporalité que celle qui génère les rythmes de la granulation. Ces différentes temporalités peuvent aussi être modulées (*accelerando*, *rallentando*, etc.) par d'autres processus internes ou générés par des éléments externes (analyses, capteurs). Leurs dynamiques s'organisent aussi par des points de rendez-vous ou d'opposition. On voit que différentes temporalités doivent coexister dans des relations complexes[115].

Processus sonores et processus informatiques

L'utilisation de différents types de processus de génération temporelle, leur dépendance à un tempo global ou à des tempi locaux et le contrôle de leurs comportements à travers leur instanciation, leur durée de vie, leur interaction et leurs modifications dynamiques pendant leur exécution nécessitent une structuration très forte.

Je rejoins ici l'approche développée par Horacio Vaggione qui propose de structurer ces processus à travers des réseaux d'objets et d'échelles temporelles. Je réalise la mise en œuvre de ces notions directement à travers la structuration en objet Antescofo et leurs méthodes qui permettent de gérer ces processus.

---

[115] Ce type d'agencement peut être réalisé avec un langage tel que Antescofo, où différents tempi peuvent être modulés par d'autres tempi (par exemple le tempo d'un processus électronique peut se régler sur le tempo d'un musicien).

Ce type de composition entre plusieurs processus temporels multi-échelles parallèles, et les multi-interactions imbriquées aussi à différentes échelles (avec un instrumentiste, avec des systèmes de captation gestuelle, avec des analyses des flux par descripteurs audio, avec des données provenant d'autres médias temporels, etc.) fait émerger une sonorité d'ordre supérieur. Mais elle nécessite une pratique et une maîtrise des systèmes d'écriture permettant d'agencer tous ces composants.

Donner une cohérence musicale à tous ces éléments en parallèle devient vite complexe et arriver à un résultat musical convaincant prend du temps, à moins de se contenter du résultat émergeant et de prendre l'acte de créer ces réseaux d'interactions comme l'acte de composition en soit. Ce n'est pas mon cas. Dans ma recherche musicale, le résultat sonore et musical doit prévaloir sur la théorie. Mais comme pour n'importe quel type de pratique, avec de l'entraînement et de l'expérimentation, on arrive d'une certaine manière à imaginer le rendu global résultant de ces interactions imbriquées et multiples. On réalise alors pleinement l'approche par objet et réseaux [Vag08] qui en même temps enrichit l'espace de pensée et apporte de nouvelles idées musicales conduisant à de nouvelles expérimentations.

VI.4. Utilisation d'algorithmes, de modèles formels et de calculs dans la composition

Plusieurs compositeurs et chercheurs ont accordé à travers l'histoire une importance capitale à l'utilisation des mathématiques dans la musique. On dit couramment que la musique est en partie faite et réalisée avec des relations algébriques (échelles, rythmes, hauteurs). Aujourd'hui, nous avons la possibilité d'utiliser des machines avec des ressources de calcul importantes qui permettent désormais une exploration inédite grâce à des techniques de génération et de modélisation des structures musicales.

L'idéal de la composition musicale automatique (CMA) ou « *musique de Turing* » [Vag03] qui utilise des algorithmes pour créer des formes musicales entières ne fait pas partie de mes recherches. Je crois que dans ce dispositif, la composition perd son rapport au sonore et devient un exercice de programmation plus qu'un acte de création musicale et sonore. Dans cette vision, la création du programme devient l'objectif de la composition elle-même. À mon avis, rares sont les cas où les résultats sont intéressants au niveau formel sans une intervention manuelle du compositeur : celui-ci doit le plus souvent ajuster la partition produite pour la rendre « musicale » du fait *qu'elle est caractérisée par la très faible interaction entre pensée compositionnelle et calcul (algorithmique)* [Sol07]. Xenakis résume bien cette volonté de ne pas subordonner la musique aux mathématiques :

« *Les restrictions sont d'ordre général canalisatrices plutôt qu'impérieuses. Ce sont des tendances de l'être sonore que la théorie et le calcul définissent et non pas un esclavage. Les formules mathématiques sont ainsi apprivoisées et asservies*. » [Xen81]

Penser qu'appuyer sur un bouton permettra de créer toute la musique automatiquement ne fait pas partie de mes aspirations. On peut tout de même formaliser des pensées musicales pour les modéliser par des algorithmes ou des processus. Par exemple,

partons de l'idée musicale suivante : créer une sonorité harmonique floue avec plus d'énergie dans les fréquences graves. Cette idée pourrait se réaliser en programmant un processus qui générerait des hauteurs à partir d'une distribution aléatoire de type exponentiel avec des rythmes irréguliers pour contrôler un synthétiseur. Puis on pourrait affiner ce processus en rajoutant des contraintes de hauteurs, par exemple pour utiliser seulement des subdivisions en huitième de ton avec des rythmes issus d'un tableau des durées pour chaque son, etc. L'avantage qu'il peut y avoir à réaliser ce type de processus dans un système temps réel capable de *live coding* comme AntesCollider est qu'on peut moduler ses paramètres en temps réel et ajuster à l'oreille (et à la main) les paramètres pour trouver le son adéquat en fonction du contexte musical.

« D'une façon plus pragmatique, nous pouvons dire que les modèles sont des représentations conceptuelles, des intermédiaires qui font le lien entre le monde abstrait des idées (concepts) et le monde concret (des sons, des instruments, du cognitif), les fédérateurs entre l'imaginaire abstrait et le sensible, et aussi des manières de faire. » (Mikhail Malt [Pot09])

Une position intéressante est développée par Mikhaïl Malt avec son concept de *solfège des modèles*. Pour lui, la maîtrise d'un modèle génératif quelconque est un savoir-faire de compositeur qui s'apparente au solfège classique.

« *Il s'agit d'un solfège au sens d'aptitude à relier le comportement de deux espaces de caractéristiques différentes en tenant compte des particularités de chacun.* [...] *Dans notre cas précis, un "solfège de modèles" est la capacité d'interpréter, ou de relier le comportement d'un modèle formel (ou d'une représentation) à une situation musicale.* [...] *Il est aussi important de signaler que nous ne parlons pas non plus de ce qui est communément désigné sous l'anglicisme de "sonification" de données, soit la conversion de paramètres générés par un modèle formel quelconque dans des paramètres de l'espace musical tels que les hauteurs, des durées ou des rythmes.* [...] *Si nous pensons à l'utilisation de modèles de hasard par des compositeurs aussi différents que Xenakis, Cage, Birtwistle, Murail ou Ferneyhough, il est possible de se rendre compte comment un même modèle appliqué à des espaces, à des contextes et dans des esthétiques différentes produit des résultats aussi distincts, même si un invariant formel demeure.*

La tâche du compositeur est de bien connaître ses modèles et d'évaluer la pertinence et le contexte musical de leur utilisation, le modèle se transforme ainsi en un interprète de la pensée du créateur. » Mikhail Malt [Pot09]

L'utilisation d'algorithmes est primordiale : ils m'aident à modéliser, à formaliser et à créer des passages avec certaines caractéristiques rythmiques, contrapuntiques, des sons synthétiques, des textures, des couches, des orchestrations, des mouvements dans l'espace, etc. Je les utilise comme des éléments de mon espace de pensée et de ma palette sonore compositionnelle en faisant des événements composables qui interagissent entre eux pour créer et faire émerger la morphologie du discours musical, au même titre que les hauteurs, des sons inharmoniques ou des silences.

Cette utilisation prend pleinement sa place dans le double jeu entre écoute et technique, dans cette boucle de rétroaction fondamentale qui relie l'expérimentation et le processus de composition. Cette expérimentation nécessaire accueille les imprévus ou les accidents qui, par exemple à cause d'une erreur dans les paramètres initiaux d'un processus ou un programme, induisent parfois des résultats inattendus mais qui permettent à l'imagination de rebondir et d'échapper à l'a priori. Quelquefois, *« la machine rend des réponses à une question que vous n'avez pas posée »* [BouGre88].

L'algorithme fonctionne donc à la fois comme un outil technique remplissant une fonction bien définie, comme un objet de mon espace de pensée musicale et comme une notion heuristique dans le *workflow* de la création.

Il est alors intéressant de noter que certaines notions, utilisées comme heuristique pour imaginer de nouveaux modèles de calcul ou bien pour concevoir de nouveaux algorithmes, se transportent *de facto* dans le domaine musical où elles gardent leur pouvoir heuristique. La notion d'agent, par exemple, en est une bonne illustration dans le développement de nouvelles approches génératives, par exemple avec la notion d'automate cellulaire. Le domaine de la vie artificielle, dont l'objectif est de créer des systèmes artificiels s'inspirant des systèmes vivants, est un autre pourvoyeur de métaphores et d'analogies heuristiques qui permettent d'imaginer d'autres formes. Dans ce contexte, l'artiste devient un « méta-créateur » [Whi04] : il développe le système qui crée l'œuvre ou une partie d'elle. L'émergence est l'une des caractéristiques de cette *A-life art* où l'artiste donne les règles à suivre qui peuvent être arbitrairement simples, mais dont l'application répétée peut engendrer une grande complexité lui rendant le résultat imprévisible [Whi04]. Dans le domaine des musiques interactives, c'est le comportement même de l'algorithme et des agents qui peut être modifié en temps réel à partir de différents *inputs*, comme des descripteurs du jeu instrumental issu de l'analyse audio ou de la captation gestuelle.

VI.5. La composition : un artisanat et une palette en constante augmentation à maîtriser

Dans cet univers compositionnel avec une palette augmentée par les technologies informatiques comme la partition centralisée, on peut assez facilement rester bloqué ou se perdre. Plus la palette est grande, plus la confrontation avec la page blanche sera traumatique si l'on n'a pas a priori dans l'esprit des idées compositionnelles, sonores et musicales fortes.

C'est le parcours et le métier du compositeur qui lui permettent de connaître sa palette et d'imaginer des mondes sonores en fonction des possibilités offertes. C'est toujours cette expérience, tirée de l'expérimentation et de l'assimilation des rétroactions techniques et musicales, qui lui permet d'augmenter cette palette pour faire face à de nouveaux besoins musicaux. Ainsi, une dialectique se déploie entre l'outil et les besoins, l'un modelant l'autre.

On pourrait imaginer une autre approche, de nature évolutionniste, où la musique et l'outil seraient largement indépendants : le compositeur partirait de cette palette

augmentée et commencerait à expérimenter pour ne garder que les résultats jugés satisfaisants vis-à-vis de son objectif musical et de son idée globale de la composition, quitte à les organiser et les composer a posteriori.

Dans mon parcours de création de sons et de formes musicales, en accord avec cet esprit de liberté musicale que j'ai défini, la composition devient un artisanat et une construction « manuelle », un « tricotage » qui se fait petit à petit, en effaçant, en revenant en arrière, en anticipant un besoin futur, dans un aller-retour perpétuel entre l'outil et le résultat sonore, pour passer d'un état à un autre, se transformer et se renouveler en permanence, pour sculpter la musique et lui donner la fraîcheur et la spontanéité dont elle se nourrit.

VI.6. Écriture du son et de la forme

« *L'instant est à la fois matériau et forme : s'il est impossible de distinguer en son sein une idée concrète — thème, motif, cellule, série ou, plus généralement, un élément premier clairement délimité — d'un développement, c'est qu'il est saisissable globalement comme seul l'est le matériau ; d'un autre côté, il est fabriqué de toute pièce à la manière d'une forme. Or, cette fusion peut aussi être nommée son, le son étant à la fois forme et matière : de toute évidence, le son est marqué par sa matérialité ; mais, simultanément, si on l'isole pour lui-même, il n'est guère possible de le résumer à autre chose qu'à lui-même, ni d'en extraire un quelconque élément, sauf à le mutiler — il n'est pas une matière mise en forme. Enfin, son et instant parachèvent l'immanence qui évacue le temps : ils sont autosuffisants et ne nécessitent, pour se réaliser, aucune mémoire de ce qui précède ou aucune anticipation de ce qui suit; ils constituent un monde clos, un univers replié sur lui-même et, par conséquent, ne renvoient à rien d'autre qu'à eux-mêmes.* […] *La spatialisation de la musique va de pair ou entraîne la mutation majeure de ce siècle où la musique cesse d'être l'art des sons (sous-entendu : de la combinatoire des sons) pour devenir l'art de la synthèse du son : le repli dans l'instant qui découle de la spatialisation est synonyme de construction intérieure du son.* » [Sol98]

L'un des aspects passionnants de l'ère actuelle est la possibilité de composer le son lui-même avec une palette riche de timbres et couleurs dans un contexte de liberté où le matériau n'est plus limité ni maîtrisé par le carcan d'un système formel (par exemple tonal). Comme déjà mentionné, nous avons aujourd'hui à disposition une pléthore de synthétiseurs, de techniques de synthèses numériques et ces outils semblent s'ouvrir encore à de nouvelles potentialités avec par exemple les apports de l'intelligence artificielle. Cette explosion des potentialités concerne aussi les instruments acoustiques avec le développement des modes de jeux pour les instruments de l'orchestre. Il en résulte pour la musique mixte une richesse non seulement du timbre sonore mais des interactions qui en découlent.

Ma recherche actuelle sur la composition se centre principalement sur la composition du son dans un contexte où les notions de matériau et forme s'effacent au profit de leur fusion. On passe ainsi de *composer avec le son* à *composer le son*. C'est le son lui-

même qui va se déployer dans le temps et l'espace et qui fera émerger la forme. Dans ce contexte, l'exploration des sons est fondamentale et disposer de systèmes qui permettent cette exploration est indispensable pour adresser non seulement le timbre mais aussi sa modification continue dans le temps (voir le chapitre VII sur la synthèse spatiale).

Ce besoin explique la focalisation de ma recherche, au niveau électroacoustique, sur la librairie AntesCollider qui permet d'un côté d'avoir un système de synthèse puissant et dynamique, et d'un autre un langage pour contrôler ces synthèses de différentes façons et les faire évoluer dans le temps et l'espace pour faire émerger un état sonore global.

Les objets (acteurs) du langage Antescofo et leurs expressivités vont permettre de créer facilement différents systèmes de synthèse pour l'exploration du son. Dans ce contexte, ces objets sont générateurs de micro- et macro-forme grâce à leurs capacités à être transformés dynamiquement dans différentes dimensions, ils deviennent multiformes, multi-échelles, multimodaux, etc. Ces objets peuvent aussi se comporter comme des agents qui vont créer leur propre vie ou interagir entre eux. Chaque objet avec ses propriétés musicales et polyphoniques donnera naissance au son et à la forme de la composition grâce à leur écriture dans la partition électronique centralisée.

La figure suivante présente un extrait de la partition électronique de la pièce audiovisuelle *Las Pintas*. Il s'agit de l'objet utilisé pour créer des morphologies granulaires HOA_grain_rot_rand_lfo_rev2. Cet objet a 25 paramètres qui peuvent être modulés en temps réel pour réaliser différents mouvements de granulation dans l'espace en HOA. Ces objets génèrent en même temps le timbre, le rythme, la texture, la spatialisation, l'amplitude et la réverbération du son pour produire une morphologie spécifique. La superposition des objets de synthèse crée en même temps le son et la forme.

```
8559  NOTE 60 1 Mort_gran2
8560
8561  obj::crea_track_HOA("lfo_rot7_rev", "group_hoa2", amp = 3, encoder = false)
8562
8563  $gran_rand_lfo_rot6_rev := obj::HOA_grain_rot_rand_lfo_rev2("armonicos7.aif", "perc",
8564      1.305, 1.31, 0.03, 0.07, 0.001, 0.07, 0.1, 0.8, 1, 3, 0.7, 0, 10, 360, 10, 60, "linear", 1, 15, "lfo_rot7_rev", 2)
8565
8566  whenever mort_gran2_amp ($mort_gran2_amp == $mort_gran2_amp)
8567  {
8568      $tracks("lfo_rot7_rev").amp($mort_gran2_amp)
8569  }
8570  whenever mort_gran2_ryth_min ($mort_gran2_ryth_min == $mort_gran2_ryth_min)
8571  {
8572      $gran_rand_lfo_rot6_rev.ryth_min($mort_gran2_ryth_min)
8573  }
8574  whenever mort_gran2_env_dur ($mort_gran2_env_dur == $mort_gran2_env_dur)
8575  {
8576      $gran_rand_lfo_rot6_rev.env_dur($mort_gran2_env_dur)
8577  }
8578
8579  NOTE 60 1 Mort_gran3
8580
8581  obj::crea_track_HOA("lfo_rot8_rev", "group_hoa3", amp = 3, encoder = false)
8582
8583  $gran_rand_lfo_rot7_rev := obj::HOA_grain_rot_rand_lfo_rev2("atube+disto+Conformal.aif", "rond",
8584      1.305, 1.31, 0.03, 0.07, 0.001, 0.07, 0.1, 0.8, 1, 3, 0.7, 0, 10, 360, 10, 60, "linear", 2, 3, "lfo_rot8_rev", 3)
8585
8586  $tracks("lfo_rot8_rev").rand_lfo_basic_track_pos_z(-2, 2, 0, 120, "linear")
8587
8588  whenever mort_gran3_amp ($mort_gran3_amp == $mort_gran3_amp)
8589  {
8590      $tracks("lfo_rot8_rev").amp($mort_gran3_amp)
8591  }
8592
8593  NOTE 60 1 Mort_gran4
8594
8595  obj::crea_track_HOA("lfo_rot9_rev", "group_hoa4", amp = 3, encoder = false)
8596
8597  $gran_rand_lfo_rot8_rev := obj::HOA_grain_rot_rand_lfo_rev2("balayage.aif", "m",
8598      1.305, 1.31, 0.03, 0.07, 0.001, 0.07, 0.1, 0.8, 1, 3, 0.7, 0, 10, 360, 10, 60, "linear", 3, 0, "lfo_rot9_rev", 4)
8599
8600  whenever mort_gran4 ($mort_gran4_amp == $mort_gran4_amp)
8601  {
8602      // print $mort_gran4_amp
8603      $tracks("lfo_rot9_rev").amp($mort_gran4_amp)
8604  }
```

Fig. 6.2 Extrait de la partition électronique de *Las Pintas*, pièce audiovisuelle. Cet extrait présente la superposition de 3 acteurs pour la création de morphologies sonores à partir de la synthèse granulaire spatiale en HOA. Chaque objet est instancié avec des paramètres différents pour générer différentes textures. Ces paramètres peuvent être modulés pour changer leur comportement (lignes 8566-8577) en temps réel pendant la performance par l'instrumentiste électronique. La superposition de plusieurs acteurs génère la sonorité globale recherchée pour ce passage de la composition.

VI.7. Au-delà de la notion de temps réel/différé

« *La prise de position pour le temps différé ou pour le temps réel, en d'autres termes la répartition des événements constitués de valeurs absolues ou relatives, doit désormais cesser d'être un choix technologique pour devenir un choix compositionnel. De ce point de vue, la synthèse, définie comme un calcul de paramètres totalement préétablis, est une technique relevant complètement du temps différé puisqu'il n'y a aucune détection ni analyse d'un signal en temps réel.* » Philippe Manoury, *Les partitions virtuelles* [Man97]

Aujourd'hui, avec l'accès à des ordinateurs de plus en plus puissants au niveau du calcul, la distinction historique entre temps réel et temps différé devient de plus en plus floue. Ce brouillage est à l'œuvre dans plusieurs directions.

La puissance de calcul permet à présent d'intégrer certaines opérations de CAO lors de la performance, et des fonctionnalités qui étaient conçues pour le temps différé, comme celles offertes par OpenMusic, peuvent être utilisées en temps réel avec *bach* dans Max

par exemple. Il existe toujours aujourd'hui des processus qui n'arrivent pas à être calculés en temps réel, comme la spatialisation de masses sonores dans un système Ambisonic d'ordre élevé (au-delà de l'ordre 5). Mais avec des processeurs (CPU) de plus en plus rapides et la multiplication des cœurs (multiprocesseur), ces limitations se rétrécissent, ce qui permet d'intégrer de plus en plus de processus qui auparavant étaient seulement disponibles en temps différé.

Par ailleurs, des logiciels conçus pour le temps réel peuvent être utilisés en temps différé. Les fonctionnalités propres à « l'écriture sur papier », comme revenir en arrière, effacer, recomposer, enregistrer, éditer, etc., peuvent à présent se faire dans des langages informatiques temps réel comme Antescofo ou la librairie AntesCollider dans la partition électronique avec la fonction *live coding*. Le travail compositionnel avec ces logiciels permet alors une écoute « en direct » du résultat des processus quelle que soit leur nature (génératif, déterministe ou stochastique). L'acte de composer précède toujours le temps de la performance : il ne s'agit pas d'une improvisation mais de composer avec des outils qui réagissent en même temps qu'on les manipule, d'une façon analogue à l'utilisation du piano comme aide à l'écriture pour orchestre.

Enfin, l'exemple des musiques électroacoustiques en temps réel (un exemple sera présenté au chapitre VIII) est un exemple de composition écrite mais qui se recrée en temps réel (au lieu de diffuser un enregistrement). Ce schéma confère une flexibilité qui permet par exemple d'adapter la pièce très finement à un lieu d'écoute ou d'altérer des paramètres et des comportements avant ou pendant le jeu.

Maquettes, écoute intérieure et concrète

La possibilité de faire de la CAO en temps réel, avec par exemple des maquettes à la manière d'OpenMusic intégrées à des langages temps réel de plus en plus sophistiqués et offrant des possibilités de traitement symbolique, permet de retrouver dans le temps différé ce qui en avait été exclu : l'écoute concrète.

Dans ma démarche, l'écoute concrète est fondamentale ; je peux imaginer des processus, des mouvements dans l'espace, des morphologies sonores qui se transforment et les programmer dans la partition électronique, mais ces idées doivent absolument être validées par l'écoute. On peut en effet avoir des idées de prime abord séduisantes sur la manière de créer des formes sonores, mais lorsqu'on réalise ces idées à l'aide d'un programme informatique, le résultat ne correspond pas toujours à ce que l'on avait imaginé. Souvent, il faut retravailler l'idée pour que sa réalisation corresponde à l'impulsion de base et surtout qu'elle devienne intéressante dans un contexte musical. La question de la perception et de l'appréhension sonore reste pour moi fondamentale et c'est pourquoi j'ai besoin de cette écoute. Je ne peux pas dérouler un concept, si élégant ou élaboré qu'il puisse être, sans un retour perceptif et sans éprouver l'impact cognitif déclenché par sa perception.

VI.8. Un compositeur doit-il apprendre à programmer ?

C'est une question qu'on m'a posée à plusieurs reprises et cette thèse défend l'usage de la programmation comme partition électronique.

On peut légitimement se poser la question du rapport entre composition et programmation informatique. De nombreux compositeurs trouvent une réponse à leurs besoins dans l'offre existante de logiciels de montage, de composition et d'interaction, sans avoir besoin de recourir à la programmation.

Le cursus annuel de composition à l'Ircam offre un bon exemple des attitudes possibles face à l'informatique musicale. Les étudiants pratiquent cette discipline pendant toute une année, avec de nombreux travaux pratiques, pour aboutir en fin d'études à la composition d'une pièce mixte en utilisant des programmes temps réel dans Max. Après ces études, certains compositeurs continuent à pratiquer l'informatique et même à étendre leurs compétences pour réaliser des pièces avec une grande composante technologique. D'autres, en revanche, abandonnent complètement la composante technologique ou s'adresseront par la suite à un RIM pour réaliser la partie électronique de leurs compositions.

La maîtrise d'un langage de programmation est similaire à l'apprentissage d'une nouvelle langue ou d'un nouvel instrument qui nécessite un investissement se comptant en années. La programmation a une courbe d'apprentissage longue et, pour certains, c'est un monde aride trop éloigné des problématiques artistique et musicale. Il faut d'une certaine façon aimer se confronter à une machine et aux problématiques logiques de la programmation pour passer au-delà de cette aridité et y percevoir un langage permettant d'exprimer, de travailler et de concrétiser une idée musicale.

Le passage entre programmation et composition n'est pas toujours évident car toutes les deux ont besoin de travail et de temps. Il faut donc trouver un bon équilibre et organiser les ressources qu'on va consacrer à l'une ou l'autre. Dans ma pratique, j'ai opté pour essayer de séparer le développement informatique de la composition, c'est-à-dire que je donne la priorité à l'un ou l'autre en fonction du temps, des commandes… Bien entendu, les deux mondes sont complètement perméables, on ne peut pas les séparer ou les isoler : pour programmer, il faut avoir en tête ce qu'on veut faire au niveau musical, et pour composer avec des partitions électroniques, on est confronté à la programmation tout au long de la composition. Pendant le processus de composition, de nouvelles idées peuvent émerger et elles doivent être programmées. Cependant, une programmation de plus bas niveau, comme le développement d'une librairie, devrait se réaliser dans un autre temps que celui de la composition, pour ne pas interférer avec la fluidité du processus créatif.

Un des principaux avantages d'être un compositeur-programmeur (même à différents niveaux de maîtrise) est de pouvoir expérimenter directement des idées musicales liées à la technique et éprouver cette rétroaction entre création, pratique et expérimentation. Aujourd'hui, cette approche est complètement intégrée à ma pratique compositionnelle et la programmation participe à l'imaginaire du compositeur. Je ne peux plus concevoir

ces deux activités l'une sans l'autre. Cette expérience de la programmation m'a aussi donné une compréhension précieuse des possibilités offertes par les technologies numériques. Et elle m'a aussi amené à travailler en tant que RIM, ce qui m'a permis de me confronter aux esthétiques et pensées de divers compositeurs.

La programmation s'est donc révélée dans mon parcours être un outil de libération. Une libération bornée par les possibilités techniques et par notre imagination, mais qui permet de s'affranchir du cadre fermé de nombreux logiciels commerciaux. Une liberté nécessaire pour expérimenter de nouvelles idées musicales.

La réponse à la question posée au début de cette section est donc positive, à la condition de ne pas opposer la programmation à la composition, de l'intégrer réellement dans les cursus de composition (au même titre que l'orchestration ou l'analyse) et de montrer toute la fécondité de la rétroaction créativité-programmation-créativité.

Cet objectif est atteignable : les compositeurs intègrent déjà complètement l'outil informatique avec l'utilisation des DAW dans leur pratique quotidienne. Des initiatives comme dans Ableton Live permettent d'ouvrir ces environnements, de montrer les bénéfices de la programmation et de simplifier l'apprentissage en s'appuyant sur des environnements déjà maîtrisés.

VII. L'espace

> « *De simple mode de représentation, l'espace a fini par absorber la musique, à tel point qu'on peut se demander si, contrairement à toute définition courante, la musique ne serait pas, de nos jours, plutôt que l'art du temps, un nouvel art de l'espace.* » [Sol98]

On considère en musique deux types d'espace : l'*espace de représentation*, un espace abstrait qui correspond aux lieux de l'écriture, de la conception musicale et de la synthèse du son dans lequel on construit et on pense les sons et la musique, et puis l'*espace physique*, l'environnement dans lequel la musique et les sons vont se propager.

On trouve plusieurs espaces de représentation dans la notation musicale, par exemple l'espace des hauteurs, des timbres ou des rythmes et leurs associations spatiales ou ce que Makis Solomos appelle la *géométrisation* de la musique :

« *Cette géométrisation renvoie à la rationalisation de la musique… Aussi, à travers les processus de géométrisation et de rationalisation, la notion d'espace de représentation s'est naturalisée en musique, permettant de la penser comme entièrement composable. C'est ainsi que, dans les années 1950-1960, est apparue, avec le sérialisme, la définition paramétrique de la musique, permettant de la décomposer en "paramètres". Aujourd'hui, on peut l'envisager comme un espace multidimensionnel, où le compositeur définit des objets symboliques justiciables d'opérations constituant une sorte de géométrie de cet espace composable.* » [Sol13]

À ce propos, Ligeti écrivait :

« *Les relations syntaxiques des moments musicaux particuliers sont transposées* […] *par notre imagination dans un espace virtuel, où les moments particuliers – éléments, figures, chaînons, parties, etc. – agissent comme des lieux ou des objets et où le devenir musical apparaît dans son écoulement total comme l'architecture dans l'espace.* » [Lig66]

Ligeti parle aussi de la notion de réversibilité comme une qualité de la pensée spatiale :

« *La permutabilité du temps est la source la plus féconde pour les associations spatiales. Alors qu'une succession irréversible, comme la progression cadentielle de l'harmonie tonale, expose le temps lui-même dans son écoulement inexorable, les figures interchangeables et renversables dans la succession temporelle présentent la qualité primordiale de l'espace : la réversibilité. Revenir à son point de départ n'est possible que dans l'espace, pas dans le temps.* » [Lig14]

Les représentations spatiales sont aujourd'hui complètement intégrées à la conception et la composition de la musique. Elles sont aussi présentes dans l'utilisation compositionnelle de l'espace physique, avec ce que l'on peut désigner par « synthèse

spatiale ». Imaginer l'espace sonore physique et ses transformations, déformations, transmutations rendues possibles grâce aux techniques de diffusion sonore spatialisée, fait de cet espace physique (et virtuel dans sa conception) un espace de représentation dans lequel on peut créer des formes sonores et musicales.

VII.1. L'espace physique

Le son est un phénomène spatial en soi puisqu'il voyage et se propage dans l'air (un fluide) sous forme d'ondes longitudinales grâce à la déformation élastique de l'air. La notion de son spatial est donc une tautologie.

Une préoccupation ancienne

L'espace physique en musique n'est pas une préoccupation ou invention récente : il est déjà présent dans plusieurs musiques traditionnelles où la disposition des musiciens dans l'espace est fondamentale. Ces dispositions ont souvent une fonctionnalité en relation directe avec des rituels ou avec l'environnement dans lesquels ils se pratiquent, par exemple en plein air ou dans des temples. Avant le XX^{e} siècle, dans la musique de tradition occidentale, la composition de l'espace physique n'a pas eu une importance fondamentale, à quelques exceptions près.

« *C'est l'autonomisation de la musique – sa défonctionnalisation, si l'on préfère – qui lui a fait perdre la pensée de l'espace, notamment dans le cadre de la musique de chambre ; quant à la musique de concert, l'espace n'a pas disparu, mais il s'est limité à la scène et au rapport frontal avec le spectateur. La reconquête de l'espace peut parfois être interprétée comme une refonctionnalisation de la musique, non pas à travers une fonction sociale, mais à travers un lieu. Mais elle relève peut-être encore plus de la rationalisation du matériau et de l'évolution vers la totalité articulée : l'espace devenant lui aussi justiciable d'une écriture, d'une construction, un pas de plus est fait en direction de cette totalité.* » [Sol13]

L'intégration à l'écriture musicale de l'espace physique comme un paramètre structural créateur de morphologies sonores est déjà manifeste au XVI^{e} siècle avec l'utilisation des particularités architecturales de la cathédrale de San Marco à Venise avec ses deux orgues et balcons. Les expériences stéréophoniques des *Cori Spezzati* de Adrian Willaert, Andrea et Giovanni Gabrieli avec les contrepoints qui passent d'un chœur à un autre dans les balcons de la cathédrale constituent déjà une écriture spatiale. Avec *Spem in alium* (1570), motet pour 40 voix séparées pour huit groupes de cinq chanteurs autour de l'audience de Thomas Thallis, ou *Deo gratias*, canon à 36 voix spatialisées de Johannes Ockeghem, on assiste à une véritable écriture des voix dans l'espace ou ce qu'on appelle aujourd'hui des sources sonores. On peut citer d'autres exemples comme la *Serenade Notturno* KV 239 (1776) pour deux orchestres et le *Notturno* KV 286 pour quatre orchestres de Mozart… Un exemple marquant plus proche de nous est la *Grande Messe des Morts* (1837) de Berlioz pour orchestre, chœur et quatre groupes d'instruments à cuivres placés dans les quatre coins du lieu de représentation. Citons aussi Debussy qui réfléchissait également à la dimension spatiale de la musique :

« *On peut entrevoir un orchestre nombreux s'augmentant encore du concours de la voix humaine [...]. Par cela même, la possibilité d'une musique construite spécialement pour le "plein air", toute en grandes lignes, en hardiesses vocales et instrumentales qui joueraient et planeraient sur la cime des arbres dans la lumière de l'air libre.* » [Deb87]

Dans les musiques électroacoustiques, la spatialisation des sons est présente dès les débuts. Le premier concert avec spatialisation en direct a été donné en 1951 avec Pierre Schaeffer et Pierre Henry, diffusion sonore en relief spatial appelé « bureau en relief ».

Une préoccupation personnelle

Mon travail sur l'espace débute en 1996 quand j'ai eu l'occasion de travailler au LIPM (Laboratoire de recherche et de production musicale) à Buenos Aires. Ce studio permettait de faire de la quadriphonie, mais seulement en temps différé avec des langages tels que CLM (Common Lisp Music[116]) ou Csound[117] tournant sur un système NeXTSTEP[118].

En 1997, avec le développement de MSP dans Max, un nouveau monde s'est ouvert pour les compositeurs n'ayant pas nécessairement accès à l'époque à des plateformes spécialisées (comme la station ISPW : Ircam Signal Processing Workstation). L'apparition de MaxMSP tournant sur un ordinateur personnel Macintosh a constitué une grande révolution pour un jeune compositeur habitant dans un pays où ce genre de recherches étaient loin d'être une préoccupation ou un paradigme dominant chez les musiciens. J'ai commencé mes premières expériences avec un système quadriphonique en temps réel et la première pièce réalisée en 1999, *Attract*, pour clarinette et électronique en temps réel, basée sur des modèles chaotiques pour la composition et le contrôle en temps réel fut une expérience marquante pour le reste de mon travail.

Mon passage par le CNSMD de Lyon et son département SONVS créé par Philippe Manoury et Denis Lorrain m'a permis de continuer l'exploration de l'espace avec des systèmes plus performants et avec des outils de spatialisation comme le Spat Ircam intégrant plus fortement les aspects perceptifs.

Un événement que je n'avais pas du tout prévu m'a amené à renouveler mon approche de la spatialisation et de l'électronique en général : la rencontre avec le compositeur Emmanuel Nunes et le travail avec Eric Daubresse en tant que RIM à l'Ircam, d'abord pour la programmation de *Lichtung III* pour grand ensemble et électronique et puis pour celle de *Einspielung I* pour violon et sa nouvelle version avec électronique en temps réel. L'écriture de l'électronique et la spatialisation dans ces deux œuvres ont eu une influence majeure dans ma façon de concevoir l'électronique et la spatialisation. Dans ces pièces, les paramètres d'une note – hauteur, rythme, intensité, agogique –

[116] https://ccrma.stanford.edu/software/clm/
[117] https://csound.com
[118] https://fr.wikipedia.org/wiki/NeXTSTEP

sont élargis avec une notion d'« espace composable » dans le traitement de l'électronique.

VII.2. Synthèse spatiale

« *Ma première tentative pour donner à la musique une plus grande liberté fut l'utilisation de sirènes dans plusieurs de mes œuvres (Amériques, Ionisation), et je pense que ce sont ces trajectoires de sons paraboliques et hyperboliques qui ont amené certains écrivains à s'emparer de ma conception spatiale de la musique, dès 1925. Zanotti Bianco, par exemple, qui écrivait dans* The Arts*, parlait à cette époque de "masses de sons coulées dans l'espace" et de "grandes masses dans l'espace astral". Bien sûr, il ne s'agissait encore que d'un "trompe l'oreille", d'une illusion auditive ; on pourrait en parler, mais cela n'existait pas encore »*, Edgar Varèse [Var83].

Le concept de synthèse spatiale vient de l'utilisation de la notion de perception spatiale (localisation absolue, relative, mouvement et trajectoires dans l'espace d'un son) pour créer ou transformer des morphologies sonores ou des structures musicales en fonction de leur position ou de leur déplacement dans un environnement imaginaire (espace virtuel en 3D) ou réel (salle de concert).

Ces idées ont commencé à s'élaborer plus précisément à partir des musiques des années 1950 avec la figure d'Edgar Varèse, pionnier dans l'idée de composer l'espace comme une « projection sonore ». Par exemple, Varèse écrit à propos d'*Intégrales* :

« *Les Intégrales furent conçues pour une projection spatiale. Je les construisis pour certains moyens acoustiques qui n'existaient pas encore, mais qui, je le savais, pouvaient être réalisés et seraient utilisés tôt ou tard… […] Par projection j'entends la sensation qui nous est donnée par certains blocs de sons, je pourrais dire "rayons de son" si proche est cette sensation de celle produite par les rayons de lumière qu'émettrait une puissante torche d'exploration. Pour l'oreille comme pour l'œil, ce phénomène donne un sentiment de prolongation, de voyage dans l'espace.* » [Var83]

D'après Pierre Boulez :

« *Je ne puis, quant à moi, me résoudre à une vue aussi simpliste ; la répartition spatiale me paraît mériter une écriture [plus] raffinée* [...]. *Elle ne doit pas seulement distribuer des ensembles éloignés suivant des figures géométriques simples, lesquels arrivent toujours, en fin de compte, à s'inscrire dans un cercle ou une ellipse : elle doit aussi, et plus encore, disposer la microstructure de ces ensembles* […]. *Il est bien certain que l'indice de répartition* (l'espace) *ne joue pas seulement dans la durée sur les durées, mais aussi bien sur les hauteurs et les dynamiques, et sur les timbres.* » [Bou63]

Karlheinz Stockhausen a beaucoup expérimenté, notamment dans sa pièce électronique *Gesang der Jünglinger,* dans des pièces instrumentales comme *Gruppen* pour trois groupes instrumentaux ou encore dans *Carré* où la construction spatiale devient un paramètre à part entière. Dans ces pièces, l'écriture de l'espace est directement liée à l'écriture instrumentale puisque pour faire passer un son d'un endroit à l'autre par des fondus enchaînés, il faut réfléchir au mouvement spatial, son orchestration et sa

temporalité. Dans son article écrit en 1958, « Musique dans l'espace », il écrit par exemple :

« *La ressemblance des trois formations orchestrales entre elles résulte de la volonté de faire circuler des groupes de sons dans l'espace, d'un corps sonore à l'autre, et de segmenter simultanément des structures sonores semblables entre elles ; chaque orchestre doit pouvoir interpeller les autres, leur répondre ou leur faire écho.* »

En relation avec l'espace comme paramètre musical à part entière, il écrivait :

« *Dans le cas particulier d'une organisation spatiale unidimensionnelle des sources acoustiques, nous pouvons à présent mettre en relation les proportions de hauteur, de durée, de timbre et de sonie avec celle du lieu du son. Nous aboutissons ainsi au concept de "mélodie spatiale" (Raum-Melodie).* » [Sto17]

Avec les premiers essais de John Chowning [Cho71] de réverbérations artificielles pour simuler des espaces par ordinateur, une nouvelle dimension s'est ouverte au traitement du son dans l'espace. À l'Ircam, le développement dès le début des années 1990 du Spat est une avancée technologique dans la manipulation de la perception spatiale d'un son qui a permis de composer avec l'espace, et tout particulièrement dans les musiques mixtes [CNW15]. Le Spat est un système modulaire temps réel intégré en tant que librairie dans le logiciel Max. Depuis sa création, il a été utilisé pour la spatialisation d'un grand nombre de pièces réalisées à l'Ircam et ailleurs et est passé par différentes étapes de développement. Au début, le système était limité à une entrée stéréo et une douzaine des sorties audio ; aujourd'hui, il peut gérer des centaines de canaux audio et plusieurs formats de spatialisation.

Depuis sa première version, le Spat fonctionne sur un paradigme de source sonore, c'est-à-dire que le système reçoit un signal mono ou stéréo en entrée représentant une source qu'on va pouvoir placer dans un espace virtuel indépendamment de la disposition particulière des haut-parleurs dans la salle. Comme pour les autres paramètres de l'électronique, il devient possible d'écrire la position et les mouvements des sources dans l'espace dans une partition électronique. Dans ce paradigme de sources spatialisées, deux concepts se dégagent, celui de la position fixe et celui de trajectoire spatiale.

Avec des techniques numériques de plus en plus élaborées (Ambisonic, WFS [DAMONI03], VBAP) et la multiplication des lieux de diffusion équipés (espace de projection de l'Ircam, Kubus du ZKM, Satosphère à la SAT, etc.), l'espace est devenu aujourd'hui un élément incontournable dans la création musicale contemporaine, un paramètre de la composition qui impacte la structure, la compréhension et l'écoute de l'œuvre.

Si la problématique de la spatialisation du son est présente assez tôt dans mes compositions mixtes et électroacoustiques, l'idée de synthèse spatiale se manifeste plus précisément avec ma pièce *Gravita*, pour alto, système de captation gestuelle et électronique temps réel, puis dans la pièce *M-brana* pour percussions, contrebasse, système de captation gestuelle et électronique temps réel. Dans ces pièces, les positions

des sources sonores (issues des traitements en temps réel des instruments) sont contrôlées par des modèles physiques (fondés sur les librairies PMPD[119] et MSD[120]). Ces modèles sont eux-mêmes contrôlés par la captation gestuelle. Dans ma pièce électroacoustique par ailleurs, j'utilise la notion de déformation spatiale, une réfraction qui se traduit par une transformation des sources en fonction de la position dans laquelle elles se trouvent dans l'espace. Quand une ou plusieurs sources sonores se déplacent, elles sont soumises à différents traitements qui dépendent de la région qu'elles traversent. Dans cette pièce, la synthèse granulaire [Gab47] est répartie dans l'espace et dans des zones en 2D.

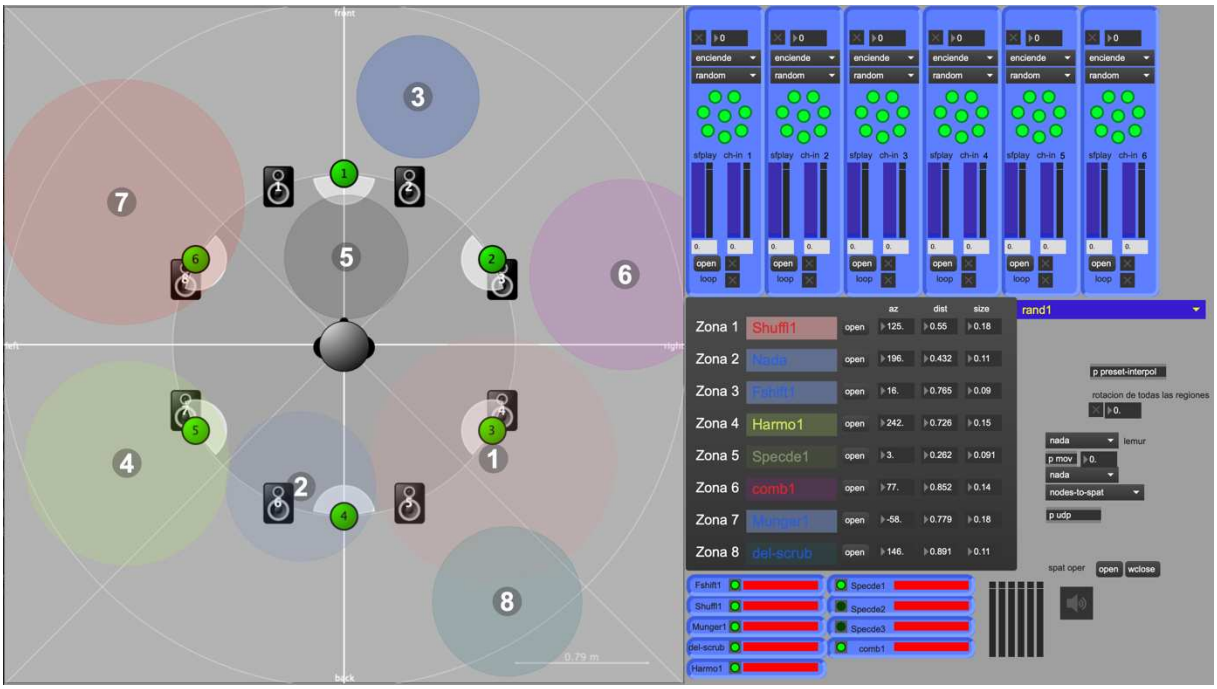


Fig. 7.1 Programme de réfraction du son en fonction de sa position spatiale (réalisé en Max/Spat). On peut choisir pour chaque zone en 2D une transformation spécifique, quand une source sonore passe par une zone spécifique, elle sera transformée en fonction du traitement assigné à la zone. Ces zones circulaires à gauche de 1 à 8, peuvent changer dynamiquement de position et de taille dans le temps.

Avec la librairie AntesCollider, les possibilités de traitements et de synthèses spatiales s'ouvrent vers de nouvelles explorations grâce à la flexibilité, le dynamisme et l'intégration de différents algorithmes de génération morphologiques dans l'espace physique, mais aussi grâce à l'intégration des techniques de spatialisation (VBAP, HOA, WFS…). L'écriture de la spatialisation et la restitution spatiale bénéficient grandement de ces techniques puisqu'on ne va pas écrire pour une configuration spécifique de haut-parleurs, mais pour un espace virtuel qui pourra par la suite être diffusé dans n'importe quelle configuration de haut-parleurs, analogue à ce qu'on

[119] http://drpichon.free.fr/pmpd/
[120] http://drpichon.free.fr/msd/

appelle dans le cinéma le *mixage orienté objet*[121] (des sources sonores avec des métadonnées spécifiant leurs positions et mouvements pour être diffusées dans différentes configurations de haut-parleurs dans les salles de cinéma).

Si le paradigme de source sonore est largement utilisé pour la spatialisation des sons avec des trajectoires ou des positions fixes, une nouvelle approche émerge avec la notion d'*objet sonore spatial*[122]. Le terme d'*objet sonore spatial* décrit une construction morphologique présentant une étendue spatiale plus complexe qu'une source sonore ponctuelle localisée à une position fixe ou bien se déplaçant sur une trajectoire. Il désigne des structures sonores composites avec des relations entre ses composants, par exemple des masses ou des textures sonores, qui à leur tour peuvent se métamorphoser dans l'espace/temps. La composition et l'écriture de ces *objets sonores spatiaux*, puis leur perception, sont conditionnées par les réflexions du son dans l'espace. La morphologie sonore avec tous ses paramètres ne peut donc pas être séparée de la position, de la vitesse de déplacement, de la distance, de l'orientation, du lieu (réel ou virtuel) et de sa réverbération.

Dans la librairie AntesCollider, il n'y a plus besoin de définir des sources sonores puisque les ressources nécessaires sont automatiquement et dynamiquement allouées lors de la création de l'*acteur*[123] associé à l'objet spatial. Cet aspect technique donne une grande liberté dans la création des structures sonores spatiales et facilite le maintien d'une cohérence entre ses composants et l'espace physique. Il est aussi assez facile d'intégrer différents algorithmes pour la création de trajectoires complexes. Les figures 2 et 3 illustrent l'intégration de la librairie Trajectory Score Library de Nadir Babouri dans AntesCollider pour la création de trajectoires en 2D et 3D. Dans la figure 4 suivante, on voit quelques exemples de création algorithmique de sources sonores par un contrôle global des mouvements spatiaux, notamment par des modèles physiques (mécanique newtonienne) ou relevant de la vie artificielle comme les nuées d'oiseaux [Rey87].

---

[121] T. Sporer, J. Plogsties et S. Brix, *CARROUSO – An European Approach to 3D-Audio*, Audio Engineering Society Convention 110, mai 2001.

[122] Le terme d'objet n'est pas utilisé ici dans le même sens que dans la notion de mixage orienté objet, même si techniquement le mixage objet peut intervenir dans la réalisation d'un objet spatial.

[123] Rappelons qu'un « acteur » est un objet dont les calculs s'effectuent en parallèle à ceux des autres acteurs. Cette notion est introduite dans le chapitre sur Antescofo.

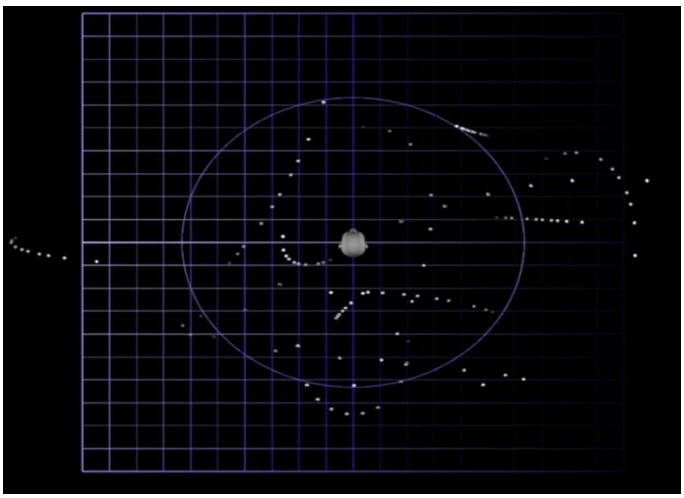


Fig. 7.2 Une capture d'écran des différentes sources dans un environnement ambisonique réalisé dans AntesCollider avec l'intégration de la librairie programmée à l'aide de la librairie Antescofo Trajectory Score Library originalement conçue pour contrôler le Spat Ircam.

```
// 2D
obj::crea_track_HOA("track_traj_2D_1", "group_hoa1", [["Impulse_Pluck", "freq", @rand_range(900, 1130), "freq_imp"
$tracks("track_traj_2D_1").traj_lib_ellipse(loop_num = 20, speed=4)

obj::crea_track_HOA("track_traj_2D_2", "group_hoa1", [["Impulse_Pluck", "freq", @rand_range(900, 1130), "freq_imp"
$tracks("track_traj_2D_2").traj_lib_sin(2, -2, 0.2, 3, 0., 2, loop_num = 20, elev=0.1)

obj::crea_track_HOA("track_traj_2D_3", "group_hoa1", [["Impulse_Pluck", "freq", @rand_range(900, 1130), "freq_imp"
$tracks("track_traj_2D_3").traj_lib_lissajou(loop_num = 20)
$tracks("track_traj_2D_3").traj_lib_lissajou(loop_num = 20, b=2)
$tracks("track_traj_2D_3").traj_lib_lissajou(loop_num = 20, m=5)

obj::crea_track_HOA("track_traj_2D_4", "group_hoa2", [["TestSynth3", "freq", @rand_range(150, 444), "amp", -8]])
$tracks("track_traj_2D_4").traj_lib_lissajou_bis(k=7, loop_num = 20, elev=0.05)

obj::crea_track_HOA("track_traj_2D_5", "group_hoa2", [["TestSynth3", "freq", @rand_range(150, 444), "amp", -8]])
$tracks("track_traj_2D_5").traj_lib_deltoide(loop_num = 20, elev=0.05)

obj::crea_track_HOA("track_traj_2D_6", "group_hoa2", [["BassSynth1", "freq", @rand_range(30, 60), "fmrange", 1.5,
$tracks("track_traj_2D_6").traj_lib_rosace(loop_num = 20, elev=0.5)
// $tracks("track_traj_2D_6").free()
obj::crea_track_HOA("track_traj_2D_7", "group_hoa3", [["BassSynth1", "freq", @rand_range(30, 60), "fmrange", 1.5,
$tracks("track_traj_2D_7").traj_lib_rosace_var(loop_num = 20, elev=0.5)

obj::crea_track_HOA("track_traj_2D_8", "group_hoa3", [["BassSynth1", "freq", @rand_range(30, 60), "fmrange", 1.5,
$tracks("track_traj_2D_8").traj_lib_gauss(iniT=-2, targeT=0.5, k = 0.2, loop_num = 20)

obj::crea_track_HOA("track_traj_2D_9", "group_hoa3", [["BassSynth1", "freq", @rand_range(30, 60), "fmrange", 1.5,
$tracks("track_traj_2D_9").traj_lib_hypocycloide1(loop_num = 20, elev=0.1)

obj::crea_track_HOA("track_traj_2D_10", "group_hoa4", [["BassSynth1", "freq", @rand_range(30, 60), "fmrange", 1.5,
$tracks("track_traj_2D_10").traj_lib_hypocycloide2(loop_num = 20, elev=0.1)

obj::crea_track_HOA("track_traj_2D_11", "group_hoa4", [["BassSynth1", "freq", @rand_range(30, 60), "fmrange", 1.5,
$tracks("track_traj_2D_11").traj_lib_curvoide(loop_num = 20, elev=0.1)

obj::crea_track_HOA("track_traj_2D_12", "group_hoa4", [["BassSynth1", "freq", @rand_range(30, 60), "fmrange", 1.5,
$tracks("track_traj_2D_12").traj_lib_hypo_var(loop_num = 20, elev=0.1)

obj::crea_track_HOA("track_traj_2D_13", "group_hoa5", [["BassSynth1", "freq", @rand_range(30, 60), "fmrange", 1.5,
$tracks("track_traj_2D_13").traj_lib_epi(m = $pi/2, speed =20, loop_num = 20)

// 3D
obj::crea_track_HOA("track_traj_3D_1", "group_hoa5", [["Impulse_Pluck", "freq", @rand_range(900, 1130), "freq_imp"
$tracks("track_traj_3D_1").traj_lib_generic3D(loop_num = 20, speed=4)

obj::crea_track_HOA("track_traj_3D_2", "group_hoa5", [["Impulse_Pluck", "freq", @rand_range(900, 1130), "freq_imp"
$tracks("track_traj_3D_2").traj_lib_helicoide3D(loop_num = 20)

obj::crea_track_HOA("track_traj_3D_3", "group_hoa6", [["Impulse_Pluck", "freq", @rand_range(900, 1130), "freq_imp"
$tracks("track_traj_3D_3").traj_lib_couronne3D(loop_num = 20)

obj::crea_track_HOA("track_traj_3D_4", "group_hoa6", [["TestSynth3", "freq", @rand_range(150, 444), "amp", -8]])
$tracks("track_traj_3D_4").traj_lib_clelie3D(loop_num = 20, offsetX = 0, offsetY = 0)
```

Fig. 7.3 Code AntesCollider pour la création des trajectoires à partir de la librairie Trajectory Score Library. Chaque *track* crea_track_HOA peut être spatialisé avec une trajectoire créée par une fonction mathématique, comme ellipse, lissajou, gausse, etc. L'avantage par rapport à d'autres systèmes est qu'il n'y a pas besoin de spécifier le numéro de la source que l'on va traiter puisque la trajectoire s'applique directement à un *track* déterminé.

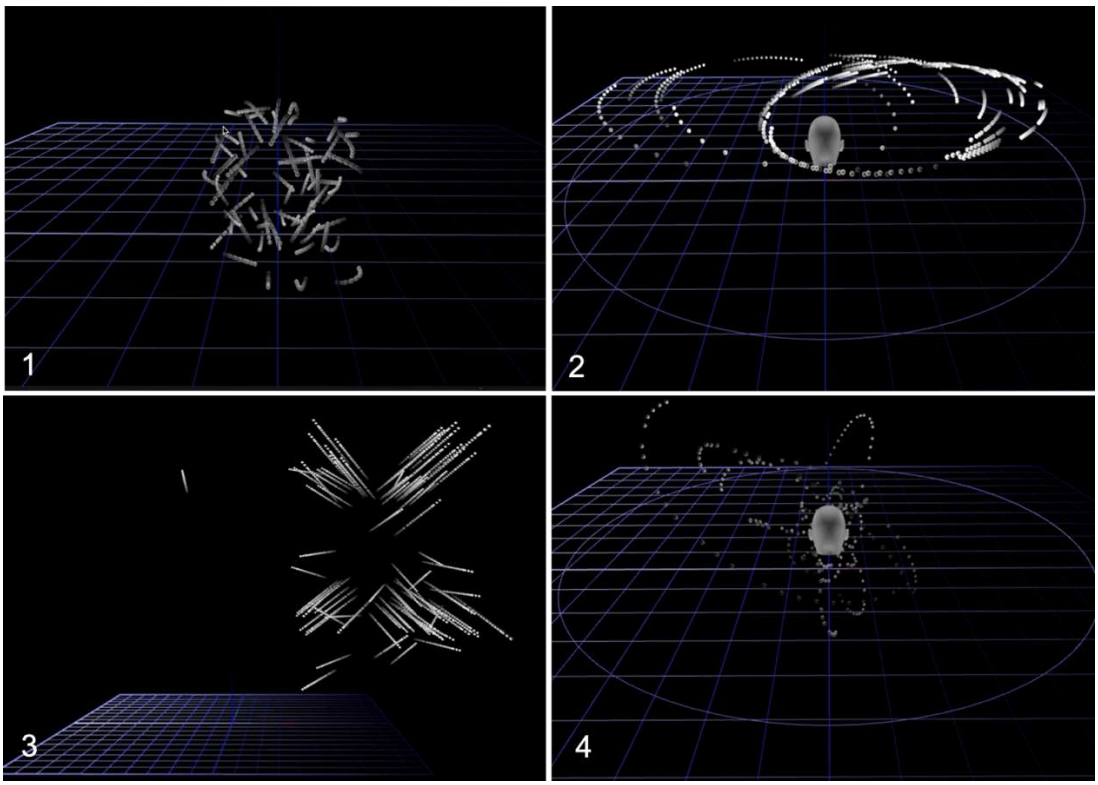


Fig. 7.4 Visualisation des sources dans un environnement ambisonique en 3D dans l'interface *antescollider_spatial_interface*. L'image n° 1 est une capture d'écran d'un système de particules qui se déplacent aléatoirement dans un cube virtuel. L'image n° 2 représente 90 sources qui suivent une trajectoire orbitale avec un type de synthèse qui peut changer dynamiquement ainsi que tous ses paramètres. L'image n° 3 est la sonification/visualisation d'un algorithme de simulation de nuées d'oiseaux (*flocking birds* ou *boids*) en 3D. L'image n° 4 est un système de particules en 3D avec des mouvements aléatoires circulaires qui peuvent, par exemple, simuler les déplacements d'insectes autour d'un auditeur. Les paramètres des modèles physiques peuvent être modulés en temps réel pour changer le comportement du système.

```
NOTE 60 1 HOA_orbits

// decodeur hexa
obj::mix_group_HOA("group_hoa1", "localhost", "hexa", 3)

// visualisation
$gui_enabled := true

// création des tracks
obj::crea_track_HOA("track_hoa1", "group_hoa1", [["Impulse_Pluck", "freq", 548.83, "freq_imp", 10.7, "del", 0.006, "amp", -3]])
$tracks("track_hoa1").ambi_orbit(0, 0.06)   // méthode orbit pour créer des orbites <angle de départ, vitesse de déplacement>

obj::crea_track_HOA("track_hoa2", "group_hoa1", [["Impulse_Pluck", "freq", 600, "freq_imp", 20, "del", 0.0005, "amp", -3]])
$tracks("track_hoa2").ambi_orbit(0, 0.04)

obj::crea_track_HOA("track_hoa2_1", "group_hoa1", [["Impulse_Pluck", "freq", 300, "freq_imp", 20, "del", 0.00051, "amp", -3]])
$tracks("track_hoa2_1").ambi_orbit(0, 0.033)

obj::crea_track_HOA("track_hoa3", "group_hoa1", amp = 6, [["Impulse_Pluck", "freq", 400, "freq_imp", 30, "del", 0.0002, "amp", 0]])
$tracks("track_hoa3").ambi_orbit(0, 0.01)

obj::crea_track_HOA("track_hoa4", "group_hoa1", [["TestSynth3", "freq", 123, "amp", -8]])
$tracks("track_hoa4").ambi_orbit(60, 0.01)

obj::crea_track_HOA("track_hoa5", "group_hoa1", [["TestSynth3", "freq", 234, "amp", -8]])
$tracks("track_hoa5").ambi_orbit(90, 0.08)

obj::crea_track_HOA("track_hoa6", "group_hoa1", [["TestSynth3", "freq", 345, "amp", -8]])
$tracks("track_hoa6").ambi_orbit(120, 0.067)

obj::crea_track_HOA("track_hoa7", "group_hoa1", [["TestSynth3", "freq", 456, "amp", -8]])
$tracks("track_hoa7").ambi_orbit(160, 0.055)
```

Fig. 7.5 Extrait du code AntesCollider pour la création de synthèses spatialisées suivant une trajectoire orbitale (image n° 2 de la figure précédente-. Chaque fois qu'on instancie un nouvel objet `obj::crea_track_HOA`, une nouvelle source est créée automatiquement. La méthode `.ambi_orbit` va appliquer un algorithme de génération d'orbites pour contrôler la position et la vitesse de chaque synthèse définie avec un dosage d'effet doppler dans chaque *track* (source).

```
1   NOTE 1 60 rand_paricules_3D_HOA
2
3   $hoa_order := 4 // hoa order
4   $nbMasses := 20 // nombre de particules
5   $mass_mass := 300. // la masse de chaque particule
6   $limit := 0.5 // pour initialisation
7
8   // 3D tab
9   $masses := []
10  $out_pos_final := [  0 | ($nbMasses*3) ]
11
12  // audio group decoder
13  obj::mix_group_HOA("group1", "localhost", "studio1", $hoa_order)
14
15  //recupère le bus hoa du mix_group_HOA "group1" dans la variable $hoa_bus
16  $hoa_bus := $groups("group1").$hoa_bus
17
18  // création d'une table de synthèses en fonction du nombre de paricules
19  $synths := [ [ ["Form_HOA_"+$hoa_order, "mul", 1/($i+0.00001), "freq", (1/($i+1)*600)+150, "globTBus", $hoa_bus ]] | $i in $nbMasses ]
20
21  // track HOA pour la création du groupe de synthèses
22  obj::crea_track_HOA("gaz_molecules", "group1", fade_in = 1, amp = 0, $synths.flatten(1), encoder = false)
23
24  forall $iter in $nbMasses
25  {
26      $coords := [$limit.rand2(), $limit.rand2(), $limit.rand2()]*0.8 // random init coords XYZ
27      $masses.push_back(obj::Mass3D($mass_mass, $coords))
28  }
29  // limites dans une sphère de rayon = 5
30  $limit_sphere := obj::Sphere3D(rMax = 5, kN = -0.5)
31
32  // Ambient3D genère des variations de positions aléatoires
33  $ambient := obj::Ambient3D()
34
35  group rand_paricules_3D
36  {
37      loop 0.02
38      {
39          @local $i := 0
40          forall $mss in $masses
41          {
42              $mss.trig()
43              $mss.inter_ambient($ambient)
44              $mss.inter_sphere($limit_sphere)
45              $out_pos_final[$i*3] := $mss.$position_out[0]
46              $out_pos_final[$i*3+1] := $mss.$position_out[1]
47              $out_pos_final[$i*3+2] := $mss.$position_out[2]
48              $tracks("gaz_molecules").set_single("Form_HOA_"+$hoa_order, $i, ["mul", $mss.$velocity_out[2].scale_compiled(0, 0.1, 0, 0.01, 1)
49              .clip_compiled(0, 0.1), "bpf", $mss.$velocity_out[2].scale_compiled(0, 0.1, 100, 5000, 1), "x", ($mss.$position_out[0]+$HOA_global_x), "y",
50              ($mss.$position_out[1]+$HOA_global_y), "z", ($mss.$position_out[2]+$HOA_global_z)])
51              $i += 1
52          }
53          of_molecules $out_pos_final // OSC vers visualisation
54      }
55  }
```

Fig. 7.6 Code dans le langage Antescofo avec la libraire AntesCollider et des modèles physiques pour la création du système de particules (image n° 4 de la figure précédente). De la ligne 3 à 6, on définit des variables globales pour l'ordre HOA, le nombre de particules, leur masse et les limites spatiales. À la ligne 27, on définit un tableau des masses avec l'objet Mass3D, puis à la ligne 30, les limites dans une sphère. La ligne 33 montre l'utilisation d'un acteur (objet) ambient pour générer des mouvements aléatoires. Les lignes 35 à 54 définissent l'itération qui va calculer un nouvel état du système tous les 0.2 pulsations (ou secondes avec un BPM de 60).

L'une des principales applications de la localisation d'une source sonore dans l'espace est la synthèse granulaire spatiale en 2D [Kim05] ou 3D [Fon13] et ses dérivés comme la synthèse concatenative [Sch00] ou *mosaicing* [ZiPa01], notamment dans un environnement ambisonique HOA [Mar09] [DeSh09]. Ces techniques de synthèse étendues pour intégrer l'espace physique, vont permettre de sculpter le son non seulement au niveau du timbre, mais aussi de l'espace et du temps. En effet, plusieurs grains peuvent être utilisés simultanément pour reconstituer une morphologie spécifique stationnaire ou évolutive dans le temps. Ces systèmes granulaires peuvent aussi reconstituer une ambiance sonore en traitant chaque grain de manière indépendante. Cette approche peut aussi être utilisée pour recréer virtuellement des ambiances synthétiques imaginaires ou réelles en 3D comme la rumeur de la ville, la présence de la pluie, le crépitement du feu, le vent, etc.

La synthèse granulaire spatiale permet de composer l'espace d'une manière très dynamique et précise car il est possible de définir le type de timbre, d'intensité,

d'enveloppe, de durée et de position spatiale 3D de chaque grain pour créer différentes typologies de sons dans l'espace. Au niveau spatial, on peut créer des mouvements individuels ou collectifs. Par exemple, un son qui se crée dans un point de l'espace va pouvoir exploser ou, à l'envers, une masse dispersée peut imploser en un point. Ces mouvements de masses sonores peuvent voyager tout en transformant continûment ses paramètres, réalisant ainsi des *morphings* spatiaux (interpolation d'une forme vers une autre sur le plan de l'espace et du timbre). On peut aussi différencier différents groupes de grains pour créer des sonorités complexes et leur appliquer différents traitements comme des réverbérations pour simuler des espaces virtuels et donner plus de réalisme aux sons dans un contexte déterminé. L'utilisation des réverbérations par convolution (*Impulse Response*) dans le domaine ambisonique ouvre également un vaste champ d'expérimentation pour le son immersif.

Mon exploration de ces techniques granulaires spatiales a débuté il y a longtemps dans mes pièces électroacoustiques et mixtes, mais leur utilisation avec les techniques HOA (et orientées objet) s'approfondit dans la pièce électroacoustique *Curvatura II.* Ces problématiques sont encore davantage traitées dans la pièce audiovisuelle *Las Pintas.* Ces pièces ont pu être développées grâce au système d'écriture et de synthèse offert par la librairie AntesCollider. Ces fonctionnalités sont aussi utilisées dans le domaine des pièces mixtes, notamment dans la pièce pour ensemble instrumental et électronique en temps réel en cours de développement.

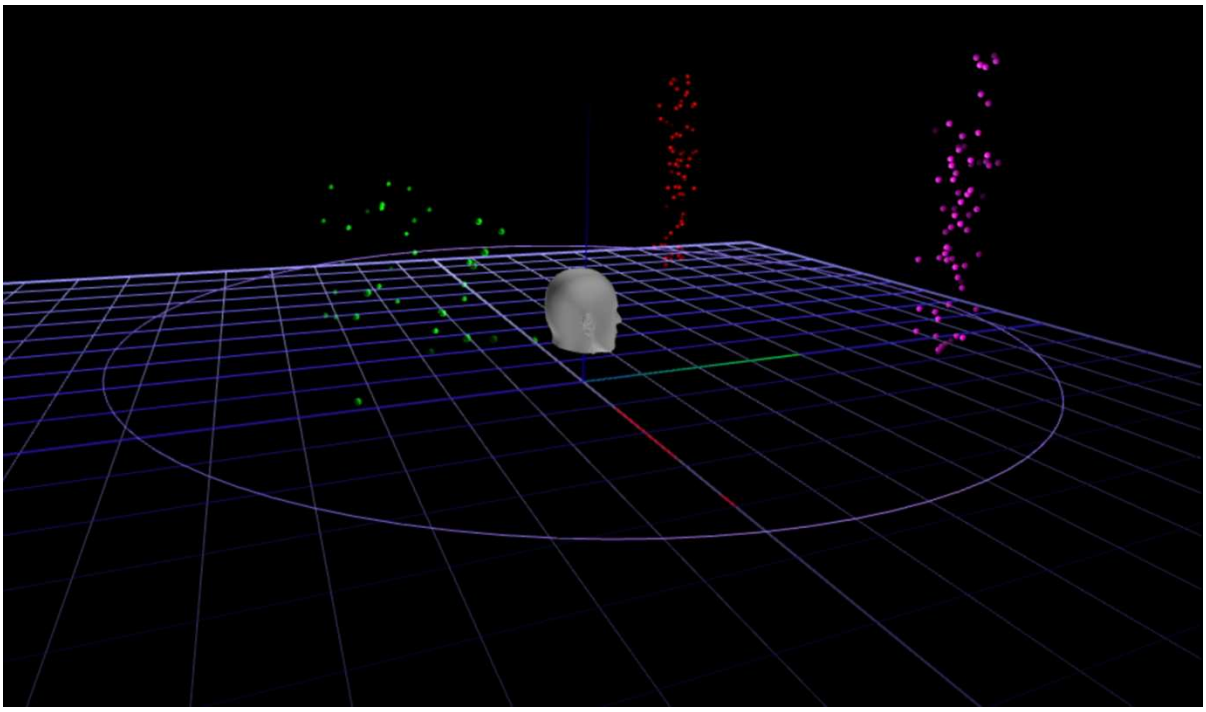


Fig. 7.7 Visualisation de trois groupes de synthèse granulaire implémentés dans AntesCollider. La synthèse granulaire spatiale permet un traitement très expressif de l'espace grâce à ses multiples possibilités de contrôle du timbre et de l'espace.

La synthèse concatenative par corpus [Sch07], technique de synthèse assez répandue ces dernières années, permet de composer le son à partir d'une base de données de sons

préenregistrés, segmentés et spatialement indexés par l'analyse de descripteurs audio comme la hauteur, l'énergie, l'harmonicité, le taux de bruit, etc. Ce corpus est exploité par un algorithme qui choisit les unités sonores du corpus les plus proches du son visé à partir des descripteurs[124]. Dans cette approche, on peut composer et explorer l'espace par rapport à des qualités sonores. Par exemple, on peut organiser la diffusion des sons dans l'espace physique en fonction de leur distribution dans l'espace des descripteurs. La flexibilité du système AntesCollider permet d'imaginer des organisations arbitrairement sophistiquées.

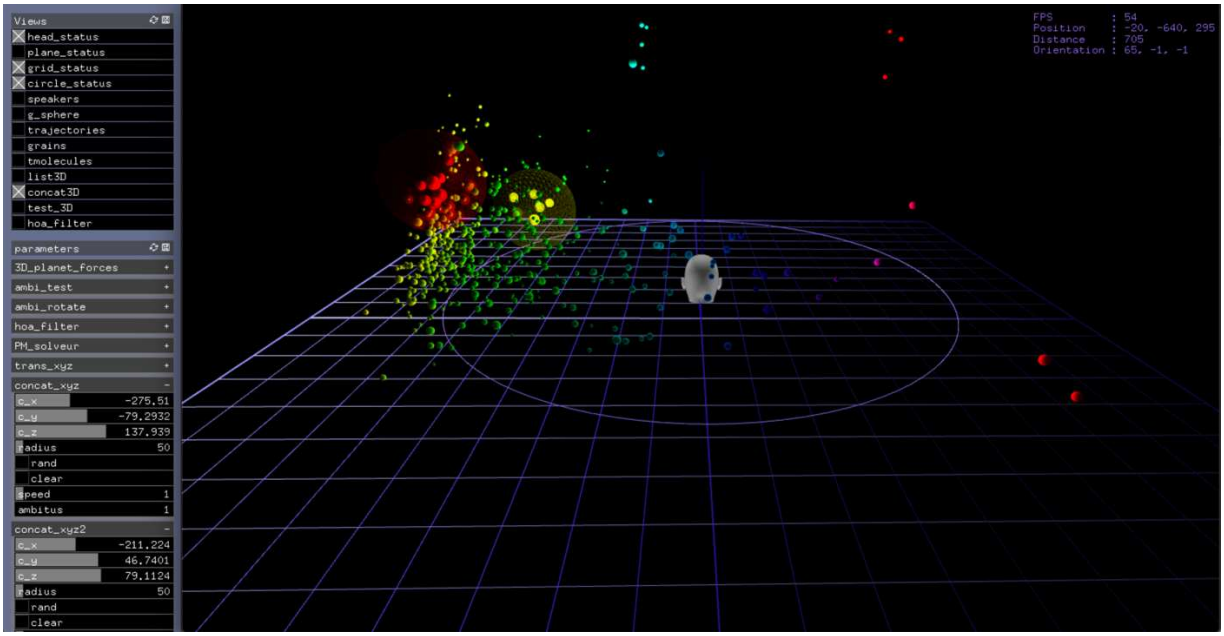


Fig. 7.8 Exemple de synthèse concatenative par analyse et segmentation « à la CataRT » en 3D. Plusieurs curseurs (têtes de lecture) en 3D représentés par une sphère peuvent déclencher les segments avec l'algorithme KNN implémenté dans le langage Antescofo.

Au-delà de la technique qui sera utilisée pour reproduire la spatialisation ou la perception spatiale, l'important est d'utiliser cet espace physique virtuel pour créer et transformer le son ou les morphologies musicales.

Une des motivations pour la création de la librairie AntesCollider est de permettre une grande liberté afin d'imaginer des scènes sonores spatiales complexes et une écriture expressive et fine de l'espace. Les *acteurs* du langage sont utilisés pour définir aussi bien la synthèse sonore que pour définir une trajectoire. Ce découplage permet une combinatoire très riche. Plusieurs approches ont été explorées pour la spécification des paramètres de contrôles ou bien des trajectoires. L'une des premières expériences a été l'utilisation des modèles physiques de contrôle PMPD (portés comme librairie dans Antescofo).

[124] AntesCollider utilise l'algorithme KNN (*k-nearest neighbors algorithm*) qui permet de retrouver efficacement tous les points dans un rayon donné d'un autre point (voir le chapitre II).

Dans ce contexte, on peut aussi intégrer des techniques mises en œuvre dans certaines pièces d'Emmanuel Nunes, liant la position spatiale à un grain ou à une enveloppe.

Les approches à base de modèles physiques et leur sonification avec des systèmes particulaires comme PMPD, et l'introduction de solveurs (voir chapitre sur Antescofo) de systèmes hybrides (équations différentielles plus événements discrets) dans AntesCollider, ouvrent de nouvelles possibilités pour la synthèse et l'écriture de l'espace, possibilités qui n'ont été que très partiellement explorées.

VII.3. Projet *Stück für die Schwerkraft* avec la compagnie de théâtre suisse ultra

La composition et l'écriture de l'espace peuvent aussi s'articuler dans d'autres contextes, espaces et environnements que celui d'une composition musicale. La composition spatiale peut aussi s'écrire pour un espace concret avec des objets physiques réels, comme c'est le cas du projet *Stück für die Schwerkraft* (*Pièce pour la gravité)*, idée originale de la compagnie de théâtre ultra de Lucerne.

En partant de l'idée de pesanteur, de force gravitationnelle et d'objets qui chutent du ciel et se dirigent à différentes vitesses vers le bas (le centre de la terre), la compagnie a imaginé une machine capable de lâcher sur scène des objets depuis les cintres. Il s'agit d'une machine conçue par Thomas Köppel qui comporte 400 trappes dont l'ouverture de chacune est contrôlée par ordinateur. Dans ce dispositif, la scène du théâtre devient un espace concret de composition audiovisuelle spatiale dans lequel des objets tombent à différentes vitesses selon leur géométrie et leur densité. La notion de composition de l'espace ou d'écriture spatiale prend un sens littéral : il faut écrire le mouvement de chute des objets physique afin de régler les sons produits quand chaque objet percute le sol. La conception musicale et compositionnelle est donc directement liée à l'espace physique de la scène et aux trajectoires des objets.

Le travail de création de la pièce s'est articulé autour de deux résidences, la première pour définir le contexte, la dramaturgie, les objets à jeter et la programmation d'un système d'écriture simple, expressif et facile à mettre en œuvre. La deuxième résidence a été consacrée à la composition de la partie musicale/temporelle avec l'écriture de la pièce proprement dite.

Pour ce genre de projet d'écriture, on voit comment un langage de programmation du temps tel que Antescofo peut s'adapter assez facilement au contrôle de la machine. Dépassant la simple spécification de la séquence temporelle des ouvertures des trappes, le langage permet de créer des modes de jeux particuliers grâce aux processus. On peut spécifier par exemple des séquences d'ouvertures suivant des lignes ou des figures arbitraires associées à des rythmes, des accords, des contrepoints, etc., et on peut composer ces figures – par exemple en parallèle ou en séquence – ou bien les transformer temporellement – par exemple les accélérer ou bien les ralentir. La combinaison de ces modes permet une écriture riche, dynamique et fluide, bien plus expressive que la programmation de bas niveau consistant à ouvrir ou fermer les trappes.

Au niveau technique, la machine fonctionne avec 400 électroaimants qui sont pilotés par un *Raspberry Pi*. Le *Raspberry Pi* communique avec l'ordinateur de la régie centrale par OSC à l'aide d'un câble RJ-45.

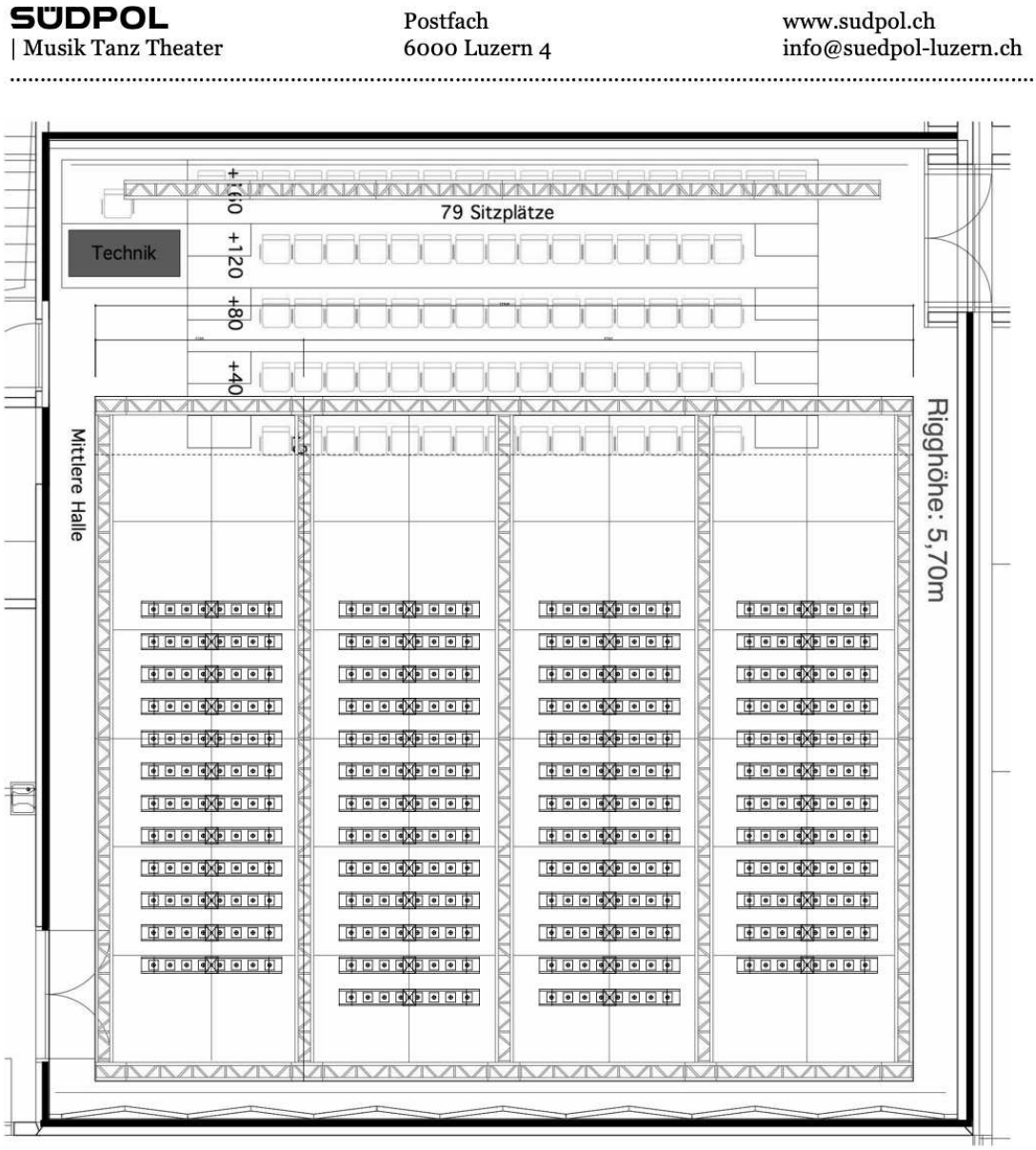


Fig. 7.9 Schéma de la disposition des trappes au plafond de la scène du théâtre.

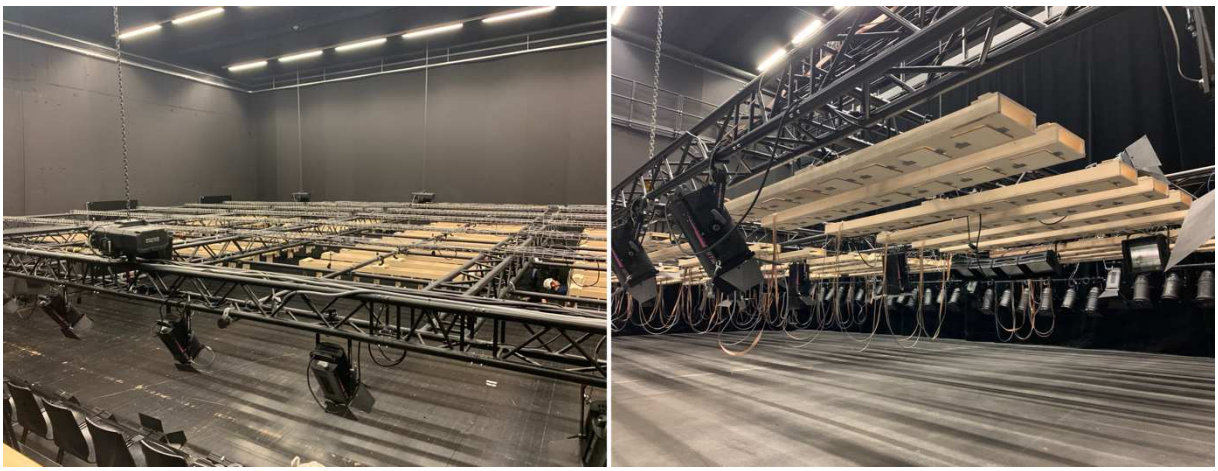


Fig. 7.10 Photos de la structure de la machine et son installation dans la scène de la salle de théâtre Südpol à Lucerne. Il s'agit de 50 cages en bois avec 8 trappes dont l'ouverture est contrôlée par un électroaimant. Chaque trappe est remplie avec des objets de différents matériaux.

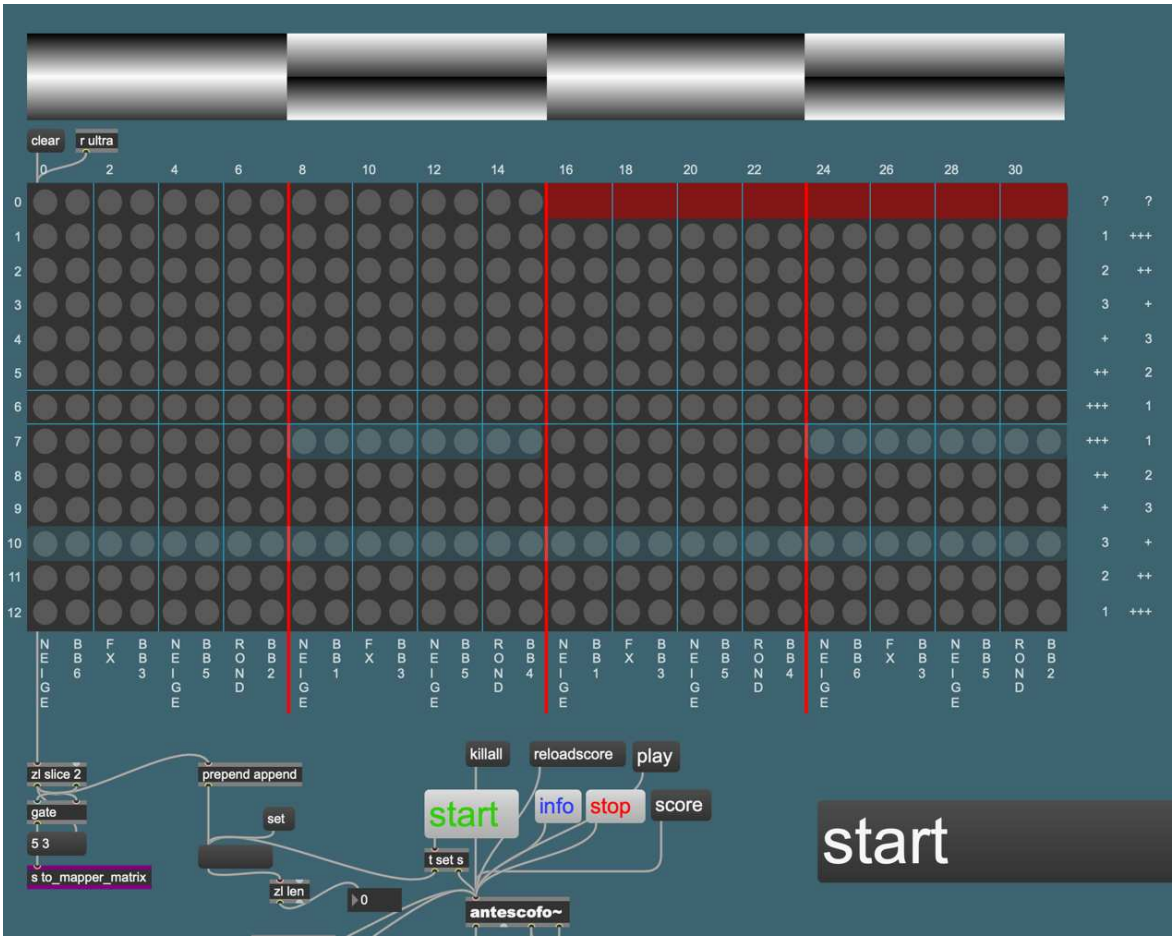


Fig. 7.11 Capture d'écran du patch Max de visualisation et composition du système de 400 trappes. L'interface permet de situer les différentes trappes dans l'espace et voir le contenu de chacune. Il y a des matières « neige », différentes tailles de billes, des rondelles en métal et « fx » qui correspond à différents éléments comme des punaises, petits objets en métal, des papillons en métal, etc. Les panneaux de la machine sont séparés en 4 colonnes de gauche à droite avec des coordonnées pour chaque trappe. La première colonne a plus de densité d'éléments au centre, la deuxième plus de densité aux bords, la troisième plus au centre et la quatrième plus aux bords.

```
@proc_def colum($colum, $duree)
{
    @local $idx := 0, $ctl
    {
        Loop $duree
        {
        $ctl := [$colum, $idx]
        to_ultra $ctl  1
        $idx += 1
        } until ($idx == 10)

    }
}

@proc_def range($range, $init, $end, $duree)
{
    @local $idx := $init, $ctl
    {
        Loop $duree
        {

        $ctl := [$idx, $range]
        to_ultra $ctl  1
        if($init<$end)
        {
            $idx += 1
        }
        else
        {
            $idx -= 1
        }

        } until ($idx == $end)

    }
}
```

```
@proc_def play($list, $duree)
{
    @local $idx := 0, $res

        loop $duree
        {
            $res := [$list[$idx], $list[$idx+1]]
            to_ultra $res
            $idx += 2
        } until ($idx > $list.size()-1)

}
```

```
@proc_def line($x_init, $y_init, $x_end, $y_end, $duree)
{
    @local $idx := 0, $x, $y, $end, $x_dist, $y_dist, $ctl, $end2
    {
        $x_dist := $x_init - $x_end
        $y_dist := $y_init - $y_end

        if(@abs($x_dist) > @abs($y_dist))
        {
            $end := $x_dist
            $end2 := @abs($x_dist)
        }
        else
        {
            $end := $y_dist
            $end2 := @abs($y_dist)
        }
        $x := $x_dist/$end
        $y := $y_dist/$end
        Loop $duree
        {
            $ctl := [@round($idx*$x)+$x_init, @round($idx*$y)+$y_init]
            to_ultra $ctl  1
            if($end<0)
            {
                $idx += 1
            }
            else
            {
                $idx -= 1
            }
        } until (@abs($idx) > $end2)
    }
}
```

```
@proc_def carre($x_init, $y_init, $x_end, $y_end, $duree)
{
    @local $ctl, $x_dist, $y_dist, $list, $idx := 0, $send_list := []

        $x_dist := @abs($x_init - $x_end)
        $y_dist := @abs($y_init - $y_end)

        $list := [ [[$i+$x_init, $j+$y_init] | $i in $x_dist+1 ] | $j in $y_dist+1]
        $list := $list.flatten(1)

        loop $duree
        {
            to_ultra ($list[$idx]) 1
            $send_list.push_back($list[$idx])
            $idx += 1
        } during [$list.size() #]
        to_ultra ($send_list.flatten())

}
```

Fig. 7.12 Quelques processus pour jouer la machine avec différents modes pour créer des figures, par exemple des colonnes, des lignes, des régions avec des rythmes ou en accords.

```
group
{
    group @tempo 45
    {
        label INIT
        ::play0([25, 6])        //BB6 1
        30 ::play0([15, 6])     //BB4 1
        40 ::play0([3, 12])     //BB3 1
        20 ::play0([24, 6])     //*neige 1
        20 ::play0([20, 1])     //*neige 1
        20 ::play0([4, 11])     //*neige 2
        17 ::play0([20, 2])     //*neige 1
        19 ::play0([8, 5])      //*neige 2
        4 ::play0([28, 6])      //*neige 1
        4 ::play0([4, 1])       //*neige 1
        //174 / 02:54
        40 ::play0([8, 6])      //*neige 1
        4 ::play0([16, 12])     //*neige 1
        4 ::play0([0, 1])       //*neige 1
        4 ::play0([20, 12])     //*neige 1
        4 ::play0([0, 12])      //*neige 1
        4 ::play0([16, 1])      //*neige 1
        4 ::play0([12, 6])      //*neige 1
        4 ::play0([13, 6])      //*BB5 1
        2 ::play0([19, 11])     //*BB3 2
    }
    //mini premier klimax
    +=> 20 ::play0([0, 2])      //neige 2
    3 ::play0([8, 8])       //neige 2
    5.5 ::play0([8, 4])     //neige 3
    7.5 ::play([19,1, 19,3, 19,5, 19,9], 0.58)      //BB3 1 / BB3 3 / BB3 ++ / BB3 +
    //rytm
    30 ::play([0, 10, 3, 10, 6, 10, 9, 10, 12, 10, 15, 10, 18, 10, 21, 10, 24, 10, 27, 10,
    2 ::play0([31, 7])                              //*BB add 1
    // 2 ::play0([21, 6])                               //*BB6 +++
    2 ::play0([11, 0])                              //*BB add back 1
    2 ::play0([13, 7])                              //*BB add 1
    2 ::play0([12, 7])                              //*BB add 1
    2 ::play0([12, 0])                              //*BB add back 1
    2 ::play0([1, 1])                               //*BB6 1
    2 ::play0([8, 0])                               //*BB add back 1
    2 ::play0([11, 7])                              //*BB add 1
    2 ::play0([27, 6])                              //*BB3 1
    2 ::play0([13, 0])                              //*BB add back 1
    2 ::play0([27, 7])                              //*BB add 1
    2 ::play0([14, 0])                              //*BB add back 1
    2 ::play0([17, 1])                              //*BB1 1
    2 ::play0([5, 0])                               //*BB add back 1
    2 ::play0([28, 7])                              //*BB add 1
    2 ::play0([4, 0])                               //*BB add back 1
    2 ::play0([19, 12])                             //*BB3 1
    2 ::play0([14, 7])                              //*BB add 1
    2 ::play0([15, 0])                              //*BB add back 1
    2 ::play0([3, 1])                               //*BB3 1
    2 ::play0([6, 0])                               //*BB add back 1
```

Fig. 7.13 Extrait du début de la partition électronique de la pièce *Stück für die Schwerkraft* pour 400 trappes. Le tempo du premier group à la ligne 426 est variable pour accélérer ou ralentir le début de la pièce. À la ligne 451, on retrouve un opérateur de continuation « +=> » pour commencer à jouer juste après la fin du premier groupe d'actions. Toutes les fonctionnalités du langage Antescofo vont permettre de créer très facilement un environnement de programmation pour contrôler la machine et ses 400 trappes.

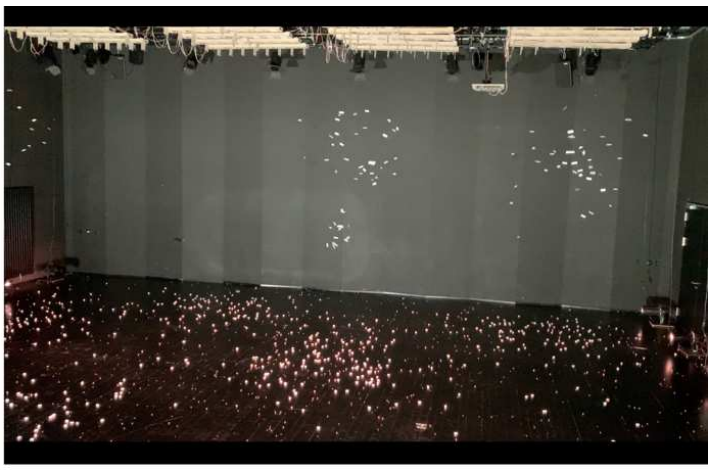


Fig. 7.14 Différents états du dispositif lors de *Stück für die Schwerkraft*. En haut, le début de la pièce, deux états intermédiaires au cours de la performance puis l'état final avec tous les objets par terre.

VIII. Musiques acousmatiques

VIII.1. Orchestration de l'électronique dans les pièces électroacoustiques

Dans ma pièce acousmatique *Fond diffus*[125], mon but était de confronter le monde du son électroacoustique et de la synthèse à une approche orchestrale de l'écriture de l'électronique. Cet objectif implique une écriture précise de la superposition de couches sonores avec plusieurs échelles temporelles ainsi qu'une écriture très détaillée de la synthèse sonore et de la spatialisation.

Ce travail déjà ancien a été le point de départ d'une réflexion plus profonde sur la notation de l'électronique dans un langage de programmation textuel plutôt qu'à travers la représentation graphique des événements musicaux sur une ligne temporelle dans une station audionumérique (DAW). La figure suivante présente un extrait de la session.

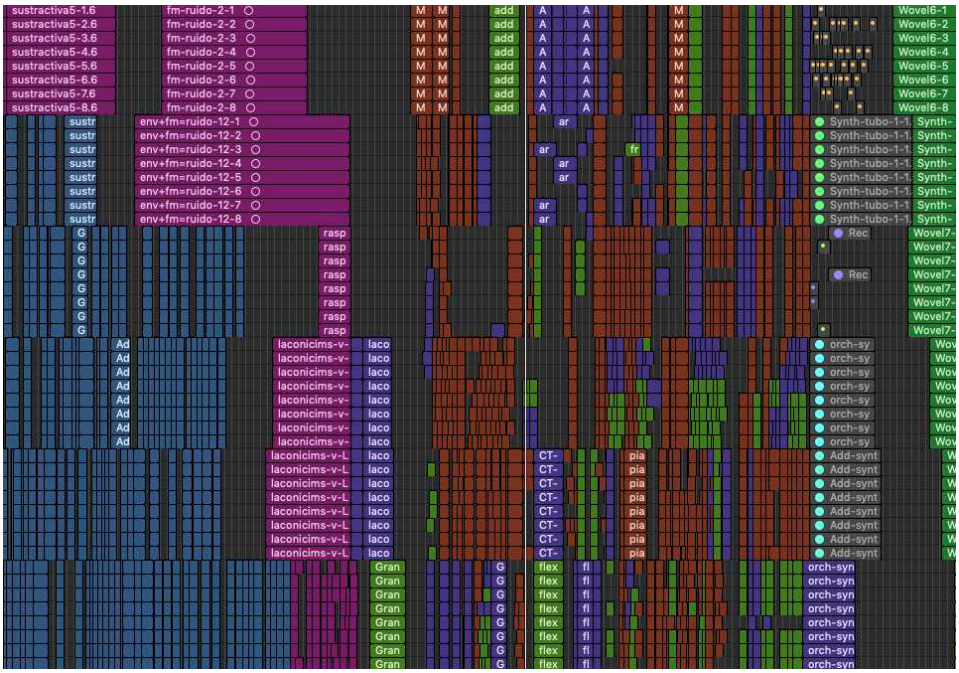


Fig. 8.1 Capture d'écran de la session Logic de la pièce électroacoustique *Fond diffus* avec plus de 150 canaux audio.

La composition dans un DAW est conditionnée par l'utilisation d'une vue graphique et des proportions spatiales relatives des clips qui représentent les sons. La structure interne des sons est fixe et ne peut pas être modifiée une fois les sons enregistrés.

[125] *Fond diffus*, pièce électroacoustique 24 canaux créée au ZKM à Karlsruhe en 2012, Prix résidence Musiques & Recherches.

Lorsque l'on calcule le rendu final de la pièce, l'espace de diffusion et de réverbération est fixé.

Toutes ces considérations et ces limitations rendent impossible le développement de pièces électroacoustiques en temps réel dans lequel le matériel électroacoustique doit pouvoir être modifié à tout moment avec des superpositions temporelles et des interactions complexes.

Ces limitations m'ont amené à utiliser AntesCollider pour pallier ces problèmes.

VIII.2. Pièces électroacoustiques jouées en live, entre écriture et improvisation

La musique électroacoustiques ou acousmatique se caractérise principalement par une composition en *temps différé*, hors temps d'exécution, et fixée sur bande ou sur un support informatique. Bien que cette musique puisse présenter une grande liberté (rythmes, timbres, morphologies…) dans sa conception/composition, le résultat est fixé sur un support et ne peut pas être modifié en cours de diffusion. Il est certes possible d'interpréter la diffusion avec des haut-parleurs, comme c'est le cas des œuvres jouées sur un acousmonium, pratique répandue dans le milieu de la musique acousmatique. Cette interprétation consiste principalement à moduler la diffusion d'une bande en choisissant les haut-parleurs de diffusion dans un ensemble réunissant différents types et différents modèles (acousmonium). Ce choix permet de jouer sur la position, la spatialisation ou encore le timbre et apporte un relief supplémentaire à l'écoute. Si la musique se renouvelle à chaque fois du fait de son interprétation, celle-ci reste limitée à une petite partie du contenu musical.

L'idée de jouer la musique électronique en « live » redonne à la musique son caractère d'art de l'instant (avec tout ce que cela implique : spontanéité, accidents, interactions de l'environnement, etc.). Dans le domaine de la musique électronique live, on trouve principalement des musiques improvisées (ou pseudo-improvisées) ou encore des musiques avec une composante très écrite dans laquelle seuls quelques paramètres sont variables et seront modifiés au cours de l'interprétation de la pièce. Il y a aussi des cas dans lesquels la musique est entièrement écrite et interprétée par un instrumentiste électronique comme dans le cas d'une partition de musique acoustique écrite.

Ma démarche dans la pièce acousmatique *Curvatura II*, et surtout dans la pièce audiovisuelle *Las Pintas*, se positionne à la frontière de ces différentes modalités dans le sens où bien que tout soit complètement écrit dans une partition électronique, l'interprète peut à tout moment intervenir et orienter le déroulement musical dans la limite de certaines modifications rendues possibles à certains endroits spécifiques de la partition. La caractéristique principale de ces pièces électroacoustiques est qu'elles sont générées en temps réel par l'ordinateur, ce qui leur confère une très grande liberté et flexibilité lors de la diffusion. Voici quelques modifications possibles au cours de la diffusion par un interprète ou via d'autres dispositifs :

- modification des processus de génération d'événements musicaux, par exemple plus ou moins de densité d'éléments ou modification des synthèses sonores et des traitements ;

- modification des synthèses et des traitements à partir des données de l'extérieur comme de la vidéo (dans le cas de la pièce *Las Pintas*) ;

- modification de la spatialisation pendant la performance ;

- modification de l'espace sonore à travers le changement des réverbérations locales et globales de la composition ;

- modification du dispositif de diffusion, par exemple en changeant le décodeur ambisonique pour l'adapter au système qui sera utilisé pendant la performance[126].

VIII.3. *Curvatura II*

Curvatura II est une composition acousmatique générée en temps réel dans un système de spatialisation HOA. Elle est d'abord une exploration, une recherche sur la création de différentes superpositions de couches sonores et sur la transformation du matériau sonore à travers le temps et l'espace. La courbure du temps réalisée à travers la déformation continuelle des tempi (*accelerandos*, *rallentandos*, modulations) et des paramètres de la synthèse du son ainsi que de leurs superpositions génèrent aussi bien la micro- que la macro-structure de la pièce. La superposition de ces éléments en constante transformation et leurs mutations donne naissance à des agrégats sonores à partir desquels émerge un effet sonore et perceptif global qui, en même temps, construit la directionnalité et la trajectoire du discours musical.

Au niveau technique, la pièce est entièrement écrite dans le langage de programmation Antescofo en utilisant la librairie AntesCollider, ce qui permet, d'une part, de jouer avec des transformations et des superpositions temporelles en temps réel et, d'autre part, d'avoir un contrôle fin de tous les paramètres de synthèse et de spatialisation du son. Cette écriture et cette génération en temps réel vont permettre aussi de modifier des paramètres comme la spatialisation pour adapter la diffusion à différentes configurations et systèmes de diffusion, mais aussi pour moduler des paramètres perceptifs comme les réverbérations. Cette approche temps réel permet la prise en compte de paramètres inconnus lors de la composition comme la taille de la salle ou des égalisations et autres traitements spécifiques en fonction de l'acoustique du lieu. La pièce peut de cette façon se réadapter à chaque fois au lieu où elle sera reproduite, ce qui amène une flexibilité qui n'est pas possible avec une musique fixée.

[126] Ceci est rendu possible car toutes les descriptions spatiales sont exprimées en coordonnées cartésiennes ou polaires indépendamment du système de diffusion et donc transposables sur n'importe quel système.

Au niveau rythmique, la pièce réalise des superpositions de différents tempi qui peuvent être statiques, en *accelerando* ou en *rallentando*. Dans certains passages, il peut y avoir jusqu'à une trentaine de voix en parallèle, chacune avec ses propres rythme et tempo. Des mouvements plus globaux peuvent être réalisés en modulant un ou plusieurs tempi par un autre tempo global qui à son tour peut lui aussi être modulé. Cette possibilité faisant dépendre des tempi « fils » d'un tempo « parent » permet d'organiser les interactions temporelles en hiérarchies, les niveaux hauts permettant d'avoir un contrôle plus global et les niveaux bas, un contrôle plus fin sur les fils.

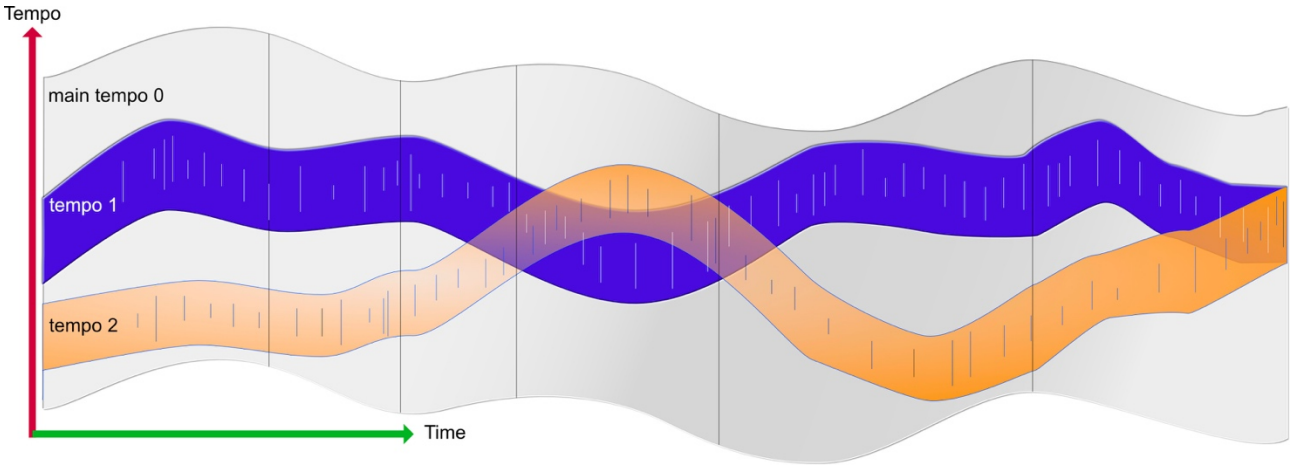


Fig. 8.2 Représentation graphique de la modulation des tempi pour les événements et les processus musicaux ; le tempo 1 et le tempo 2 (fils) ont leur propre tempo qui module dans le temps. Le tempo 0, le tempo global « parent » à son tour module les deux tempi « fils » 1 et 2, ce qui permet de générer des modulations imbriquées avec une relation et une interaction entre les tempi.

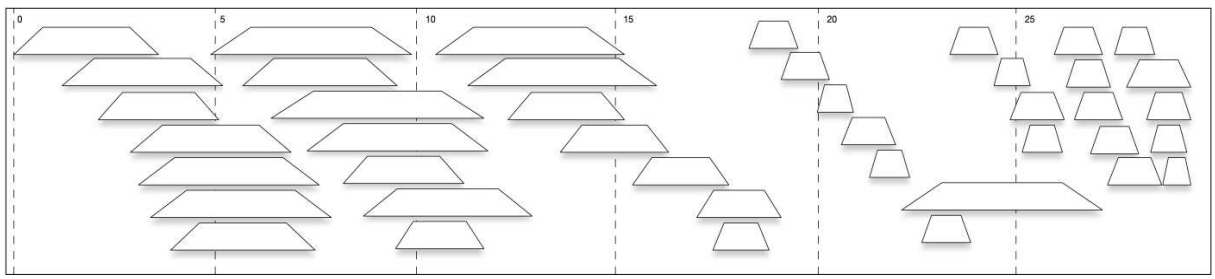


Fig. 8.3 Représentation graphique d'un extrait de 30 secondes de *Curvatura II* d'une couche temporelle. Chaque losange représente une synthèse granulaire qui joue des rythmes générés à partir de probabilités de manière fixe, en *accelerando* ou en *rallentando*.

```
482  NOTE 60 4.2 e3_ribatutos4
483
484              5 print ribatutoC1_note1
485              obj::crea_track_HOA("obj_synth6_4", 0, 1, -100, [["TAddic_5_8", "preset", "synth6_2", "lag", 0], ["TGranInterpExt", "preset", "
486              $tracks("obj_synth6_4").ryth_prob([["TGranInterpExt", "t_trig", 1], ["TPan8", "pos", 0]], 1, 1, 800)
487              $tracks("obj_synth6_4").line_param("TAddic_5_8", "transp", 5, 2.5, 6)
488              $tracks("obj_synth6_4").amp([-6, 6.5, "circ", 1, 1.5, -118])
489              $tracks("obj_synth6_4").change_proc_tempo([800, 6.5, "circ", 1200, 1.5, 100])
490
491          5.5 print ribatutoB5_note2
492          $tracks("obj_synth6_3").line_param("TAddic_5_8", "transp", 2, 4, 5)
493          $tracks("obj_synth6_3").amp([-10, 5.5, "circ", 2, 1.5, -118])
494          $tracks("obj_synth6_3").change_proc_tempo([700, 5.5, "circ", 1500, 1.5, 800])
495
496      4.5 print ribatutoA8_note3
497      $tracks("obj_synth6_2").line_param("TAddic_5_8", "transp", 3, 1.5, 5.5)
498      $tracks("obj_synth6_2").amp([-10, 5.5, "circ", 0, 1.5, -118])
499      $tracks("obj_synth6_2").change_proc_tempo([700, 5.5, "circ", 1500, 1.5, 800])
500
501              5 print ribatutoC1_note4
502              $tracks("obj_synth6_4").line_param("TAddic_5_8", "transp", 4, 2.2, 6.5)
503              $tracks("obj_synth6_4").amp([-6, 6.5, "circ", 1, 2.5, -118])
504              $tracks("obj_synth6_4").change_proc_tempo([1000, 6.5, "circ", 400, 2.5, 1000])
505
506          6.3 print ribatutoB6_note5
507          $tracks("obj_synth6_3").line_param("TAddic_5_8", "transp", 2.5, 3, 1.5)
508          $tracks("obj_synth6_3").amp([-20, 1.5, "circ", 10, 2.5, -118])
509          $tracks("obj_synth6_3").change_proc_tempo([1500, 1.5, "circ", 2000, 2.5, 800])
510
511      print ribatutoA9_note6
512      $tracks("obj_synth6_2").line_param("TAddic_5_8", "transp", 3, 2.2, 3.5)
513      $tracks("obj_synth6_2").amp([-30, 3.5, "circ", 5, 1.5, -118])
514      $tracks("obj_synth6_2").change_proc_tempo([700, 3.5, "circ", 1500, 1.5, 800])
515
516              4 print ribatutoC1_note7
517              $tracks("obj_synth6_4").line_param("TAddic_5_8", "transp", 1.2, 2.2, 1.5)
518              $tracks("obj_synth6_4").amp([-30, 1.5, "circ", 8, 1.5, -118])
519              $tracks("obj_synth6_4").change_proc_tempo([2000, 1.5, "circ", 1400, 1.5, 1000])
520
521          0.7 print ribatutoB6_note8
522          $tracks("obj_synth6_3").line_param("TAddic_5_8", "transp", 2., 1.3, 1.5)
523          $tracks("obj_synth6_3").amp([-40, 1.5, "circ", 5, 2.5, -118])
524          $tracks("obj_synth6_3").change_proc_tempo([1500, 1.5, "circ", 3000, 2.5, 800])
525
526
527  NOTE 60 8.2 e3_ribatutos5
528
529          /// continuar con elementos estaticos a nivel tempo
530                  1 print _note10
531                  obj::crea_track_HOA("obj_synth6_5", 0, 1, -40, [["TAddic_5_8", "preset", "synth6_2", "lag", 0], ["TGranInterpExt", "preset"
532                  $tracks("obj_synth6_5").ryth_prob([["TGranInterpExt", "t_trig", 1], ["TPan8", "pos", 0]], 1, 1, 2500)
533                  $tracks("obj_synth6_5").set("TAddic_5_8", ["transp", 3.5])
534                  $tracks("obj_synth6_5").amp([-40, 2.5, "circ", 5, 2.5, -118])
535
536      1.5 print ribatutoA8_note11
537      $tracks("obj_synth6_2").set("TAddic_5_8", ["transp", 3.])
538      $tracks("obj_synth6_2").change_proc_tempo(1000)
539      $tracks("obj_synth6_2").amp([-40, 2.1, "circ", 10, 1.5, -118])
540      $tracks("obj_synth6_2").change_proc_tempo([1000, 2.1, "circ", 3000, 1.5, 800])
```

Fig. 8.4 Extrait de la partition électronique de *Curvatura II*, pièce entièrement réalisée avec la librairie AntesCollider. Chaque *track* qui génère des séquences rythmiques avec différents types de synthèse est modulée par des changements de tempo (ex. lignes 489, 494, 504…) avec la méthode `.change_proc_tempo`. Les tempi sont modulés à travers des BPF (*Break Point Functions*), mais ils peuvent aussi être modulés de l'extérieur ou à partir d'un tempo « parent ».

IX. Musiques mixtes

IX.1. La captation gestuelle

Le geste comme producteur des sons électroniques existe depuis un siècle (Thérémine 1919, Ondes Martenot 1927). À l'Ircam et dans la tradition des musiques mixtes, les recherches commencent dans les années 1980, notamment avec les travaux de Barry Vercoe et du flûtiste Larry Beauregard avec une flûte MIDI qui incorporait des interrupteurs optiques pour connaître l'état d'ouverture ou de fermeture des clés. L'objectif était de réaliser un suivi de partition, identifié comme un composant clé pour réaliser un accompagnement automatique[127].

À partir des années 2000, avec l'avènement de matériels plus accessibles et miniaturisés pour la captation des gestes des instrumentistes, l'utilisation de ces technologies de captation a commencé à intéresser directement les compositeurs. Des musiciens s'y sont aussi intéressés afin d'étendre les possibilités de jeux de leur instrument pour en faire des « instruments augmentés ».

Comme le microphone pour la captation du signal audio, la captation gestuelle dans la pratique des musiques mixtes a permis une description en temps réel des événements musicaux joués par les instrumentistes. Cette description peut être analysée et utilisée pour différentes opérations de production du son. La captation gestuelle vient ainsi compléter l'analyse du jeu instrumental avec des paramètres que les seules données audio ne peuvent pas fournir – comme la vitesse et le sens du mouvement des mains, la préparation et l'accompagnement du geste instrumental (par exemple chez les percussionnistes), les mouvements dans l'air, les déplacements sur la scène, les positions relatives et absolues des mains ou des archets pour les cordes, etc.

Les capteurs

Parmi les différents types de capteurs (optiques, mécaniques, magnétiques, etc.) utilisés en musique mixte, j'ai surtout travaillé avec des IMU (*Inertial Measurement Unit*) comme les accéléromètres ou les gyroscopes, mais aussi avec les magnétomètres permettant de caractériser une orientation absolue. Les capteurs optiques, notamment avec les capteurs infrarouges (comme la Kinect, caméra conçue pour capter les mouvements du corps), ont aussi beaucoup été utilisés pour compléter les données d'accéléromètre et obtenir une position absolue des mains ou du corps du performeur (musicien, danseur ou comédien).

[127] https://www.youtube.com/watch?v=vOYky8MmrEU

Fig. 9.1 À gauche, quelques capteurs mécaniques ; à droite, un capteur photographique infrarouge Kinect version 2.

Mon travail sur la captation gestuelle a débuté en 2005 avec la composition de *Gravita* pour alto et électronique en temps réel, pièce créée à l'espace de projection de l'Ircam lors du concert cursus en 2006. Dans cette pièce, l'idée était d'utiliser un accéléromètre sur trois axes pour mesurer la vitesse de déplacement de l'archet de l'altiste pour contrôler des traitements électroniques ainsi que pour exciter un modèle physique de corde qui a son tour produisait, à travers le mapping des modes de vibrations du modèle vers la synthèse, un traitement en temps réel de l'électronique.

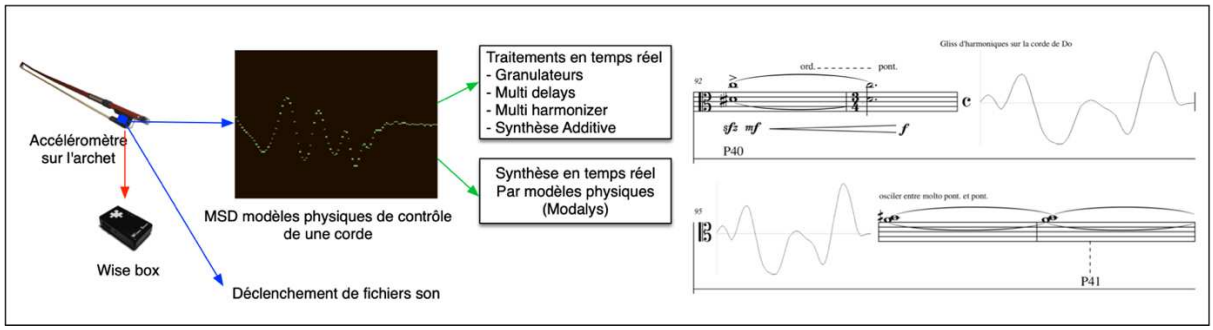


Fig. 9.2 À gauche, schéma de représentation du capteur placé sur l'archet de l'alto pour déclencher des fichiers son et pour exciter un modèle physique masse-ressort créé avec la librairie MSD. Le modèle physique va contrôler à son tour différents traitements et synthèses. À droite, un extrait de la partition avec des mouvements rapides de l'archet pour générer des glissandos d'harmoniques et en même temps exciter le modèle physique.

Pour cette pièce, j'ai utilisé un microcontrôleur Wise Box (2005) qui se connecte sans fil à l'ordinateur via le protocole OSC. Étant donné le prix élevé de ces microcontrôleurs à l'époque, j'ai décidé par la suite de les fabriquer moi-même en utilisant du matériel déjà existant et de l'adapter à mes besoins. J'ai développé et programmé trois systèmes à partir de 2007, les deux premiers basés sur des microcontrôleurs MakingThings avec le protocole xbee pour l'envoi des données sans fil. Le troisième système a été développé à partir de microcontrôleurs Arduino et

consistait en deux microcontrôleurs, un pour l'émission des données depuis un capteur numérique 9 axes (3 accéléromètres, 3 gyroscopes et 3 boussoles) et un deuxième pour recevoir les données et les envoyer via OSC en Ethernet vers l'ordinateur de concert. Avec ces capteurs, j'ai réalisé les compositions *M-brana* pour percussions, contrebasse, captation gestuelle et électronique temps réel, et *Amas* pour hautbois, violon, guitare, percussions, contrebasse et électroniques avec système de captation gestuelle.

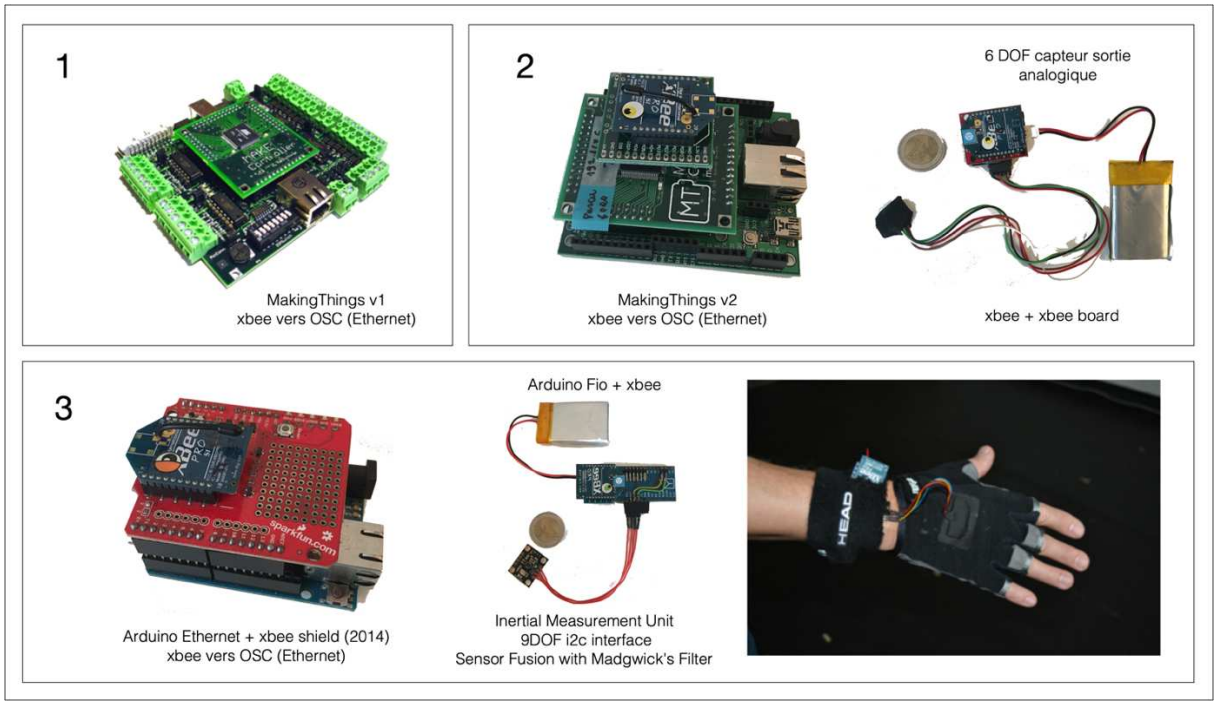


Fig. 9.3 Trois systèmes de captation gestuelle DIY (*do-it-yourself*). Versions 1 et 2 avec des microcontrôleurs MakingThings permettant la réception sans fil (xbee) depuis des capteurs analogiques via le protocole OSC. La version 3 est un système développé à partir de microcontrôleurs Arduino. Dans ce système, les capteurs sont numériques et le filtre de fusion des données est réalisé dans le microcontrôleur Arduino Fio.

Par la suite, j'ai utilisé le capteur RIoT[128] développé par Emmanuel Fléty à l'Ircam qui intègre un capteur 9 DOF avec un microprocesseur intégré et programmable avec le logiciel *Energia* (un environnement similaire à l'IDE Arduino). Le module RIoT, diffusé aujourd'hui par BITalino, embarque un capteur 9 axes de ST Microelectronics avec 3 accéléromètres, 3 gyroscopes et 3 magnétomètres, tous de précision 16 bits. Les données sont envoyées sans fil (en Wifi) via le protocole OSC directement vers un réseau local. Le cœur de la carte est un module Wifi Texas Instrument avec un processeur Cortex ARM 32 bits qui exécute le programme et s'occupe de la pile Ethernet/WAN. Il intègre aussi par défaut une implémentation du filtre Madgwick pour fusionner les données des différents capteurs et obtenir les angles d'orientation (*yaw, pitch, roll*) avec une latence d'environ 5 ms. J'ai utilisé ce module pour les

[128] http://ismm.ircam.fr/riot/

compositions *Homotopy* pour percussions, captation gestuelle et percussions, ainsi que pour le projet *GeKiPe*. Les avantages de ce module sont :

- sa taille réduite ;
- les fonctions *plug and play* ;
- l'envoi de l'OSC directement vers l'ordinateur hôte ;
- la programmation locale (sur le contrôleur) de processus permettant d'avoir une meilleure précision de la captation (débruitage, fusion de données, détection de *kicks*, calcul de la direction du mouvement, accélération, position angulaire, etc.).

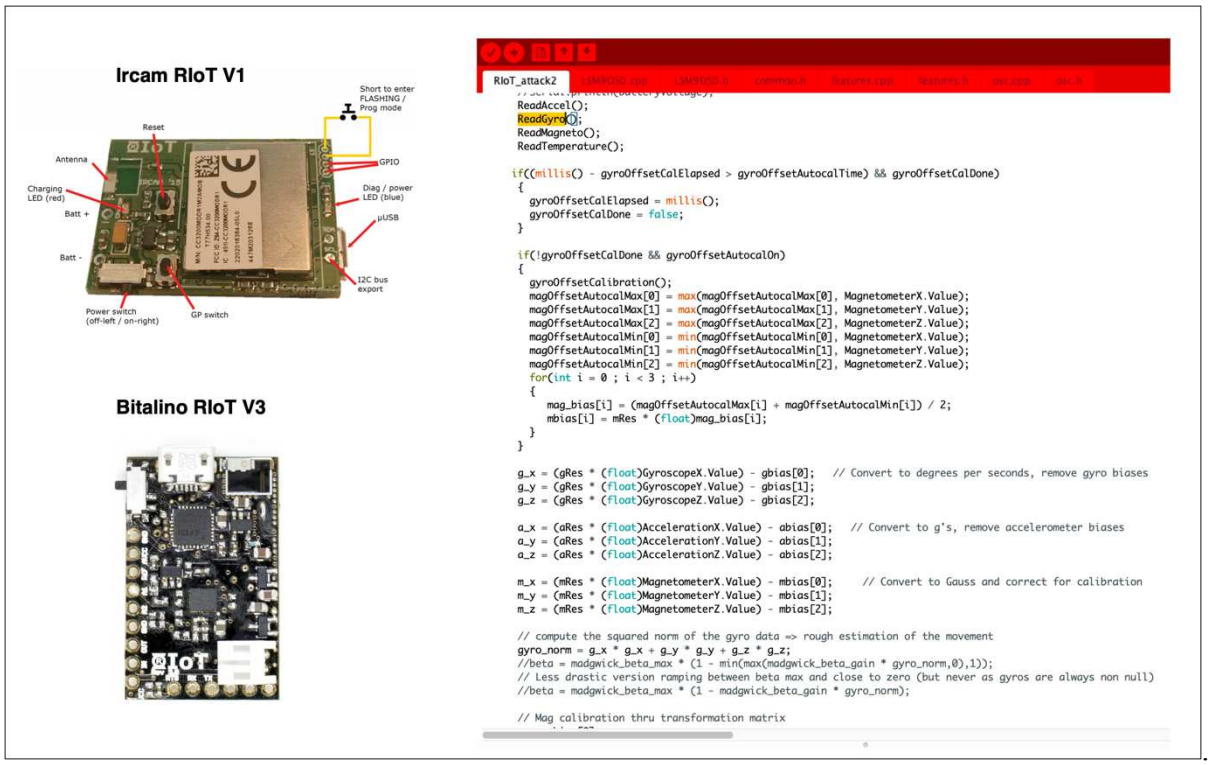


Fig. 9.4 En haut à gauche, le capteur RIoT développé à l'Ircam dans sa version 1 et en bas, la version 3 distribuée par Bitalino[129]. À droite, une partie du code de programmation du module dans l'interface Energia. Cette interface lui confère une grande flexibilité puisqu'on va pouvoir customiser et rajouter de nouveaux traitements dans le module lui-même pour avoir une meilleure précision et envoyer par Wifi dans un réseau local en OSC des données déjà prétraitées.

Souvent, dans les pièces interactives qui utilisent des capteurs IMU, leur positionnement vise les parties du corps humain dont les mouvements sont les plus expressifs, par exemple les mains des instrumentistes. De plus, placer les capteurs sur des gants est depuis longtemps assez courant, par exemple chez les artistes Laetitia

[129] https://plux.info/kits/376-bitalino-r-iot-810121007.html

Sonami, Marcelo Wanderley, Pierre-Yves Fortier, Michel Waisvisz, Georges Aperghis, Imogen, etc.

Gestes

La plupart des activités musicales (par exemple l'interprétation, la direction d'orchestre, la danse) impliquent des mouvements ou des gestes du corps. Les gestes musicaux peuvent être étudiés en fonction de leurs aspects spatiaux ou fonctionnels, de leur utilisation dans les performances (comme outils de communication ou de contrôle) ou selon des fins artistiques métaphoriques. Les progrès récents de l'informatique, de l'électronique et des capteurs ont suscité un intérêt croissant pour la conception de nouvelles interfaces musicales permettant aux chercheurs et aux artistes d'aborder les questions relatives au mouvement et aux gestes dans un contexte musical.

Les gestes musicaux peuvent être interprétés comme l'intersection entre des actions observables et des images mentales [Del88]. Ils peuvent être étudiés à différents niveaux en allant du purement fonctionnel au purement symbolique, que nous les considérions comme efficaces (produisant du son), sous la forme d'accompagnement (par exemple en soutenant le geste efficace) ou en fonction d'indices plus figuratifs [CaWa00]. Une définition analogue propose qu'un « *geste est un mouvement ou un changement d'état qui devient marqué comme significatif par un agent. […] Pour qu'un mouvement ou un son soit (devenu) un geste, il doit être pris intentionnellement par un interprète, qui peut ou non être impliqué dans la production sonore réelle d'une performance, de manière à le doter des attributs de la signification humaine.* » [GrKi06] En d'autres termes, les gestes musicaux doivent être significatifs et porteurs d'informations importantes (communication, contrôle, métaphore).

Dans ma démarche d'utilisation des systèmes de captation gestuelle pour la composition des musiques interactives, je me suis intéressé fondamentalement à deux typologies de gestes :

- les gestes instrumentaux propres à la pratique instrumentale experte acquis et travaillés pendant des années au sein d'une tradition d'enseignement et d'apprentissage ; et
- les gestes non instrumentaux, qui n'ont pas une relation directe avec le jeu instrumental, par exemple les mouvements dans l'air ou les gestes à visée chorégraphiques.

Gestes instrumentaux

Le geste instrumental est caractérisé par une interaction physique directe entre le geste sur l'instrument et la production du son. Des études spécifiques sur le geste instrumental ont élaboré différentes définitions, par exemple les catégories gestuelles basées sur la notion de continuum énergétique de Claude Cadoz [CLF81] ou la typologie de gestes de François Delalande [Del88] à partir de l'analyse de l'interprétation de Glenn Gould. Ces catégories, par exemple les gestes d'excitation, de modification, de sélection, de polarisation et de maintien chez Cadoz ou les gestes

effecteurs, accompagnateurs et évocateurs chez Delalande, sont très importantes à prendre en compte pendant la composition pour déterminer les informations pertinentes à capter sur l'instrument acoustique. Par exemple, quand on pose un accéléromètre sur l'archet d'un instrument à cordes, il faut prendre en compte ces catégories pour savoir quel doit être le résultat sonore de l'électronique. Si on prend en compte ces catégories et leur association à un résultat électronique (*mapping*), le choix des gestes peut influer directement sur l'écriture instrumentale et le déroulement de la composition. Des trémolos plus ou moins serrés vont produire des gestes bien différents des pizzicatos, et ces derniers dépendent aussi de l'instrument (de sa taille dans la famille des cordes).

Amas pour hautbois, violon, guitare, percussions, contrebasse et électronique avec système de captation gestuelle, créée au festival Archipel de Genève par l'ensemble Vortex en 2012 est un exemple de pièce où l'électronique est réalisée en grande partie à partir des gestes des musiciens.

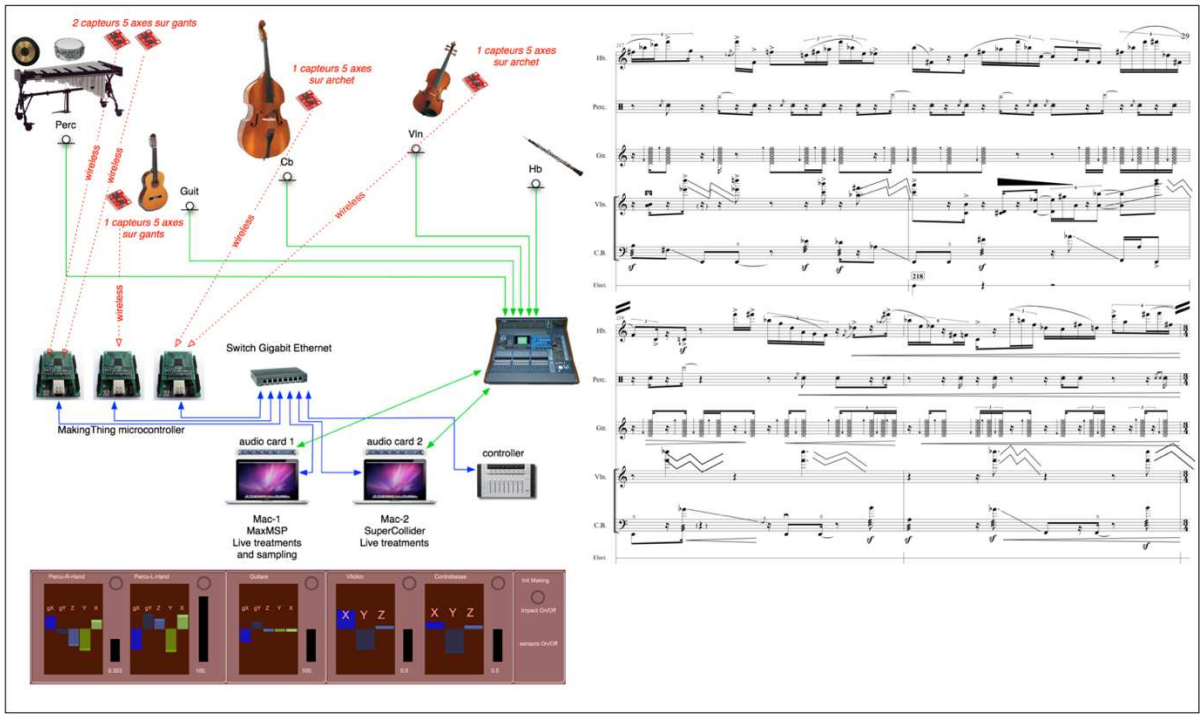


Fig. 9.5 Représentation de quelques éléments de *Amas* comme exemple d'une pièce réalisée avec des gestes instrumentaux. À gauche, le schéma technique du réseau de capteurs des instruments ; en bas, la visualisation des données de captation dans le logiciel Max. À droite, un extrait de la partition.

Gestes non instrumentaux

Les gestes non instrumentaux peuvent être aussi riches et permettent de produire des sonorités et modes de jeux différents de ceux produits par les gestes instrumentaux. Ils sont plus difficiles à intégrer car il faut les composer en fonction du résultat sonore, mais ils se superposent aussi aux gestes instrumentaux qu'il faut préserver, ce qui limite les possibilités de l'instrumentiste. Dans le projet *GeKiPe*, ils deviennent un paramètre supplémentaire non seulement au niveau sonore, mais aussi visuel à prendre en compte pendant l'élaboration de la composition. Ces gestes doivent aussi s'adapter à

l'instrumentiste, par exemple un percussionniste aura plus de possibilités de mouvements dans l'espace qu'un contrebassiste.

Parmi les gestes non instrumentaux faits avec le mouvement des mains, on peut avoir des gestes de pantomime [RiSc91] où l'instrumentiste imite le jeu instrumental comme dans certains passages des pièces *Homotopy* et surtout *Hypersphères* où le percussionniste n'a pas un instrument physique et joue des percussions dans l'air. Les autres gestes non instrumentaux vont être induits par leurs relations avec le résultat sonore à obtenir et le *mapping* vers les synthétiseurs, par exemple des mouvements vers le haut pour changer une hauteur ou vers l'avant ou l'arrière pour changer la dynamique. Il peut aussi y avoir des mouvements chorégraphiques comme des cercles, des zigzags, des spirales, des mouvements brusques du corps tout entier, etc. Dans cette catégorie, on peut trouver des gestes aussi bien continus que discrets ou un mélange des deux :

- Les gestes continus se caractérisent par des mouvements continus, analogues en musique à des mouvements graduels comme des *glissandi* ou des *crescendi*. Au niveau de l'électronique, ces mouvements vont servir non seulement à faire des contrôles continus sur les paramètres de l'électronique, mais aussi à naviguer (en 1D, 2D ou 3D), par exemple dans une base d'échantillons comme dans la synthèse granulaire ou concatenative. Ces mouvements peuvent aussi être subdivisés, par exemple avec des seuils. J'ai souvent mis en œuvre l'utilisation d'un quadrillage de l'espace par une caméra Kinect afin de localiser différentes opérations en un ensemble discret de régions (projet *GeKiPe*).
- Les gestes discrets sont utilisés comme déclencheurs (*triggers*). Ils correspondent à des mouvements brusques et rapides (*kicks*) et se caractérisent à partir de la dérivée de l'accélération.

Mapping

Le terme *mapping*, largement utilisé en musique, consiste à transposer et mettre en correspondance un domaine vers un autre. Ce terme est couramment utilisé dans d'autres disciplines artistiques ; on parle de *mapping* pour la projection de lumières ou de vidéos sur des volumes présentant un relief comme des façades d'églises, ou plus généralement des objets en 3D. Dans la musique, ce terme désigne plus spécifiquement la liaison entre les données issues de la captation gestuelle et les paramètres de la synthèse électronique.

Le *mapping* existe depuis l'apparition des instruments électroniques (DMI) puisqu'il existe une séparation entre le geste de contrôle et la production sonore. Cette séparation n'existe pas pour les instruments acoustiques puisque l'interface gestuelle fait généralement partie du système de production du son, et la relation de causalité entre geste de contrôle et production sonore est statiquement prédéfinie [MiWa06]. Dans les DMI, cette relation doit être créée et un même geste peut être mappé à différents paramètres et produire, selon le *mapping* choisi, des résultats très différents. Les entrées peuvent être n'importe quelle donnée et la sortie une nouvelle valeur adaptée

aux paramètres de contrôle de l'instrument numérique ou du traitement électronique qu'il va contrôler.

Il existe plusieurs types de *mapping* ; on peut les séparer en deux grandes catégories : le *mapping direct* et le *mapping indirect*.

Le *mapping direct* peut être divisé en deux catégories qui peuvent coexister en parallèle, une même donnée de captation peut être utilisée pour les deux cas en même temps :

- contrôles directs unidimensionnels : la captation gestuelle issue des données des capteurs est mappée directement vers un ou plusieurs paramètres d'un synthétiseur. Ces données vont être mises à l'échelle (normalisation et adaptation des plages de variation du paramètre d'entrée à celles du paramètre de sortie) et éventuellement transformées par une fonction de transfert (une NIM dans le langage Antescofo). Par exemple, se déplacer dans un buffer audio avec les données d'inclinaison de la main (utilisation pour la commande de SuperVP ou de la synthèse granulaire) ;
- contrôles directs multidimensionnels : une ou plusieurs données de captation vont servir à faire des interpolations entre différents états d'un ou plusieurs modules de synthèse (interpolation de préréglages). Par exemple, la vitesse de mouvement de la main va changer toutes les fréquences d'une synthèse additive en fonction d'états prédéfinis.

Dans le *mapping indirect*, les données de captation gestuelles vont être utilisées non pas pour contrôler directement un paramètre, mais constituent un moyen d'interagir avec un algorithme qui, lui, contrôle des paramètres sonores. Dans cette catégorie, on peut citer les modèles physiques qui seront excités et modifiés par la captation de gestes ou des algorithmes d'apprentissage supervisé (*machine learning*) comme les réseaux de neurones où l'on va apprendre à la machine à reconnaître différents gestes qui viendront moduler différents types de traitements. Parmi ces systèmes, on peut citer Mubu[130] (*Machine learning and gesture analyse*), la famille d'objets Max développée à l'Ircam où l'on trouve des systèmes pour suivi de geste (*gesture follower*[131]) ou des objets de reconnaissance gestuelle. Citons aussi le logiciel Wekinator[132] développé par Rebecca Fiebrink[133].

IX.4. *Homotopy*

Homotopy est une pièce pour percussions, captation gestuelle et électronique temps réel, dont la première version a été créée à Shizuoka, au Japon, en 2016 par Thierry

[130] https://forum.ircam.fr/projects/detail/mubu/
[131] http://ismm.ircam.fr/gesture-follower/
[132] http://www.wekinator.org
[133] https://www.doc.gold.ac.uk/~mas01rf/homepage/

Miroglio. La version finale a été créée à la Biennale de Venise en septembre 2017 puis a été redéveloppée avec AntesCollider.

Cette composition a été conçue grâce à une étroite collaboration avec le percussionniste Thierry Miroglio. La collaboration avec l'instrumentiste est absolument fondamentale pour travailler et composer avec des systèmes de captation gestuelle et électronique en temps réel.

En mathématiques, une homotopie (du grec *homos* : même, semblable et *topos* : lieu) est une déformation continue entre deux chemins. Cette définition se réfère directement à l'une des principales idées de la pièce qui consiste à transformer les sons réels des instruments de percussion de façon continue à travers les gestes du percussionniste, en utilisant des capteurs IMU 9 axes. Pour ce faire, différentes techniques d'analyse ont été utilisées telles que l'analyse des pics spectraux pour générer un son resynthétisé et modifié en temps réel en utilisant diverses techniques de synthèse comme la synthèse additive, soustractive, granulaire, par table d'onde, concaténative, etc.

La composition est structurée en trois parties correspondantes à trois zones sur scène, chacune avec ses propres caractéristiques de timbre, de rythme et d'électronique. Le percussionniste se déplace successivement de la zone 1 à la zone 2 puis 3. Le choix des instruments est fonction du timbre, pour bien caractériser chaque zone et sa sonorité. Une des contraintes de l'instrumentation est aussi de ne pas utiliser de grands instruments (à part le tam-tam) pour que la pièce soit la plus portable possible afin de faciliter sa performance dans différents contextes et différents lieux.

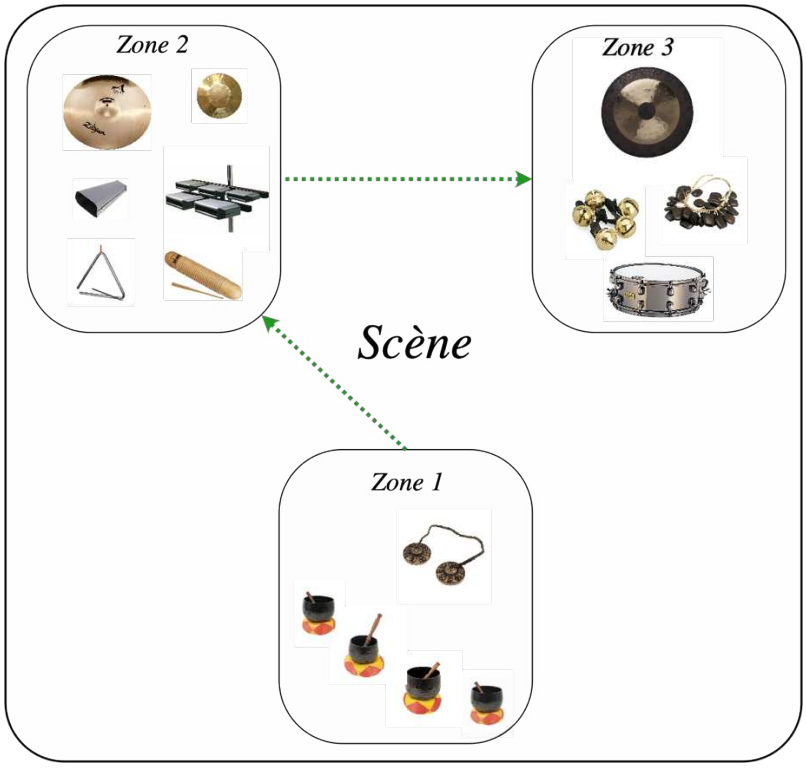
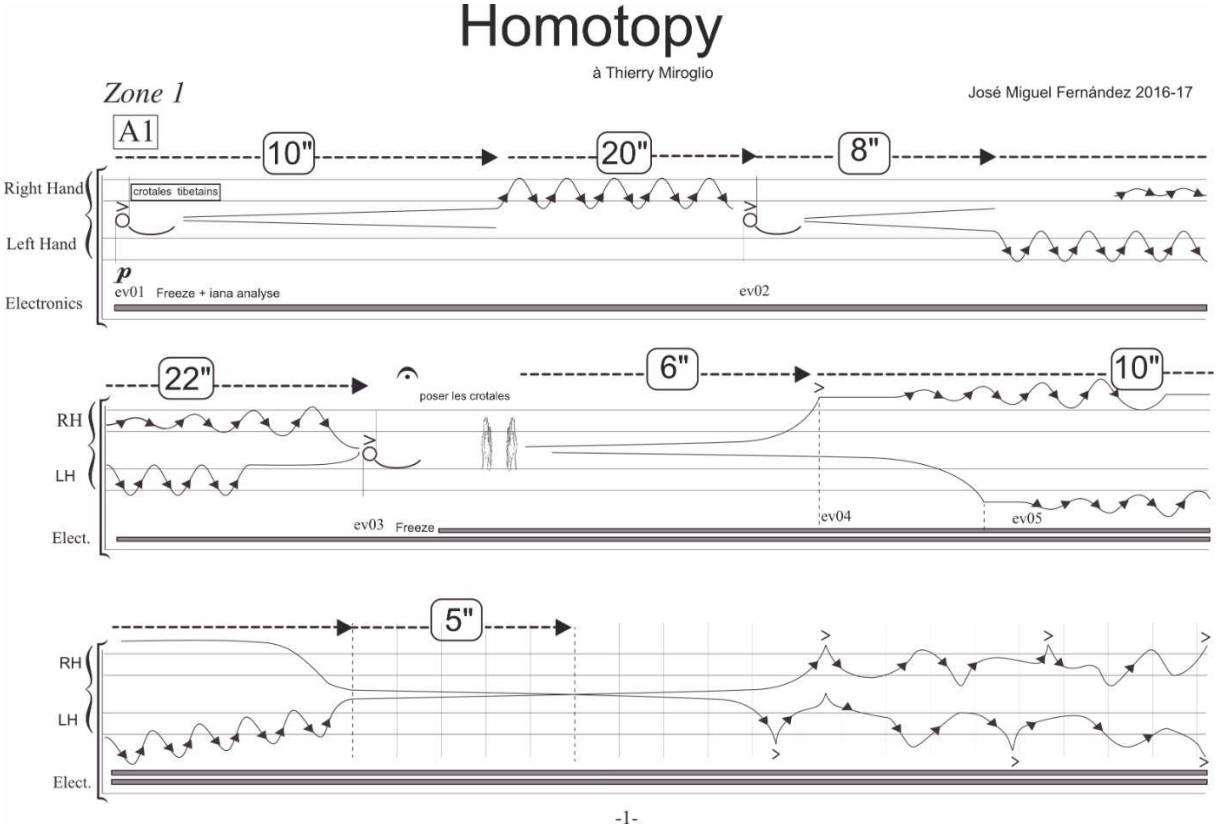
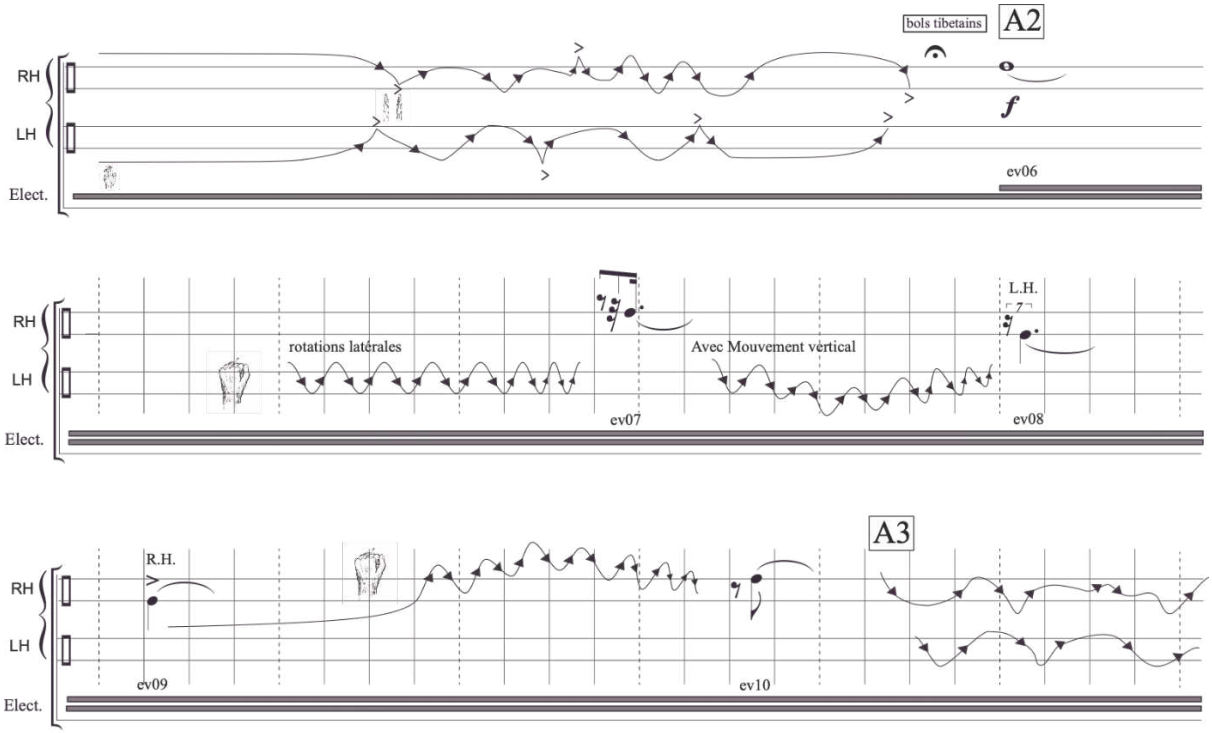


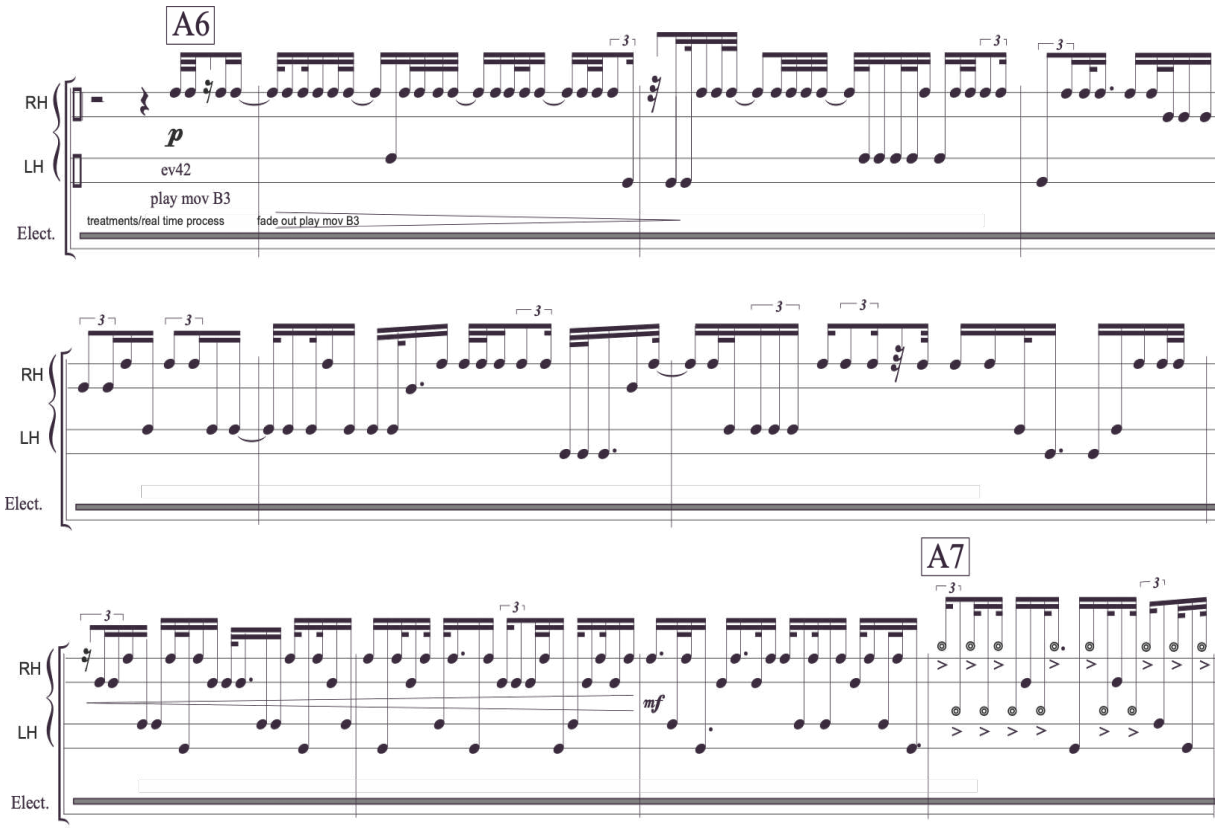
Fig. 9.6 Schéma des trois parties placées dans trois zones de la scène où le percussionniste se déplace pendant la performance.

Sur le plan gestuel, la pièce utilise des gestes instrumentaux et non instrumentaux comme la pantomime pour des jeux de percussions dans l'air et d'autres mouvements. L'écriture des gestes non instrumentaux reprend les descriptions de gestes développées pour des pièces du répertoire, notamment celles de Thierry de Mey dans *Light Music* et *Musique de table*.



-1-


bols tibetains
A2
RH
LH
Elect.
f
ev06
rotations latérales
Avec Mouvement vertical
L.H.
ev07
ev08
A3
R.H.
ev09
ev10



-4-

Fig. 9.7 Pages 1, 2 et 4 de *Homotopy* pour percussions, captation gestuelle et électronique en temps réel. Dans les pages 1 et 2, on peut voir l'écriture des gestes non instrumentaux avec des graphiques représentant les mouvements des mains droite et gauche, les *kicks* représentés par une attaque « > ». Dans la page 4, à la fin de la troisième portée, les notes avec un « ◎ » représentent un jeu dans l'air de sons de bols tibétains synthétiques réalisés en temps réel dans SuperCollider.

Sur le plan technique, la performance est réalisée avec deux capteurs RIoT sur des gants. Les capteurs sont programmés pour envoyer différents types de données directement en OSC vers l'ordinateur via un réseau Wifi.

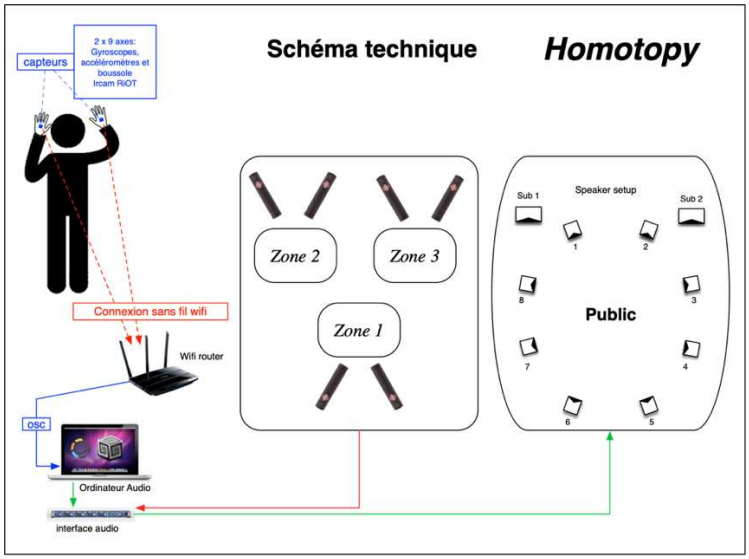


Fig. 9.8 Schéma avec les différents composants techniques de la pièce *Homotopy*.

Au niveau électroacoustique, la pièce est composée pour un système de diffusion octophonique. Elle intègre plusieurs systèmes réalisés dans la première version de la librairie AntesCollider avec un suivi de percussions réalisé grâce à une analyse d'*onset* (attaques des notes) et l'intégration des données de captation directement du capteur RIoT vers le langage Antescofo.

Suivi de percussions : comme déjà évoqué, Antescofo dans sa version actuelle n'est capable que de suivre des notes harmoniques, il n'est donc pas en mesure de suivre des sons plus complexes ou inharmoniques comme ceux des percussions. Pour implémenter un suivi de percussions, j'ai utilisé l'UGen *onsets* dans SuperCollider basée sur une analyse FFT pour détecter les attaques. Cette UGen envoie directement par OSC l'information de la détection des attaques au langage Antescofo qui va utiliser cette information pour avancer séquentiellement dans la partition électronique. Ce suivi, à la différence du suivi audio, ne comporte pas d'estimation de la position du jeu dans la partition, ce qui ne permet pas de rattraper automatiquement les fautes ou les mauvaises détections. On peut cependant se débrouiller pour sécuriser l'avancement dans la pièce. Dans la zone 2, par exemple, les coups dans l'air avec une inclinaison vers le haut vont déclencher des événements, mais servent aussi de points de resynchronisation pour le suivi de percussions : Antescofo attendra en effet ces gestes (*kick* avec inclinaison vers le haut) pour continuer à avancer, sans en oublier un seul.

```
233   oscrecv R_perc 87653 "/R_perc" $R_perc
234   oscrecv L_perc 87653 "/L_perc" $L_perc
235
236   oscrecv R_perc 87653 "/R_perc2" $R_perc2
237   oscrecv L_perc 87653 "/L_perc2" $L_perc2
238
239   oscrecv R_perc 87653 "/R_perc_dir" $R_perc_dir_plano, $R_perc_dir
240   oscrecv L_perc 87653 "/L_perc_dir" $L_perc_dir_plano, $L_perc_dir
241   oscrecv R_perc 87653 "/R_intensity" $R_intensity
242   oscrecv L_perc 87653 "/L_intensity" $L_intensity
243   oscrecv R_perc 87653 "/R_mov" $R_mov
244   oscrecv L_perc 87653 "/L_mov" $L_mov
245   oscrecv R_perc 87653 "/R_mov_dir" $R_mov_dir_plano, $R_mov_dir
246   oscrecv L_perc 87653 "/L_mov_dir" $L_mov_dir_plano, $L_mov_dir
247   oscrecv R_perc 87653 "/R_rota_trill" $R_rota_trill
248   oscrecv L_perc 87653 "/L_rota_trill" $L_rota_trill
249   oscrecv R_incli_front 87653 "/R_incli_front" $R_incli_front
250   oscrecv L_incli_front 87653 "/L_incli_front" $L_incli_front
251   oscrecv R_incli_lateral 87653 "/R_incli_lateral" $R_incli_lateral
252   oscrecv L_incli_lateral 87653 "/L_incli_lateral" $L_incli_lateral
```

Fig. 9.9 Réception des données de captation gestuelle provenant du capteur RIoT vers le langage Antescofo à travers la primitive `oscrecv`. Chaque donnée est assignée à une variable qui sera par la suite appelée pour réaliser le *mapping*.

```
2999  @obj_def sc_onset2($track, $inst, $del, $antirebond) // del in ms
3000  {
3001      @local $actif, $action, $onebang, $pause //, $nom, $inc
3002      @init
3003      {
3004          $actif := 1
3005          $onebang := 1
3006          $pause := 1
3007          crea_track8 $track 0 0.005 0 AudioInput input $inst #-> TBandPass #preset percu3 #-> FftOnsetReplay2 thresh 0.05 relaxtime 0.1 id $inst
3008      }
3009      @abort
3010      {
3011          crea_track8 $track off 0.01
3012      }
3013      @fun_def actif($y) {
3014            $actif := $y
3015      }
3016      @fun_def relaxtime($relaxt) {
3017          crea_track8 $track set 02_FFftOnsetReplay2 relaxtime $relaxt
3018      }
3019      @fun_def antirebond($anti) {
3020          $antirebond := $anti
3021      }
3022      @fun_def thresh($thresh) {
3023          crea_track8 $track set 02_FFftOnsetReplay2 thresh $thresh
3024      }
3025      @whenever ($pause && $actif&& ($onsetdetect == $inst))
3026      {
3027          if ($stop_detect){
3028              ($del)ms iana_anal bang
3029              $nextevent := 1
3030              $actif := 0
3031              $onsetdetect := -1
3032              ($antirebond)s $actif := 1 //antirebond
3033              print active onsetdetect $antirebond
3034          }
3035      }
3036  }
3037
3292  @proc_def ::onset_nextevent($on_off, $ev)
3293  @abort{
3294      print kill onset_nextevent $ev
3295  }
3296  {
3297      whenever($on_off && ($nextevent == 1))
3298      {
3299          if ($stop_detect){
3300              antescofo::nextlabel
3301          }
3302
3303      }
3304      print on onset_nextevent $ev
3305  }
```

Fig. 9.10 Code Antescofo de l'objet `sc_onset2` pour lancer une analyse de détection d'*onsets* dans SuperCollider. Cet objet a plusieurs méthodes qui permettent de contrôler les différents paramètres de l'analyse. Plus bas (à la ligne 3292), il y a un processus simple qui, à la réception de la variable globale $nextevent en provenance de l'objet d'analyse, déclenche une commande interne `antecofo::nextlabel` qui déclenchera l'événement suivant pour réaliser le suivi et le séquencement automatique.

Dans la première partie de la pièce, l'idée principale est d'analyser les sons des instruments de percussion et de réaliser ces analyses en temps réel à chaque fois qu'un instrument est joué. Les données d'analyse sont calculées avec l'objet Max développé à l'Ircam *iana~* [134] (pour l'analyse et l'extraction des composants perceptifs significatifs du son en temps réel). Elles alimentent une synthèse additive dans SuperCollider (scsynth de 20 x 3 partiels, chaque partiel contenant trois composantes qui peuvent dévier pour créer des battements).

```
6605  NOTE 60 1 prepare //prepare first onset
6606
6607  iana_fft-size 1024 //fft size for iana analisys
6608
6609  //analisis onsetdetection crotale tibetaine
6610  //                      sc_onset2($track, $inst, $del, $antirebond)
6611  $onsetdetect_01 := obj::sc_onset2("perc1", $percuZ1, 50, 0.5) //local
6612
6613  $onsetdetect_01.antirebond(15)
6614
6615  //activer le "nextlabel"
6616  $onset_nextevent := ::onset_nextevent(1, "init")
6617
6618
6619  NOTE 60 1 ev01
6620
6621  //                 iana_addsynth_one($track, $fade_in, $amp, $dev, FADER)
6622  //                                   ($track, $fade_in, $amp, $dev, $fad)
6623  $iana_ev01 := obj::iana_addsynth_one("iana_ev01", 4, 3, 1, "Add1") // desv era -3 error
6624  0.1 $iana_ev01.spat_circ(1, 0, 5) //dir, offset, tempo
6625
6626  7 ::line_track8_param_multi("iana_ev01", "00_TAddic_20_8", "d", [1, 4, 20]) //enchainement pour "d" de fixe au
6627  //                    map_gest_t3($name, $gest_in, $track, $module, $param, $min, $max)
6628  4 $iana_ev01_map1 := {@map_gest_t3($iana_ev01, $R_incli_lateral, "00_TAddic_20_8", "d", 6, 100)}
6629  $iana_ev01.add_macro($iana_ev01_map1)
6630
6631  0.1 $iana_ev01_map2 := {@map_gest_t3($iana_ev01, $R_incli_lateral, "00_TAddic_20_8", "shift", -20, 40)}
6632  $iana_ev01.add_macro($iana_ev01_map2)
```

Fig. 9.11 Code Antescofo correspondant aux deux premiers événements de la pièce. Le premier, `prepare`, servira à initier le suivi de percussion en instanciant l'objet `sc_onset2` et le processus `::onset_nextevent` pour faire avancer les événements automatiquement. Le second, ev01, lance une analyse de pics spectraux avec l'analyseur iana pour faire de la resynthèse par synthèse additive à partir des sons du crotale et des bols tibétains. La captation gestuelle grâce au *mapping* (lignes 6628 à 6631) modulera les paramètres « d » (déviation aléatoire des partiels) dans le premier et fera un *shift* (décalage) de fréquences dans le second.

[134] https://forum.ircam.fr/projects/detail/max-sound-box/

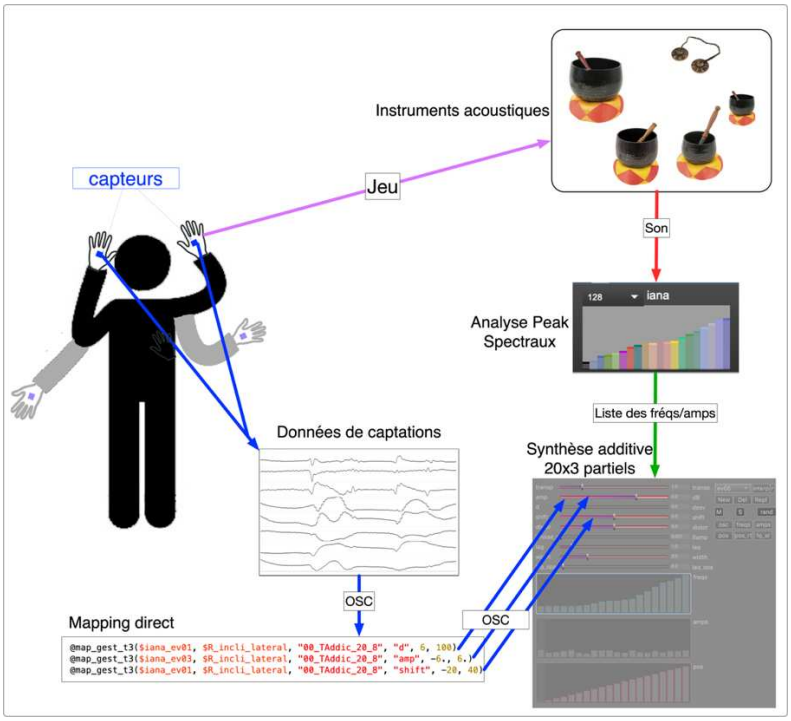
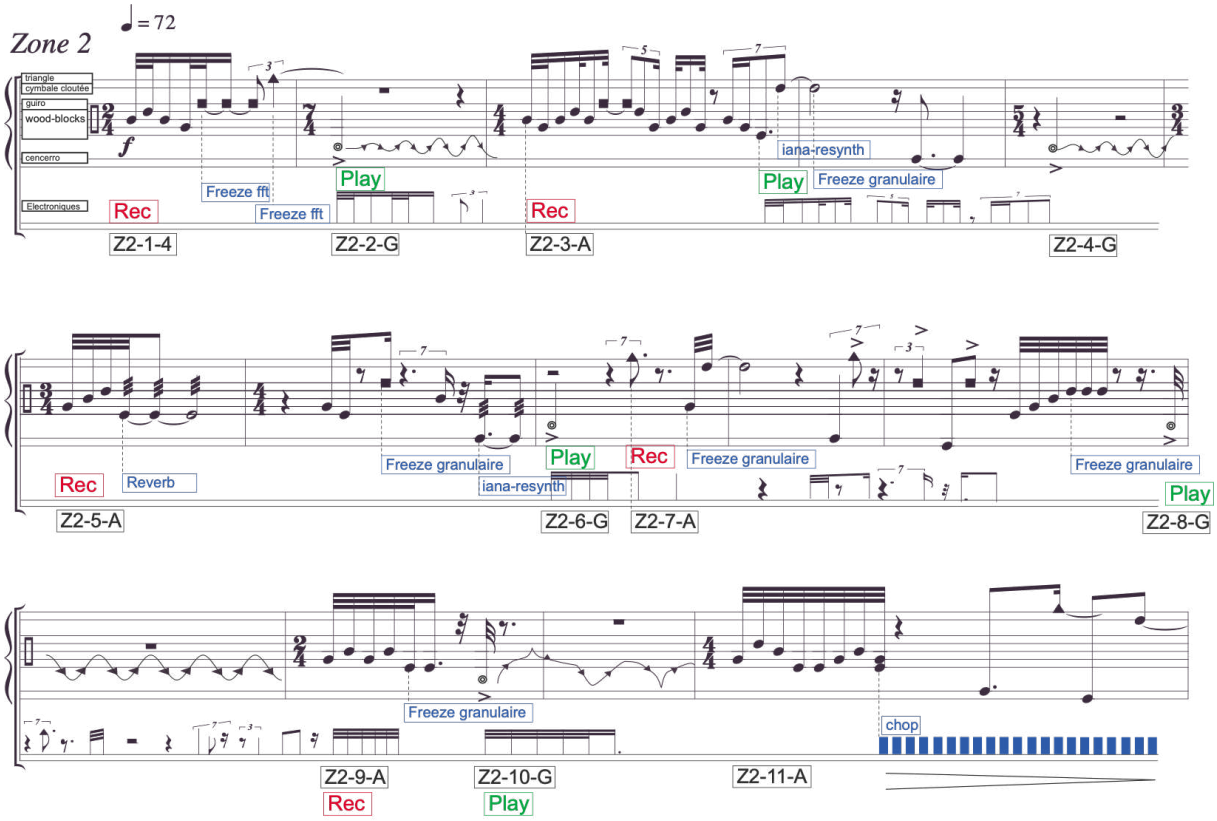


Fig. 9.12 Schéma d'interaction et *mapping* entre le jeu instrumental, l'analyse/resynthèse et la modulation de la resynthèse à partir de la captation gestuelle. L'idée est de moduler la synthèse issue de l'analyse de l'instrument avec les mouvements gestuels des mains. Le *mapping* est réalisé directement dans le langage Antescofo, ce qui permet une grande flexibilité et des contrôles dynamiques. Dans cet exemple, c'est `$R_incli_lateral` (la variable qui est assignée à l'inclinaison latérale de la main droite) qui va contrôler les paramètres « d », « amp » (amplitude) et « shift ». Toutes les connexions entre les différents éléments se font par le protocole OSC.

Dans la zone 2 de la pièce, les durées de chaque rythme sont enregistrées dans un tableau Antescofo à partir des informations fournies par le détecteur d'attaques *onset*. Ces durées enregistrées pendant l'exécution de la pièce sont utilisées plus tard pour créer une réponse de l'électronique avec des sons de percussions synthétiques.



-6-

Fig. 9.13 Page 6, début de la zone 2 de *Homotopy*. Grâce au système de suivi de percussion, on peut écrire l'électronique d'une façon assez précise. Les enregistrements des rythmes sont représentés par « Rec » (en rouge) et la lecture avec « Play » (en vert). Les autres traitements en temps réel tels que les Freezes fft, Freezes granulaires, iana-resynth, Reverb, etc. (en bleu) sont aussi modulés par la captation gestuelle. La note « ◎ » qui représente un *kick* joué dans l'air va déclencher des événements et va aussi permettre au système de suivi de percussions de se resynchroniser.

```
505   $map_rec_rythm := map{}
506   @proc_def ::rec_rythm_onset($name)
507   {
508       @local $start_rec, $last_val, $tab, $last_date, $data, $action, $date, $dummy
509       $start_rec := false
510
511       whenever ($onsetdetect==$onsetdetect)
512       {
513           if($start_rec)
514               {
515                   $dummy := @push_back($tab, $NOW - $last_date)
516                   $last_date := $NOW
517               }else
518               {
519                   $last_date := $NOW
520                   $tab := [0]
521                   $map_rec_rythm := @add_pair($map_rec_rythm, $name, $tab) // add NIM to dico
522                   $start_rec := true
523               }
524       }
525   }


549   @proc_def ::play_rythm_samp($name, $tpo, $amp)
550   {
551       @local $seq, $dur, $inc
552       $inc := 0
553       $seq := $map_rec_rythm($name)
554       $dur := $seq[]
555
556       group @tempo $tpo
557       {
558           loop $dur
559           {
560               playsample_solo_G @choose($samples_zone2) @rand_int(8) $amp @rand_range(0.8, 1.2) Samples
561               $inc := $inc+1
562               $dur := $seq[$inc]
563           } until ((@size($seq)-1)==$inc)
564       }
565   }
```

Fig. 9.14 Code Antescofo de deux processus, pour enregistrer des rythmes et les rejouer (plus tard) avec des sons de synthèse. Les rythmes sont insérés dans un tableau avec le processus ::rec_rythm_onset grâce à la déclaration *whenever* qui reçoit les détections d'*onset*. Chaque séquence de rythmes est à son tour enregistrée dans le dictionnaire $map_rec_ryhm (une *map* dans le langage Antescofo) pour l'utiliser par la suite. Le processus ::play_ryhm_samp va permettre de rejouer les séquences rythmiques stockées dans le dictionnaire à différents tempi.

```
7893  NOTE 60 1 Z2_4_G
7894
7895  0.1 group
7896
7897      1 $next_envent_Zone2 := ::onset_nextevent(1, "Z2_5")
7898      $onsetdetect_Zone2.antirebond(0.01)
7899      0.5 $rec_cel3 := ::rec_rythm_onset("cel_3")
7900  }
7901  NOTE 60 1 Z2_5A1
7902  print 1
7903  NOTE 60 1 Z2_5A2
7904  print 2
7905  NOTE 60 1 Z2_5A3
7906  print 3
7907  $onsetdetect_Zone2.antirebond(2)
7908  NOTE 60 1 Z2_5A4
7909  print 4
7910  group
7911  {
7912      crea_track8 Z2_5A4 0 1 0 AudioInput input $percuZ2 #-> TAdCVerb #preset estremo-rev-init #-> TPan8 pos 0.5 #group "Add1"
7913      ::line_track8_param_multi("Z2_5A4", "02_TPan8", "pos", [0.5, 3, 1])
7914      0.5 crea_track8 Z2_5A4 set 00_AudioInput amp -120
7915  }
7916  2 $onsetdetect_Zone2.antirebond(0.01)
7917  NOTE 60 1 Z2_5A5
7918  print 5
7919  NOTE 60 1 Z2_5A6
7920  print 6
7921  $onsetdetect_Zone2.antirebond(0.5)
7922  $Z2_6_G_Trig := ::L_perc_one_range_nextlabel("Z2_6_G", $min_incl_front, 1.0) //declanche l evenement avec kick main gauche
7923
7924  NOTE 60 1 Z2_5A7
7925  print 7
7926  crea_track8 gran_5_7 0 2 0 AudioInput input $percuZ2 #-> TGranFreeze8_8in rate 1 buf @buff_gran() trate 16.5 tratedev 106.43
7927  crea_track8 gran_3_16 off 10
7928  0.1 $onsetdetect_Zone2.antirebond(0.01)
7929  NOTE 60 1 Z2_5A8
7930  print 8
7931  $onsetdetect_Zone2.antirebond(2)
7932  NOTE 60 1 Z2_5A9
7933  print 9
7934  group iana_Z2_5A9
7935  {
7936      $iana_evZ2_5A9 := obj::iana_addsynth_one("iana_evZ2_5A9", 4, -3, 10, "Add1")
7937      $iana_evZ2_5A9.spat_circ(1, 0, 5) //dir, offset, tempo
7938      $iana_evZ2_5A9_map1 := {@map_gest_t3($iana_evZ2_5A9, $L_incli_lateral, "00_TAddic_20_8", "d", 6, 20)}
7939      $iana_evZ2_5A9.add_macro($iana_evZ2_5A9_map1)
7940      $iana_evZ2_5A9_map2 := {@map_gest_t3($iana_evZ2_5A9, $L_incli_lateral, "00_TAddic_20_8", "shift", -20, 20)}
7941      $iana_evZ2_5A9.add_macro($iana_evZ2_5A9_map2)
7942  }
7943
7944  NOTE 60 1 Z2_6_G
7945  ::play_rythm_samp("cel_3", 60, -3)
```

Fig. 9.15 Extrait de la partition électronique de *Homotopy*. Dans cet extrait, on peut apercevoir différentes actions de l'électronique comme l'enregistrement des séquences rythmiques en temps réel (ligne 7899) avec le processus `::rec_rythm_onset` («cel_3») et la lecture plus tard avec le processus `::play_ryhm_samp` («cel_3», 60, -3) (ligne 7945). Le déclenchement d'un événement avec un *kick* dans l'air avec le processus `::L_perc_one_range_nextlabel` (ligne 7922). L'activation d'un *freeze* granulaire (ligne 7922) et le traitement du son direct avec une réverbération (ligne 7912) à des moments précis de la partition. Une analyse/resynthèse modifiée par la captation gestuelle est définie par le groupe iana_Z2_5A9 (des lignes 7934 à 7941).

X. Expériences audiovisuelles

Le désir d'unir l'image et le son existe depuis longtemps. On peut citer par exemple les écrits de Kandinsky[135] où il donne à la musique une place fondamentale dans sa conception de la peinture, ou de Walter Ruttmann, pionner du « cinéma absolu », qui imagine un art qui se situe « *aux franges de la peinture et de la musique*[136] ». Ou encore le concept de synesthésie quand Baudelaire écrit dans son poème *Correspondances :* « *les parfums, les couleurs et les sons se répondent*[137] » ou quand Schönberg dit :

« *Ce qui est décisif, c'est qu'un processus spirituel, qui a sans aucun doute sa source dans l'action, est exprimé non seulement à travers les gestes, le mouvement et la musique, mais aussi à travers les couleurs et la lumière.* […] *Qu'à partir de valeurs de lumière et de tons de couleur particuliers, on peut pour ainsi dire construire des figures et des formes (Gestalten) semblables aux formes, aux figures et aux motifs de la musique.*[138] *»*

Citons aussi l'exemple de *Prométhée*, *Poème du feu* de Scriabine, pièce dans laquelle il avait prévu de faire appel à un clavier de couleurs pour interpréter la partie des lumières en utilisant des projections lumineuses à l'aide d'ampoules colorées. Cette pièce a par la suite inspirée plusieurs artistes. Michel Chion parle de *synchrèse* :

« *Soudure irrésistible et spontanée qui se produit entre un phénomène sonore et un phénomène visuel ponctuel lorsque ceux-ci tombent en même temps, cela indépendamment de toute logique rationnelle. […] La synchrèse permet aussi de jouer d'effets de contradiction et de décalage, effets qui, sans elle, conduiraient à une pure et simple désolidarisation de l'"audio" et du "visuel".* »

À partir du XX^e^ siècle, plusieurs expérimentations ont abordé ce rapport son et image et étudié la perception multimodale. Parfois appelé aussi « musiques visuelles », ces nouvelles formes mettent en scène des rapports entre formes abstraites et son, n'ont cessé de se développer comme les travaux de Norman McLaren, des frères John et James Whitney (qui emploient un système à pendules pour générer une bande son optique entièrement synthétique pour la « musique audiovisuelle »), de Nam June Paik ou de Stein et Woody Vasulka, pour ne citer que quelques-uns des exemples marquants.

L'usage simultané de plusieurs médias n'est pas un phénomène récent et plusieurs expériences d'union et synchronisation entre le son et l'image ont été faites au cours des XX^e^ et XXI^e^ siècles. Ce rapprochement tient à plusieurs facteurs, y compris

---

[135] W. Kandinsky, Du Spirituel dans l'art et dans la peinture en particulier.
[136] W. Ruttmann, *Peindre avec le temps*.
[137] C. Baudelaire, *Les fleurs du mal*.
[138] A. Schönberg, Conférence de Breslau sur « Die glückliche Hand » 1928.

techniques : les sons et les images animées sont intiment liés dès le début des développements techniques permettant leur enregistrement (les premières expériences avec des bandes sonores optiques sont antérieures aux bandes magnétiques) [COES07].

Ces développements technique et expérimentations continuent aujourd'hui avec des systèmes numériques et il existe une multitude de systèmes de génération d'images/sons en temps réel tels que Processing[139] ou openFrameworks[140] (ce dernier sera utilisé pour la plupart de mes expériences audiovisuelles) ou de programmation graphique *data flow* tels que Max (avec les librairies Nato, SoftVNS, Jitter, etc.), vvvv[141], Houdini[142] ou encore parmi les plus utilisés TouchDesigner[143].

Cependant, l'utilisation de la vidéo en temps réel dans la musique contemporaine n'est pas une pratique très courante, même s'il y a dans la programmation des festivals aujourd'hui quelques exceptions. Dans mon travail, l'intégration de la vidéo est venue assez naturellement, tout d'abord avec quelques expériences avec la librairie Nato dans Max et par la suite dans Jitter et SoftVNS.

Avec les projets GeKiPe et la pièce *Las Pintas* tous les deux audiovisuels, j'ai rencontré des artistes vidéo avec lesquels je partageais des problématiques communes autour du travail sur la forme et le développement technologique. Ces artistes vidéo utilisent aussi des librairies et logiciels ouverts où il faut programmer le système, ce qui nous permis, dans la lignée du code-partition présenté dans ce travail, d'accéder à une liberté technique et artistique qui n'aurait pas été accessible avec les systèmes classiques de montage fondés sur une *timeline* explicite.

L'utilisation de ces logiciels nous ramène à des problématiques communes, par exemple l'utilisation d'algorithmes pour la création et surtout la composition temporelle avec un système flexible et programmable. Pour cela, des langages temporels tels que Antescofo qui proviennent du monde musical peuvent aussi beaucoup aider, non seulement pour l'interaction entre les deux médias mais aussi pour la composition de la vidéo. Le dialogue entre les deux médias est un champ de recherche passionnant et avec beaucoup de possibilités de développement artistique. Aujourd'hui, à partir de différentes analyses de chaque média, on peut « sonifier » les données vidéo et « visualiser » les données audio en temps réel, ce qui induit de nouveaux rapports que nous avons expérimenté dans la pièce *Las Pintas*.

X.1. Projet *GeKiPe*

Le projet *GeKiPe* (Geste, Kinect, Percussions) est un projet de recherche-création soutenu par la Haute école spécialisée de Suisse occidentale et développé au sein de la Haute école de musique de Genève en partenariat avec l'association Flashback et la

[139] https://processing.org
[140] https://openframeworks.cc
[141] https://vvvv.org
[142] https://www.sidefx.com/products/houdini/
[143] https://derivative.ca

collaboration de l'Ircam pour les capteurs RIoT. L'équipe de travail est constituée de Philippe Spiesser comme interprète/performeur/percussionniste, Thomas Köppel artiste vidéo et programmeur du système vidéo, Alexander Vert compositeur et directeur de l'association Flashback de Perpignan et moi-même en tant que compositeur, concepteur et programmeur du dispositif de captation gestuelle et de la partie informatique musicale.

Les efforts de recherche se sont concentrés sur plusieurs domaines d'expertise essentiels à l'expression artistique et musicale : le contrôle des composantes du geste par l'interprète, la relation entre le geste et la synthèse sonore, le développement technologique de l'interface, l'intégration des données et le design visuel.

Construit comme une approche interdisciplinaire impliquant des instrumentistes professionnels, des compositeurs, des réalisateurs en informatique musicale et des artistes visuels, *GeKiPe* vise des applications musicales et audiovisuelles concrètes, avec une attention particulière aux qualités sonores, visuelles et gestuelles.

Le système *GeKiPe* permet à l'interprète de déclencher des sons et des images pendant ses performances, et de les contrôler par des gestes corporels capturés par les capteurs et des caméras Kinect. En utilisant leur corps entier comme un instrument de musique, les interprètes peuvent dessiner des trajectoires spatiales qui se traduisent par une composition visuelle et sonore sur une scène virtuelle. Différents compositeurs extérieurs au projet sont invités à écrire des compositions originales à l'aide de *GeKiPe*, validant la conception du système, mais influençant aussi son développement en fonction de leurs besoins. La recherche technologique, la recherche musicale et la création ont été étroitement liées tout au long du processus de collaboration.

Le projet *GeKiPe* est constitué pour l'instant de deux spectacles, *SCULPT* et *Crossing Points*.

Architecture et captation

Le système de captation gestuelle utilisé pour ce projet repose sur une caméra infrarouge Kinect (v.2) et des capteurs RIoT sur chaque main du performeur. La caméra Kinect va capter grâce à sa caméra infrarouge les positions absolues en 3D du squelette du performeur avec les positions des mains, du corps et son déplacement dans l'espace. La localisation peut être relative – par exemple, les mains sont localisées par rapport au corps du performeur – ou absolue – le corps du performeur sur scène. Les capteurs RIoT permettent d'acquérir la position relative des mains avec une précision suffisante pour caractériser des mouvements fins avec une latence d'environ 5 millisecondes. Cette faible latence est très importante pour une bonne interaction avec un percussionniste qui a l'habitude du retour instantané fourni par les instruments traditionnels. La Kinect va permettre de compléter ces informations avec les positions absolues des mains et du corps du performeur.

Les données gestuelles sont traitées par différents algorithmes pour contrôler en parallèle le son et la vidéo. Ces algorithmes peuvent être configurés dynamiquement ou personnalisés en fonction des projets compositionnels.

Comme modèle pour la performance, nous avons produit un ensemble de données gestuelles basées sur les mouvements exécutés par des percussionnistes sur différents instruments traditionnels. De plus, les gestes ont été enregistrés en utilisant différentes frappes de la main (rebond, bloqué, brossé, secoué).

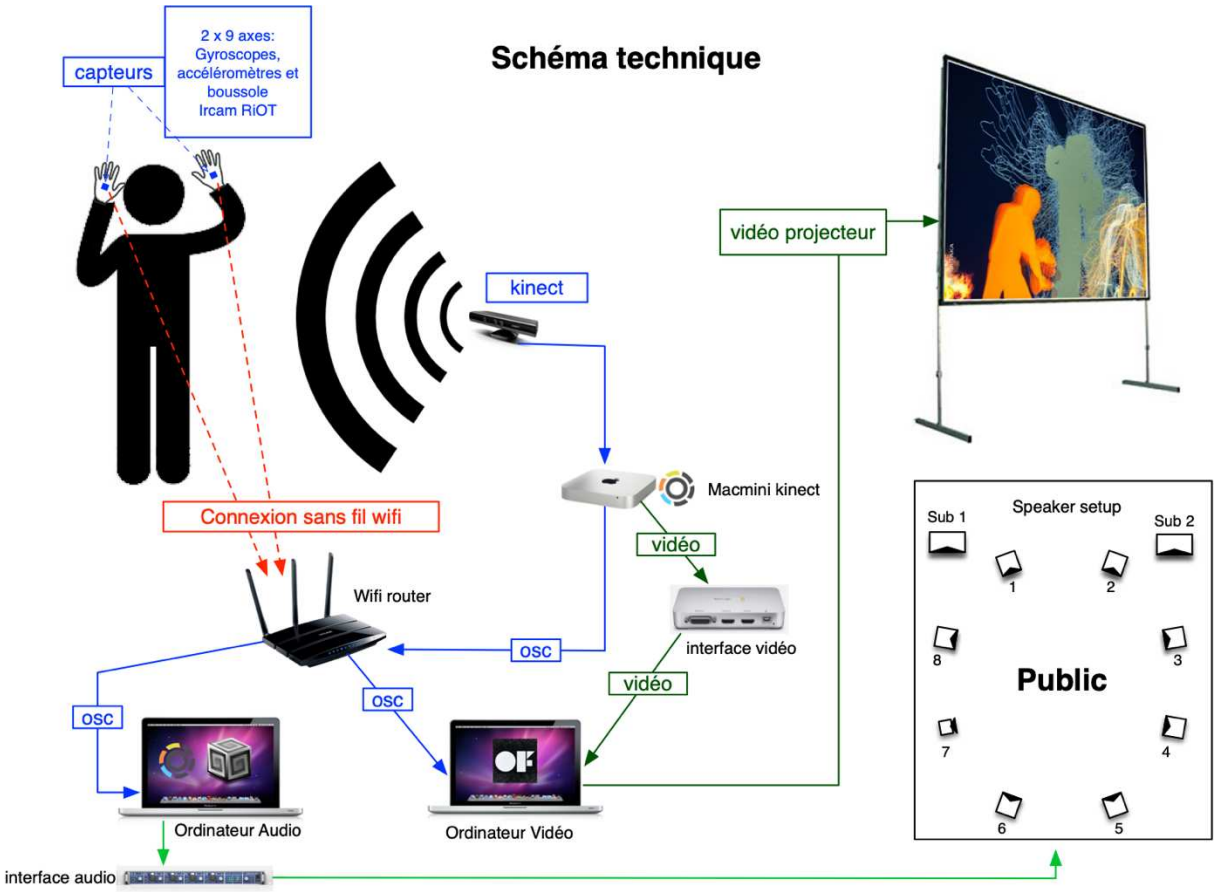


Fig. 10.1 Schéma technique de différents éléments du projet *GeKiPe* pour le spectacle multimédia *SCULPT*. Les données de captation gestuelle provenant de la Kinect (à travers un MacMini) et des capteurs RIoT sont partagées dans un réseau local entre l'ordinateur dédié à la synthèse et aux traitements audio et celui de génération de vidéo en temps réel. Le système de diffusion est une couronne octophonique horizontale.

Mapping dynamique

Le *mapping* est réalisé dans le langage Antescofo avec l'utilisation de la librairie AntesCollider. Différents types de générateurs sonores ont été créés (synthèse additive et soustractive, granulaire, concatenative, SuperVP, modulateurs, vocodeur de phase, motifs rythmiques) et mappés en temps réel depuis la Kinect et les capteurs sur la base des données gestuelles de l'interprète en contrôlant de manière continue les différents paramètres sonores tels que la hauteur, la modulation, la spatialisation, la dynamique, la vitesse des changements des paramètres, etc.

Le *mapping* dynamique se réfère à la possibilité dans le langage Antescofo de pouvoir changer dynamiquement et à la volée les *mapping* entre la captation gestuelle et les modules de synthèse et de traitement. Cela permet au performeur de jouer des éléments différents suivant le moment de la pièce et assure l'avancement dans l'exécution de l'œuvre. Chaque *mapping* peut aussi changer de comportement en cours d'exécution,

par exemple on peut moduler la plage de la sortie de chaque *mapping* pour changer dynamiquement le minimum ou le maximum du contrôle sur un paramètre de la synthèse.

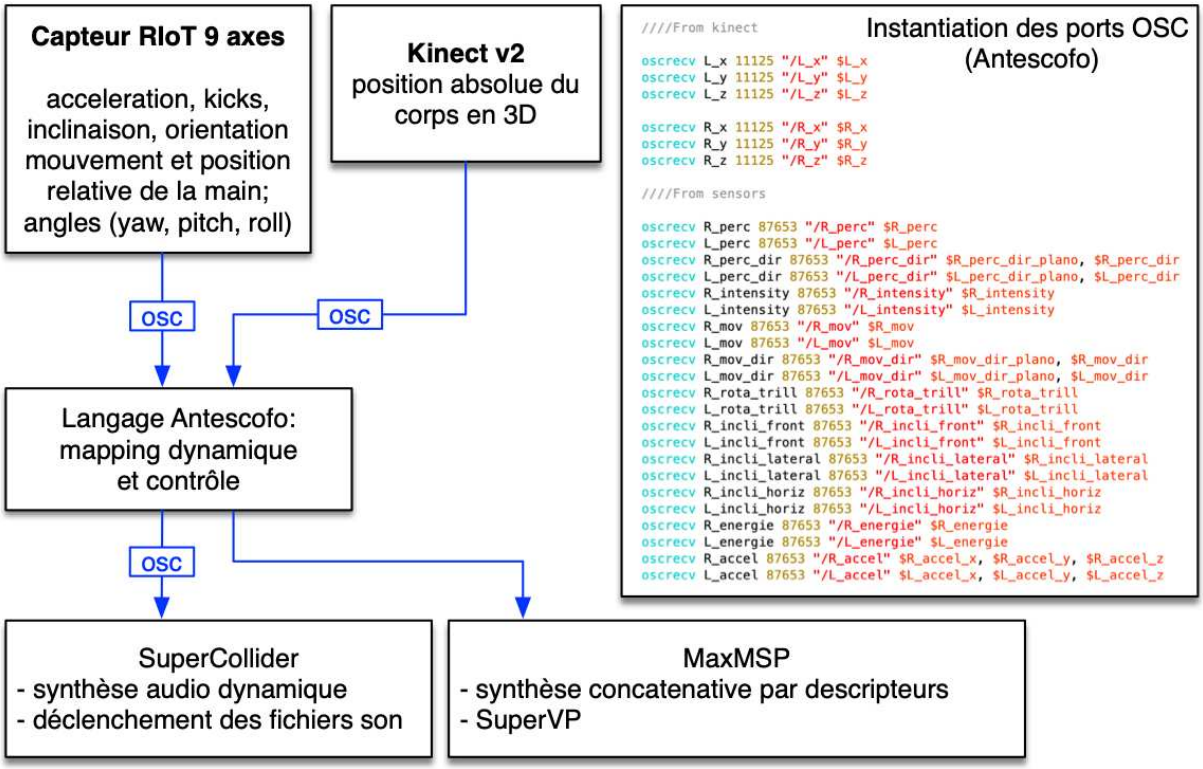


Fig. 10.2 Schéma représentant le *mapping* entre la captation gestuelle et la synthèse sonore. Les données de captation de la Kinect et les capteurs sont envoyées via OSC et réceptionnées dans le langage Antescofo à travers la commande oscrecv (à droite) qui crée dynamiquement les accès aux données OSC. Le langage Antescofo fera les différents *mapping* en fonction de la composition et d'une façon dynamique, comme pour les processus et objets (acteurs) du langage, il sera possible de les créer, les transformer et les détruire pendant l'exécution de la pièce.

Dans la suite de cette section, je présenterai les deux pièces du spectacle *SCULPT*, *Hypersphère* et *Le Silence* d'Alexander Vert pour présenter un autre type d'approche dans le suivi de geste, à partir des données de captation.

SCULPT est un spectacle multimédia d'une heure, mélangeant musique, vidéo et performance, comprenant deux performances : *Le Silence* d'Alexander Vert et *Hypersphère* de moi-même. La vidéo est réalisée par Thomas Köppel. Les compositeurs, l'artiste vidéo et le percussionniste ont travaillé ensemble pour produire le spectacle. Le public voit l'interprète principal, Philippe Spiesser, jouer des percussions invisibles en dessinant des mouvements dans l'espace de la scène, avec son corps, ses mains et ses jambes. Les sons et les images déclenchés par ces trajectoires peuvent être modulés tout au long du spectacle.

X.2. *Hypersphère*

La pièce a été spécifiquement composée pour un percussionniste bien que, ne faisant pas appel à des percussions réelles, la pièce pourrait en principe être interprétée par un

performeur musicien (pas nécessairement percussionniste). Écrite pour le percussionniste Philippe Spiesser, cette pièce est une nouvelle étape dans ma recherche qui vise à explorer les liens entre captation gestuelle, production (synthèse) du son et images de synthèse en temps réel.

Le titre de la pièce fait référence à l'hypersphère, un objet mathématique à n-dimensions utilisé entre autres pour calculer les rotations des capteurs et déduire leurs positions. Dans le discours musical, il s'agit de parcourir des espaces sphériques multidimensionnels dans lesquels différents sons et techniques de synthèse vont être déclenchés, révélés et modulés par la captation gestuelle. La synthèse d'images et les mouvements dans l'espace du percussionniste sont aussi en relation étroite avec ces sphères, leurs déformations et les sons spatialisés à travers un système multicanal octophonique.

C'est une pièce qu'on pourrait qualifier de générative avec une grande composante d'improvisation à partir d'instructions données au percussionniste. Même si l'intégralité du déroulement séquentiel de la pièce est écrite dans la partition électronique, l'interprète a pleine liberté pour utiliser des gestes à sa convenance et dilater ou accélérer la durée et la typologie des gestes et par conséquent de la pièce. La composition est entièrement écrite dans la librairie AntesCollider et tous les sons déclenchés sont synthétiques, générés et traités en temps réel. La pièce est structurée en 30 séquences. Il n'y a pas de partition instrumentale à proprement parler, mais des instructions de gestes à réaliser. Ces gestes sont directement issus d'un important travail effectué avec en collaboration avec l'instrumentiste pour trouver une gestuelle adaptée et expressive, apte à contrôler la musicalité des sons de synthèses. Voici quelques exemples de la liste de séquences :

- Ev01_Introduction : il s'agit de réaliser des mouvements lents avec les mains. Les mains sont posées en bas puis elles commencent à entrer dans la zone de synthèse en allant vers l'avant pour dessiner une sphère avec les mains devant.
- Ev02_Rec_init_R : faire des gestes plutôt répétitifs en boucle avec la main droite pendant quatre secondes.
- Ev03_Rec_init_L : faire des gestes plutôt répétitifs en boucle avec la main gauche pendant cinq secondes.
- Ev06_Freeze_court : commencer à faire des *kicks* (mouvements brusques percussifs dans l'air). Ces mouvements vont déclencher des *freezes* granulaires avec des rythmes périodiques rapides.
- Ev16_Synth_bajo2_distor_horz : distorsion du son grave avec inclinaison frontale des mains.

Toutes les interactions entre la captation gestuelle et la génération du son sont réalisées à l'aide de *mapping* dynamiques. Dans certaines parties de la pièce, il peut y avoir jusqu'à 15 *mapping* parallèles pour contrôler différents types de synthèses et

traitements. Par exemple, dans la première séquence, une synthèse additive avec cinq partiels (200 Hz, 333 Hz, 427 Hz, 616 Hz, 757 Hz) est générée par AntesCollider. Quelques règles de *mapping* ont été préétablies et permettent d'assurer une certaine cohérence pour le percussionniste :

- inclinaison de la main gauche captée par les capteurs RIoT : modulation de fréquence ;
- mouvement sur l'axe X par la main gauche, capté par la Kinect : spatialisation du son ;
- mouvement sur l'axe Z par la main gauche, capté par la Kinect : modulation d'amplitude ;
- mouvement sur l'axe Y par la main gauche, capté par la Kinect : transposition.

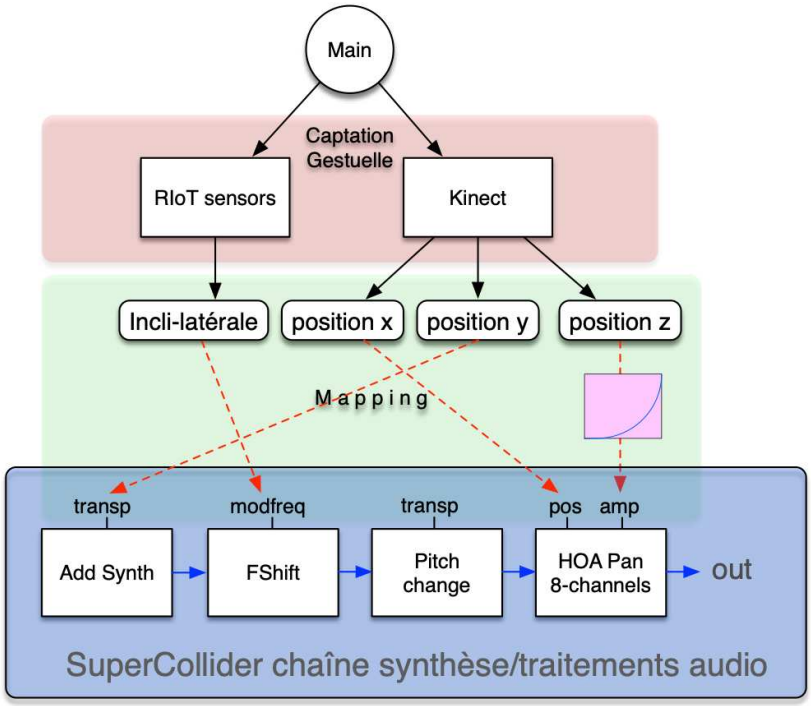


Fig. 10.3 Exemple de synthèse audio et du *mapping* déclaré dans le langage Antescofo. AntesCollider instancie une synthèse audio et crée les connexions et les *mapping* entre les capteurs/Kinect (capture de mouvements de la main gauche) et la synthèse à un moment donné de la partition. Les *mapping* peuvent aussi avoir des fonctions de transfert, par exemple avec « position z » pour contrôler les amplitudes.

```
3132  ///////// Group HOA
3133  obj::mix_group_HOA("group_hoa1", "localhost", "octo", 4)
3134
3135  ///////// MAIN GAUCHE
3136  // Synth 1
3137  obj::crea_track_HOA("synth1", "group_hoa1", amp = 0, fade_in = 10, doopler = 0,
3138      [["TAddic_5_8", "preset", "init1"], ["TMFxShift2", "preset", "init1"], ["TMFxPitch2", "preset", "init1"]])
3139  // Reverb Aux
3140  obj::crea_aux_HOA("aux_1", "group_hoa1", ["TGVerb_8", "preset", "init1"])
3141  // connect_aux synth1 aux_1 10
3142  $aux("aux_1").connect("synth1", fade_in = 10)
3143  // Synth 2
3144  obj::crea_track_HOA("synth1_z_in", "group_hoa1", amp = 0, fade_in = 10, doopler = 0,
3145      [["TAddic_5_8", "preset", "init_z_in"], ["TMFxShift2", "preset", "init_z_in"], ["TMFxPitch2", "preset", "init_z_in"]])
3146
3147  // mapping
3148  $tracks("synth1").map_gesture_index($L_incli_lateral, "TMFxShift2", 0, "modfreq", 1, 60)
3149  $tracks("synth1").map_gesture_ambi_pol_angle($L_x, 0.3, -180, 180)
3150  $tracks("synth1").map_gesture_func_amp($L_z, $quad_out, -40, 20)
3151  $tracks("synth1").map_gesture($L_y, "TAddic_5_8", "transp", 2, 0.1)
3152  $tracks("synth1_z_in").map_gesture_index($L_incli_lateral, "TMFxShift2", 0, "modfreq", -900, -1900)
3153  $tracks("synth1_z_in").map_gesture_ambi_pol_angle($L_x, 0.3, -160, 200)
3154  $tracks("synth1_z_in").map_gesture_func_amp($L_z, $exp_out_del, -40, 20)
3155  $tracks("synth1_z_in").map_gesture($L_y, "TAddic_5_8", "transp", 0.1, 2.)
3156
3157  ///////// MAIN DROITE
3158  // Synth 1
3159  obj::crea_track_HOA("synth2", "group_hoa1", amp = 0, fade_in = 10, doopler = 0,
3160      [["TAddic_5_8", "preset", "init2"], ["TMFxShift2", "preset", "init2"], ["TMFxPitch2", "preset", "init2"]])
3161  // connect_aux synth2 aux_1 10
3162  $aux("aux_1").connect("synth2", fade_in = 10)
3163  // Synth 2
3164  obj::crea_track_HOA("synth2_z_in", "group_hoa1", amp = 0, fade_in = 10, doopler = 0,
3165      [["TAddic_5_8", "preset", "init_z_in"], ["TMFxShift2", "preset", "init_z_in"], ["TMFxPitch2", "preset", "init_z_in"]])
3166
3167  // mapping
3168  $tracks("synth2").map_gesture_index($R_incli_lateral, "TMFxShift2", 0, "modfreq", 1, 60)
3169  $tracks("synth2").map_gesture_ambi_pol_angle($R_x, 0.3, -200, 160)
3170  $tracks("synth2").map_gesture_func_amp($R_z, $exp_out, -40, 20)
3171  $tracks("synth2").map_gesture($R_y, "TAddic_5_8", "transp", 2., 0.1)
3172  $tracks("synth2_z_in").map_gesture_index($R_incli_lateral, "TMFxShift2", 0, "modfreq", -2000, -1100)
3173  $tracks("synth2_z_in").map_gesture_ambi_pol_angle($R_x, 0.3, -160, 200)
3174  $tracks("synth2_z_in").map_gesture_func_amp($R_z, $exp_out, -40, 25)
3175  $tracks("synth2_z_in").map_gesture($R_y, "TAddic_5_8", "transp", 2., 0.1)
```

Fig. 10.4 Code de la première séquence, création du `mix_group_HOA`, des *tracks* avec les chaînes de synthèse et traitement, et les *mapping* depuis la Kinect et les capteurs vers la synthèse. Le *mapping* de la figure précédente est écrit entre les lignes 3148-3151. Dans cette séquence, il y a 8 *mapping* par main pour contrôler 4 chaînes de synthèses en parallèle.

Les paramètres audio mappés aux gestes peuvent donc changer au fur et à mesure de l'évolution de la performance de manière écrite (composée) ou improvisée, puisque ces *mapping* peuvent être créés et supprimés à la volée. Le langage Antescofo permet également d'enregistrer, à travers les NIM, une ou plusieurs séquences de mouvements (données gestuelles multidimensionnelles) dans un dictionnaire (MAP). Ces enregistrements des données de captation gestuelle sont utilisés par la suite pour générer de nouveaux matériaux et couches sonores qui se superposent au jeu en direct de l'interprète. Ces données enregistrées peuvent aussi être transformées (une NIM peut être transformée algorithmiquement de multiples façons) pour créer des *accelerandos*, des *rallentandos*, et générer différents types d'étirements temporels sur des données.

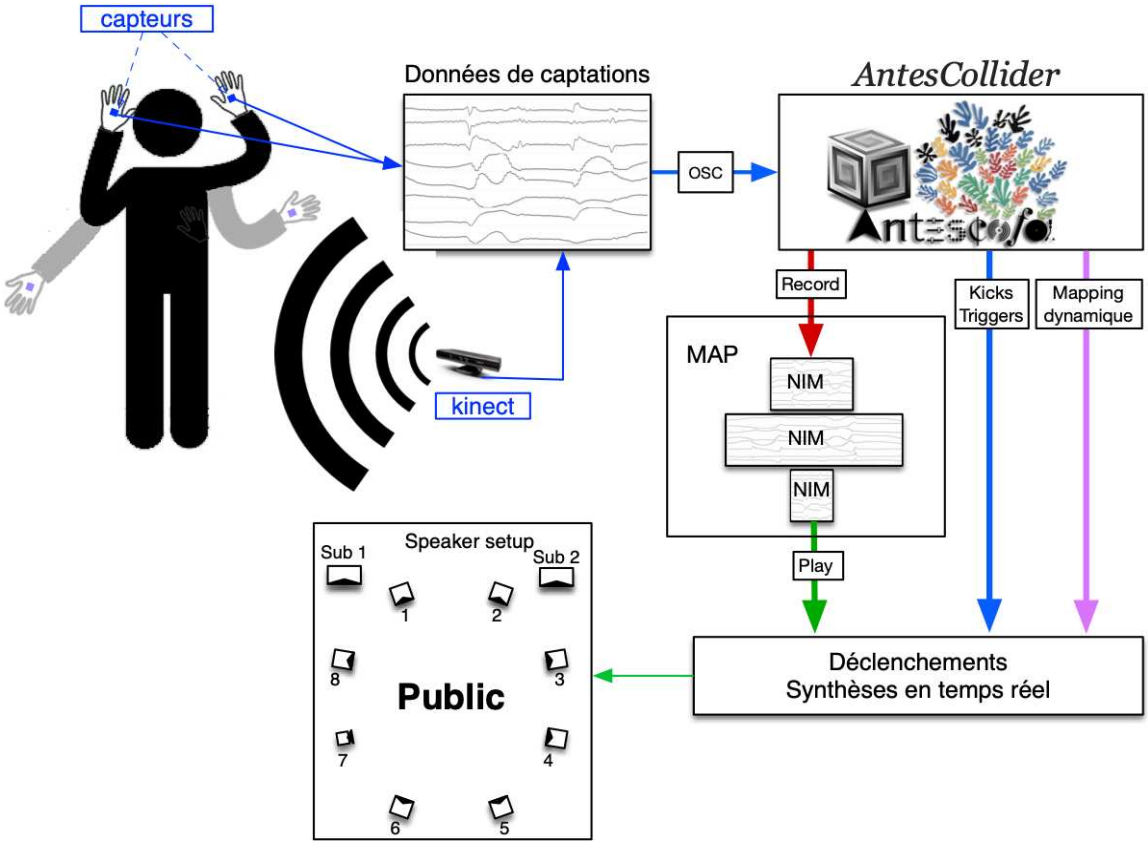


Fig. 10.5 Schéma représentant l'interaction dans *Hypersphère* et les traitements des données de captation gestuelle. Les données rentrent dans Antescofo qui coordonne les différentes applications des données pour générer l'électronique en temps réel. Les données seront enregistrées et mappées vers différents processus de synthèse et de traitements.

```
468  @proc_def ::rec_gest_R2_all($name, $dur)
469  @abort
470  {
471      $map_rec_gest_dur := @add_pair($map_rec_gest_dur, $name, $NOW - $last_date) // add dur to dico
472      print ($NOW - $last_date)
473      $map_rec_gest($name).push_back(@max_val($map_rec_gest($name)), 1, @min_val($map_rec_gest($name)))
474      abort $action
475  }
476  {
477      @local $start_rec, $last_val, $nim, $last_date, $data, $action, $accum, $date, $dummy
478      $accum := 0.0
479      $start_rec := false
480      $last_val := [0, 0, 0, 0, 0, 0, 0]
481      $data := $last_val
482      $last_date := $NOW
483
484      $action :=  {whenever ($R_incli_front==$R_incli_front)
485          {
486              if($start_rec)
487              {
488                  $data := [$R_x, $R_y, $R_z, $R_incli_front, $R_incli_lateral, $R_intensity]
489                  $nim.push_back($last_val, $NOW - $last_date, $data)
490                  $last_val := $data
491                  $last_date := $NOW
492              }else { // init
493                  $nim := NIM {0, $last_val 0 $data }
494                  $map_rec_gest.add_pair($name, $nim) // add NIM to dico
495                  $start_rec := true
496                  $last_val := [$R_x, $R_y, $R_z, $R_incli_front, $R_incli_lateral, $R_intensity]
497                  $last_date := $NOW
498               }
499
500          }
501      }
502      $dur abort $action
503      1 print final (@max_key($map_rec_gest($name))[0])
504  }


798  @proc_def ::play_gest_loop_R2_del($name_gest, $synth, $list_gest, $modules, $params, $scales, $nim_fonction, $tpo, $dur, $nloop)
799  {
800      @local $nim_rec, $min, $max, $inc, $idx
801
802      $inc := 0
803      $nim_rec := $map_rec_gest($name_gest)
804      $min := @min_key($nim_rec)[0]
805      $max := @max_key($nim_rec)[0]
806
807      loop $dur @tempo $tpo
808      {
809          group
810          {
811              curve slider @Grain := 0.01s, @Action := {
812                  forall $idx in @size($list_gest)
813                  {
814                      $tracks($synth).set($modules[$idx], [$params[$idx],
815                          @scale($map_func($nim_fonction[$idx])($nim_rec($x)[$gestes_index_R($list_gest[$idx])]),
816                              0., 1., $scales[$idx][0], $scales[$idx][1], 1)])
817                  }
818              }
819              { $x {      {($min)}
820                  ($dur)  {($dur)}}
821              }
822          }
823          $inc := $inc+1
824          print ("loop"+$name_gest) $inc
825
826          if ($inc >= $nloop){
827              crea_track8 $synth off 1
828          }
829      } until ($inc >= $nloop)
830      $max print fin ("loop "+$name_gest)
831  }
```

Fig. 10.6 Code dans le langage Antescofo des processus pour enregistrer les données de captation gestuelle dans une MAP. Le processus `::rec_gest_R2_all` prend en entrée un flux de données de captation provenant de la Kinect et capteurs et les enregistre temporellement dans une NIM ($nim, ligne 489). Une fois l'enregistrement terminé, la NIM est stockée dans un dictionnaire `$map_rec_gest` MAP (ligne 494) avec un nom (*key*) pour les répertorier. Le processus `play_gest_loop_R2_del` est une des possibilités de rejouer les données en changeant par exemple le tempo. Dans ce cas, le processus joue une nouvelle synthèse à partir des données enregistrées au préalable.

```
3164  NOTE 60 1 Ev02_Rec_init_R
3165
3166  group gest1_R
3167  {
3168      $gest1_R := ::rec_gest_R2_all("gest1_R", 4)
3169      4 abort $gest1_R
3170  }
3171  5 group play_synth
3172  {
3173      // Synthèse joué par les données de captation enregistrés auparavant
3174      obj::crea_track_HOA("synth_play1", "group_hoa1", amp = 0, fade_in = 10, doopler = 0,
3175          [["TAddic_5_8", "preset", "test3"], ["TMFxShift2", "preset", "test2"], ["TMFxPitch2", "preset", "test2"]])
3176      // connect_aux synth_play1 aux_1 5
3177      $aux("aux_1").connect("synth_play1", fade_in = 5)
3178
3179      // Mapping des données enregistrés vers la synthèse
3180      $play_synth1 := ::play_gest_loop_R2_del("gest1_R", "synth_play1",
3181          ["R_incli_lateral", "R_x", "R_z", "R_y"], ["01_TMFxShift2", "03_TPan8", "03_TPan8", "00_TAddic_5_8"],
3182          ["modfreq", "pos", "amp", "transp"], [[1, 60], [0., 2.], [-40, 20], [2., 0.1]], [
3183          "lin", "lin", "exp_out", "lin"], 60, 4, 6)
3184
3185      // Synthèse joué par les données de captation enregistrés auparavant
3186      obj::crea_track_HOA("synth_play1_z_in", "group_hoa1", amp = 0, fade_in = 10, doopler = 0,
3187          [["TAddic_5_8", "preset", "init_z_in"], ["TMFxShift2", "preset", "init_z_in"], ["TMFxPitch2", "preset", "init_z_in"]])
3188      // connect_aux synth_play1_z_in aux_1 5
3189      $aux("aux_1").connect("synth_play1_z_in", fade_in = 5)
3190
3191      // Mapping des données enregistrés vers la synthèse
3192      $play_synth1_z := ::play_gest_loop_R2_del("gest1_R", "synth_play1_z_in",
3193          ["R_incli_lateral", "R_x", "R_z"], ["01_TMFxShift2", "03_TPan8", "03_TPan8"],
3194          ["modfreq", "pos", "amp"], [[-2000, -1100], [0., 2.], [-50, 0]], ["lin", "lin", "exp_out_del"], 60, 4, 6)
3195  }
```

Fig. 10.7 Événement `Ev02_Rec_init_R` dans la partition électronique. Le processus `rec_gest_R2_all` enregistre une NIM nommée « gest1_R » pendant 4 temps dans le dictionnaire `$map_rec_gest`. Cinq temps plus tard (ligne 3171), une nouvelle synthèse est instanciée (ligne 3174) qui sera tout de suite contrôlée par le processus `play_gest_loop_R2_del` (ligne 3180) où sont définis les différents *mapping* et paramètres de module. Une deuxième synthèse est instanciée en même temps (ligne 3186) qui sera contrôlée par le même enregistrement des données « gest1_R ».

X.3. *Le Silence*

La deuxième composition du spectacle *SCULPT* est *Le Silence,* une composition d'Alexander Vert. Cette composition présente d'autres enjeux sur l'écriture des gestes et le système de suivi de gestes et elle a été réalisée d'une façon très différente d'*Hypersphère*. La pièce est constituée d'environ 400 échantillons audio qui sont déclenchés par un système de suivi de gestes réalisé à partir des données de la captation gestuelle de la Kinect et des RIoTs. Le percussionniste Philippe Spiesser a mémorisé, comme pour une partition traditionnelle, tous les gestes de la pièce, qui doivent être précis pour permettre le suivi gestuel implémenté dans le langage Antescofo.

Système de suivi de gestes

Pour réaliser un suivi de gestes qui puisse déclencher les 400 échantillons d'une façon séquentielle, nous avons utilisé deux types de données, les *kicks* des capteurs RIoT et un quadrillage de l'espace relatif au mouvement de l'interprète sur la scène avec la Kinect.

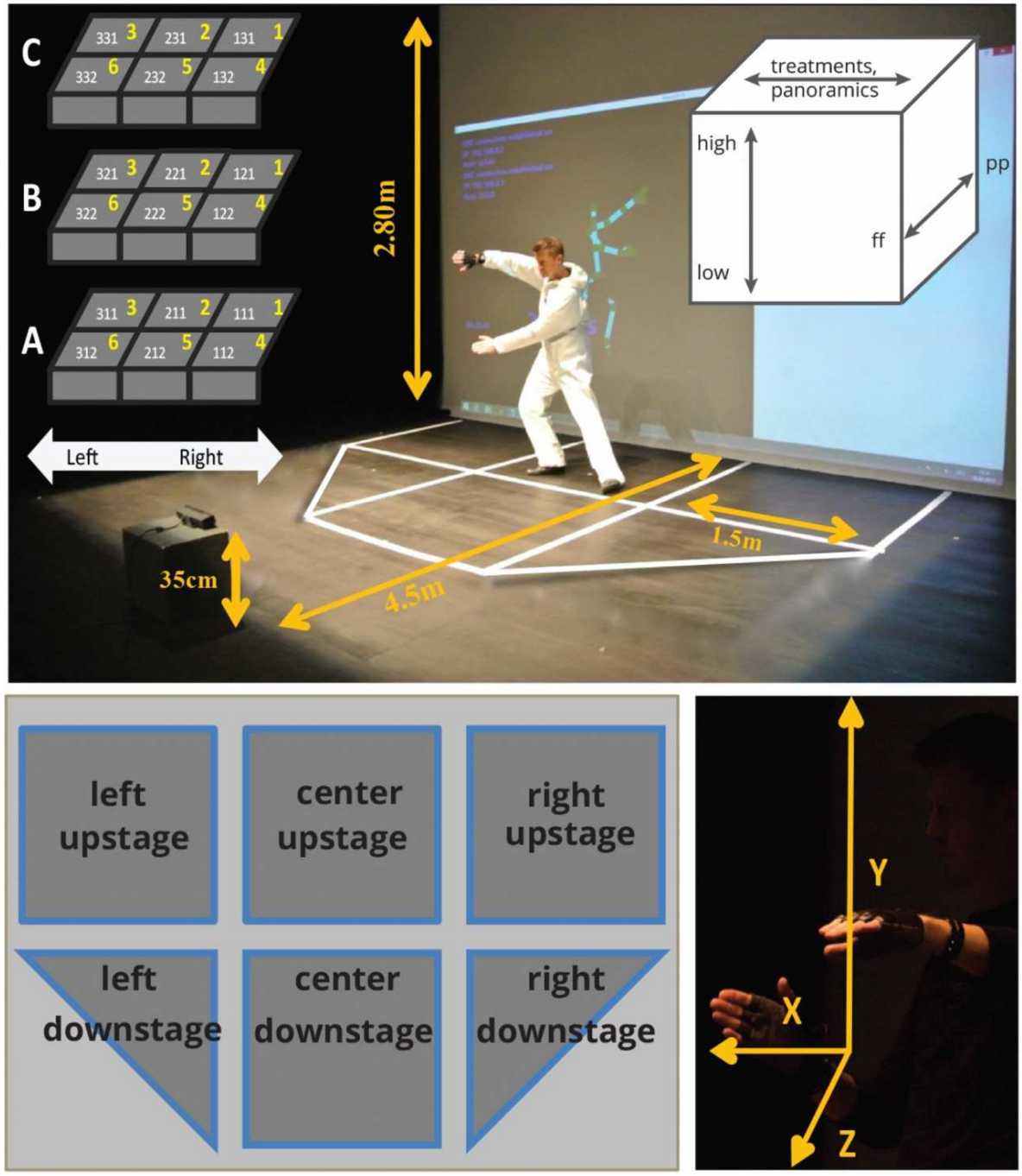


Fig. 10.8 Représentation du quadrillage virtuel de l'espace en 3D pour définir des zones grâce à la caméra Kinect. Chaque main peut se déplacer dans un cube virtuel de 18 cases (en haut à gauche) séparées en trois hauteurs (A, B et C), chacune divisée en 6 cubes. Ce quadrillage est calculé entre les positions relatives de la nuque et celles des deux mains inférées du squelette. De cette façon, le cube de 18 cases suit toujours l'interprète dans la scène. L'espace scénique aussi est quadrillé pour repérer la position du performeur sur scène.

Ce quadrillage en 18 cubes avec leurs coordonnées en combinaison avec les *kicks* pour chaque main va définir les positions et les actions spécifiques (comme des notes), ce qui va permettre de les écrire dans une séquence d'événements pour construire la composition. Celle-ci sera réalisée à partir de gestes chorégraphiques dans l'espace.

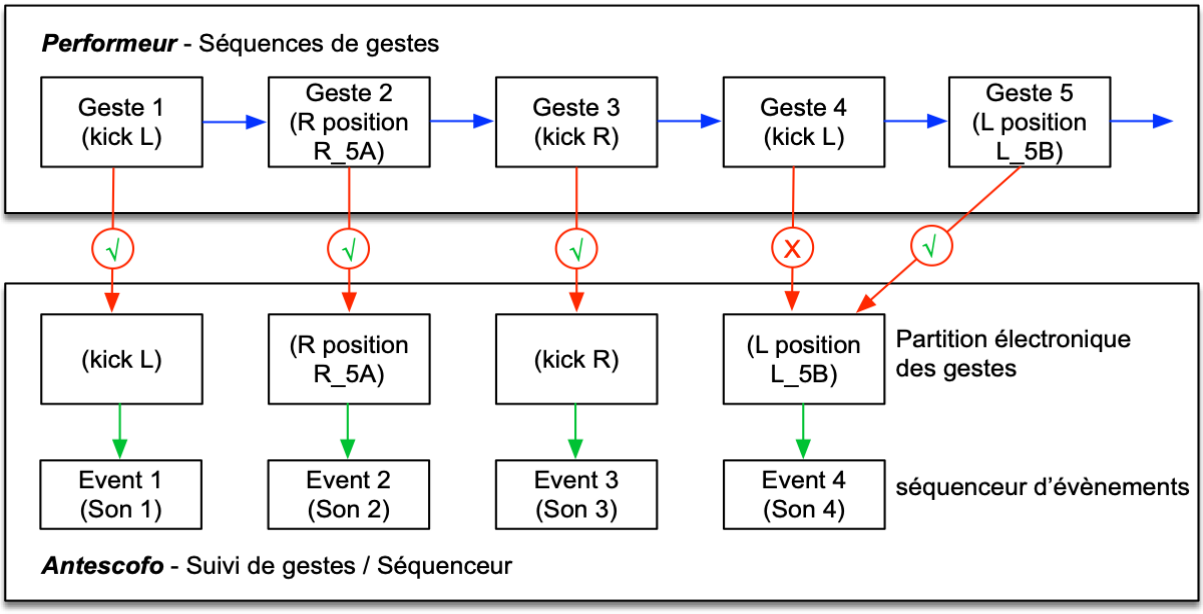


Fig. 10.9 Schéma du fonctionnement du suivi de gestes dans *Le Silence*. L'interprète joue des gestes qui sont écrits aussi bien dans la partition musicale que dans sa réalisation électronique. Le suivi de gestes programmé dans le langage Antescofo reçoit les positions et *kicks* provenant de la captation gestuelle et les compare avec le geste attendu (dans la partition électronique). Si le geste réalisé par le performeur correspond à celui de la partition électronique, le système déclenche l'événement correspondant. Si le geste ne correspond pas, par exemple le « Geste 4 » du performeur, le système ne fait rien et attend le bon geste, dans cet exemple le « Geste 5 ».

Écriture de la partition pour l'interprète

Pour l'écriture de la partition, nous avons élaboré un système d'écriture à base d'un lexique de gestes musicaux en référence aux mouvements des mains, aux *kicks* et aux 18 cubes repérés par la caméra Kinect. Les gestes sont transcrits dans une notation traditionnelle afin qu'elle puisse être lue et comprise facilement par d'autres interprètes. Cette notation introduit deux portées pour les gestes effectués par la main droite et par la main gauche. Chaque portée compte trois lignes correspondant aux trois zones dans l'axe vertical. Le type de geste (main ouverte ou fermée, direction, intensité du mouvement, etc.) est noté à l'aide de symboles dédiés.

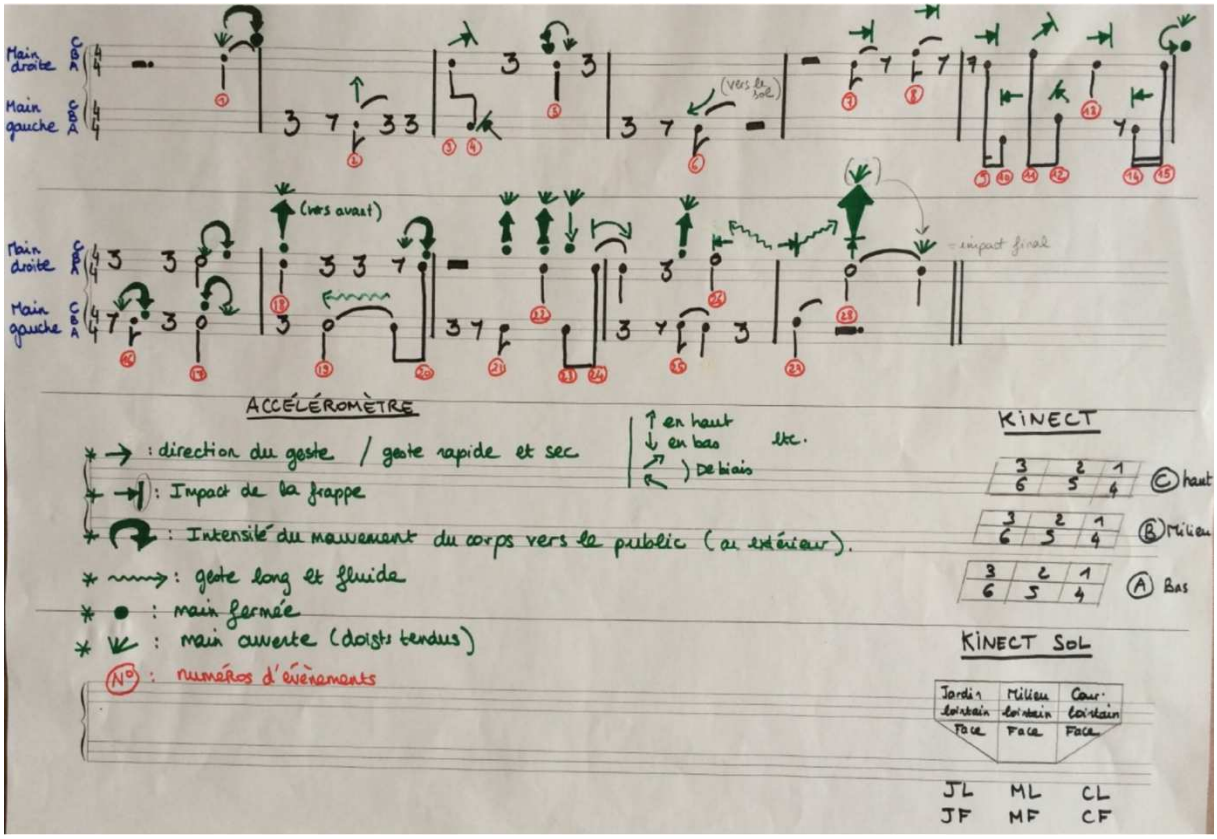


Fig. 10.10 Exemple d'une partition basée sur le quadrillage réalisé avec la Kinect et les *kicks* des capteurs RIoT. Chaque portée de la partition est séparée en trois lignes correspondant à une main et sa position dans l'espace.

Mapping de la partie vidéo

Le moteur de rendu visuel a été programmé par Thomas Köppel avec la librairie C++ open source openFrameworks. Une partie du moteur est une simulation de fluide réalisée en calculant un champ vectoriel représentant les forces en action en utilisant l'analyse du mouvement sur de multiples images capturées (technique de flux optique). Le comportement de ce champ vectoriel a été programmé en fonction de différents ensembles de paramètres tels que la densité, la gravité ou les perturbations.

Les images vidéo capturées peuvent être utilisées comme des entrées visuelles brutes et peuvent être modifiées par différents effets visuels (par exemple, flou de mouvement, flou temporel, retard). En même temps, ces images sont vectorisées (en utilisant la bibliothèque OpenCV[144]) et analysées pour la reconnaissance des contours dans l'image (par exemple pour extraire le corps de l'interprète). Ces données d'images vectorielles peuvent ensuite être utilisées pour le traitement algorithmique (bruit de Perlin) de la déformation de l'image initiale.

Comme pour les données de captation gestuelle décrites précédemment, les données vectorielles d'image et de mouvement peuvent être enregistrées dans des *buffers* (tampons) d'images pour les rejouer plus tard dans la performance. Trois têtes de

[144] https://opencv.org

lecture peuvent accéder aux *buffers* préenregistrés et produire soit une sortie visuelle directe, soit une sortie de données pour le *mapping* des paramètres. Une matrice de routage de données dynamique permet de faire correspondre n'importe quelle donnée entrante (données des capteurs, analyse sonore, données des *buffers* vidéo) à n'importe quel paramètre de rendu d'image (par exemple comportement des fluides, effets visuels, traitements algorithmiques). Les paramètres et les données du *mapping* sont stockés dans des préréglages qui sont utilisés pendant la performance en relation avec le suivi de gestes. Tous les paramètres sont aussi modifiables en temps réel, ce qui permet de jouer et d'improviser avec les visuels en temps réel pendant l'exécution de la pièce.

X.4. *Crossing Points*, pour percussions, captation gestuelle, électronique et vidéo en temps réel

Crossing Points est le deuxième spectacle du projet GeKiPe. Il est composé de deux pièces, *Mad Max* de Pierre Jodlowski et *Crossing Points*, une composition collaborative d'Alexander Vert, Thomas Köppel et José Miguel Fernández. La partie instrumentale de *Crossing Points* est constituée d'un set de cinq toms plus un tom grave (pour l'introduction de la pièce). La partie instrumentale a été écrite à deux mains, les deux compositeurs du projet ont composé des sections séparément qui ont ensuite été juxtaposées pour donner la forme générale de la pièce.

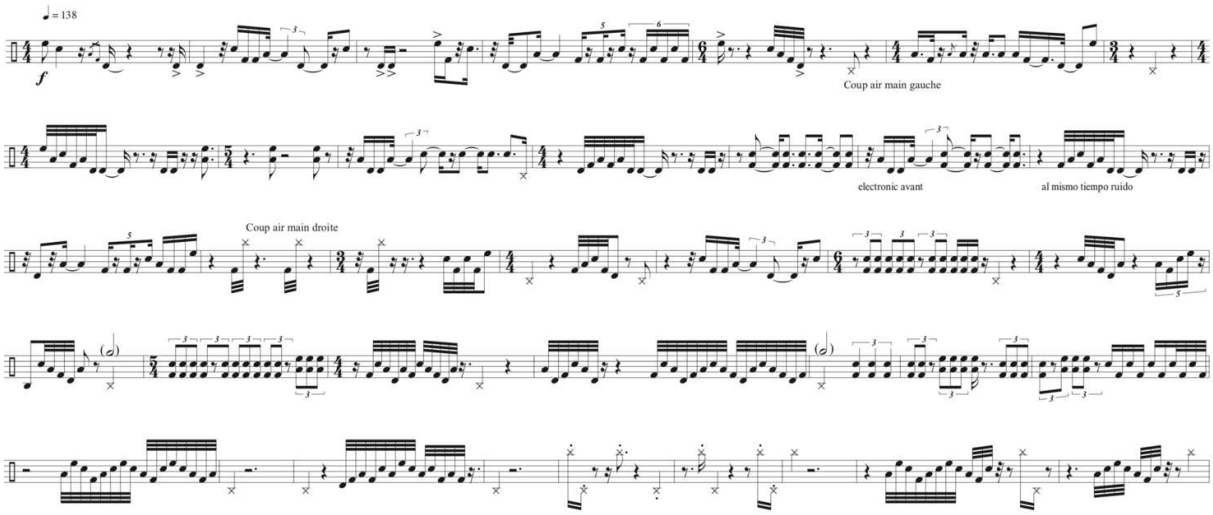


Fig. 10.11 Extrait de la partition de *Crossing Points*. Cette partie implique à la fois un jeu instrumental et des gestes mimétiques « dans l'air » (notés avec le symbole x) pour déclencher des sons de synthèse.

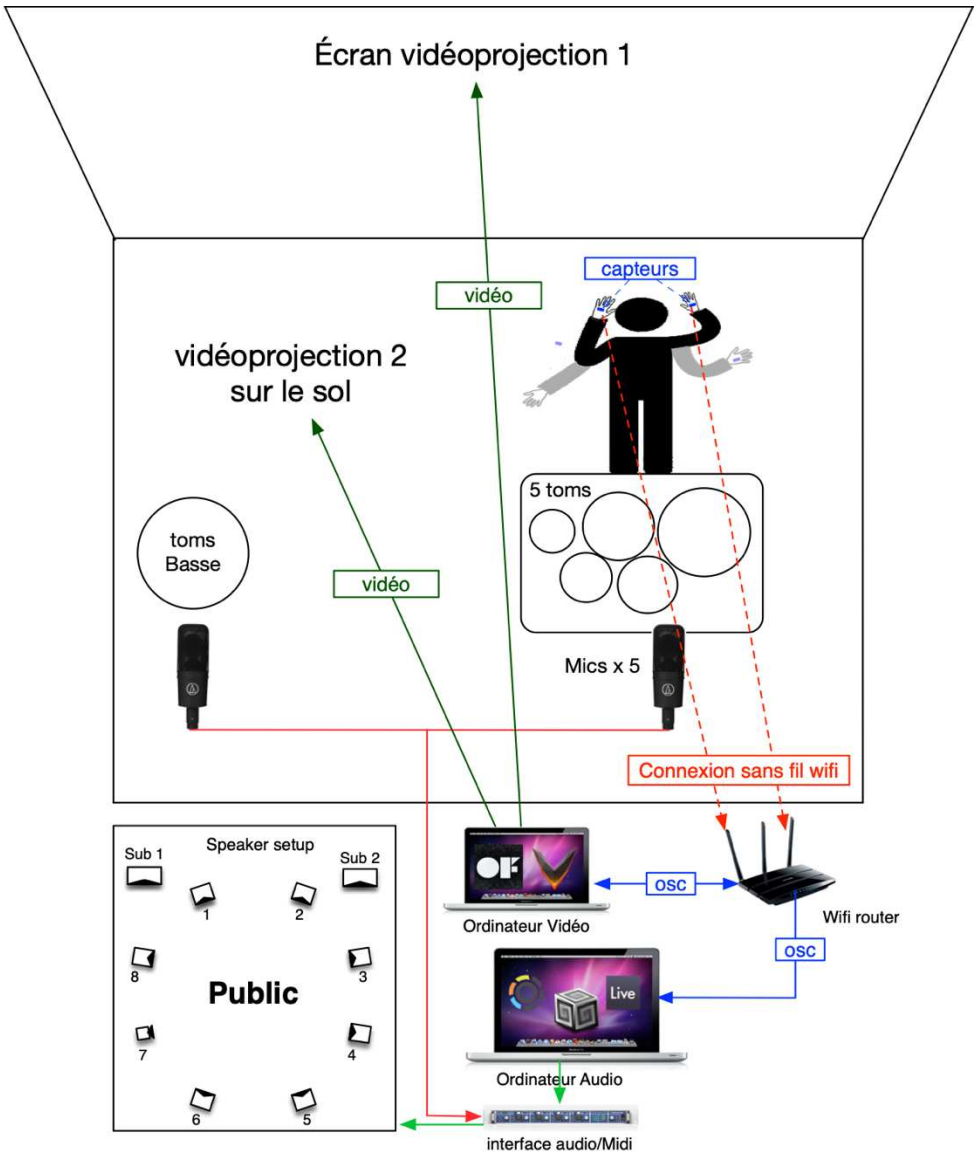


Fig. 10.12 Schémas des différents éléments de la pièce *Crossing Points* pour 6 toms, captation gestuelle, électronique et vidéo en temps réel. La partie audio est constituée de la librairie AntesCollider pour la synthèse et le traitement, de Max pour la synthèse concatenative par descripteurs et de Live pour la synchronisation entre les fichiers audio et la vidéo dans un système de diffusion octophonique. La partie vidéo est basée sur un système de particules programmé dans openFrameworks et contrôlé à partir du séquenceur Vezér. La vidéo est générée en temps réel et projetée sur deux écrans, l'un en fond de scène et l'autre sur le sol pour créer un « L ».

Synthèse concatenative

La synthèse concatenative par descripteur audio est un des différents traitements et synthèses contrôlés par la captation gestuelle utilisés dans la pièce. Des fichiers audio avec différents timbres et morphologies vont être analysés par des descripteurs audio grâce à la bibliothèque Mubu[145] dans Max. Les fichiers sont analysés et segmentés soit

[145] https://forum.ircam.fr/projects/detail/mubu/

de manière régulière (*chop*), soit par détection d'attaque (*onset*). Les gestes vont être mappés vers les valeurs de différents descripteurs. Par exemple, nous avons utilisé le *mapping* de l'intensité d'un geste vers la *loudness* dans une requête à une base de données, l'inclinaison de la main vers haut pour le *f0* ou le *centroïde*[146] ou encore une combinaison de plusieurs *mapping*. Cette technique permet de mapper facilement des gestes vers les caractéristiques perceptives des sons, ce qui rend identifiable l'association d'une sonorité au type de geste utilisé.

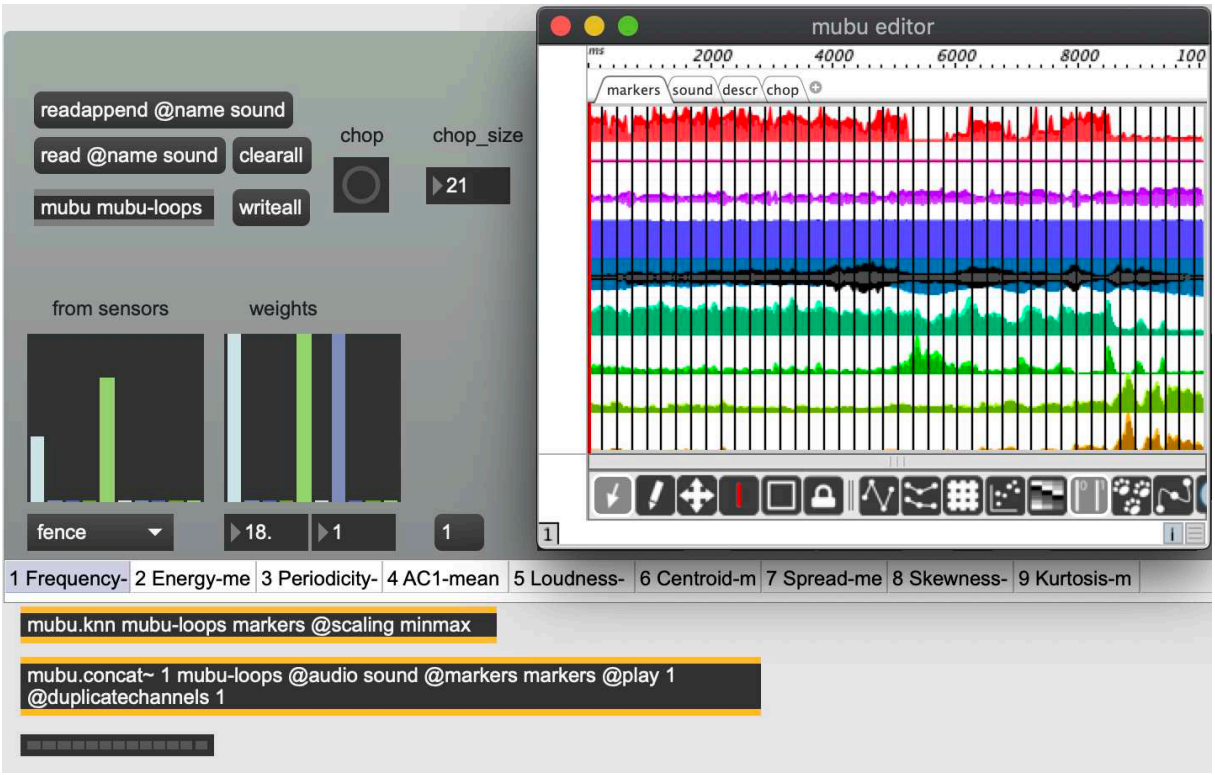


Fig. 10.13 Extrait d'un patch Max pour faire de la synthèse concatenative par descripteur. Dans cette capture, on peut voir le son analysé par 9 descripteurs audio différents et segmenté régulièrement (*chop*) par segments de 21 ms. Dans les paramètres de recherche d'échantillons les plus semblables par l'algorithme de KNN, on va pouvoir donner des poids à chaque descripteur. Dans cet exemple, les descripteurs utilisés sont *Frequency*, *Loudness-mean* et *Spread-mean*. Dans la partie en entrée « from sensors », on va pouvoir mapper différentes données de captation vers les descripteurs sélectionnés avec différentes valeurs pour chercher différents segments de fichiers.

La partie vidéo programmée dans la librairie C++ openFrameworks repose sur un système qui peut être contrôlé en temps réel par les données de captation, d'analyses audio ou par un contrôleur physique. Le système fonctionne en *shaders,* ce qui permet une grande optimisation. En effet, tous les calculs sont alors réalisés par la carte graphique (*Graphics Processing Unit*, GPU). Le système permet de créer des milliers de particules affichées par deux vidéoprojecteurs.

[146] Centre de masse spectrale, caractérise perceptivement par la brillance d'un son.

Fig. 10.14 Quatre photos pendant la performance de *Crossing Points*. On peut apercevoir le système particulaire avec des milliers des particules qui circulent dans les deux projections sur le fond et le sol de la scène.

Ateliers pédagogiques

Parallèlement aux processus de recherche et de création, *GeKiPe* permet une approche pédagogique de la performance musicale multimédia en s'appuyant sur des aspects interdisciplinaires musicaux, visuels, chorégraphiques et technologiques. Les ateliers *GeKiPe* ont permis aux étudiants de mettre en relation le mouvement du corps avec des événements sonores et musicaux programmables, dans un environnement visuel également évolutif. Sur le plan éducatif, *GeKiPe* permet une initiation aux arts du spectacle à travers plusieurs pratiques artistiques : musique (enregistrement et montage

sonore), danse (chorégraphie improvisée ou guidée), art visuel (en relation avec le son ou le mouvement).

L'écoute directe, encouragée par la forme du dispositif, déclenche spontanément chez les élèves un processus gestuel. L'aspect expérimental et les similitudes des approches utilisées dans le design du son, de l'image et du mouvement permettent d'établir des ponts entre différentes formes d'art (danse, arts plastiques, musique, vidéo). En utilisant des scénarios interactifs amenant les étudiants à contrôler des paramètres spécifiques, *GeKiPe* a prouvé qu'il favorisait une meilleure compréhension des principes sous-jacents à la composition musicale et visuelle, et qu'il constituait un outil éducatif puissant tant du point de vue artistique et technologique que directement en tant que support d'apprentissage et d'enseignement.

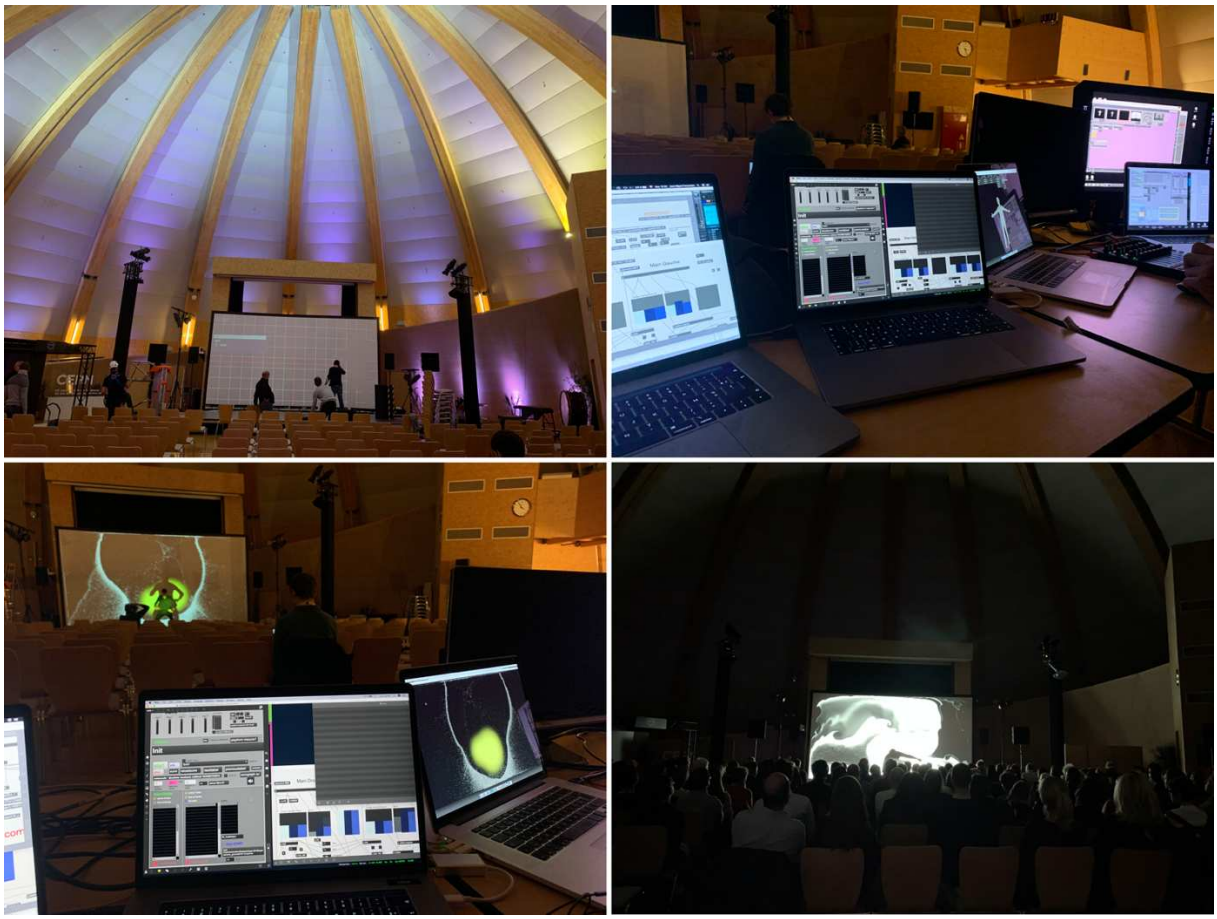


Fig. 10.15 Quelques photos du spectacle *SCULPT* lors du concert au Globe du CERN [147] à Genève le 14 novembre 2019.

[147] https://home.cern/fr

X.5. *Las Pintas*, œuvre audiovisuelle en temps réel

Las Pintas, œuvre audiovisuelle en temps réel, est une commande de la Société des arts technologiques de Montréal (SAT). Elle a été créée dans sa première version à la SAT en octobre 2019 dans la Satosphère[148] (dôme hémisphérique immersif qui combine une projection vidéo en 360° reposant sur 8 vidéoprojecteurs et un système ambisonique avec 157 haut-parleurs). La pièce a été reprise le 5 mars 2020 au Centre Pompidou dans le cadre du concert Ircam Live dans une configuration de vidéoprojection frontale, et un système ambisonique de 29 haut-parleurs. La pièce est une collaboration avec l'artiste vidéo Raphaël Foulon qui vise à explorer la dialectique des rapports image/son dans le contexte de systèmes interactifs génératifs immersifs. Le dispositif repose sur le couplage à tout instant de deux systèmes génératifs, l'un dédié au son et l'autre à l'image, guidés en temps réel par deux interprètes.

Si les relations entre images animées et sons sont souvent abordées dans le contexte du cinéma, elles se trouvent considérablement renouvelées par les autres arts audiovisuels, les installations vidéo, le VJing, les scénographies contemporaines et les nouveaux dispositifs de diffusion. La projection à 360° bouscule la notion de hors-champs. La spatialisation sonore favorise potentiellement une écoute causale, mais l'approche générative — qui conduit à des structures formelles plutôt que représentationnelles tant sur le plan sonore que visuel —, encourage une écoute réduite et conduit à un brouillage des catégories diégétiques et extradiégétiques, de « in » et de « off ».

Une des problématiques questionnées par *Las Pintas* est la congruence image/son. En essayant d'échapper aux rapports littéraux de redondance, d'opposition, de complémentarité ou d'indifférence, l'objectif est de créer une fusion perceptive entre les modalités sonores et visuelles tout en évitant les rapports de subordination d'un média sur l'autre. Les solutions explorées sont ancrées dans le couplage des deux systèmes génératifs, chacun communiquant à l'autre des « paramètres phénoménologiques » du médium généré plutôt que des descripteurs caractérisant son mode de production, sa structure ou son fonctionnement.

Ces paramètres phénoménologiques décrivent des caractéristiques sensibles de l'image ou du son produit, comme la densité graphique, la distribution des couleurs, la localisation spatiale des objets graphiques et leur vitesse de changement, etc. pour l'image, et le spectre, la rugosité, l'harmonicité, le tempo, etc. pour le son. Ce ne sont pas les paramètres de contrôle qui permettent de produire l'image ou le son mais des descripteurs du résultat.

La logique de fabrication propre à un système génératif reste donc indépendante de l'autre, et la symétrie du dispositif, qui se traduit par la rétroaction d'un système sur l'autre, permet d'évacuer les rapports de causalité trop évidents au profit de la codétermination que permet l'interaction bidirectionnelle. C'est en jouant sur le caractère plus ou moins direct du *mapping* des descripteurs de l'un des médias sur le

[148] https://sat.qc.ca/fr/satosphere#

fonctionnement du système de production de l'autre que l'on peut accroître ou diminuer la perception de la congruence entre les deux modalités.

Cette approche reste cependant cantonnée à une échelle locale de la matière visuelle et sonore. Elle ne permet pas de répondre à des enjeux plus globaux impliqués dans l'appréhension générale de la performance : comment gérer une polyphonie, instaurer un arc dramatique, installer et rendre perceptible une forme (au sens musical), spécifier une intrigue temporelle commune aux deux médias ? L'approche mise en œuvre par *Las Pintas* repose sur une partition centralisée qui permet de définir, au sein d'un support unique, les différentes parties de la performance, leurs enchaînements et leurs articulations. Cette partition audiovisuelle est exécutable : elle correspond à un programme temps réel écrit dans un langage dédié permettant d'exprimer les relations réglant la succession, la simultanéité et la durée des différents systèmes génératifs utilisés au cours de la performance. C'est à ce niveau que sont traitées les notions de plans et de scènes, de point de vue et de point d'écoute, et que s'élabore la dynamique de la performance.

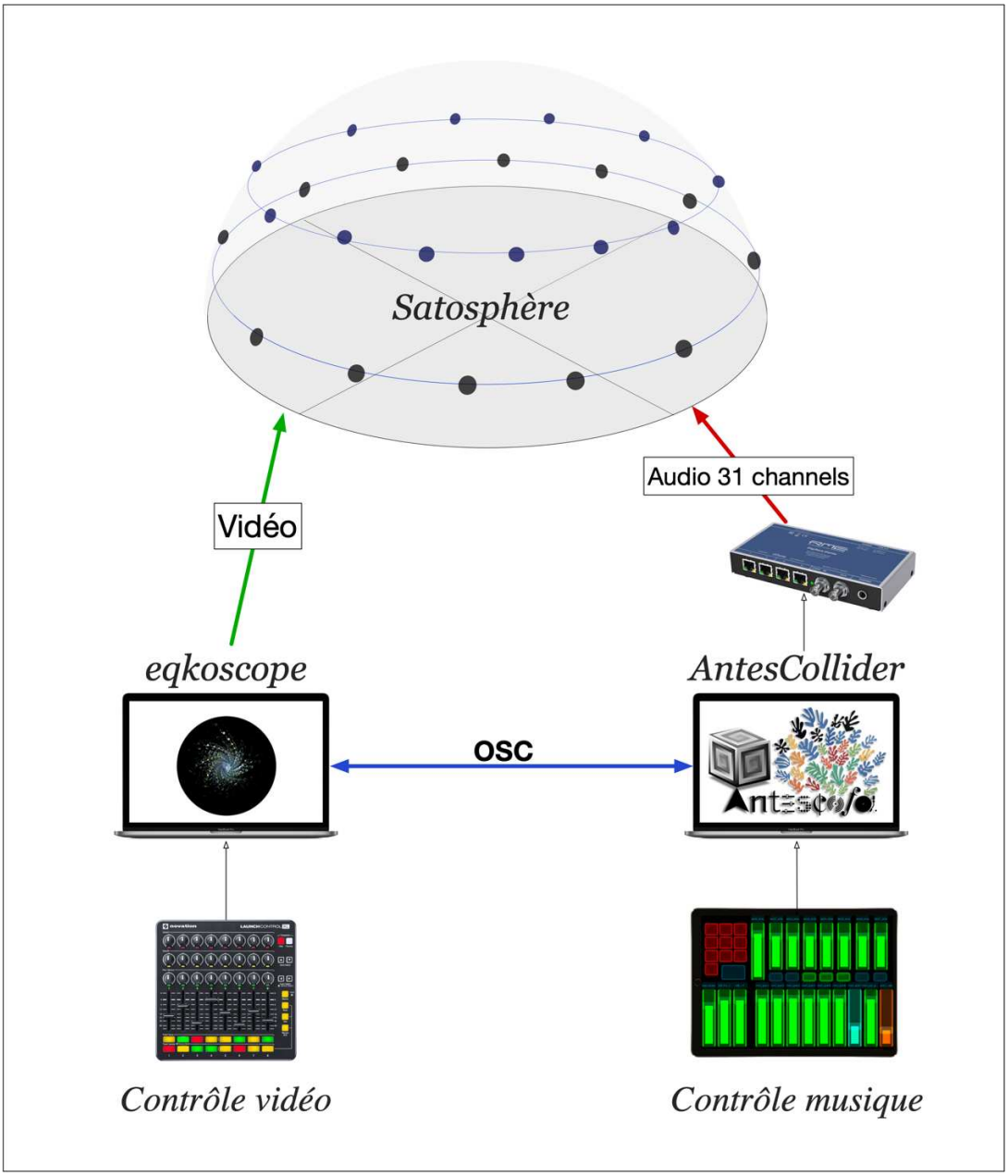


Fig. 10.16 Schéma du dispositif utilisé dans *Las Pintas*. La pièce est interprétée en temps réel par un instrumentiste vidéo qui manipule des images avec le logiciel eqkoscope (R. Foulon) et un musicien qui déclenche et manipule le son pendant l'exécution de la pièce avec AntesCollider. La connexion OSC bidirectionnelle permet d'envoyer mutuellement des informations pour transformer les images et l'audio. Dans sa version originale, *Las Pintas* a été écrite pour la Satosphère de la SAT avec un système hémisphérique en 360° et un système ambisonique de 157 haut-parleurs distribués en 31 canaux.

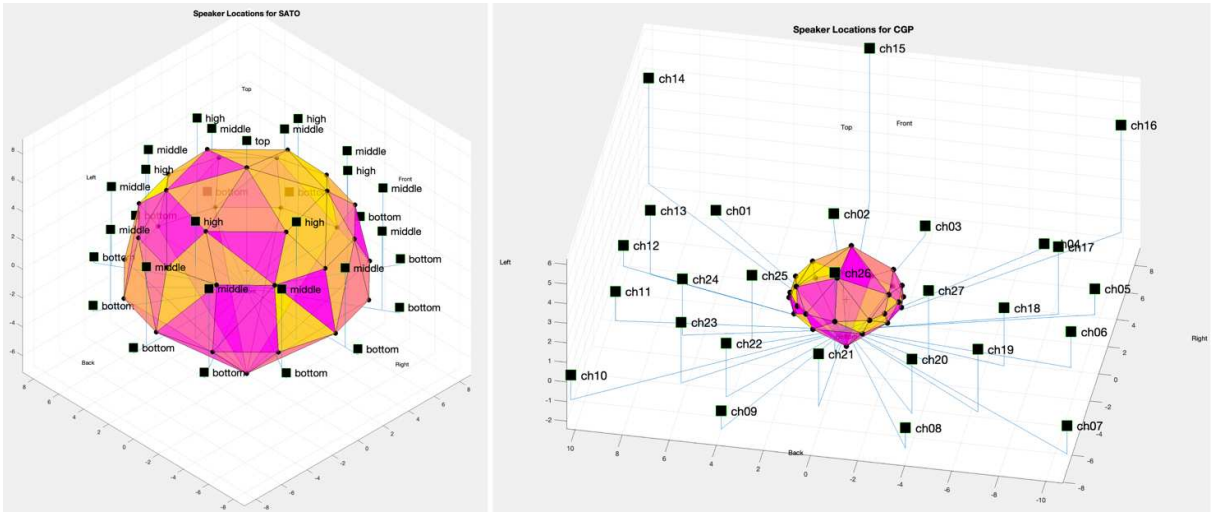
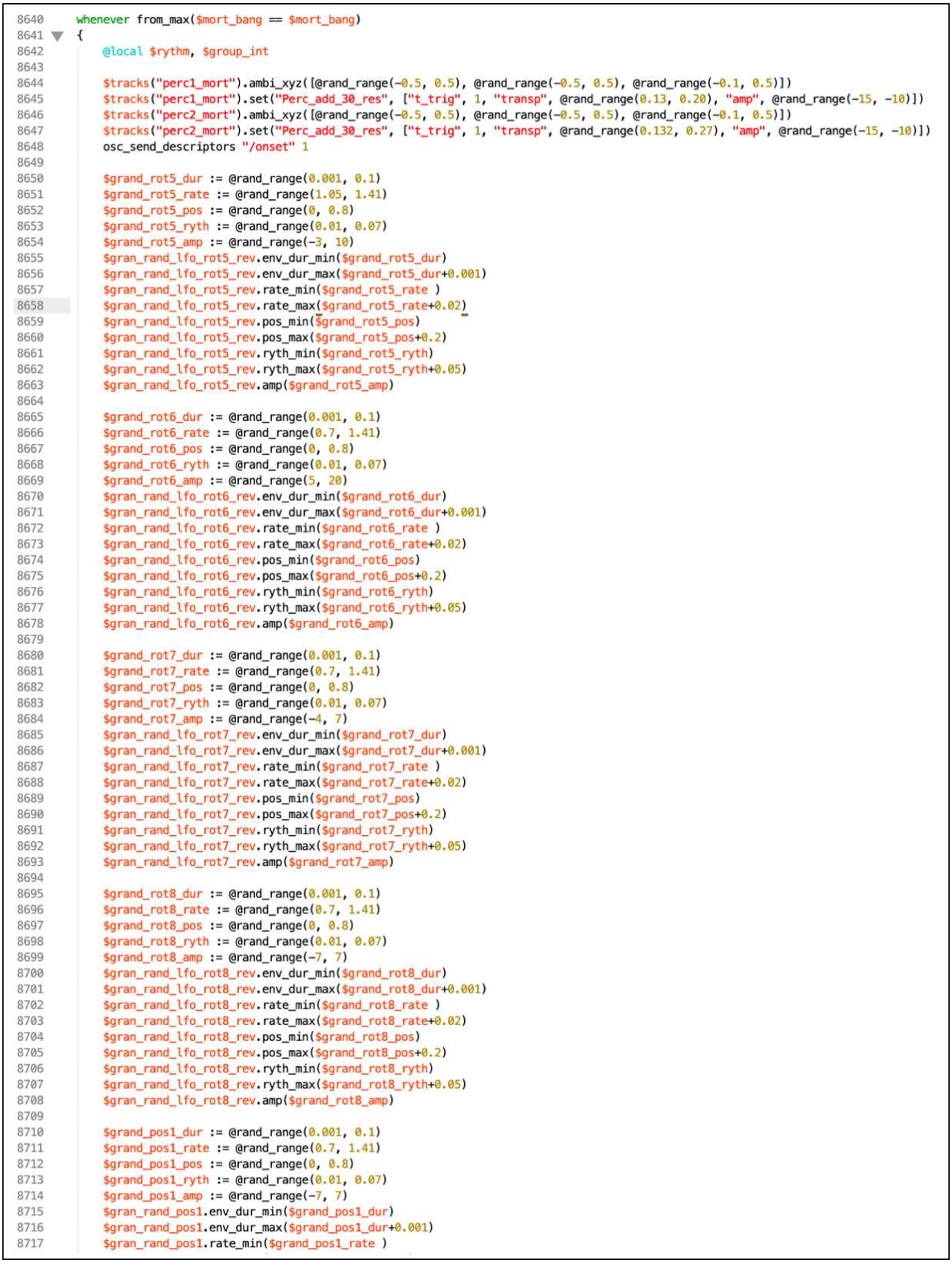


Fig. 10.17 *Plot* en 3D dans le logiciel MATLAB de deux décodeurs ambisoniques réalisés avec la librairie pour FAUST Ambisonic Decoder Toolbox[149] pour les concerts à la Satosphère de la SAT et à la grande salle du Centre Pompidou.

```
8069  obj::crea_track_HOA("track_hoa8_2", "group_hoa2", amp = -13, fade_in = 10, doopler = 1, [["Snow", "amp", 0, "speed", 0.5, "tone", 0.05, "density", 0.6, "blur", 0.1], ["ParamEq", "fre
8070  $tracks("track_hoa8_2").ambi_rand_lfo_sphere(0.9, -100, -40, 0, 360, 0, 0, "linear", 20, id= 6, host = "192.168.1.33", port = 9999, descripteur = ["Centroid"], scale = [0.15, 0.30, 0
8071  $tracks("track_hoa8_2").rand_lfo("Snow", "speed", 0.01, 2, 0.1, "linear", 20)
8072  $tracks("track_hoa8_2").rand_lfo("Snow", "density", 0.01, 1, 0.1, "linear", 20)
8073  $tracks("track_hoa8_2").doopler_lev(10)
8074
8075  10 obj::crea_track_HOA("track_hoa9_2", "group_hoa2", amp = -17, fade_in = 10, doopler = 1, [["Snow", "amp", 0, "speed", 1, "tone", 0.059, "density", 0.3, "blur", 0.25], ["ParamEq", "
8076  $tracks("track_hoa9_2").ambi_rand_lfo_sphere(0.9, -100, -40, 0, 360, 0, 0, "linear", 20, id= 7, host = "192.168.1.33", port = 9999, descripteur = ["Centroid"], scale = [0.15, 0.30, 0
8077  $tracks("track_hoa9_2").rand_lfo("Snow", "speed", 0.01, 2, 0.1, "linear", 20)
8078  $tracks("track_hoa9_2").rand_lfo("Snow", "density", 0.01, 1, 0.1, "linear", 20)
8079  $tracks("track_hoa9_2").doopler_lev(10)
8080
8081  8 obj::crea_track_HOA("track_hoa10_2", "group_hoa2", amp = -16, fade_in = 10, doopler = 1, [["Snow", "amp", 0, "speed", 1.3, "tone", 0.045, "density", 0.3, "blur", 0.28], ["ParamEq",
8082  $tracks("track_hoa10_2").ambi_rand_lfo_sphere(0.9, -100, -40, 0, 360, 0, 0, "linear", 20, id= 8, host = "192.168.1.33", port = 9999, descripteur = ["Centroid"], scale = [0.15, 0.30,
8083  $tracks("track_hoa10_2").rand_lfo("Snow", "speed", 0.01, 2, 0.1, "linear", 10)
8084  $tracks("track_hoa10_2").rand_lfo("Snow", "density", 0.01, 1, 0.1, "linear", 20)
8085  $tracks("track_hoa10_2").doopler_lev(10)
8086
8087  11 obj::crea_track_HOA("track_hoa11_2", "group_hoa2", amp = -12, fade_in = 10, doopler = 1, [["Snow", "amp", 0, "speed", 1.3, "tone", 0.035, "density", 0.3, "blur", 0.3], ["ParamEq",
8088  $tracks("track_hoa11_2").ambi_rand_lfo_sphere(0.9, -100, -40, 0, 360, 0, 0, "linear", 20, id= 9, host = "192.168.1.33", port = 9999, descripteur = ["Centroid"], scale = [0.15, 0.30,
8089  $tracks("track_hoa11_2").rand_lfo("Snow", "speed", 0.01, 2, 0.1, "linear", 400)
8090  $tracks("track_hoa11_2").rand_lfo("Snow", "density", 0.01, 1, 0.1, "linear", 20)
8091  $tracks("track_hoa11_2").doopler_lev(10)
8092
8093  12 obj::crea_track_HOA("track_hoa12_2", "group_hoa2", amp = -17, fade_in = 10, doopler = 1, [["Snow", "amp", 0, "speed", 1.3, "tone", 0.08, "density", 0.3, "blur", 0.2], ["ParamEq",
8094  $tracks("track_hoa12_2").ambi_rand_lfo_sphere(0.9, -100, -40, 0, 360, 0, 0, "linear", 10, id= 10, host = "192.168.1.33", port = 9999, descripteur = ["Centroid"], scale = [0.15, 0.30,
8095  $tracks("track_hoa12_2").rand_lfo("Snow", "speed", 0.01, 2, 0.1, "linear", 400)
8096  $tracks("track_hoa12_2").rand_lfo("Snow", "density", 0.01, 1, 0.1, "linear", 20)
8097  $tracks("track_hoa12_2").doopler_lev(10)
8098
8099  15 obj::crea_track_HOA("track_hoa13_2", "group_hoa2", amp = -14, fade_in = 10, doopler = 1, [["Snow", "amp", 0, "speed", 1.3, "tone", 0.2, "density", 0.3, "blur", 0.05], ["ParamEq",
8100  $tracks("track_hoa13_2").ambi_rand_lfo_sphere(0.9, -100, -40, 0, 360, 0, 0, "linear", 12, id= 11, host = "192.168.1.33", port = 9999, descripteur = ["Centroid"], scale = [0.15, 0.30,
8101  $tracks("track_hoa13_2").rand_lfo("Snow", "speed", 0.01, 2, 0.1, "linear", 400)
8102  $tracks("track_hoa13_2").rand_lfo("Snow", "density", 0.01, 1, 0.1, "linear", 20)
8103  $tracks("track_hoa13_2").rand_lfo("Snow", "amp", -40, -20, -26, "linear", 60)
8104  $tracks("track_hoa13_2").doopler_lev(10)
8105
8106  5 obj::crea_track_HOA("track_hoa8_3", "group_hoa3", amp = 6, fade_in = 10, doopler = 1, [["Snow", "amp", -20, "speed", 0.5, "tone", 0.02, "density", 1.6, "blur", 0.05], ["ParamEq", "
8107  $tracks("track_hoa8_3").ambi_rand_lfo_sphere(0.9, -100, -40, 0, 360, 0, 0, "linear", 10, id= 12, host = "192.168.1.33", port = 9999, descripteur = ["Loudness"])
8108  $tracks("track_hoa8_3").rand_lfo("Snow", "speed", 0.01, 2, 0.1, "linear", 100)
8109  $tracks("track_hoa8_3").rand_lfo("Snow", "tone", 0.02, 0.0205, 0.02, "linear", 100)
8110  $tracks("track_hoa8_3").rand_lfo("Snow", "density", 1, 4, 0.1, "linear", 20)
8111  $tracks("track_hoa8_3").doopler_lev(10)
8112
8113  7 obj::crea_track_HOA("track_hoa9_3", "group_hoa3", amp = 8, fade_in = 10, doopler = 1, [["Snow", "amp", -20, "speed", 1, "tone", 0.003, "density", 0.3, "blur", 0.08], ["ParamEq", "f
8114  $tracks("track_hoa9_3").ambi_rand_lfo_sphere(0.99, -100, -40, 0, 360, 0, 0, "linear", 13, id= 13, host = "192.168.1.33", port = 9999, descripteur = ["Loudness"])
8115  $tracks("track_hoa9_3").rand_lfo("Snow", "speed", 0.01, 2, 0.1, "linear", 100)
8116  $tracks("track_hoa9_3").rand_lfo("Snow", "tone", 0.03, 0.0305, 0.02, "linear", 100)
8117  $tracks("track_hoa9_3").rand_lfo("Snow", "density", 2, 6, 0.1, "linear", 20)
8118  $tracks("track_hoa9_3").rand_lfo("Snow", "amp", -20, -40, -20, "linear", 40)
8119  $tracks("track_hoa9_3").doopler_lev(10)
```

Fig. 10.18 Extrait d'une partition électronique de *Las Pintas*. Dans cette séquence, différentes synthèses sont instanciées séquentiellement. Chacune se déploie indépendamment avec des trajectoires spatiales et des paramètres différents. La sortie audio de chaque *track* AntesCollider est analysée en temps réel par des descripteurs audio, dans ce cas Centroid (lignes 8071, 8076, 8082…) et Loudness (lignes 8107, 8114…). Les données d'analyse par descripteur sont envoyées vers l'ordinateur de rendu vidéo et servent d'entrées au logiciel eqkoscope.

[149] https://bitbucket.org/ambidecodertoolbox/adt/src/master/



```
8640    whenever from_max($mort_bang == $mort_bang)
8641    {
8642        @local $rythm, $group_int
8643
8644        $tracks("perc1_mort").ambi_xyz([@rand_range(-0.5, 0.5), @rand_range(-0.5, 0.5), @rand_range(-0.1, 0.5)])
8645        $tracks("perc1_mort").set("Perc_add_30_res", ["t_trig", 1, "transp", @rand_range(0.13, 0.20), "amp", @rand_range(-15, -10)])
8646        $tracks("perc2_mort").ambi_xyz([@rand_range(-0.5, 0.5), @rand_range(-0.5, 0.5), @rand_range(-0.1, 0.5)])
8647        $tracks("perc2_mort").set("Perc_add_30_res", ["t_trig", 1, "transp", @rand_range(0.132, 0.27), "amp", @rand_range(-15, -10)])
8648        osc_send_descriptors "/onset" 1
8649
8650        $grand_rot5_dur := @rand_range(0.001, 0.1)
8651        $grand_rot5_rate := @rand_range(1.05, 1.41)
8652        $grand_rot5_pos := @rand_range(0, 0.8)
8653        $grand_rot5_ryth := @rand_range(0.01, 0.07)
8654        $grand_rot5_amp := @rand_range(-3, 10)
8655        $gran_rand_lfo_rot5_rev.env_dur_min($grand_rot5_dur)
8656        $gran_rand_lfo_rot5_rev.env_dur_max($grand_rot5_dur+0.001)
8657        $gran_rand_lfo_rot5_rev.rate_min($grand_rot5_rate )
8658        $gran_rand_lfo_rot5_rev.rate_max($grand_rot5_rate+0.02)
8659        $gran_rand_lfo_rot5_rev.pos_min($grand_rot5_pos)
8660        $gran_rand_lfo_rot5_rev.pos_max($grand_rot5_pos+0.2)
8661        $gran_rand_lfo_rot5_rev.ryth_min($grand_rot5_ryth)
8662        $gran_rand_lfo_rot5_rev.ryth_max($grand_rot5_ryth+0.05)
8663        $gran_rand_lfo_rot5_rev.amp($grand_rot5_amp)
8664
8665        $grand_rot6_dur := @rand_range(0.001, 0.1)
8666        $grand_rot6_rate := @rand_range(0.7, 1.41)
8667        $grand_rot6_pos := @rand_range(0, 0.8)
8668        $grand_rot6_ryth := @rand_range(0.01, 0.07)
8669        $grand_rot6_amp := @rand_range(5, 20)
8670        $gran_rand_lfo_rot6_rev.env_dur_min($grand_rot6_dur)
8671        $gran_rand_lfo_rot6_rev.env_dur_max($grand_rot6_dur+0.001)
8672        $gran_rand_lfo_rot6_rev.rate_min($grand_rot6_rate )
8673        $gran_rand_lfo_rot6_rev.rate_max($grand_rot6_rate+0.02)
8674        $gran_rand_lfo_rot6_rev.pos_min($grand_rot6_pos)
8675        $gran_rand_lfo_rot6_rev.pos_max($grand_rot6_pos+0.2)
8676        $gran_rand_lfo_rot6_rev.ryth_min($grand_rot6_ryth)
8677        $gran_rand_lfo_rot6_rev.ryth_max($grand_rot6_ryth+0.05)
8678        $gran_rand_lfo_rot6_rev.amp($grand_rot6_amp)
8679
8680        $grand_rot7_dur := @rand_range(0.001, 0.1)
8681        $grand_rot7_rate := @rand_range(0.7, 1.41)
8682        $grand_rot7_pos := @rand_range(0, 0.8)
8683        $grand_rot7_ryth := @rand_range(0.01, 0.07)
8684        $grand_rot7_amp := @rand_range(-4, 7)
8685        $gran_rand_lfo_rot7_rev.env_dur_min($grand_rot7_dur)
8686        $gran_rand_lfo_rot7_rev.env_dur_max($grand_rot7_dur+0.001)
8687        $gran_rand_lfo_rot7_rev.rate_min($grand_rot7_rate )
8688        $gran_rand_lfo_rot7_rev.rate_max($grand_rot7_rate+0.02)
8689        $gran_rand_lfo_rot7_rev.pos_min($grand_rot7_pos)
8690        $gran_rand_lfo_rot7_rev.pos_max($grand_rot7_pos+0.2)
8691        $gran_rand_lfo_rot7_rev.ryth_min($grand_rot7_ryth)
8692        $gran_rand_lfo_rot7_rev.ryth_max($grand_rot7_ryth+0.05)
8693        $gran_rand_lfo_rot7_rev.amp($grand_rot7_amp)
8694
8695        $grand_rot8_dur := @rand_range(0.001, 0.1)
8696        $grand_rot8_rate := @rand_range(0.7, 1.41)
8697        $grand_rot8_pos := @rand_range(0, 0.8)
8698        $grand_rot8_ryth := @rand_range(0.01, 0.07)
8699        $grand_rot8_amp := @rand_range(-7, 7)
8700        $gran_rand_lfo_rot8_rev.env_dur_min($grand_rot8_dur)
8701        $gran_rand_lfo_rot8_rev.env_dur_max($grand_rot8_dur+0.001)
8702        $gran_rand_lfo_rot8_rev.rate_min($grand_rot8_rate )
8703        $gran_rand_lfo_rot8_rev.rate_max($grand_rot8_rate+0.02)
8704        $gran_rand_lfo_rot8_rev.pos_min($grand_rot8_pos)
8705        $gran_rand_lfo_rot8_rev.pos_max($grand_rot8_pos+0.2)
8706        $gran_rand_lfo_rot8_rev.ryth_min($grand_rot8_ryth)
8707        $gran_rand_lfo_rot8_rev.ryth_max($grand_rot8_ryth+0.05)
8708        $gran_rand_lfo_rot8_rev.amp($grand_rot8_amp)
8709
8710        $grand_pos1_dur := @rand_range(0.001, 0.1)
8711        $grand_pos1_rate := @rand_range(0.7, 1.41)
8712        $grand_pos1_pos := @rand_range(0, 0.8)
8713        $grand_pos1_ryth := @rand_range(0.01, 0.07)
8714        $grand_pos1_amp := @rand_range(-7, 7)
8715        $gran_rand_pos1.env_dur_min($grand_pos1_dur)
8716        $gran_rand_pos1.env_dur_max($grand_pos1_dur+0.001)
8717        $gran_rand_pos1.rate_min($grand_pos1_rate )
```

Fig. 10.19 Extrait de la partition électronique de *Las Pintas*. Dans cette partie, on peut voir comment tous les paramètres des synthèses sont modifiés en faisant varier aléatoirement leurs paramètres sur le déclenchement d'un *trigger* (*Whenever*) à partir du contrôleur Lemur dans un iPad.

Système vidéo eqkoscope

L'eqkoscope est un logiciel de génération vidéo basé sur openFrameworks et diffusé sous licence libre (GPL v2). Il est développé depuis 2012 par Raphaël Foulon et est disponible en téléchargement gratuit depuis octobre 2018. Conçu principalement pour les performances live, il est doté d'un moteur de rendu orienté temps réel. Il est pourvu de nombreuses entrées/sorties standard (MIDI, OSC, TCP/IP, audio) afin de permettre une intégration avec des pièces musicales. Il est en outre possible de le contrôler à partir d'une grande variété d'interfaces : joystick, manettes, leapmotion, capteurs de mouvement. Il a été utilisé dans le cadre de concerts, représentations théâtrales contemporaines, installations audiovisuelles, postproduction et *mapping* vidéo.

L'utilisation d'un outil de création vidéo dédié permet de conserver une indépendance face à l'utilisation de logiciels ou bibliothèques d'effets commerciaux. Ainsi, nous développons un univers graphique unique qui n'est pas teinté par l'utilisation d'effets spéciaux présents dans la palette offerte par les outils de création « grand public » (Resolume, Modul8, MadMapper, AfterEffects).

Le logiciel a été conçu autour d'une boucle de larsen. Ici, le larsen est numérique : il est l'émulation d'un système de larsen analogique. Une caméra virtuelle effectue des captures de l'état interne de la carte graphique, cette caméra peut être déplacée dans un espace 3D. Un dispositif de diffusion virtuel permet de reproduire le rendu en termes de colorimétrie et de distorsions spatiales d'un écran ou d'un tube cathodique. Enfin, il est possible d'intégrer à la boucle de larsen des effets vidéo codés sous la forme de *shaders* en langage GLSL. Ces effets sont d'ordre colorimétrique (niveaux, saturation), issus des chaînes de postproduction classiques (netteté, scintillement, kaléidoscopes), de l'émulation (*glitch-art*, distorsions analogiques) ou des arts numériques (tunnels, *compositing* complexe).

Dans le cadre de la création des « univers » dans *Las Pintas*, nous avons exploité le côté chaotique et organique du larsen vidéo pour aboutir à la création de formes de vie numériques, dont les paramètres peuvent être finement contrôlés grâce aux fonctionnalités de l'eqkoscope et d'Antescofo.

La proposition s'inscrit dans la culture du larsen vidéo qui est peu à peu redécouverte ces dernières années, en l'abordant sous l'angle moderne d'un contrôle numérique précis. Elle se démarque également en termes de format : le larsen vidéo tel que nous l'envisageons est généré à partir d'un noyau circulaire, c'est-à-dire que les dynamiques mises en œuvre favorisent la génération de visuels sous la forme de disque (contrairement aux formats rectangulaires traditionnellement utilisés dans le cinéma et la performance vidéo). Ainsi, notre proposition sera idéalement diffusée dans un système de rendu audiovisuel hémisphérique de type *full dome*. Des systèmes plus simples permettant une projection vidéo frontale ou zénithale pourront évidemment être envisagés, grâce à la souplesse et la puissance offertes par les technologies de projection à courte focale.

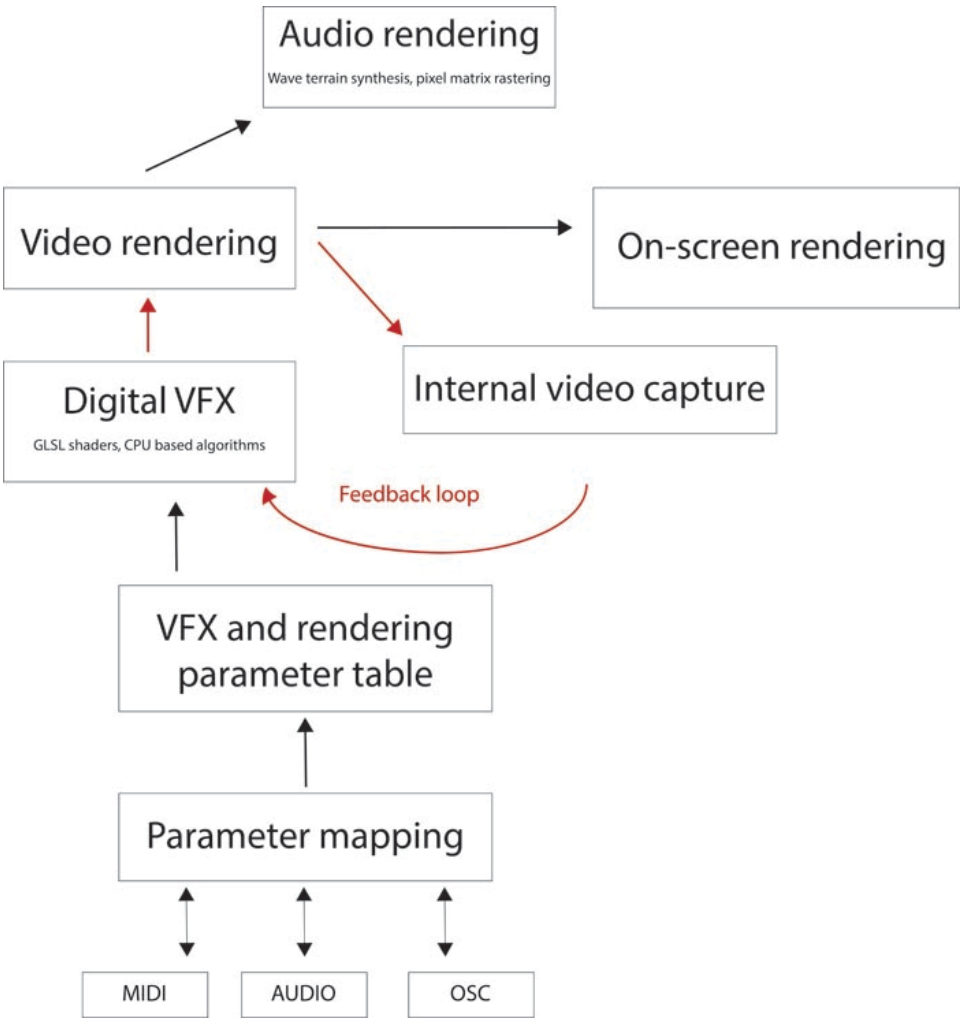


Fig. 10.20 Architecture logicielle de l'eqkoscope, articulée autour d'une boucle de larsen vidéo numérique

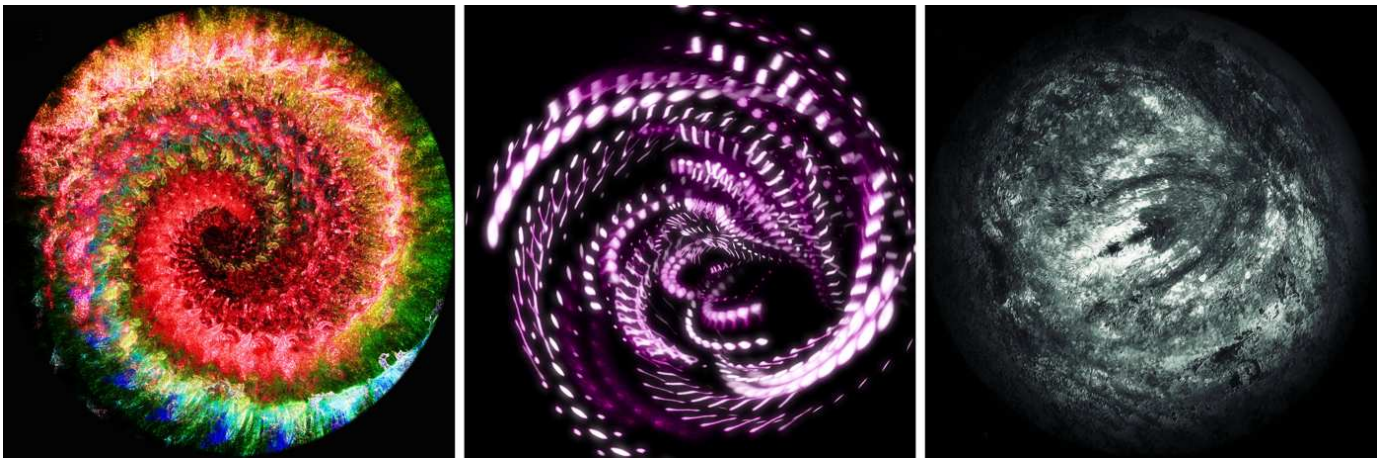


Fig. 10.21 Captures de trois instants vidéo de *Las Pintas* réalisés avec le logiciel eqkoscope.

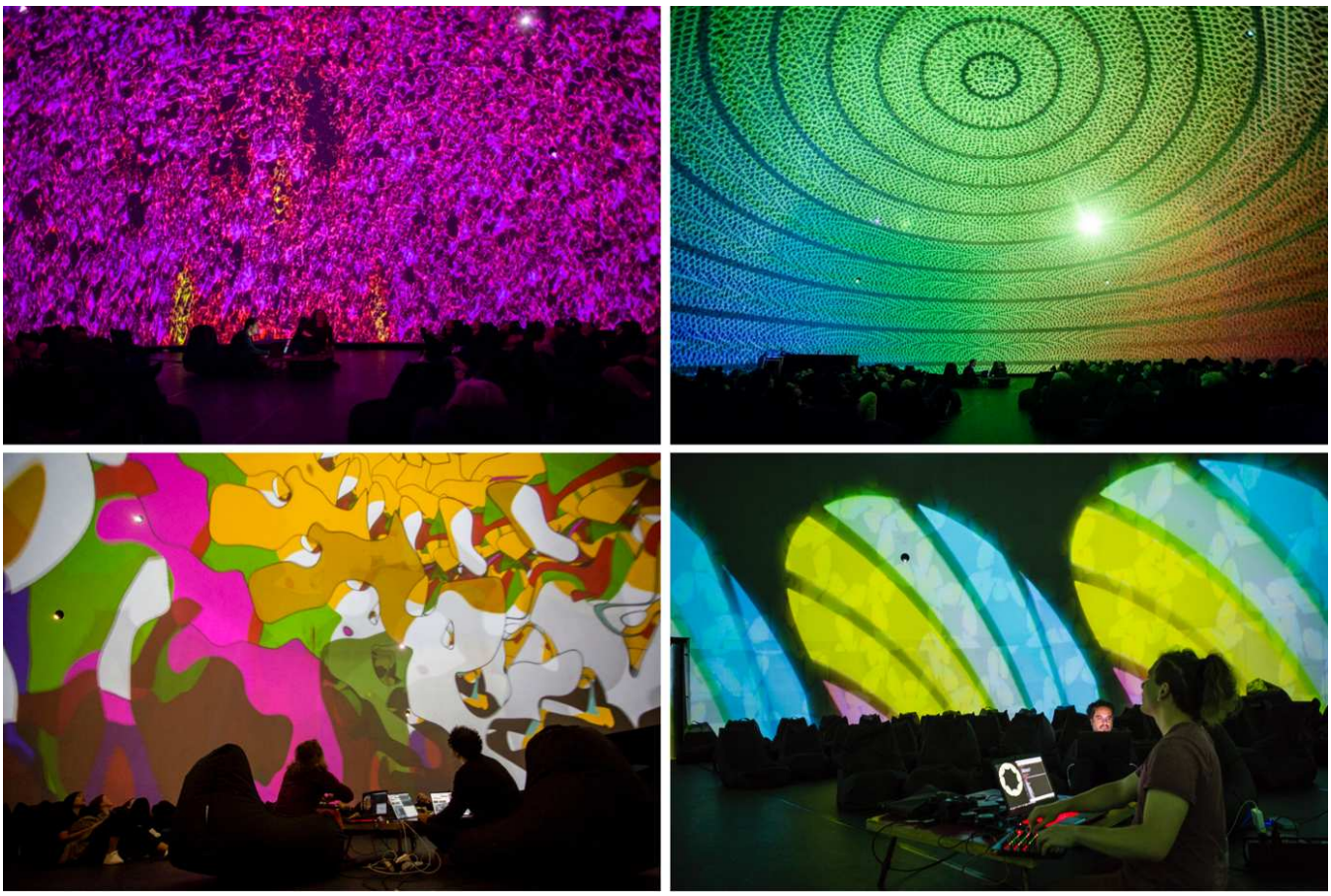


Fig. 10.22 Photos de la performance à la Satosphère de la SAT à Montréal.

Conclusion

Ce travail de thèse en recherche artistique au sein de l'équipe Représentations musicales au laboratoire STMS et à l'Ircam, a été une vraie motivation pour formaliser mon travail et explorer de nouvelles approches aussi bien au niveau artistique que technique. La possibilité de pouvoir travailler et d'échanger des savoir-faire musicaux et technologiques dans un environnement de recherche a été stimulant et a nourri ma pensée et ma pratique musicale.

Ce travail m'a permis de questionner et d'affiner mon positionnement sur différents aspects de la création musicale contemporaine enrichie aujourd'hui avec des moyens technologiques en interaction avec d'autres médias, notamment la vidéo. Il m'a aussi permis de concrétiser plusieurs idées relatives à la notation de l'électronique en relation avec la notion de partitions électroniques centralisées dans le contexte des musiques interactives. Ces idées technologiques et musicales se sont concrétisées par le développement de la librairie AntesCollider qui vise à intégrer le système Antescofo en tant que suivi de partition, et surtout comme langage de programmation avec un système de synthèse audio performant et dynamique tel que SuperCollider.

Comparée aux techniques précédentes telles que les *cue-list* ou les *presets* statiques, l'expressivité du langage Antescofo avec l'utilisation des structures de contrôle de haut niveau comme les acteurs (objets), permet un renouvellement de l'écriture de l'électronique temps réel pour aller vers une véritable notation de l'écriture, augmentée et dynamique. Cette piste est prometteuse et me fait entrevoir de nouveaux paradigmes d'écriture de l'électronique en se détachant des représentations graphiques et de leurs interfaces.

La composition de *Curvatura II*, composée textuellement en langage Antescofo/AntesCollider a constitué une expérimentation révélatrice. Ce travail m'a convaincu de tout l'intérêt et de la fécondité musicale de trois propositions :

- un programme peut aussi jouer le rôle d'une partition (pour l'électronique) ;
- écrire des partitions directement dans un langage de programmation est porteur de nouvelles idées musicales et peut jouer le même rôle d'espace de pensée que la notation traditionnelle ;
- la réalisation par un programme d'une partition électronique apporte une flexibilité et ouvre une liberté musicale inégalée.

Cette liberté est fondamentale dans ma recherche et dans ma pratique de compositeur, même si en contrepartie, il faut apprendre et maîtriser un nouveau langage informatique textuel.

Ce travail n'en est qu'à ses débuts et il y a sûrement de nouvelles fonctionnalités dédiées à développer pour l'écriture des musiques interactives et leurs relations avec d'autres médias temporels, en particulier si on pense à des applications comme l'opéra, le théâtre, la danse, etc.

Une recherche pratique avec ces systèmes est nécessaire, par exemple pour faciliter la lisibilité et la compréhension de la polyphonie dans une représentation textuelle qui est essentiellement monodimensionnelle.

Une piste évoquée au chapitre V.12. est de développer des représentations hybrides – graphique et textuelle – en permettant le passage automatique de l'une à l'autre. Cette piste avait été initiée par l'outil Ascograph. Nous avons aussi évoqué l'intérêt de représentations graphiques alternatives pour les interprètes (musiciens, chefs d'orchestre, danseurs, etc.) comme pour les musicologues (analyse de l'œuvre).

J'ai pu expérimenter la notion de partition électronique centralisée dans plusieurs compositions. Combiner un langage synchrone réactif et temporisé à un moteur audio dans un même environnement dynamique s'est avéré beaucoup plus expressif que tous les systèmes que j'ai pu pratiquer auparavant en tant que compositeur et RIM. Cette intégration donne une flexibilité nouvelle au paradigme des musiques interactives, étend l'espace de pensée du compositeur et offre un support incontestable pour organiser et réaliser le réseau complexe d'interactions que ce genre musical exige. Plusieurs œuvres ont été réalisées avec la librairie et plusieurs techniques électroacoustiques ont été implémentées, surtout dans le domaine de la spatialisation (synthèse concatenative spatiale par exemple). L'écriture de l'électronique et l'expérimentation dans la librairie AntesCollider se sont révélées non seulement efficaces mais porteuses de nouvelles idées aussi bien au niveau musical que technique.

Le travail sur l'écriture et la composition de l'espace (« synthèse spatiale ») avec ce système ne fait que commencer, plusieurs possibilités de création sont ouvertes, notamment avec l'analyse du champ sonore et l'utilisation des descripteurs pour composer l'espace. La capacité à ne plus penser et composer l'espace en tant que sources indépendantes qui se déplacent mais de le concevoir et l'imaginer d'une façon organique est un axe de recherche fascinant que je continuerai à développer aussi bien théoriquement, musicalement que techniquement.

L'utilisation des systèmes de captation gestuelle permet aussi de nouvelles approches dans la composition de l'interaction qui, à mon avis, n'ont pas été exploitées du fait d'un manque de systèmes de *mapping* efficaces et flexibles. C'est une pratique qui n'est pas trop répandue dans le milieu de la musique, sûrement à cause de la pérennité des dispositifs et au fait que pour arriver à des résultats convaincants, il faut beaucoup d'expérimentations et du temps avec le performer, etc.

Les expériences audiovisuelles que j'ai eu l'opportunité de développer entre génération musicale et vidéo en temps réel dans une performance comme dans la pièce *Las Pintas* ont été fort stimulantes. La construction d'une synesthésie entre les deux médias, la mise en parallèle au niveau formel de problématiques propres à chaque domaine, et l'articulation des deux médias en évitant les rapports de subordination, sont aussi pour moi un champ de recherche artistique riche et avec plein des possibilités que j'espère pouvoir continuer à développer.

J'ai aussi évoqué la problématique de l'obsolescence de la technologie et de son avancement rapide et implacable, ce qui nous oblige à une adaptation permanente pour

laquelle il semble que les outils et concepts sont obsolètes avant même de les avoir pratiqués. Cet état de fait implique une constante évolution qui va à l'encontre d'une pratique à long terme qui permettrait de « maîtriser » l'outil, comme cela a été le cas avec les instruments acoustiques traditionnels et leur stabilisation qui a permis de créer un répertoire riche. L'utilisation de systèmes génériques et open source (qui ne seront pas soumis à la loi du marché) ne garantissent pas une pérennité absolue, mais ils favorisent un renouvellement plus incrémental, l'intégration progressive de nouvelles fonctionnalités (tant qu'elles ne nécessitent pas des changements radicaux d'infrastructures), ce qui favorise la rétrocompatibilité et une pérennité sur le long terme. Il faut peut-être aussi penser, concevoir et accepter la composition comme un acte artistique éphémère, sans se soucier de la pérennité, ce qui serait une façon abrupte de se libérer de cette question.

Enfin, si en tant que compositeur, cette thèse m'a permis de mener une recherche personnelle passionnante, qui a nourri ma pensée musicale et m'a amené à explorer de nouveaux mondes sonores et techniques, elle a aussi été l'occasion de partager ces réflexions et ce travail à travers des concerts, des discussions, des cours, des projets collaboratifs, et à travers des outils open source mis à la disposition de la communauté des compositeurs, musiciens et créateurs sonores.

Bibliographie

[Ago98] Carlos Agon, « OpenMusic : Un langage visuel pour la composition musicale assistée par ordinateur », PhD thesis, University of Paris 6, 1998.

[AgAsBr06] Carlos Agon, Gérard Assayag, Jean Bresson (Eds.), The OM Composer's Book, Editions Delatour/IRCAM, 2006-2018 (3 volumes).

[AG12] Andrea Agostini, Daniele Ghisi, "bach: an environment for computer-aided composition in Max", In Proceedings of the International Computer Music Conference, 2012.

[AgGhGi19] Andrea Agostini, Daniele Ghisi, Jean-Louis Giavitto, "Programming in style with bach (extended version)", 14th International Symposium on Computer Music Multidisciplinary Research, 2019, Marseille, France.

[Ass98] Gérard Assayag, "Computer assisted composition today", in The First Symposium on Music and Computers, Corfu, Greece, 1998.

[Bay93] François Bayle, *Musique acousmatique, propositions... positions*, Buchet Chastel, 1993.

[Ber06] Luciano Berio, « Remembering the Future », based on lectures delivered in 1993 and 1994, Cambridge, MA: Harvard University Press, 2006.

[BG92] Gérard Berry and Georges Gonthier, "The Esterel Synchronous Programming Language: Design, Semantics, Implementation", *Sci. Comput. Program.*, 19(2), 87-152, 1992.

[Ble10] T. Blechmann, "Supernova, a multiprocessor-aware synthesis server for SuperCollider", in Proceedings of the Linux Audio Conference, 2010, p. 141-146.

[Bou63] Pierre Boulez, *Penser la musique aujourd'hui*, Gallimard, 1963.

[BouGre88] Pierre Boulez, Patrick Greussay, « Et la musique, entretiens avec Philippe Manoury », Machines Virtuelles, Traverses 44-45, Centre de création industrielle, Centre Georges Pompidou, Paris, 1988, p. 133.

[BrBiCaSc17] Jean Bresson, Dimitri Bouche, Thibaut Carpentier, Diemo Schwarz, Jérémie Garcia, "Next-generation Computer-aided Composition Environment: A New Implementation of OpenMusic", International Computer Music Conference (ICMC'17), 2017, Shanghai, China.

[BuCoPo16] Grigore Burloiu, Arshia Cont, Clément Poncelet, “A visual framework for dynamic mixed music notation”, *Journal of New Music Research*, Taylor & Francis (Routledge), 2016.

[CLF81] Claude Cadoz, Annie Luciani et Jean-Loup Florens, « Synthèse musicale par simulation des mécanismes instrumentaux, transducteurs gestuels rétroactifs pour l’étude du jeu instrumental », *Revue d’acoustique*, 4.59 (1981), p. 279-292 (cf. p. 52, 95).

[CaWa00] Claude Cadoz, Marcelo M. Wanderley, « Gesture - Music ». Marcelo Wanderley et Marc Battier, Ircam- Centre Pompidou. Trends in Gestural Control of Music, 2000.

[CNW15] Thibaut Carpentier, Markus Noisternig, Olivier Warusfel, “Twenty Years of Ircam Spat: Looking Back, Looking Forward”, 41st International Computer Music Conference (ICMC), Sept. 2015, Denton, TX, United States. p. 70-277.

[Chi98] Michel Chion, *Le son*, coll. fac. cinéma-image, Nathan université, Paris, 1998.

[Cho71] J. Chowning, “The Simulation of Moving SoundSources”, *Journal of the Audio Engineering Society*, 19, p. 2-6, 1971.

[CCG14] Thomas Coffy, Jean-Louis Giavitto, Arshia Cont, “AscoGraph: A User Interface for Sequencing and Score Following for Interactive Music”, ICMC 2014, 40th International Computer Music Conference, Sept. 2014, Athens, Greece.

[COES07] Nick Collins, Julio d’Escrivan, *Electronic Music*, Cambridge University Press, 2007.

[CMM09] Nicholas J. Conard, Maria Malina et Susanne C. Münzel, “New flutes document the earliest musical tradition in southwestern Germany”, *Nature*, 460.7256 (2009), p. 737 (cf. p. 18, 43).

[Cuv16] Philippe Cuvillier, On temporal coherency of probabilistic models for audio-to-score alignment. Sound [cs.SD]. Université Pierre et Marie Curie – Paris VI, 2016.

[BZS10] Frédéric Bevilacqua, Bruno Zamborlin, Anthony Sypniewski, Norbert Schnell, Fabrice Guédy, Nicolas Rasamimanana, “Continuous realtime gesture following and recognition”, Gesture in embodied communication and human-computer interaction, p. 73-84, Springer, 2010.

[DiS13] Jean-Louis Di Santo, « L’acousmoscribe », In Journée d’étude #1. GT-Notation, MINT-OMF, 2013.

[DiSh03] Agostino Di Schipio, “Sound is the interface”: from interactive to ecosystemic signal processing, *Organised Sound*, vol. 8, n° 3, p. 269-277.

[Deb87] Claude Debussy, *Monsieur Croche et autres écrits*, Gallimard, 1987.

[Del88] François Delalande, « Le Geste, outil d'analyse : quelques enseignements d'une recherche sur la gestique de Glenn Gould », *Analyse musicale* 10 (1988), p. 43-46 (cf. p. 54).

[FOL12] Dominique Fober, Yann Orlarey, Stéphane Letz, "INScore - An Environment for the Design of Live Music Scores", Linux Audio Conference, 2012, Stanford, United States. p. 47-54. hal-02158817

[MC02] James McCartney, "Rethinking the computer music language: SuperCollider", *Computer Music Journal*, vol. 26, n° 4, p. 61-68, 2002.

[WRFR97] Matthew Wright, Adrian Freed *et al.*, "Open SoundControl: A New Protocol for Communicating with Sound Synthesizers", ICMC, 1997 (cf. p. 138).

[Con08] Arshia Cont, "Antescofo: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music", In Proceedings of International Computer Music Conference (ICMC), Belfast, Irlande du Nord, August 2008.

[Con10] Arshia Cont, "A Coupled Duration-Focused Architecture for Real-Time Music-to-Score Alignment", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6), 974-987, 2010.

[DeSh09] Etienne Deleflie, Greg Schiemer, "Spatial Grains: Imbuing Granular Particles With Spatial-Domain Information", in Proceedings of ACMC09, Improvise, The Australasian Computer Music Conference, Queensland University of Technology, 2009.

[Der97] Jacques Derrida, *De la grammatologie*, Minuit, 1967.

J. Echeveste, J.-L. Giavitto, A. Cont, "A Dynamic Timed-Language for Computer-Human Musical Interaction", https://hal. inria.fr/hal-00917469/document, Tech. Rep., 2013.

[Gab47] Dennis Gabor, "Acoustical Quanta and the Theory of Hearing", *Nature*, vol. 159, n° 4044, 1947.

[GIAG19] Jean-Louis Giavitto, Andrea Agostini, "Bell, a textual language for the bach library", ICMC 2019 - International Computer Music Conference, juin 2019, New York, United States.

J.-L. Giavitto, J.-M. Echeveste, A. Cont, P. Cuvillier, "Time, Timelines and Temporal Scopes in the Antescofo DSL v1. 0", International Computer Music Conference (ICMC), 2017.

[FEGIDO19] J. M. Fernandez, J.-L. Giavitto, P. Donat-Bouillud, “AntesCollider: Control and Signal Processing in the Same Score”, ICMC 2019 - International Computer Music Conference, New York 2019

[Fon13] Nuno Fonseca, “3D Particle Systems for audio Applications”, DAFx-13, proc. of the 16th Int. Conference on Digital Audio Effects, Maynooth, 2013.

[Ger73] Michael A. Gerzon, “Periphony: With-height sound reproduction”, *Journal of the Audio Engineering Society*, 21(1), 2-10, 1973.

[Gri08] Gérard Grisey, « Devenir du son », *Écrits*, Éditions MF, 2008.

[GrPe96] T.R.G. Green, M. Petre, “Usability analysis of visual programming environments: a ‘Cognitive dimensions’ framework”, *J. Vis. Lang. Comput*. 7 (2) (1996) 131-174.

[GrKi06] Antony Gritten and Elaine King, *Music and gesture*, Ashgate Publishing, Ltd., 2006.

[GoB18] Rama Gottfried, Jean Bresson, “Symbolist: An Open Authoring Environment for End-user Symbolic Notation”, International Conference on Technologies for Music Notation and Representation (TENOR’18), 2018, Montréal, Canada.

[HCRP91] N. Halbwachs, P. Caspi, P. Raymond, D. Pilaud, “The Synchronous Data Flow Programming Language LUSTRE”, Proceedings of the IEEE, 79(9), 1305-1320, 1991.

[Bou00] R. C. Boulanger *et al*., The Csound book: perspectives in software synthesis, sound design, signal processing, and programming, MIT press, 2000.

[Kim05] D. Kim-Boyle, “Sound Spatialization with Particle Systems”, in Proc. 8th International Conference on Digital Audio Effects (DAFX-05), Madrid 2005, p. 65-68 ; Proceedings of the Journées d’informatique musicale Conference, Paris 2005, p. 143-146.

[LaSi87] J.H. Larkin, H.A. Simon, “Why a diagram is (sometimes) worth ten thousand words”, *Cogn. Sci*., 11 (1987) 65-99.

[Lem19] Serge Lemouton, Riccardo Borghesi, Sampo Haapamäki, Frédéric Bevilacqua, Emmanuel Fléty, “Following Orchestra Conductors: the IDEA Open Movement Dataset”, Moco 2019 Proceedings, 2019.

[Lig14] György Ligeti, *Écrits sur la musique et les musiciens*, Éditions Contrechamps, 2014.

[Lig66] György Ligeti, “Form in der Neuen Musik”, Darmstädter Beiträge zur Neuen Musik, 1966.

[LBH09] T. Lossius, P. Balthazar, T. de la Hogue, “DBAP Distance-Based Amplitude Panning”, in Proceedings of the International Computer Music Conference, Montréal, 2009.

[Mal99] D.G. Malham, “Higher order ambisonic systems for the spatialisation of sound”, ICMC99, p. 1-4, Beijing.

[Man97] Phillipe Manoury, « Les partitions virtuelles, Considérations (toujours actuelles) sur l’état de la musique en temps réel », http://www.philippemanoury.com/?p=340, 1997

[Man03] Phillipe Manoury, Marc Battier, Alain Bonardi, Serge Lemouton. « Les musiques électroniques de Philippe Manoury », Ircam, Paris, 2003.

[Man12] Phillipe Manoury, *La musique du temps réel, Entretiens avec Omer Corlaix et Jean-Guillaume Ledrun*, Éditions MF, 2012.

[Mar09] Nicholas Mariette, « Ambigrainer - a Higher Order Ambisonic granulator in pd », Ambisonics symposium, 2009.

[MRM90] Gérard Marino, Jean-Michel Raczinski, Marie Hélène Serra, “The new upic system”, Proceedings of the 1990 International Computer Music Conference, p. 249-252, 1990.

[MiWa06] Eduardo Miranda et Marcelo Wanderley, “New Digital Musical Instruments: Control and Interaction Beyond the Keyboard”, *The computer music and digital audio series*, vol. 21, 2006, Middelton, Wiscounsin: A-R Editions, Inc.

[Nel15] Netttl, Bruno, “The Study of Ethnomusicology: Thirty-Three Discussions”, Urbana: University of Illinois Press, 2015.

[DGCNO16] P. Donat-Bouillud, J.-L. Giavitto, A. Cont, N. Schmidt, Y. Orlarey, “Embedding native audio-processing in a score following system with quasi sample accuracy”, in ICMC 2016, 42th International Computer Music Conference, 2016.

Y. Orlarey, D. Fober, S. Letz, “Syntactical and semantical aspects of Faust”, Soft Computing, vol. 8, n° 9, p. 623-632, 2004.

[Orl08] Yann Orlarey, Dominique Fober et Stephane Letz, “FAUST: an Efficient Functional Approach to DSP Programming”, *New Computational Paradigms for Computer Music*, T. 290, Delatour, 2008, p. 33 (cf. p. 34, 94).

[DAMONI03] J. Daniel, S. Moreau, R. Nicol, “Further investigations of high-order ambisonics and wavefield synthesis for holophonic sound imaging”, Audio Engineering Society Convention 114, Audio Engineering Society, 2003.

[GIA19] Jean-Louis Giavitto, Andrea Agostini, Bell, a textual language for the bach library. ICMC 2019 - International Computer Music Conference, Jun 2019, New York,

[GIA14] Jean-Louis Giavitto and José Echeveste, “Real-time matching of antescofo temporal patterns”, Proceedings of the 16th International Symposium on Principles and Practice of Declarative Programming, ACM, 2014, p. 93-104.

[Ech15] J. Echeveste, Un langage de programmation pour composer l’interaction musicale : la gestion du temps et des événements dans Antescofo, thèse de doctorat.

[GrPe92] T.R.G. Green, M. Petre, “When visual programs are harder to read than textual programs”, European Conference on Cognitive Ergonomics (ECCE), San Francisco, CA, USA, 1992.

[GRLE17] Florian Grond and Pierre Lecomte, “Higher Order Ambisonics for SuperCollider”, Linux audio conference 2017, proceedings, 2017.

[Mag19] Thor Magnusson, *Sonic writing: technologies of material, symbolic, and signal inscriptions*, Bloomsbury Academic, 2019.

[LEGA15] P. Lecomte and P.-A. Gauthier, “Real-Time 3D Ambisonics using Faust, Processing, Pure Data, And OSC”, 15th International Conference on Digital Audio Effects (DAFx-15), Trondheim, Norway, 2015.

[Lev13] Fabien Lévy, *Le compositeur, son oreille et ses machines à écrire*, Vrin 2013.

[Dan84] Roger B. Dannenberg, “An On-Line Algorithm for Real-Time Accompaniment”, Proceedings of International Computer Music Conference (ICMC), p. 193-198, Paris, France, 1984.

[Pee04] Geoffroy Peepters, “A Large Set of Audio Features for Sound Description”, CUIDADO project, IRCAM, 2004.

[Pot09] Laurent Pottier, Le calcul de la musique. Composition, modèles & outils, Centre interdisciplinaire d’études et de recherche sur l’expression contemporaine (CIEREC), Travaux 44, Collection « Musique et Musicologie », publications de l’université de Saint-Étienne, 2009.

[Pou06] M. Pouzet, Lucid Synchrone, version 3, Université Paris-Sud, LRI, 2006.

[Puc91] Miller Puckette, « Combining Event and Signal Processing in the MAX Graphical Programming Environment”, Proceedings of International Computer Music Conference (ICMC), vol. 15, p. 68-77, Montréal, Canada, 1991.

[Puc96] Miller Puckette, “Pure data”, in Proceedings, International Computer Music Conference (San Francisco, CA: International Computer Music Association), 224-227.

[Puc02] Miller Puckette, “Max at Seventeen”, *Computer Music Journal*, 26, 31-43, 2002.

[Pul97] Ville Pulkki, “Virtual Sound Source Positioning Using Vector Base Amplitude Panning”, *Journal of the Audio Engineering Society*, juin 1997, vol. 45/6, p. 456.

[Rey87] Craig W. Reynolds, “Flocks, herds and schools: A distributed behavioral model”, ACM SIGGRAPH Computer Graphics, vol. 21, n° 4, juillet 1987, p. 25-34.

[RiSc91] Rimé Bernard et Schiaratura Loris, “Gesture and speech” In: R. S. Feldman & B. Rimé (Eds), *Fundamentals of nonverbal behavior*, New York & Cambridge: Cambridge Univ. Press, 1991, p. 239-281.

[Ris80] Jean-Claude Risset, *Du songe au son, entretiens avec Matthieu Guillot*, L’Harmattan, 1980.

[Ris85] Jean-Claude Risset, « Le compositeur et ses machines », Editions Esprit, n° 99 (3), mars 1985, p. 59-76.

[Roa96] Curtis Roads, *The Computer Music Tutorial*, Cambridge, MIT Press, 1996.

[Roa01] Curtis Roads, *Microsound*, Cambridge, MA: MIT Press, 2001.

[Roa15] Curtis Roads, *Composing electronic music: a new aesthetic*, Oxford University Press, 2015.

[ROW17] Charles Roberts, Graham Wakefield, “Gibberwocky: New Live-Coding Instruments for Musical Performance”, NIME proceedings, 2017.

[Sch52] Pierre Schaeffer, *À la recherche d’une musique concrète*, Éd. du Seuil, Paris, 1952.

[Sch66] Pierre Schaeffer, *Traité des objets musicaux*, Éd. du Seuil, 1966.

[Sol98] Makis Solomos, « Notes sur la spatialisation de la musique et l’émergence du son », *Le son et l’espace*, sous la direction de Hugues Genevois et Yann Orlarey, Lyon, Aléas, 1998, p. 105-125.

[Sol07] Makis Solomos, *Espaces composables. Essais sur la musique et sur la pensée musicale d’Horacio Vaggione*, L’Harmattan, 2007.

[Sol13] Makis Solomos, *De la musique au son - L’émergence du son dans la musique des XX^e^-XXI^e^ siècles*, (Æsthetica) (French Edition), Presses universitaires de Rennes, 2013.

[Sti10] Bernard Stiegler, *For a new critique of political economy*, Cambridge Malden, MA: Polity, 2010.

[Sto17] Karlheinz Stockhausen, *Comment le temps passe. Essais sur la musique 1952-1961*, Éditions Contrechamps, 2017.

[Sch00] Diemo Schwarz, "A system for data-driven concatenative sound synthesis", DAFX00 Proceedings, Verona, 2000.

[Sch07] Diemo Schwarz, "Corpus-based concatenative synthesis", Signal Processing Magazine, IEEE, 24(2), 92-104, 2007.

[Vag95] Horacio Vaggione, « Esthétique et musique électroacoustique », Actes de l'Académie internationale de musique électroacoustique, vol. I. Bourges: Éditions Mnémosyne (1995), p. 101-108.

[Vag03] Horacio Vaggione, « Composition musicale et moyens informatiques : questions d'approche », in M. Solomos, A. Soulez, H. Vaggione (2003), p. 91-116.

[Vag08] Horacio Vaggione, « Composition musicale : représentations, granularités, émergences », Intellectica, 2008/1-2, 48-49, p. 155-174.

[Van17] Annette Vande Gorne, « Traité d'écriture sur support », *Lien, revue d'esthétique musicale*, 2017.

[Var83] Edagar Varèse (1954), *Écrits*, textes réunis et présentés par Louise Hirbour, Paris, Christian Bourgois, 1983.

[Var66] Edgard Varese and Chou Wen-chung, "The Liberation of Sound", *Perspectives of New Music*, vol. 5, n° 1 (Autumn-Winter, 1966), p. 11-19.

[Ver84] Barry Vercoe, "The Synthetic Performer in the Context of Live Performance", Proceedings of International Computer Music Conference (ICMC), p. 199-200, 1984.

[Whi04] Mitchell Whitelaw, *Metacreation: Art and Artificial life*, The MIT Press.

[WC03] Ge Wang et Perry R. Cook, "ChucK: A Concurrent, On-the-fly, Audio Programming Language", ICMC 2003 (cf. p. 23).

[Whi97] K. N. Whitley, "Visual programming languages and the empirical evidence for and against", *J. Vis. Lang. Comput*. 8 (1) (1997) 109-142.

[Whi97] K. N. Whitley, B.A.F., "Visual programming: the outlook from academia and industry", The Seventh Workshop on Empirical Studies of Programmers, Alexandria, VA, USA, 1997.

[Wil11] Scott Wilson, David Cottle and Nick Collins, *The SuperCollider Book*, The MIT Press, 2011.

[Xen81] Iannis Xenakis, *Musiques formelles*, Stock, 1981.

[Zat13] Laura Zattra, « Les origines du nom de RIM (Réalisateur en informatique musicale) », Actes des Journées d'informatique musicale (JIM 2013), Saint-Denis, 2013, p. 113-120.

[Zic98] D. Zicarelli, "An extensible real-time signal processing environment for Max", 1998.

[ZiPa01] Aymeric Zils, François Pachet, « Musical mosaicking", Proceedings of the COST G-6 Conference on Digital Audio Effects (DaFx-01), p. 39-44, Limerick, U.K. University of Limerick, (cited on p. 28.), 2001.

Glossaire

CNSMD Conservatoire national supérieur de musique et danse

CPU *Central processing Unit*, processeur central d'un ordinateur.

CAO Composition assistée par ordinateur.

DAW *Digital Audio Workstation*, Station audionumérique.

DMI *Digital Musical Instrument*, instrument de musique numérique.

DSP *Digital Signal Processor*, Processeur de signal numérique.

FAUST *Functional AUdio Stream*, langage de programmation fonctionnelle développé par le GRAME, dédié à l'implémentation d'algorithmes de traitement du signal.

GRAME Groupe de réalisation et de recherche appliquée en musique électroacoustique, Centre national de création musicale, créé à Lyon en 1982.

GRM Groupe de recherches musicales, Centre de recherche musicale dans le domaine du son et des musiques électroacoustiques fondé par Pierre Schaeffer en 1958 à Paris.

GUI *Graphical User Interface*.

HOA *Higth Order Ambisonic*, Ambisonie d'ordre supérieur, technique d'encodage et de reproduction sonore pour la spatialisation du son.

IA Intelligence artificielle.

Ircam Institut de recherche et coordination acoustique/musique, fondé par Pierre Boulez en 1970 à Paris.

ISPW *Ircam Signal Processing Workstation*. L'ISPW, développée en 1989 à l'Ircam, était une plateforme matérielle DSP faisant tourner un serveur de calcul audio temps réel, pilotable par le logiciel Max.

JSON *Java Script Object Notation*.

LFO *Low Frequency Oscillator*.

Open source Logiciel dont le code source est disponible et libre.

MAO Musique assistée par ordinateur, ensemble des pratiques utilisant l'informatique à des fins de composition et d'interprétation musicales.

Mapping Terme utilisé pour désigner la mise en relation de variables d'entrées (des données issues de capteurs) à des variables de sortie (des paramètres de synthèse audio).

MIDI *Musical Instrument Digital Interface*, protocole de communication et format de fichier créés en 1984, dédiés à la musique et utilisés pour la communication entre instruments électroniques, contrôleurs, séquenceurs et logiciels de musique.

MIR *Music Information Retrieval*.

MSD *Mass Spring Damping*, système de modèles physiques de contrôle développé par Nicolas Montgermont.

MSP *Max Signal Processing*, extension DSP, basée sur FTS, ajoutée à Max en 1997 et permettant le traitement du signal en temps réel directement dans Max.

MUSIC-N Fait référence à une famille de programmes de musique par ordinateur et de langages de programmation issus ou influencés par le programme MUSIC, écrit par Max Mathews en 1957 aux Bell Labs. MUSIC a été le premier programme informatique à générer des formes d'ondes audionumériques par synthèse directe.

NeXTSTEP Système d'exploitation pour le machines NeXT dans les années 1990.

NIME *New Interfaces for Musical Expression*, Nouvelles interfaces pour l'expression musicale, désignant aussi bien ces interfaces que la conférence éponyme qui leur est consacrée.

OS *Operating System* (système d'exploitation).

OSC *Open Sound Control*, format de transmission de données entre ordinateurs, synthétiseurs, robots ou tout autre matériel ou logiciel compatible, conçu pour le contrôle en temps réel. Il utilise le réseau au travers des protocoles UDP ou TCP et apporte des améliorations en termes de rapidité et flexibilité par rapport à l'ancienne norme MIDI.

PMPD *Physical Modelling for pd,* collection d'objets PD pour faire un système masse-ressort, développé par Cyril Henry.

RIM Réalisateur en informatique musicale, personne chargée d'aider un compositeur dans la partie technique, en particulier informatique, d'une œuvre musicale comprenant une part d'électronique.

SAT Société des arts technologiques, organisme culturel montréalais où se trouve la Satosphère, salle de concert en dôme hémisphérique en 360°.

SONVS Ancien département de composition au CNSMD de Lyon.

TENOR Conférence international sur les technologies de la notation et de la représentation musicale, conférence annuelle créée en 2015 à la suite d'un groupe de travail de l'AFIM sur la question de la notation musicale.

UPIC Unité polyagogique informatique du CEMAMu.

VBAP *Vector Base Amplitude Panning*, est une méthode créée par Ville Pulkki pour placer dans l'espace des sources virtuelles.

WFS *Wave Field Synthesis*, technique de rendu audio spatial basée sur la production de fronts d'ondes artificiels, synthétisés par un grand nombre de haut-parleurs à commande individuelle. De tels fronts d'ondes semblent provenir d'un point de départ virtuel, la source virtuelle ou source fictive. Contrairement aux techniques traditionnelles de spatialisation telles que le son stéréo ou *surround*, la localisation des sources virtuelles dans WFS ne dépend pas de la position de l'auditeur et ne change pas avec elle.