UNIVERSITE PARIS VIII

VINCENNES - ST-DENIS

MASTER II

MENTION : MUSIQUE

SPECIALITE : MUSICOLOGIE, CREATION, MUSIQUE ET SOCIETE

PARCOURS : THEORIE ET RECHERCHE

DIRECTRICE : ANNE SEDES.

Des interfaces pour la mise en espace du son avec la bibliothèque HOA.

PARIS ELIOTT

Année 2012-2013

TABLE DES MATIÈRES

Je tiens à remercier l'ensemble de l'équipe du CICM, Anne Sèdes, ma directrice de mémoire, Alain Bonardi, ainsi que mes précieux collaborateurs au sein du projet HOA, Julien Colafrancesco et Pierre Guillot, avec qui j'ai pris beaucoup de plaisir à travailler et sans qui aucune des réalisations présentées ici n'auraient pu naître.

Je remercie aussi la plateforme Arts Sciences Technologies et la MSH Paris-Nord qui nous ont accueillis dans leurs locaux une année durant pour mener à bien ce projet.

Introduction

La bibliothèque HOA (*High Order Ambisonics*[1]) est un ensemble de codes, ayant permis l'élaboration de moteurs sonores et d'interfaces graphiques dédiés à la mise en espace du son aux moyens des techniques ambisoniques d'ordre supérieur. L'ambisonie, se basant sur une représentation physique du champ sonore en harmoniques sphériques ou circulaires, est une technique de spatialisation particulière qui peut sembler, aux premiers abords, assez complexe à prendre en main pour un musicien. Ces techniques constituent néanmoins des vecteurs évidents de création musicale qui laissent envisager de nombreuses opérations relatives à la restitution, la synthèse, ou la transformation de champs sonores. Par l'intermédiaire d'interfaces graphiques d'interaction et de représentation originales et adaptées au cadre de l'ambisonie, la bibliothèque HOA propose donc aux musiciens et aux compositeurs de s'approprier ces techniques grâce à une approche renouvelée de la spatialisation ; permettant la manipulation, la mise en place et la représentation de traitements de l'espace et du son tels que l'encodage de sources sonores ponctuelles, la distorsion de la perspective [Daniel, 2001] ou encore le filtrage spatial.

Cette bibliothèque est issue de deux projets de recherche entrepris par le CICM[2] en 2012 et 2013 dans le cadre du LABEX Arts H2H de l'Université Paris-VIII auxquels nous avons pris part, d'abord comme stagiaire puis en tant qu'ingénieur d'étude. Ces projets, intitulés « *La spatialisation du son par les musiciens, pour les musiciens* » et « *Des interfaces pour la mise en espace du son* », ont permis de réaliser plusieurs mises en œuvre[3] des traitements sonores et interfaces graphiques à destination d'environnements de

[1] Ambisonie d'ordre supérieur.

[2] Centre de recherche en informatique et Création Musicale, http://cicm.mshparisnord.org/.

[3] L'ensemble des mises en œuvres de la bibliothèque est téléchargeable sur le site du projet HOA : http://www.mshparisnord.fr/hoalibrary/.

programmation visuelle tels que MaxMSP[4] ou Pure Data[5], ou encore sous la forme de plugins VST[6].

Dans le domaine des musiques assistées par ordinateur, les interfaces sont destinées à créer un lien étroit entre l'utilisateur et les algorithmes de traitement du signal. Elles constituent donc en ce sens une couche indispensable au contrôle et à la représentation des processus de mise en espace et de transformation des sons. Selon Vinet, les interfaces peuvent être classées par degré croissant d'abstraction [Vinet, 1999]. Cette typologie sommaire, reprise aussi par Couturier dans sa thèse [Couturier, 2004], nous aide à mieux cibler les enjeux de développement des différentes interfaces pour la mise en espace du son avec la bibliothèque HOA. En premier lieu viennent se placer les *objets de commande virtuels*, c'est ce que nous appellerons dans ce cadre les interfaces de contrôle. Ces objets reposent sur des représentations graphiques d'objets physiques ou imaginaires sur lesquels une action de l'utilisateur se traduit par une réaction visuelle et sonore du système en temps réel. Ces interfaces serviront la mise en espace du son à travers des modes d'interaction spécifiques permettant la manipulation du champ sonore à l'aide d'opérations réalisées principalement aux moyens d'une souris et d'un clavier d'ordinateur. Le second type d'interface est ce que couturier appelle des *indicateurs visuels*, elle permettent d'obtenir des « informations sur le déroulement de l'action et sur l'état du système » [Couturier, 2004]. Elles seront dans notre cas destinées à la représentation de données principalement sonores. Les interfaces de contrôle et de représentation qui seront présentées ici s'appuieront sur une représentation égocentrée et bidimensionnelle de l'espace en adéquation avec le modèle ambisonique en deux dimensions et l'ensemble des moteurs sonores de la bibliothèque HOA. Enfin, le dernier type d'interfaces est lié aux concepts de *programmation visuelle* destinés à « représenter les procédures intervenant dans le calcul de fonctions complexes par assemblage de modules élémentaires » [Vinet, 1999].

[4] http://cycling74.com/.

[5] http://puredata.info/.

[6] http://www.steinberg.net/.

En nous appuyant principalement sur la mise en œuvre MaxMSP de la bibliothèque HOA nous tenterons dévoiler le potentiel créatif et musical des interfaces de contrôle, de manipulation et de représentation du champ sonore facilitant la prise en main et l'appropriation par les musiciens des traitements ambisoniques d'ordre supérieurs. Nous nous intéresserons en premier lieu au contrôle du champ sonore à travers une série d'interfaces dédiées à la manipulation de sources sonores ponctuelles, ainsi qu'aux traitements plus généraux sur le champ sonore permis par l'interopérabilité de la bibliothèque entre le domaine des harmoniques circulaires et des ondes planes. Nous interrogerons ensuite les capacités de représentation graphique du champ sonore qu'offre la bibliothèque dans ces différents domaines. Enfin nous conclurons sur les interfaces dédiées à la mise en place des traitements ambisoniques, palliant les limites qu'imposent actuellement l'approche modulaire que nous avons choisi d'adopter au sein des logiciels de programmation visuelle en proposant des améliorations de son ergonomie.

1 DES INTERFACES DE CONTROLE DU CHAMP SONORE

Le contrôle de la mise en espace du son peut revêtir de multiples formes. Pottier opère une segmentation du contrôle de la spatialisation en trois catégories [Pottier, 2012]. Le premier mode de contrôle est appelé *contrôle graphique*. Il correspondrait, selon lui, au positionnement des sources, au dessin de trajectoires. Le second mode est dit *algorithmique* dans la mesure où des « formules sont utilisées pour calculer les positions des sons dans l'espace ». Le dernier mode enfin, est un contrôle de type *gestuel*, « interactif, permettant par des gestes sur des capteurs de déplacer en temps réel les sons dans l'espace » [Pottier, 2012].

S'il existe bien des situations de mise en espace des sons où seul un type de contrôle intervient, ces trois modes semblent souvent s'interpénétrer, se retrouvant fréquemment liés les uns aux autres dans la pratique, pour augmenter et améliorer l'interaction entre l'utilisateur et la matière sonore ainsi mise en espace. Le cas d'une interface telle que la propose l'application *Lemur*[7] pour tablette *iPad*[8] est par exemple typique. Elle fait intervenir les trois modes de contrôle simultanément en proposant un contrôle de type gestuel par le toucher, graphique par la représentation visuelle, et algorithmique notamment grâce à l'intégration de modèles physiques. Ce type de contrôleur peut être en ce sens particulièrement adapté au contexte d'une installation sonore où le public est par exemple amené à interagir en temps réel avec un environnement sonore. Il peut en revanche se révéler moins approprié pour contrôler un processus sonore lors de l'élaboration d'une pièce musicale demandant un contrôle plus fin de l'espace et du temps. Il semble dès lors évident de dire que le mode de contrôle de la mise en espace du son diffère en fonction du contexte de création. Aussi, nous aborderons principalement ici ce contrôle sous l'angle des interfaces graphiques proposées par la bibliothèque HOA. Ces interfaces, contrairement à celles plus génériques que l'on trouve par exemple dans MaxMSP, sont entièrement dédiées au contrôle des moteurs sonores de la bibliothèque. Elles offrent ainsi des modes d'interaction spécifiques augmentant, en nombre et en qualité, les opérations réalisables grâce à des représentations symboliques des processus mis en œuvre.

Nous aurons aussi l'occasion d'évoquer d'autres stratégies de contrôle, comme le contrôle par *mapping* gestuel ou à partir de l'extraction de paramètres sonores, en nous appuyant sur

[7] http://liine.net/en/products/lemur/.

[8] http://www.apple.com/ipad/.

des installations sonores que nous avons eu loisir de présenter lors du projet HOA, ou encore grâce à des pièces musicales qui ont été écrites aux moyens des outils logiciels de la bibliothèque.

Nous nous intéresserons en premier lieu au contrôle de la spatialisation de sources sonores ponctuelles. Celle-ci nous amènera à évoquer les principes inhérents à certains moteurs sonores élémentaires de la bibliothèque HOA, et les possibles manières de les contrôler graphiquement mais aussi algorithmiquement. Il nous semblera alors important d'élargir ensuite le *contrôle graphique* à d'autres opérations plus générales sur le champ sonore, permises par le large éventail des techniques ambisoniques, ne faisant pas forcément appel à la vision d'une construction de l'espace strictement atomique. Ces stratégies seront discutées dans la seconde partie où nous étudierons des manipulations plus globales du champ sonore notamment grâce à des modules permettant une interopérabilité entre le domaine des harmoniques circulaires et le domaine des ondes planes. Cette représentation du champ sonore nous a permis de faire éclore des traitements de l'espace et du son au fort potentiel musical tels que le filtrage spatial ou encore la *distorsion de la perspective*. Ces nouvelles opérations musicales requerront donc de nouvelles stratégies de contrôle que nous tenterons de mettre à jour grâce à des propositions innovantes.

1.1 MISE EN ESPACE DE SOURCES SONORES PONCTUELLES

La spatialisation d'une source sonore ponctuelle est le fait de positionner celle-ci virtuellement dans l'espace et dans le temps, de manière à lui donner un caractère spatial s'apparentant à un point, ou à une trajectoire. C'est sans doute la manière la plus commune d'envisager la mise en espace des sons, et sûrement aussi le mode de spatialisation des plus utilisé par les musiciens et compositeurs. Il nous semble important ici de présenter dans un premier temps les différents moteurs sonores qui permettent ce type de mise en espace au sein de la bibliothèque HOA. Nous en décrirons leur fonctionnement en évoquant les principes de base relatifs à l'encodage ambisonique d'une source sonore ponctuelle. Ceci nous permettra de mettre à jour une première interface, permettant à la fois d'offrir une représentation et un contrôle d'une série d'opérations liées à l'encodage. Il conviendra alors de rappeler les adaptations que nous avons faites et les libertés que nous avons dû prendre face au modèle ambisonique ; dont les spécificités ne permettaient par exemple pas de prendre en compte directement de paramètre de distance, paramètre pourtant nécessaire à la mise en place de ce mode attendu de spatialisation. Nous présenterons donc un moteur sonore, *hoa.map~*, qui dans la bibliothèque HOA répond de façon pertinente à ces enjeux. Nous nous intéresserons ensuite aux différentes stratégies de contrôle de cette spatialisation. Nous verrons donc en détail une série d'interfaces dédiées à ce moteur sonore en particulier, qui offriront à l'utilisateur la possibilité d'opérer une spatialisation précise tant du point de vue spatial que temporel grâce à une représentation graphique et symbolique manipulable des sources sonores. Nous proposerons enfin dans la dernière partie, d'autres modes de contrôle de type graphique, basés cette fois-ci sur une assistance algorithmique en nous appuyant sur l'état de l'art en la matière ainsi que sur les quelques prototypes que nous avons développés en nous inspirant de certains modèles physiques existants. L'utilisateur ne va donc plus jouer sur la position spatiale des sources directement, mais sur un ensemble de variables de haut-niveau qui vont faire évoluer les positions de celles-ci de façon automatique.

1.1.1 CONTROLE DE L'ENCODAGE AMBISONIQUE DE SOURCES PONCTUELLES

Nous proposons ici au lecteur d'aborder l'encodage ambisonique de sources sonores ponctuelles à travers une première représentation graphique offerte par l'objet *hoa.control*. Cet objet est un outil principalement pédagogique, destiné à familiariser l'utilisateur aux principes de bases inhérents à une série de modules de la bibliothèque HOA en proposant un contrôle sommaire des variables y étant liées. En ambisonie, l'encodage est le fait de convertir un signal dans le domaine des harmoniques sphériques ou circulaires. Les différents modules de la bibliothèque HOA, se limitent pour l'instant à des traitements en deux dimensions, et sont donc amenés à manipuler une série de signaux dépendants d'harmoniques sous leur forme circulaire. Les harmoniques sont des fonctions spatiales qui permettent de représenter mathématiquement un espace[9]. Cette représentation de l'espace permet alors d'exercer une infinité de traitements ou d'opérations sur le champ sonore avant de le restituer sur un système de haut-parleurs grâce à une opération de décodage ambisonique.

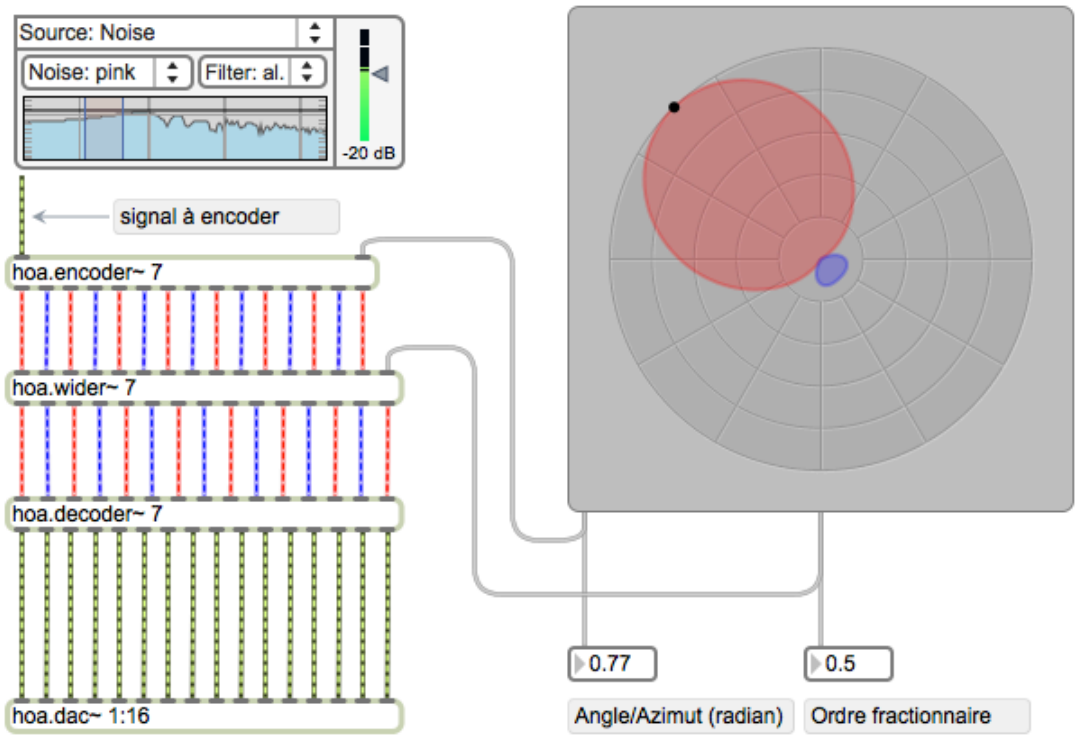

Figure 1 : patch MaxMSP représentant l'interface *hoa.control* à l'ordre 7, reliée aux modules d'encodage et d'ordre fractionnaire (de haut en bas).

[9] Nous revenons plus en détail sur cette représentation au chapitre consacré à la représentation des harmoniques circulaires.

L'objet *hoa.control* offre, à travers une vue bidimensionnelle de l'espace, une représentation graphique d'un champ sonore imaginaire sous la forme d'une somme pondérée d'harmoniques circulaires. Cette représentation permet de visualiser l'incidence d'une chaîne de traitement ambisonique correspondant à un encodage ambisonique d'un signal mesuré d'amplitude 1 pour un ordre de décomposition, une valeur d'azimut et de diminution de la résolution angulaire donnée. Cet objet autorise aussi en parallèle le contrôle des différents modules relatifs à chacune de ces opérations à travers une manipulation graphique de leur paramètre respectif [Figure 1]. Les lobes positifs, représentés en rouge sur cette interface, représentent la partie de l'espace ou le signal sera positif suite à la restitution du champ sonore sur un système de haut-parleurs ambisonique, les lobes bleus figurent l'inverse.

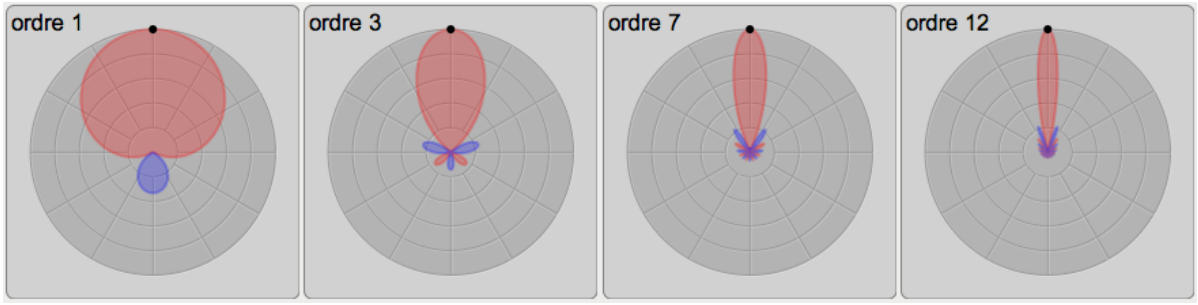

Figure 2 : représentation des harmoniques circulaires produites par l'encodage d'un signal 1, à 0 degré à l'ordre de décomposition ambisonique 1, 3, 7 et 12 (de gauche à droite).

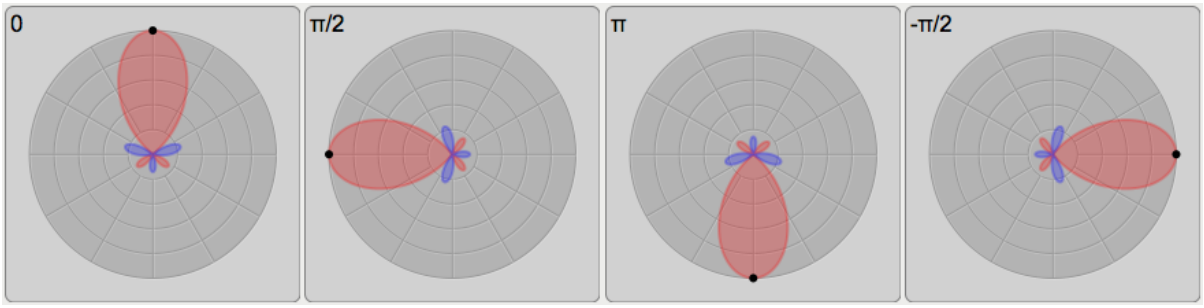

Figure 3 : représentation des harmoniques circulaires produites par l'encodage à l'ordre 3 d'un signal d'amplitude 1, aux angles suivants (exprimés en radians) : 0, π/2, π, et –π/2 (de gauche à droite).

Au sein de la bibliothèque HOA, l'encodage ambisonique d'une source sonore ponctuelle peut être réalisé grâce au moteur sonore *hoa.encoder~*. Cet objet prend comme seuls paramètres un ordre de décomposition ambisonique fixe et une valeur d'azimut variable. L'ordre de décomposition ambisonique va déterminer la résolution spatiale du champ sonore, c'est à dire avec quelle précision les sources sonores ponctuelles seront

perçues suite à sa restitution. La [Figure 2] représente quatre encodages de sources sonores pour des ordres de décomposition allant de 1 à 12. On perçoit alors assez nettement, à travers le resserrement du lobe principal, l'augmentation de la résolution angulaire de la source sonore virtuelle en fonction de l'ordre de décomposition ambisonique.

Le second paramètre propre à l'encodage ambisonique d'une source sonore ponctuelle est la variable d'azimut attendue en radian entre 0 et 2π. Notons que contrairement à la représentation mathématique, nous avons choisi, au sein des interfaces et des moteurs sonores de la bibliothèque HOA, de représenter le 0 face à l'auditeur et non pas à droite, ceci afin de correspondre à un usage plus répandu dans la pratique musicale. Le sens de rotation reste quant à lui, comme dans la représentation mathématique, antihoraire. La [Figure 3] expose une rotation antihoraire complète de la source sonore réalisée en variant le paramètre d'azimut de l'encodage, représenté par un point sur l'interface. Si cette variable est plus facilement maniable à l'aide d'une souris sur l'interface *hoa.control*, celle-ci peut tout aussi bien être gérée de façon textuelle en transmettant directement la valeur en radian à l'objet *hoa.encoder~*.

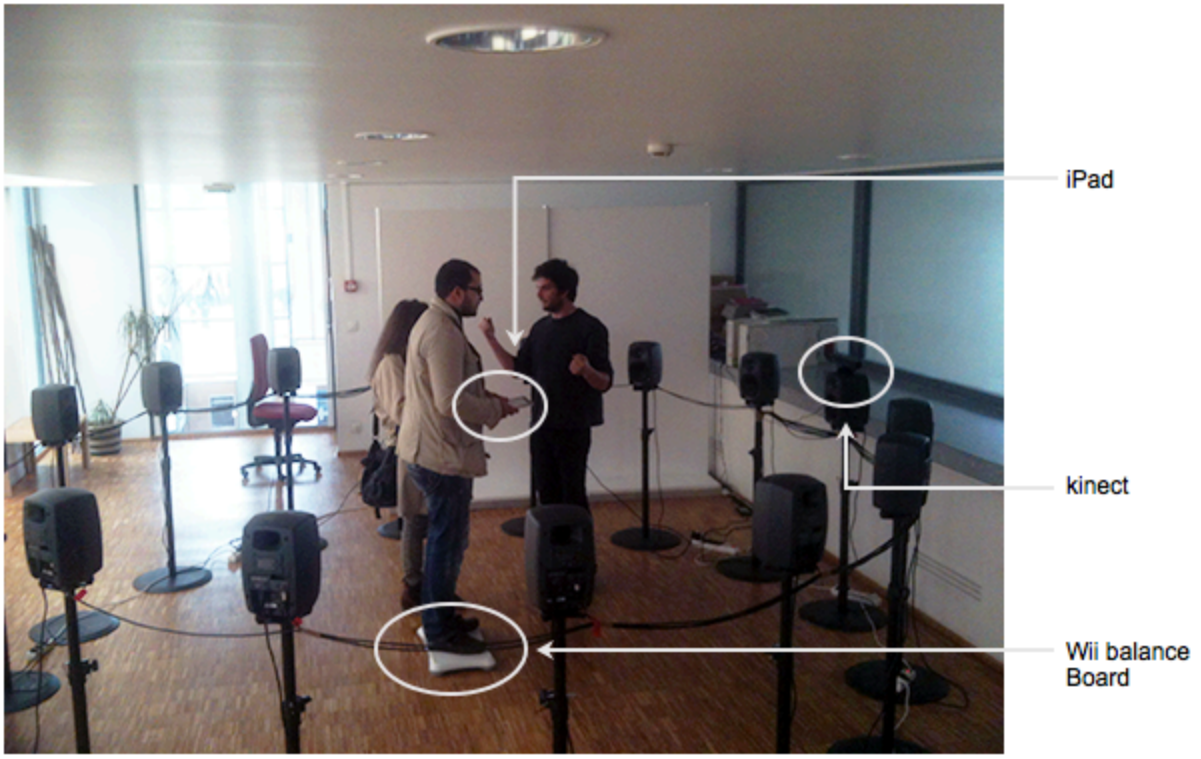

Figure 4 : photographie du dispositif de l'installation *Transduction* lors de l'événement *Savante banlieue 2012*.

La variation de l'angle d'incidence d'un encodage ambisonique représente déjà un premier traitement très intéressant d'un point de vue musical que nous avons pu exploiter, notamment à l'occasion de l'événement *savante banlieue 2012*[10]. L'installation *Transduction*[11] faisait intervenir, entre autre, un capteur gestuel de type *Wii balance Board*[12] qui agissait à la manière d'un *potentiomètre rotatif* pour l'utilisateur qui pouvait alors manier, à l'aide de variations de poids exercées sur l'objet, l'angle d'incidence d'une source sonore ponctuelle qui se déplaçait alors autour de lui [Figure 4].

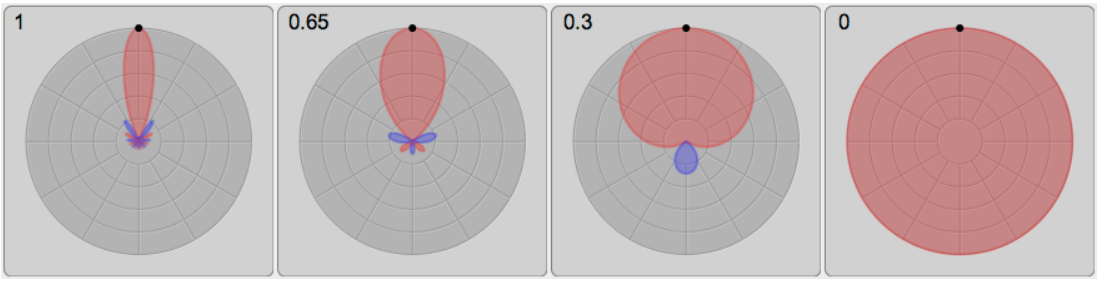

Figure 5 : représentation des harmoniques circulaires produites par l'encodage d'un signal d'amplitude 1, à 0 degré à l'ordre 7 avec des valeurs d'ordre fractionnaire successive de 1, 0.65, 0.3 et 0 (de gauche à droite).

Comme nous venons de le voir, l'encodage ambisonique d'une source sonore consiste à convertir un signal dans le domaine des harmoniques circulaires en donnant à la source sonore un angle d'incidence dont la résolution angulaire dépendra de l'ordre de décomposition ambisonique. Cet ordre de décomposition se définit au moment de l'encodage et ne peut pas être redéfini dynamiquement par la suite. Or, il nous paraissait tout à fait pertinent d'un point de vue musical de donner aux musiciens et aux compositeurs la possibilité de faire varier la résolution angulaire d'une source sonore suite à une opération d'encodage. Ce traitement se trouve donc implémenté au sein de l'objet *hoa.wider~* de la bibliothèque HOA. Grâce à un algorithme permettant la simulation d'un ordre de décomposition fractionnaire, cet objet permet désormais de moduler la résolution angulaire d'un champ sonore[13]. Pour autoriser une gestion plus aisée et musicale de cette opération, le paramètre permettant de la contrôler a été ramené entre 0 et 1. La valeur 1 correspond à la

[10] http://savantebanlieue.plainecommune.fr/.

[11] Cette installation sera évoquée à nouveaux lors des prochains chapitres.

[12] http://www.nintendo.fr/.

[13] Pour plus de précisions quant à la mise en œuvre de ce traitement, le lecteur pourra se référer à [Guillot, 2013].

résolution angulaire maximale du champ sonore tel que définie par l'ordre de décomposition lors de l'encodage. La valeur 0 correspond à un champ sonore dont la résolution serait nulle, omnidirectionnelle. L'utilisateur dispose donc d'un large paramètre de jeu entre ces deux extrêmes [Figure 5], contrôlable lui aussi à la souris à travers cette même interface.

Grâce à l'interface *hoa.control*, l'utilisateur peut donc désormais manipuler directement l'azimut de la source sonore à encoder [Figure 3] et faire varier dynamiquement sa résolution angulaire à l'aide de la souris [Figure 5] ; il peut aussi définir un ordre de décomposition [Figure 2] pour obtenir ainsi une prévisualisation du champ sonore généré par l'encodage ambisonique qu'il souhaite effectuer [Figure 1]. Cet objet, assimilable à un « potentiomètre rotatif augmenté », représente donc une première interface, particulièrement adaptée au contrôle des objets *hoa.encoder~*, *hoa.wider~*[14]. Notons aussi que cette interface, bien que très spécifique, est aussi généralisable au contrôle d'autres traitements musicaux prenant en variable un angle en radian[15]. Néanmoins, les possibilités strictement musicales offertes par cet outil restent encore quelque peu limitées. Cet objet seul ne permet par exemple pas de spatialiser plus d'une source à la fois. De plus, il manque aux moteurs sonores contrôlés la possibilité de gérer un paramètre de distance, qui se révèle être un paramètre indispensable à la spatialisation des sources sonores ponctuelles dans l'espace.

Nous avons effectivement constaté, en étudiant les usages de plusieurs musiciens et compositeurs, mais aussi en tenant compte de notre propre pratique compositionnelle, que la possibilité de placer et déplacer des sources librement dans l'espace à l'aide de simple coordonnées spatiales représentait une réelle attente de la part de ceux-ci. L'usage très fréquent de l'objet *ambipan~*[16], notamment dans le cadre de l'atelier de composition organisé chaque année par l'Université Paris-VIII[17], témoigne d'ailleurs de cet intérêt. Nous avons donc intégré, en réponse à cette demande et pour rester au plus proche des usages des

[14] Nous renvoyons le lecteur aux *patchs* d'aide respectifs de ces objets pour de plus amples informations sur leur fonctionnement, ainsi qu'à [Guillot, 2013] qui décrit avec plus de précision les principes mathématiques sous-jacents à ces traitements.

[15] Nous pensons notamment à une rotation exercée sur l'ensemble du champ sonore grâce au moteur sonore *hoa.rotate~*.

[16] Objet MaxMSP et Pure Data développé par Benoît Courribet et Rémi Mignot dans le cadre de l'ACI Jeunes Chercheurs 'Espaces Sonores'. En téléchargement sur http://cicm.mshparisnord.org/.

[17] Atelier de composition animé par le compositeur José Manuel Lopez Lopez.

musiciens et compositeurs, un moteur sonore conceptuellement et ergonomiquement proche de l'objet *ambipan~* à la série d'outils de spatialisation de la bibliothèque HOA.

La synthèse de source sonore ponctuelle comme nous avons pu le voir, dans le strict cadre ambisonique, est un traitement prenant comme paramètre un ordre de décomposition et comme variable un angle d'incidence que l'on souhaite donner à la source[18]. Pour donner à l'utilisateur la possibilité de gérer un paramètre de distance, nous avons dû sortir quelque peu du modèle acoustique pour privilégier une approche plus artistique. En s'inspirant de l'algorithme utilisé dans l'objet *ambipan~*, qui a largement été validé sur un plan artistique, nous avons donc développé le moteur sonore *hoa.map~*. Ce moteur sonore permet à l'utilisateur de placer virtuellement des sources dans un espace bidimensionnel à l'aide de coordonnées spatiales, en considérant que l'auditeur se trouve au centre (coordonnées {0,0}) d'un dispositif circulaire de haut-parleurs dont le rayon vaut arbitrairement 1.

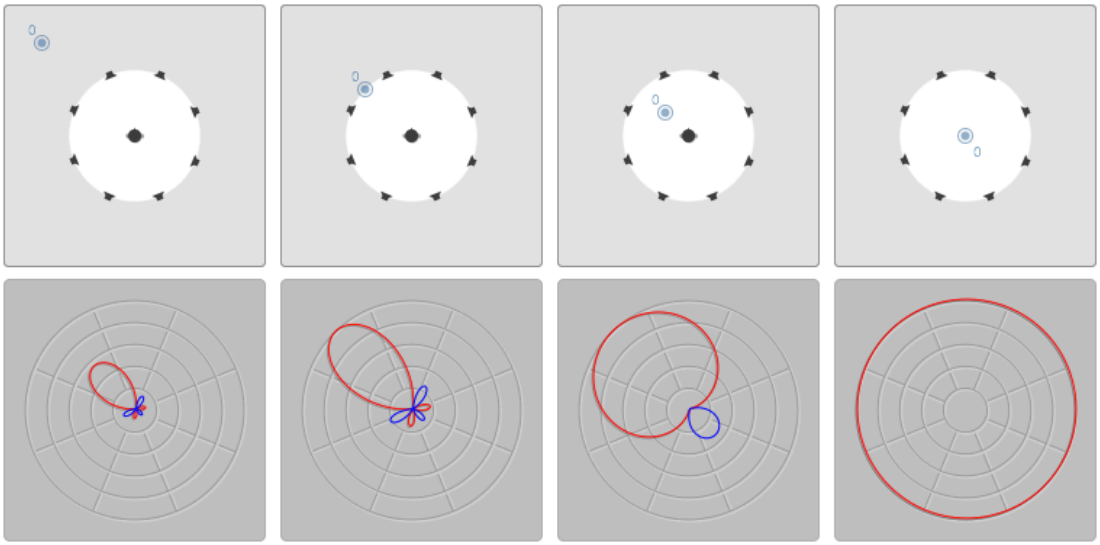

Figure 6 : sur la partie supérieure : représentation d'un auditeur placé au centre d'un dispositif de huit haut-parleurs et d'une source sonore virtuelle se rapprochant de celui-ci avec un angle d'incidence de π/4 (de gauche à droite) ; sur partie inférieure : représentation des harmoniques circulaires correspondant aux encodages successifs des sources sonores en fonction de la distance à l'ordre 3[19].

L'algorithme implémenté dans les deux moteurs sonores tire notamment parti du traitement relatif à l'ordre fractionnaire que nous avons évoqué ci-avant pour exercer une compensation

[18] Notons que l'encodage en trois dimensions prendra en plus de la variable d'ordre et d'azimut, un paramètre d'élévation de la source.

[19] Nous revenons sur l'objet *hoa.scope~* et la représentation des harmoniques circulaires dans la partie lui étant consacrée.

de la distance. Si la distance de la source par rapport à l'auditeur est supérieure au rayon du cercle de haut-parleurs [Figure 6.1], la distance est associée à l'amplitude de la source sonore. Ainsi, pour donner l'illusion que le son s'éloigne de l'auditeur, l'algorithme va diminuer l'intensité de la source sonore. Si la source se trouve sur le cercle [Figure 6.2], l'encodage se fait de façon « classique », suivant l'angle d'incidence et l'ordre de décomposition fixé. Quand la source sonore à encoder se trouve à l'intérieur du cercle de haut-parleurs, l'algorithme n'agit plus sur l'amplitude de la source mais sur sa résolution angulaire. Pour donner l'illusion que le son se rapproche de l'auditeur cette résolution décroît progressivement [Figure 6.3], jusqu'à rendre la source sonore omnidirectionnelle quand celle-ci se trouve exactement au centre du cercle [Figure 6.4].

Contrairement à l'objet *ambipan~*, l'objet *hoa.map~* permet de spatialiser une ou plusieurs sources sonores virtuelles sur un plan à l'aide d'un seul et même objet, à partir de coordonnées cartésiennes ou polaires ou encore de gérer l'état de *mute* de ces sources. L'objet *hoa.map~* par rapport à *ambipan~*, outre le fait qu'il ait été optimisé du point de vue de ses performances de calcul et augmenté de nouvelles fonctions, permet d'accéder à l'ambisonie d'ordre supérieur et donc d'obtenir une résolution angulaire potentiellement meilleure des sources sonores ainsi encodées. De plus, l'objet *hoa.map~* ne sort pas directement les signaux destinés aux haut-parleurs, mais des signaux dans le domaine des harmoniques circulaires, ce qui offre plus de flexibilité à l'utilisateur. Il offre ainsi la possibilité d'opérer par la suite des transformations sur l'ensemble du champ sonore généré par cet encodage ou encore de l'associer à un autre champ sonore synthétisé avant de restituer le tout au moment du décodage.

Nous venons de voir avec l'interface *hoa.control* qu'il devenait plus facile pour l'utilisateur de gérer un angle d'incidence ou de moduler la résolution angulaire d'une source sonore à l'aide d'une interface intermédiaire leur étant entièrement dédiés plutôt qu'en manipulant directement ces paramètres à partir d'une variable textuelle [Figure 1]. Nous verrons ici qu'il en est de même pour la gestion de coordonnées spatiales de sources sonores virtuelles au sein d'un plan.

1.1.1 CONTROLE GRAPHIQUE DE COORDONNEES SPATIALES

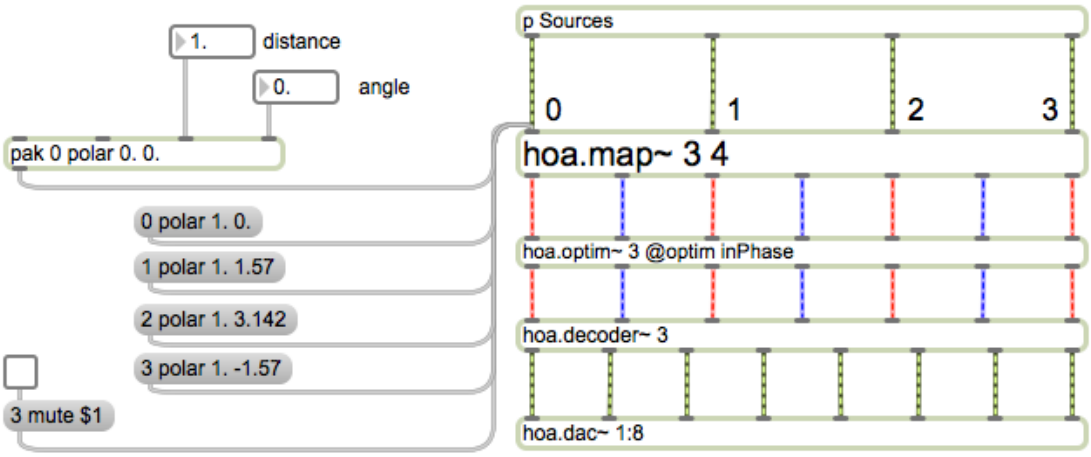

Figure 7 : le moteur sonore *hoa.map~* pour MaxMSP avec 4 sources en entrée, dont la position est contrôlée par l'intermédiaire de messages.

Le positionnement des sources sonores grâce à travers l'objet *hoa.map* nécessite d'être contrôlé grâce à des messages, (ou des signaux audio dans le cas particulier d'une source sonore unique). Pour spatialiser quatre sources sonores, comme c'est le cas ici, l'utilisateur doit donc transmettre à l'objet quatre couples de coordonnées spatiales précédés du type de coordonnées et de l'index de la source à spatialiser [Figure 7]; ce qui fait beaucoup de variables à gérer pour l'utilisateur. De plus, ces variables demeurant de type textuelles peuvent paraître quelque peu abstraites ou du moins peu maniables pour l'utilisateur. C'est pourquoi nous avons rapidement posé la nécessité d'avoir une interface dédiée à ce traitement, capable de gérer de façon transparente pour l'utilisateur l'envoi de ces messages en lui offrant une représentation graphique manipulable de la position de chaque source sonore virtuelle sur un plan.

Il existe déjà de nombreuses interfaces permettant la génération de couples de coordonnées par manipulation de points sur un plan. On pourra citer, entre autre, l'objet *ambimonitor*[20] de l'ICST [Schacher, 2010], [Schacher, Kocher, 2006], ou encore le *spat.viewer*[21] de l'IRCAM. Néanmoins, il s'est révélé pour nous indispensable de développer la nôtre pour ne pas ajouter de dépendances logicielles externes à la bibliothèque, tout en

[20] Institute for Computer Music and Sound Technology, Ambisonic library for MaxMSP : http://www.icst.net/research/downloads/ambisonics-externals-for-maxmsp/.

[21] Bibliothèque de spatialisation *Spat* de l'IRCAM pour MaxMSP : http://forumnet.ircam.fr/product/spat/.

ayant une interface qui puisse s'adapter parfaitement aux besoins spécifiques de ce moteur sonore.

Nous revenons brièvement sur les premiers prototypes développés dans le cadre du projet HOA avant de nous attarder plus en détail sur la dernière version de cette interface pour MaxMSP. Nous évoquerons enfin les mises en œuvre qui ont pu être effectuées à destination du logiciel Pure Data et sous la forme d'un plugin VST [Guillot et al., 2013].

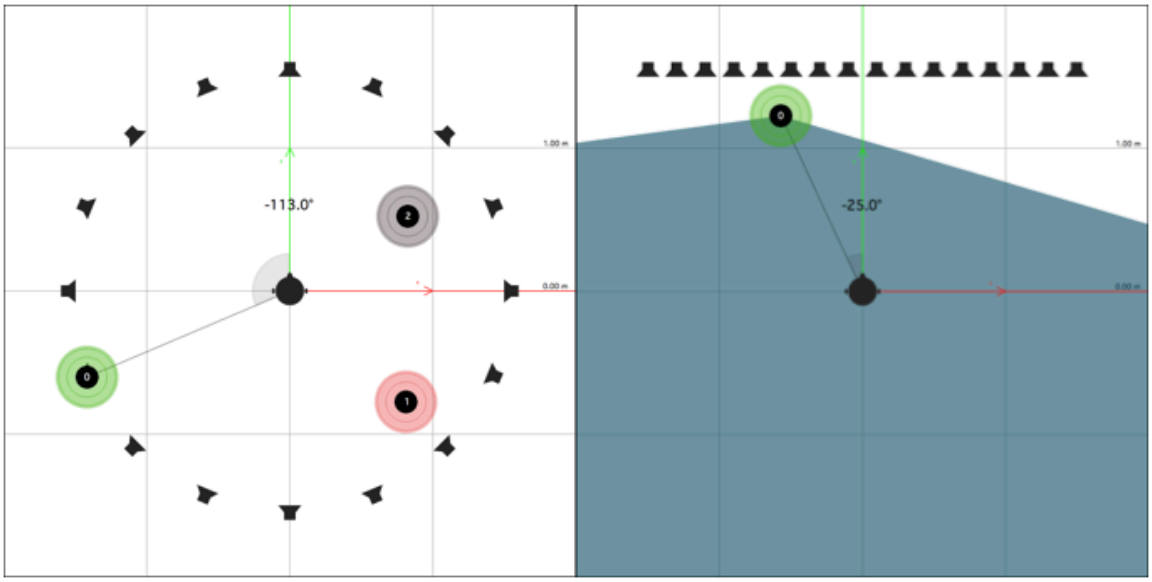

Figure 8 : premier prototype d'interface de spatialisation de sources sonores virtuelles sur un plan. Représentation d'un système de restitution de type ambisonique (à gauche), et *Wave Field Synthesis* (à droite) de 16 haut-parleurs. La partie foncée, sur la droite, représente la zone d'écoute.

La toute première interface à avoir été développée au sein du projet HOA [Figure 8], devait pouvoir représenter et donner à manipuler des sources sonores sur un espace à deux dimensions pour des systèmes de restitution sur haut-parleurs divers comme le système ambisonique ou WFS[22]. Ce premier prototype, développé en JavaScript, concluant sur beaucoup d'aspects (grille, zoom, placement des sources), mais trop limitatif et lent, a dû ensuite être revisité lors de sa première intégration à la bibliothèque HOA.

[22] Nous étudiions en effet à l'époque des techniques de spatialisation diverses et ne nous étions pas encore arrêté sur l'ambisonie.

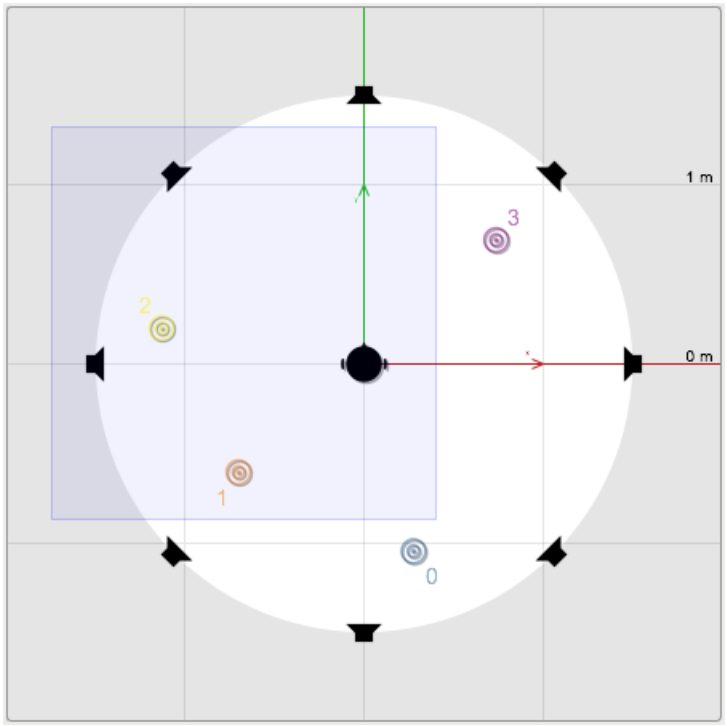

Figure 9 : deuxième version de l'interface *hoa.map* pour MaxMSP représentant quatre sources sonores virtuelles placées arbitrairement à l'intérieur d'un cercle formé par 8 haut-parleurs. En bleu, un rectangle de sélection multiple de sources.

Cette seconde version de l'interface [Figure 9], cette fois-ci développée en langage C, est une version améliorée du prototype présenté ci-dessus. On retrouve le repère orthonormé avec la même échelle et grille. En revanche, la WFS ayant été écartée des champs de recherches du projet[23], seule la représentation du système de restitution ambisonique a été retenue. Le portage du code en langage C nous a permis de tirer parti de l'API[24] de MaxMSP, notamment en terme d'intégration, de personnalisation de l'interface (couleurs, modes), ou de sauvegarde des paramètres au sein du *patch*. De plus, un certain nombre d'opérations ont pu être rendues plus facilement réalisables par l'utilisateur. Celui-ci peut désormais sélectionner plusieurs sources pour les déplacer ensemble ou contraindre le mouvement des sources à une variation d'angle ou de distance en maintenant enfoncées certaines touches du clavier ; ces opérations sont explicitées plus-après.

[23] Le système de restitution de seize canaux ne suffisait pas à donner des résultats assez probants sur le plan musical.

[24] *Application Programming Interface.*

Nous revenons à présent sur la dernière version de cette interface pour MaxMSP. Celle-ci s'inspire des deux précédents prototypes évoqués en y apportant de nouvelles améliorations.

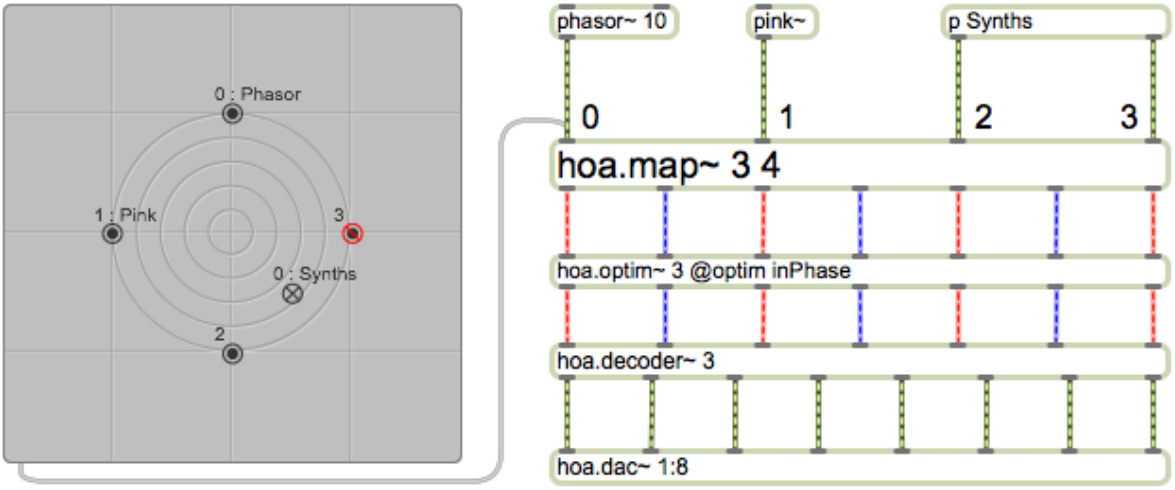

Figure 10 : patch MaxMSP dans lequel l'interface graphique *hoa.map* est reliée directement au moteur sonore *hoa.map~*. Représentation de quatre sources sonores virtuelles à une distance de 1 par rapport au centre et des angles respectifs de 0, π/2, π, -π/2 (de la source 0 à 3). La source d'index 3 indique un état de *mute* positif. La croix entourée d'un cercle indique un groupe auquel appartiennent les sources 2 et 3.

Au sein de cette nouvelle interface nous avons cherché à mettre un accent particulier sur son utilisabilité et son ergonomie, c'est-à-dire à faire en sorte que l'utilisateur ait accès à un maximum d'opérations musicales de la manière la plus facile et confortable qui soit. Ainsi, il n'y a désormais plus d'intermédiaire entre l'interface graphique et le moteur sonore qu'il contrôle [Figure 10], les deux objets se relient directement. Cette nouvelle interface, par rapport aux deux précédentes a aussi été largement épurée. Le nombre de haut-parleurs n'est plus représenté, le système de restitution ambisonique étant désormais simplement suggéré par des cercles concentriques placés au centre de l'interface. Le cercle de plus grand rayon correspond au cercle de haut-parleurs ambisonique, situé à une distance fixée à 1 par rapport à l'auditeur situé au centre, faisant ainsi ressortir la corrélation directe avec l'algorithme du moteur sonore auquel cette interface est associée. La grille, quant à elle, ne dépend plus d'une échelle en mètres comme c'était le cas précédemment, mais est relative à la distance du cercle de haut-parleurs exprimée en valeur abstraite ; tout comme le cercle, elle s'adapte en fonction du niveau de zoom variable grâce au *scroll* de la souris.

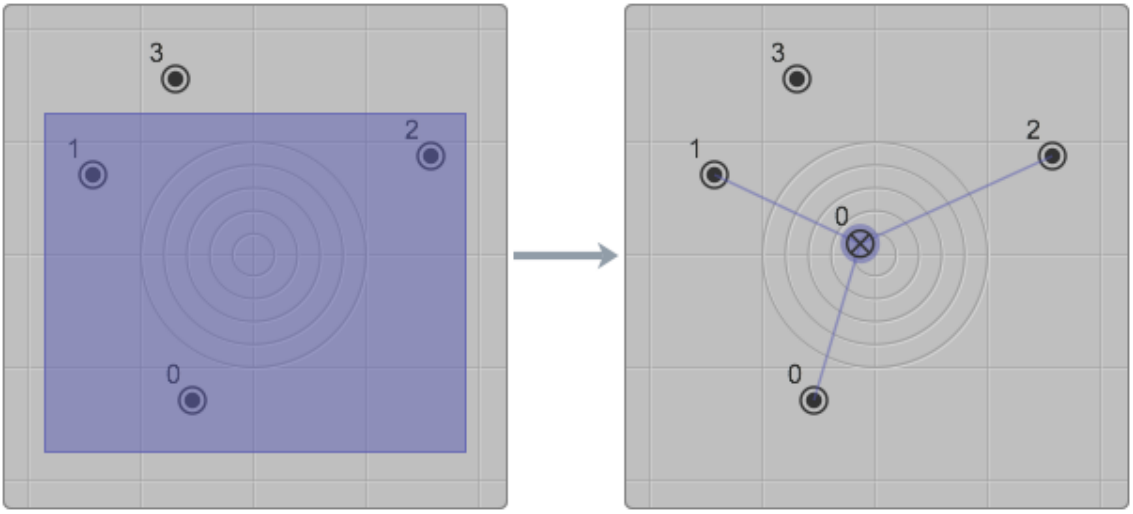

Figure 11 : représentation d'une création de groupe par sélection manuelle d'un ensemble de source.

Développé en langage C++, cet objet a pu étendre largement le principe de groupe et de source, permettant notamment à l'utilisateur de réaliser facilement des opérations de sélection ou de déplacement. Celui-ci peut regrouper, à l'aide d'une sélection à la souris, un ensemble de sources spécifiques. Sur la [Figure 11] par exemple, un groupe à l'index 0, contenant les source d'index 0 à 2, a été crée par rectangle de sélection. Il était important aussi, lors de la conception d'une interface graphique, de répondre aux usages les plus courants afin de rendre la prise en main de l'interface d'autant plus intuitive. Ainsi, l'utilisateur peut aussi utiliser la combinaison clavier assez commune et intuitive en informatique [*cmd*+*a*] afin de créer un groupe contenant automatiquement toutes les sources présentes sur la scène.

Les groupes se distinguent des sources en étant représentés graphiquement par une croix entourée d'un cercle située à l'emplacement du barycentre, calculé à partir de la position des sources qu'il contient. Des lignes apparaissent au survol d'un groupe ou d'une source avec la souris, reliant les sources aux groupes auxquels elles appartiennent afin de donner une indication à l'utilisateur sur la manipulation qu'il est sur le point d'effectuer. Tout comme les sources, les groupes sont sauvegardés avec le *patch*. Ces groupes, une fois créés, peuvent être manipulés à l'aide de la souris à partir de leur point d'ancrage directement, sans avoir à sélectionner à nouveau l'ensemble des sources ; comme on peut le voir à travers les figures qui suivent :

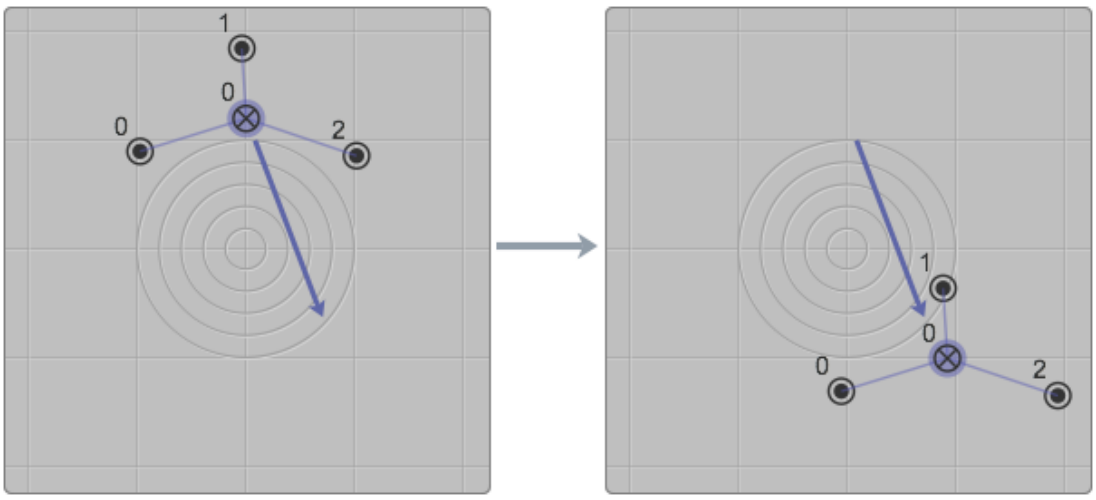

Figure 12 : représentation d'un déplacement par groupe libre.

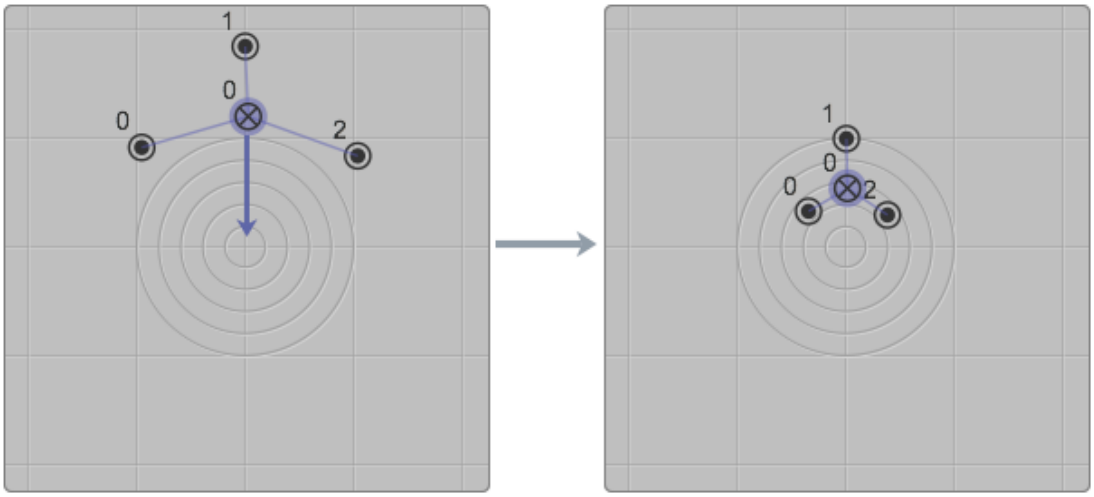

Figure 13 : représentation d'un déplacement par groupe contraint à la distance.

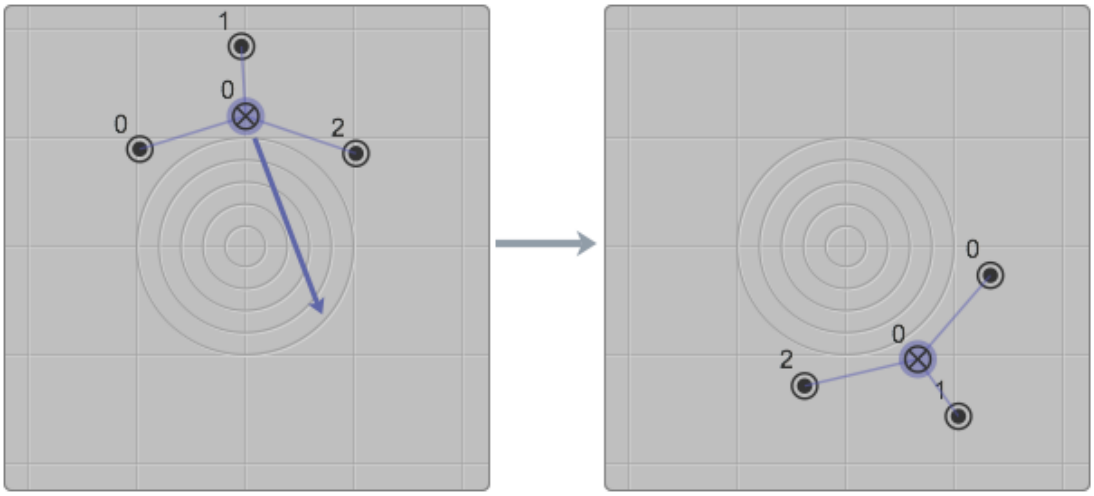

Figure 14 : représentation d'un déplacement par groupe contraint au rayon.

Cette interface graphique se révèle très intuitive et particulièrement adaptée dans le contexte d'une utilisation avec la souris ou le *trackpad* d'un ordinateur personnel, ces périphériques d'entrée fonctionnant eux aussi selon un modèle de coordonnées de type x/y. Dans une interface traditionnelle de type x/y (par exemple l'objet *pictslider* de MaxMSP), deux données sont contrôlables en même temps avec la souris (couple de coordonnées). Au sein de l'objet *hoa.map*, l'utilisateur peut désormais en contrôler une infinité grâce aux groupes. En plus du simple déplacement de sources à l'aide de groupes, qui s'apparente à une opération de translation [Figure 12], un certain nombre d'opérations de spatialisation additionnelles sont proposées à l'utilisateur comme le fait de pouvoir contraindre le déplacement d'une ou plusieurs sources à la distance par rapport au centre [Figure 13], ou d'exercer une rotation sur un ensemble de sources [Figure 14]. On peut imaginer rajouter, dans une version future de l'interface, d'autres types d'opérations qui pourront se révéler utiles et pertinente sur le plan musical comme le fait de pouvoir contraindre les sources à un mouvement horizontal ou vertical, ou encore les mêmes opérations mais à partir d'un axe ou un point de symétrie différent du centre.

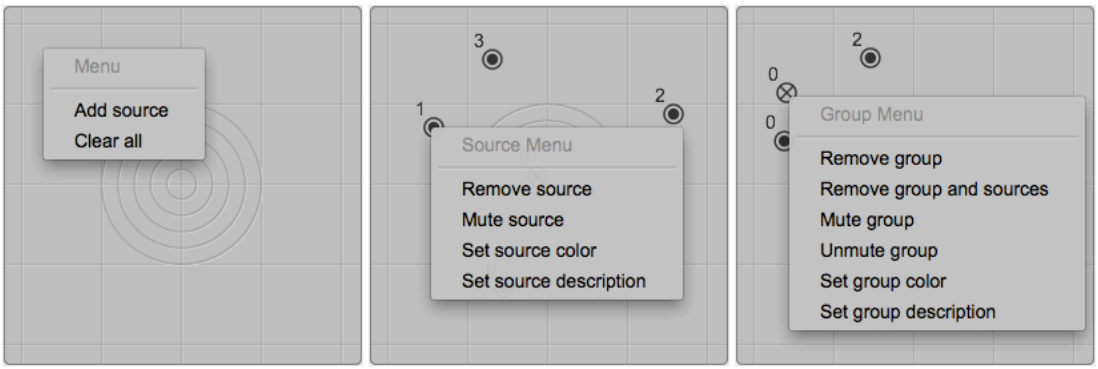

Figure 15 : représentation du menu contextuel apparaissant en fonction de l'endroit ou clique l'utilisateur. De gauche à droite : sur une zone libre, sur une source, sur un groupe.

Toutes les actions sur les données de l'interface peuvent être gérées grâce à des messages de type textuels que l'on peut lui transmettre. Néanmoins, pour améliorer d'autant plus son ergonomie, nous avons fait le choix de permettre à l'utilisateur de pouvoir gérer un maximum de ces actions ou opérations directement sur l'interface. Un menu à choix multiple apparaît au clic-droit de l'utilisateur sur l'interface. Ce menu est contextuel dans la mesure où il dépend de l'endroit sur lequel on aura cliqué pour proposer des actions différentes selon chaque cas [Figure 15]. L'utilisateur peut donc ajouter ou retirer de la scène des sources

sonores ou des groupes de sources, gérer l'état de *mute* d'une ou plusieurs sources ou encore changer la description et la couleur des groupes et des sources, tout cela très facilement.

Une autre fonction que les musiciens et compositeurs recherchent fréquemment dans une interface graphique est la possibilité de sauver et d'automatiser les paramètres qu'il manipule dans le temps. Au sein de l'environnement de programmation MaxMSP, cette fonction peut être remplie par la suite d'objets *pattr*[25]. L'objet *pattrstorage* permet de stocker plusieurs états de paramètres contenus dans un patch, il est alors possible de rappeler ou de faire des interpolations entre plusieurs états. Pour rendre l'objet graphique compatible avec ce système, le développeur doit fournir deux fonctions de type *getter/setter*, la première sera appelée lorsque le *pattrstorage* cherchera à connaître la valeur courante d'un paramètre, la seconde lorsque notre objet reçoit une nouvelle valeur pour ce paramètre. L'interpolation entre deux valeurs de paramètre est alors entièrement gérée par le *pattstorage* lui-même. Nous avons pu rendre compatible beaucoup d'objets graphiques de la bibliothèque HOA avec ce système (*hoa.space*, *hoa.recomposer*, *hoa.gain~*)[26], mais ceci n'a malheureusement pas été possible dans le cas de l'objet *hoa.map*. Les données à manipuler étaient en effet à la fois trop complexes et variables pour être laissées au soin d'un objet externe. Pour pallier cela et offrir aux musiciens et compositeurs un outil réellement puissant et complet sur le plan musical, nous avons fait le choix d'intégrer à notre interface notre propre version du *patrstorage* en recréant un maximum de fonctions similaires. Afin d'en faciliter la prise en main, ces fonctions sont accessibles grâce à des messages formatés de la même façon que pour le *pattrsorage*, l'utilisateur n'a donc pas à apprendre de nouvelle syntaxe. Comme dans le cas d'une utilisation traditionnelle de l'objet *pattrstorage*, cet outil offre donc la possibilité de stocker plusieurs états (nombre de sources, positions des sources et des groupes, états de *mute*, couleurs, descriptions), pour les rappeler ensuite ou pour créer des interpolations entre plusieurs états[27]. Un autre mode que nous avons pu mettre en place aussi, grâce à l'élaboration de ce système, est l'enregistrement et la relecture de trajectoires. L'utilisateur peut alors sauver une série d'états successifs en bougeant une ou plusieurs sources pour créer des trajectoires, il peut ensuite réenregistrer par dessus, poursuivre une trajectoire

[25] La suite d'objets *pattr* est disponible uniquement au sein du logiciel MaxMSP (*pattr, pattrstorage, autopattr…*).

[26] Nous revenons plus en détail sur ces implémentations dans les parties consacrées à chaque objet spécifique.

[27] Le lecteur peut se référer à l'aide de l'objet *hoa.map* pour obtenir les détails de son fonctionnement.

commencée, et relire le tout ensuite. Les états, comme les trajectoires, peuvent être sauvées sur le disque dur au format JSON[28] et chargés dynamiquement par la suite.

L'interface *hoa.map* dans sa version actuelle est particulièrement complète et aboutie. Elle permet d'offrir aux musiciens et compositeurs, comme nous avons pu le voir, de nombreuses opérations musicales, facilement accessibles grâce à des fonctions et des modes d'interaction bien pensés. Sa qualité et sa fiabilité ont d'ailleurs pu déjà être mises à l'épreuve au sein de plusieurs créations musicales. Le compositeur espagnol German Alonso, a par exemple pu retravailler sa pièce *ecce Saturnus (anti-Cronia)*[29], pour saxophone et électronique en temps réel, composée initialement avec la bibliothèque de spatialisation *spat* de l'IRCAM, en s'appropriant le moteur sonore *hoa.map~* et son interface graphique associée très facilement et rapidement. Plusieurs améliorations peuvent néanmoins être envisagées ; comme une meilleure gestion du temps au sein des trajectoires (enregistrement de trajectoires indépendantes et gestion d'un temps absolu ou relatif) ; gestion d'un paramètre de volume des sources qui pourrait passer par une représentation graphique de la présence de la source (grâce à la composante alpha ou la taille de la source) ; enfin, une gestion d'un historique des opérations effectuées sur l'interface afin de pouvoir annuler ou rétablir une action (ex : cmd+z/cmd+y).

Nous présentons à présent brièvement les mises en œuvre que nous avons pu effectuer à destination du logiciel Pure Data et sous la forme d'un plugin VST. Le fait de proposer ces interfaces pour différents environnements logiciels a principalement pour but d'étendre la communauté d'utilisateurs de la bibliothèque HOA en satisfaisant des usages et besoins toujours plus diversifiés qui viendrons, nous l'espérons, alimenter de nouvelles approches de l'espace et du son [Guillot et al., 2013].

La mise en œuvre des interfaces pour Pure Data est encore assez récente mais permet tout de même déjà de donner accès à la majeure partie des fonctions présentes dans la dernière version de l'objet *hoa.map* pour MaxMSP. On retrouve la possibilité d'ajouter

[28] JavaScript Object Notation.

[29] Cursus 1 de composition, IRCAM, 2013.

dynamiquement des sources au sein de l'interface, de créer des groupes, de déplacer une ou plusieurs sources librement ou contraintes à la distance ou à l'angle, la sauvegarde des positions des sources au sein du patch, ou encore la gestion de la sauvegarde et de la restauration d'états et de trajectoires.

La spatialisation de sources virtuelles au sein des logiciels de type séquenceur est depuis longtemps disponible. En revanche si beaucoup de plugins existent et semblent offrir des fonctions similaires, peu d'entre eux permettent de faire de l'ambisonie d'ordre supérieur. Grâce à la mise en œuvre que nous avons pu effectuer, sous la forme d'un plugin VST, celle-ci est désormais accessible pour le compositeur.

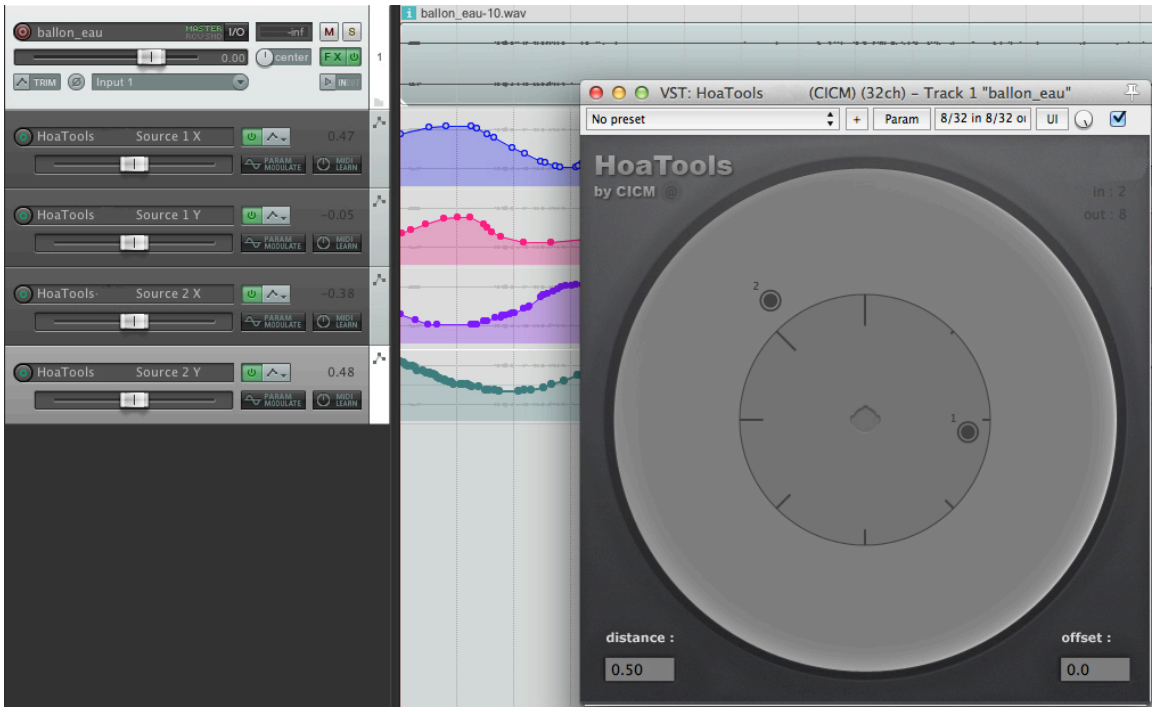

Figure 16 : le plugin *HoaMap* chargé au sein du logiciel hôte *reaper*. Représentation d'une spatialisation d'un son stéréophonique sur un système de restitution ambisonique à huit canaux ; automation des coordonnées cartésiennes des deux sources sonores virtuelles.

Ce plugin est une adaptation logicielle de la série d'objets MaxMSP présentée à la [Figure 10], il utilise les mêmes classes C++ qui ont servies à créer les objets *hoa.map~*, *hoa.optim~* et *hoa.decoder~*, la partie graphique de l'interface, quant à elle, a dû être en grande partie réécrite. Pour nous faciliter ce travail nous avons fait le choix d'utiliser la bibliothèque de

classes C++ JUCE[30], très fréquemment utilisée dans le domaine de la production de logiciels audio professionnels[31]. L'un des avantages majeurs de cet ensemble de classes est le fait de permettre le déploiement du logiciel développé, à partir d'un seul code source, sur plusieurs plateformes et sous différents formats (MacOs, Windows, Linux, VST, AudioUnit, RTAS..). De plus, on remarquera aussi beaucoup de ressemblance au niveau des fonctions de dessin entre cette bibliothèque et l'API de MaxMSP, ce qui a aussi contribué à nous faciliter le travail. Nous détaillons plus précisément les stratégies de mises en œuvre adoptées dans l'article [Guillot et al., 2013].

Le principal enjeu de développement de ce plugin était d'offrir à la communauté d'utilisateur de logiciels de type *time-line* un outil simple à prendre en main pour la mise en espace de sources sonores virtuelles au sein d'un espace à deux dimensions. Nous avons donc décidé de simplifier au maximum l'interface utilisateur. Le choix de l'ordre de décomposition ambisonique n'est par exemple pas un paramètre modifiable comme dans le cas de la mise en œuvre MaxMSP ou Pure Data, il est dépendant du nombre de sortie du plugin (ex : si le plugin a huit sorties, celui-ci fonctionnera avec un ordre de décomposition 3[32]). De la même façon, le nombre d'entrées du plugin va déterminer le nombre de sources à spatialiser ; et le nombre de sorties déterminera le nombre de haut-parleurs représenté.

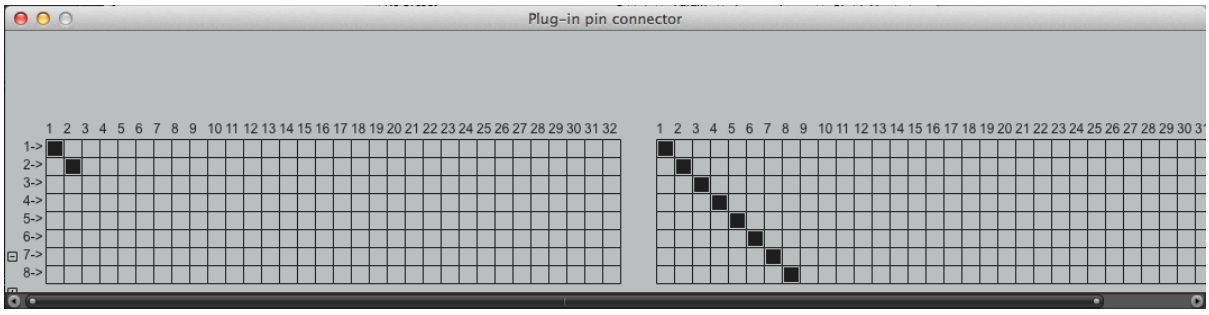

Figure 17 : représentation de la matrice de routing du plugin HoaMap utilisé à la Figure 16.

[30] http://www.juce.com/

[31] Le logiciel MaxMSP est par exemple, depuis la version 5, développé avec JUCE, ou encore la suite de plugins GRM-Tools depuis la version 3.

[32] Nous rappelons que dans le cadre d'une restitution ambisonique, le nombre de haut-parleurs minimum, en fonction d'un ordre de décomposition *n*, est égal à (2**n*+1).

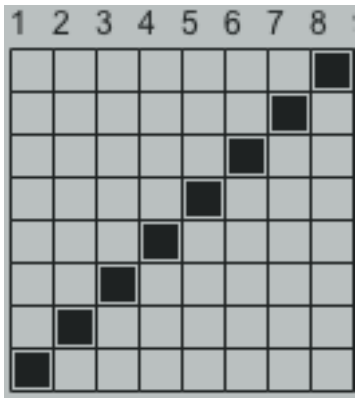

Figure 18 : représentation de la configuration d'une matrice de routing de sortie audio pouvant être utilisée pour inverser le sens de rotation des sorties du plugin HoaMap.

Au cours de notre développement nous avons eu l'occasion de tester ce plugin au sein de différentes DAW[33]. Malheureusement, nous nous sommes aperçus que beaucoup d'entre elles n'étaient pas particulièrement adaptées au traitement multicanal[34]. Nos tests se sont donc rapidement orientés vers le logiciel hôte *Reaper*[35] [Figure 15], qui semblait lui, très bien supporter les contraintes liées à l'ambisonie (pistes multicanales, *routing* audio configurable facilement). Au sein de ce logiciel hôte, l'entrée de chaque plugin possède une matrice de *routing* indépendante [Figure 16], il est donc très facile pour l'utilisateur de configurer celle-ci pour s'adapter au système de restitution qu'il souhaite utiliser. Nous sommes, pour notre part, habitués à numéroter les haut-parleurs dans le sens anti-horaire, si ce n'était pas le cas pour l'utilisateur, celui-ci pourrait alors simplement inverser la matrice de sortie pour faire marcher ce plugin avec sa configuration [Figure 17].

L'avantage de cette mise en œuvre réside aussi et surtout dans le fait de pouvoir tirer parti des facilités d'édition qui font l'une des caractéristiques des logiciels hôtes, notamment au niveau de la ligne temporelle. Nous noterons au passage que des opérations qui ne sont pas encore implémentées dans la version Pure Data ou MaxMSP comme l'historique des actions, peuvent être ici gérées par l'hôte. Tous les paramètres ayant été rendus automatisables, l'utilisateur peut dès lors enregistrer des trajectoires à l'aide de la souris puis les éditer, réécrire par dessus, les lire. Il peut de même contrôler des sources de manière algorithmique à l'aide d'un signal avec le système *side-chain* (ex : positionnement pseudo-aléatoire d'une source par gestion des coordonnées à partir de l'amplitude d'un bruit blanc ou un oscillateur).

[33] Digital Audio Workstation.

[34] ex : ableton Live, Adobe Audition.

[35] Cockos : http://www.cockos.com/reaper/.

Nous venons de voir que cet outil, disponible sous des formes différentes, dispose d'un fort potentiel musical, offrant au compositeur des opérations de mise en espace du son élaborées et facilement contrôlables. La spatialisation de sources sonores virtuelles peut alors être précise et maîtrisable autant du point de vue spatial que temporel. Néanmoins, il peut arriver aussi que, pour une création sonore particulière, le musicien ou le compositeur, ne veuille pas, ou n'ait tout simplement pas besoin, d'agir de façon aussi précise sur la position spatiale de chaque source sonore, ni même d'avoir à manipuler chaque sources sonores indépendamment. Au contraire, il peut chercher à contrôler de façon plus globale la spatialisation de la masse sonore produite par l'encodage d'un ensemble de sources, et non les trajectoires de chaque entité séparément. De même, il peut être parfois utile d'avoir à sa disposition un système de spatialisation intégrant une certaine autogestion, assurant ainsi un dynamisme inhérent au procédé de spatialisation, sans nécessiter d'action particulière de la part de l'utilisateur.

Nous tenterons donc dans la prochaine partie de trouver des réponses pertinentes à ces problématiques en nous appuyant sur l'état de l'art en matière de spatialisation algorithmique, puis nous dévoilerons quelques propositions logicielles qui sont faites dans la bibliothèque sous la forme de prototypes fonctionnels inspirés de modèles physiques existants.

1.1.2 INTERFACES GRAPHIQUES A ASSISTANCE ALGORITHMIQUE

Le contrôle algorithmique, à la différence d'un contrôle graphique, est le fait de placer des sources sonores dans l'espace par calcul, à l'aide de formules mathématiques ou de modèles physiques. Nous postulons ici que les deux modes de spatialisation peuvent aisément cohabiter au sein d'une même interface. Un modèle physique permet de simuler, de façon matérielle ou informatique, un phénomène ou un comportement qui se manifeste dans la nature. Les modèles physiques sont utilisés dans de multiples contextes autant scientifiques, industriels qu'artistiques (physique, mathématique, architecture, arts visuels..). En musique, la synthèse de sons par modélisation physique, par exemple, peut permettre d'imiter le comportement physique de l'onde sonore produite par l'excitation d'un instrument de musique avec un objet. Dans ce type de synthèse, l'utilisateur n'est pas amené à fournir des informations de bas niveau comme l'amplitude ou la fréquence des oscillateurs eux-mêmes (comme dans le cas d'une synthèse additive par exemple), mais plutôt de renseigner le modèle en fournissant des paramètres généraux, de plus haut niveaux, qui serviront à définir son comportement général (densité du matériau, dimensions de l'instrument..).

Ce qui nous intéresse ici est de voir dans quelle mesure la manipulation de paramètres de plus hauts niveaux que les simples coordonnées spatiales des sources, peut contribuer à enrichir ou renouveler l'approche de la mise en espace du son.

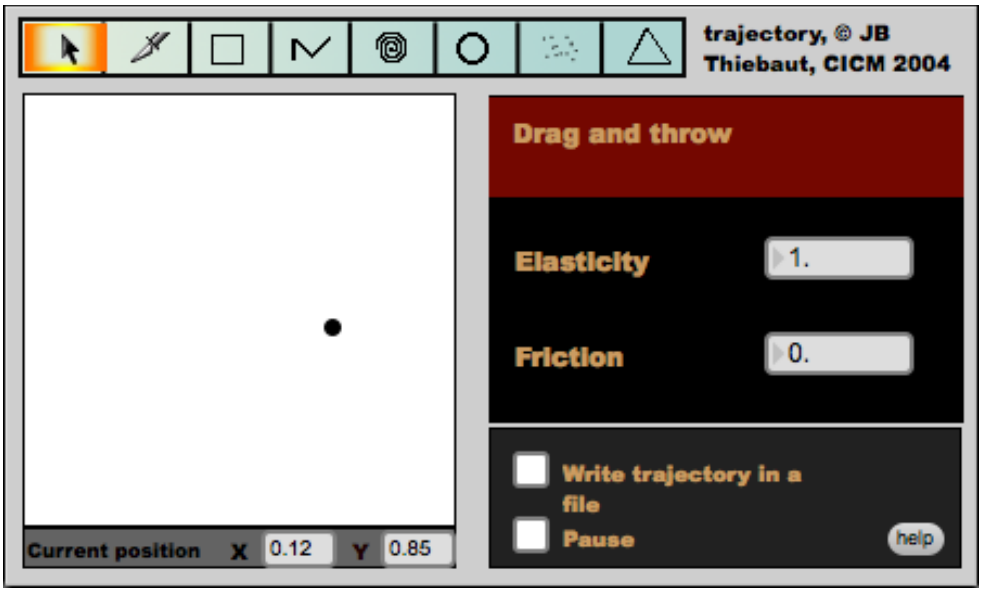

Figure 19 : représentation du mode *dragAndThrow* présent au sein du *patch trajectory* pour MaxMSP.

L'abstraction *trajectory*[36], disponible sous la forme d'un *bpatcher* au sein du logiciel MaxMSP [Figure 19], a pour principale vocation d'offrir à l'utilisateur la possibilité de dessiner des trajectoires géométriques pour spatialiser des sources sonores [Thiebaut, 2005]. Largement utilisé par les compositeurs ces dix dernières années, cet outil, destiné à se greffer à n'importe quel moteur sonore, met à la disposition de l'utilisateur une série de huit modules facilitant la création d'interpolations de coordonnées cartésiennes. Ces interpolations serviront notamment à créer des trajectoires spatiales qui pourront être interprétées par des moteurs sonores de spatialisation tels qu'*ambipan~* ou *hoa.map~*, ou encore à contrôler la position d'une source sonore individuelle au sein d'une autre interface de spatialisation telle qu'*hoa.map*.

Au cours du processus de création d'interfaces pour les moteurs sonores de la bibliothèque HOA, nous avons choisi de nous éloigner quelque peu du modèle de conception de trajectoires par dessin de formes géométriques, cette approche n'ayant pas été validée du point de vue de la perception musicale. La majorité des modules présents au sein de l'interface *trajectory*, favorisant nettement cette approche, ne trouvent donc pas leur équivalent au sein des interfaces développées dans ce cadre. Il est cependant à noter que les fonctionnalités présentes dans les modules *pen* et *Broken Line* de l'outil *trajectory* peuvent être reproduites au sein de l'interface *hoa.map* par l'intermédiaire des *slots* et de la fonction *trajectory*.

L'aspect géométrique de cet outil mis de côté, l'intérêt que nous lui portons réside surtout en sa capacité à être un générateur de mouvement. Cette interface intègre en effet quelques principes de modèles physiques basiques tels que l'élasticité, le rebond, la friction ou l'accélération qui aident à donner un caractère dynamique à la spatialisation pertinent du point de vue de la création musicale. Le mode *dragAndThrow*, illustré par la [Figure 19], permet par exemple de générer des coordonnées spatiales à partir de la simulation de la trajectoire effectuée par une balle lancée dans un espace clos, rebondissant contre les parois et décélérant à cause de la friction de l'air.

[36] Trajectory, Jean-baptiste thiebaut, CICM, 2004, http://cicm.mshparisnord.org/ rubrique téléchargement.

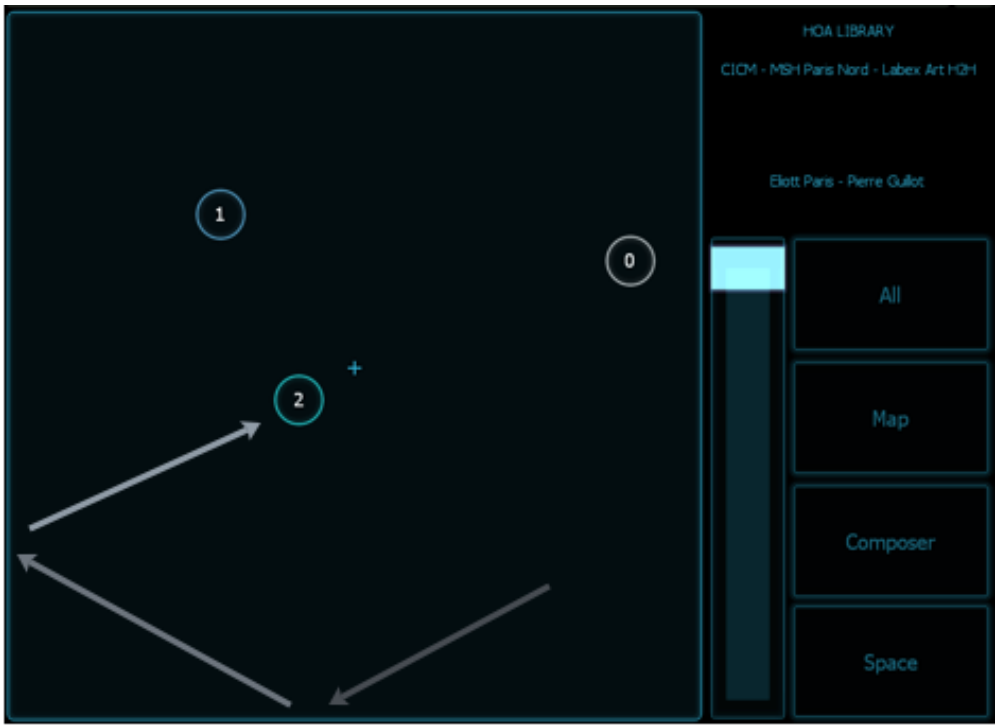

Figure 20 : représentation de la vue map de l'interface *lemur* pour *iPad* présentée au public lors de l'événement *Futur en Seine 2013*. Les balles symbolisent trois sources sonores distinctes au sein d'un espace bidimensionnel. Les flèches représentent le mouvement d'une source après que l'utilisateur l'ait relâchée.

On retrouve des comportements similaires dans beaucoup d'autres logiciels, et notamment au sein de l'application *Lemur*[37] pour *iPad*, que nous avons eu l'occasion d'utiliser lors de plusieurs installations. Lors de la présentation de la bibliothèque HOA dans le cadre de l'événement *Futur en Seine 2013*[38] et de l'installation *savante banlieue 2012*, le public était amené à interagir, entre autres, avec une interface graphique originale programmée sur *iPad* [Figure 20] communiquant en OSC[39] avec l'ordinateur pour contrôler la spatialisation. Les sources sonores, symbolisées à l'écran par des balles, étaient manipulables au touché par l'utilisateur. Celui-ci pouvait alors modifier la position d'une ou plusieurs de ces sources en les déplaçant simplement, ou en les « jetant » contre les parois virtuelles afin de les faire rebondir. Les retours du public ont été quasi unanimes quant à la qualité de cette interface et de son adéquation avec ce qui leur était donné à entendre, nous accordant sur le fait que le recours aux modèles physiques dans ce cas, aidait à donner plus de mouvement et de musicalité au champ sonore ainsi déployé sur un système de restitution au casque ou sur haut-parleurs.

[37] http://liine.net/en/products/lemur/.

[38] http://www.futur-en-seine.fr/fens2013/projet/interfaces-pour-la-mise-en-espace-du-son/.

[39] Open Sound Control.

Dans une autre mesure, on pourra citer le facteur d'agitation présent dans les dernières versions de la suite de plugins *GRM-Tools*[40]. Cette fonctionnalité ne fait pas, à proprement parler, usage de modèle physique mais permet néanmoins d'ajouter un certain dynamisme aux différents paramètres de l'interface. Par l'intermédiaire d'un facteur d'agitation global, une valeur pseudo-aléatoire est ajoutée ou retranchée à la valeur courante de chaque variable. Cette fonctionnalité pourrait très aisément être implémentée, dans le cadre de la mise en œuvre MaxMSP de la bibliothèque HOA, pour ajouter du dynamisme à une ou plusieurs sources ponctuelles statiques. Cette implémentation pourrait agir tout aussi bien en tant que greffon à l'interface *hoa.map* ou en tant qu'objet graphique indépendant.

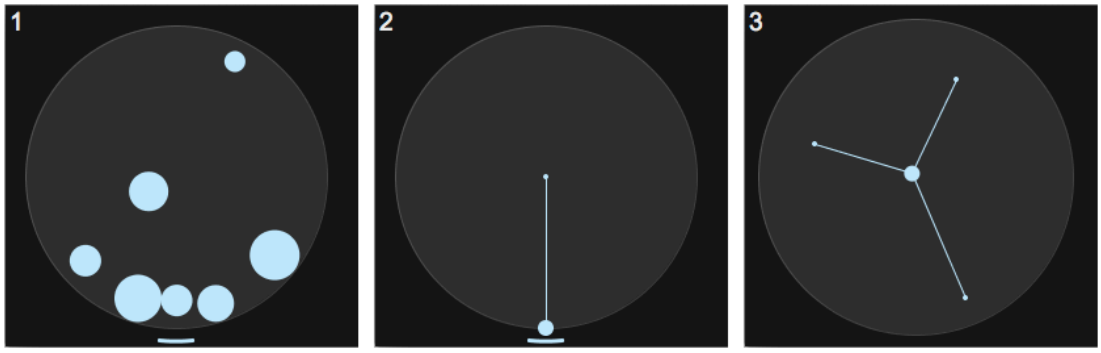

Figure 21 : représentation de trois interfaces de la bibliothèque *spatium* : *spatium.gravityBall2D* [1], *spatium.pendulum2D* [2], *spatium.springs2D* [3].

L'attrait pour les modèles physiques au sein des interfaces de spatialisation semble demeurer intact pour les utilisateurs, comme en témoigne encore la suite récente d'interface de la bibliothèque *spatium*[41] [Penha, Oliveira, 2012], qui en fait elle aussi un usage intensif [Figure 21]. Cette suite proposant notamment des modules de spatialisation générant des coordonnées spatiales par simulation de collisions, gravité ou balancement.

Le premier modèle que nous avons choisi d'implémenter au sein de la bibliothèque HOA, en tant qu'objet graphique indépendant, est celui des *boids*, qui permet de simuler le vol d'une nuée d'oiseaux. Créé en 1987 par Craig Reynolds [Reynolds, 1987], celui-ci

[40] GRM-Tools, INA-GRM, http://www.inagrm.com/grmtools.

[41] http://spatium.ruipenha.pt/interfaces/.

représente une « *alternative à l'écriture des trajectoires individuelles de chaque oiseau* »[42]. Ce modèle semblait donc apporter, pour nous, une réponse éventuelle en terme de gestion algorithmique de trajectoire de source sonore. Si celui-ci a beaucoup été employé dans les arts visuels (jeux vidéos, cinéma, installations)[43], il l'a moins été en musique. Or, si nous considérons chaque oiseau comme une source sonore potentielle, nous pouvons nous servir de ses coordonnées spatiales pour la mise en espace du son, en les transmettant à un moteur sonore comme *hoa.map~*.

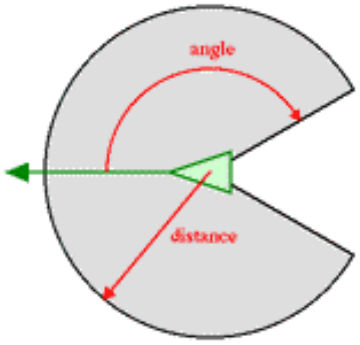

Figure 22 : représentation schématique du champ de vision d'un *boid*.

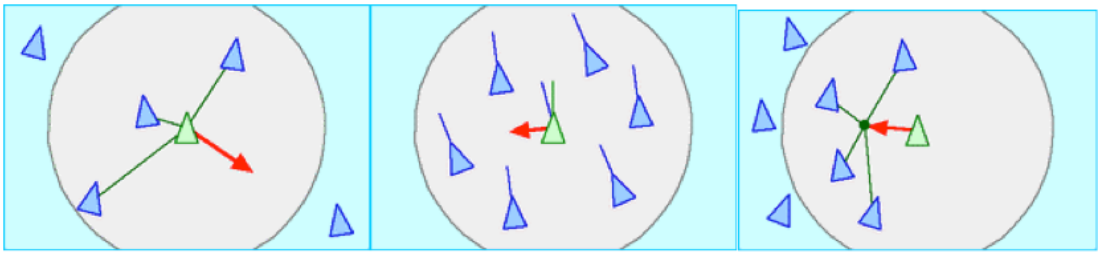

Figure 23 : figures représentant schématiquement les trois règles de base qui régissent le modèle des *boids*. Séparation, alignement, cohésion (de gauche à droite).

Le mouvement collectif des *boids*, d'apparence concerté, est une propriété émergente d'un ensemble de règles d'interaction entre les oiseaux. Chaque oiseau a un mouvement qui lui est propre. Sa trajectoire dépend à la fois des lois qui régissent la physique du mouvement, comme sa position et sa vélocité, mais aussi de sa perception locale de l'environnement dynamique. Ainsi, la trajectoire de chaque oiseau « s'adapte » en fonction des autres oiseaux

[42] Craig W. Reynolds, *Flocks, Herds and Schools : A Distributed Behavioral Model*, 1987.

[43] Le modèle des *boids* a par exemple été utilisé pour synthétiser le mouvement d'un troupeau de dinosaures dans le film Jurassic Park.

se trouvant dans son champ de vision [Figure 22], et d'un certain nombre de paramètres définis par l'utilisateur. Ces paramètres qui régissent le modèle physique des *boids* ont été définis par Reynolds telles que ci-dessous et illustrées à la [Figure 23].

- La séparation: empêcher que deux oiseaux, trop proche l'un de l'autre, n'entrent en collision.
- L'Alignement : faire en sorte que chaque oiseau s'aligne par rapport aux autres, afin qu'ils gardent tous une trajectoire commune.
- La Cohésion : faire en sorte que les oiseaux se regroupent autour du centre de masse du groupe en maintenant une certaine distance entre chaque membres.

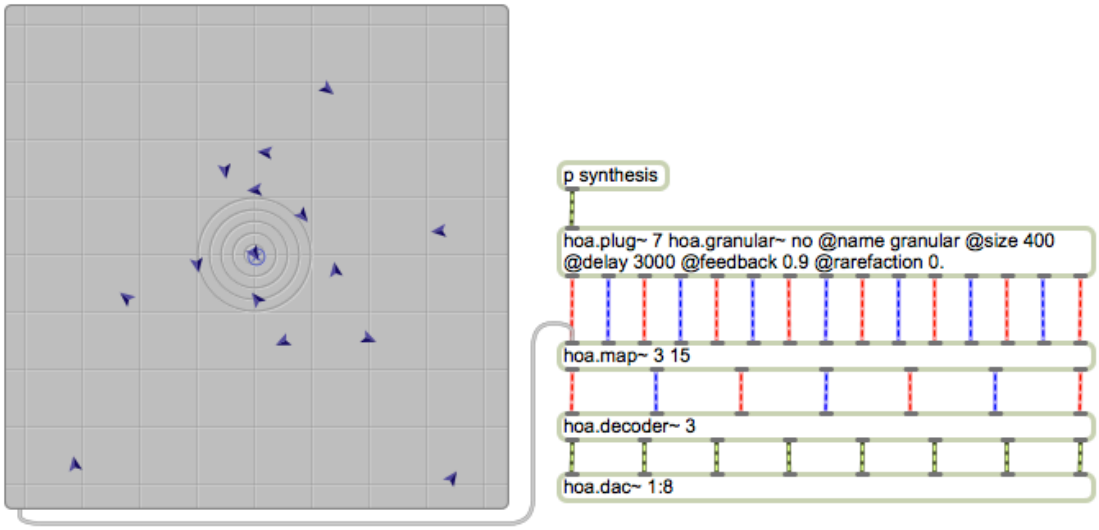

Figure 24 : extrait d'un patch MaxMSP contenant l'interface *hoa.boids* reliée au moteur sonore *hoa.map~* à l'ordre 3. Chacun des quinze oiseaux donne les informations de spatialisation pour un grain sonore.

De nombreuses implémentations graphiques des *boids* ont déjà été faites pour divers environnements logiciels comme *OpenFrameworks*[44], *processing*[45], Pure Data, ou MaxMSP. Celle que nous présentons ici a été réalisée en partie grâce au code source de l'objet *jit.boids*2d[46]. L'interface *hoa.boids* propose toujours une aide à la gestion de coordonnées spatiales de sources sonores virtuelles au sein d'un espace à deux dimensions. Elle reste parfaitement compatible avec le moteur sonore *hoa.map~* présenté dans la partie précédente,

[44] http://www.openframeworks.cc/.

[45] Daniel Shiffman, http://processing.org/examples/flocking.html

[46] Jash, André Sier, Eric Singer, Wesley Smith, 2005-2010, disponible pour MaxMSP et Pure Data.

comme en témoigne la [Figure 24] qui montre un patch dans lequel des grains sonores sont encodés grâce à la position spatiale de chaque oiseau.

La représentation graphique du plan au sein cette interface hérite de celle de l'interface *hoa.map* (grille, zoom, échelle, cercles concentriques représentant le cercle de haut-parleur). La représentation de la source sonore virtuelle quant à elle, a été remplacée par une flèche symbolisant un oiseau, dont l'orientation, calculée à partir du vecteur vélocité de chaque agent, donne une indication sur sa direction.

Contrairement à l'interface *hoa.map*, dans laquelle on contrôlait précisément la position spatiale de chaque source individuellement, ici, l'utilisateur est invité à faire varier des paramètres qui agissent sur le comportement global du modèle physique, l'interface *hoa.boids* gérant ensuite de façon « autonome » l'animation de chacun de ses agents.

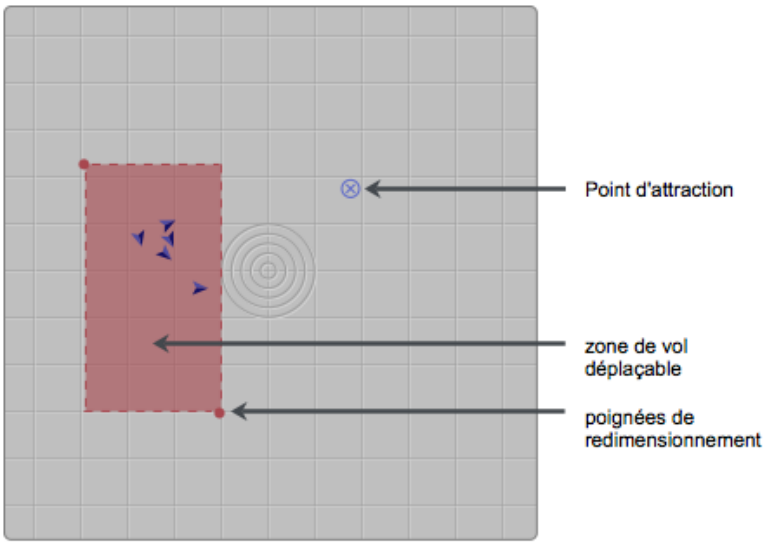

Figure 25 : représentation des types d'interaction avec l'interface *hoa.boids* permises par la souris.

Les interactions permises par la souris avec l'interface graphique sont pour l'instant peu nombreuses, l'utilisateur peut zoomer, définir le point d'attraction des oiseaux ainsi que la zone de vol au sein de laquelle sera contraint le mouvement des oiseaux (déplacement, et redimensionnement à l'aide de poignées dans les coins supérieur gauche et inférieur droit) comme le montre la [Figure 25].

Les variables, facilement manipulables et pertinentes d'un point de vue musical sont entre autres : l'attraction, la cohésion, la séparation, la vitesse, et l'alignement. Le taux

d'attraction varie entre 0 (attraction nulle) et 1 (attraction maximale). Le paramètre de cohésion, lui aussi situé entre 0 et 1, permettra de séparer ou regrouper les oiseaux les uns par rapport aux autres, tandis que le paramètre de séparation permettra de garder une certaine distance entre eux. Faire varier le paramètre d'alignement produira l'effet de passer d'un mouvement des oiseaux d'apparence chaotique à un mouvement où les oiseaux semblent se suivre les uns les autres. Enfin l'utilisateur pourra accélérer ou décélérer le mouvement global des *boids* grâce au paramètre de vitesse.

Cette interface a été rendue compatible avec le système de *preset/pattrstorage* de MaxMSP afin d'offrir à l'utilisateur la possibilité de sauver facilement plusieurs états de paramètres pour les rappeler plus tard ou de créer des interpolations dans le temps. Ce système facilite donc la gestion des nombreuses variables à l'aide d'un seul *macro paramètre* que représente l'index de l'état à rappeler.

Cette version n'étant qu'une première ébauche, beaucoup d'optimisations et d'améliorations peuvent lui être apportées. Le modèle physique des *boids* peut être augmenté de plusieurs comportements existant déjà dans la littérature notamment l'intégration de prédateurs et d'obstacles [Reynolds, 1988], [Aengus, 2005]. Si nous avons choisi de ne pas implémenter le comportement de prédation, car jugé sans réelle utilité musicale, le fait de pouvoir placer des obstacles que les oiseaux auraient à éviter au sein de l'interface semble être une proposition pertinente à faire au compositeur. On pourrait de même imaginer augmenter ce modèle en permettant à l'utilisateur de définir plusieurs points d'attraction, avec de possibles poids différents, ou encore de pouvoir dessiner des trajectoires ou des chemins préférés que les oiseaux devraient plus avoir tendance à suivre. En l'état actuel, l'objet procure en sortie les coordonnées spatiales de chaque oiseau présent sur la scène, ainsi que les coordonnées spatiales du barycentre calculé à partir de ces positions. On pourrait imaginer fournir aussi à l'utilisateur, dans une version future de l'interface, des informations sur la vélocité de chaque oiseau, sa direction, mais aussi d'autres informations liées à la masse d'oiseaux elle-même, comme l'aire totale qu'elle occupe dans l'espace, l'oiseau le plus excentré, la direction moyenne de la masse. Il serait alors possible de croiser ces données avec tout type de paramètres musicaux pour créer de nouvelles interactions musicales à partir de ce modèle.

Les premiers tests réalisés avec cet outil sont assez concluants d'un point de vue musical. Il semble par exemple très approprié dans le contexte d'une utilisation avec un capteur de poids comme la *Wii balance Board*[47] relié à la position du point d'attraction des *boids*. Cette interface permet donc bien d'ajouter un certain dynamisme à la spatialisation de sources sonores ponctuelles tout en maintenant des paramètres facilement manipulables et à fort potentiel musical. Elle nous encourage à continuer à expérimenter et développer d'autres interfaces en ce sens.

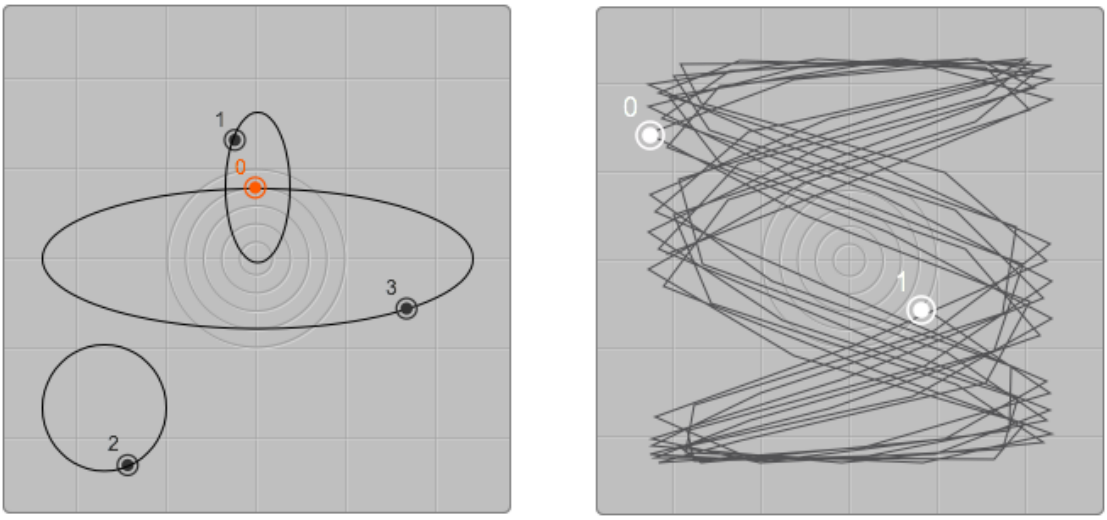

Figure 26 : représentation schématique du projet d'interface *hoa.galaxy*, quatre sources observant des trajectoires elliptiques, la source d'index 1 est un satellite de la source d'index 0 (à gauche). Représentation schématique du projet d'interface *hoa.lissajou* (à droite).

Le mouvement circulaire est un des plus efficaces en termes de spatialisation, et sûrement aussi un des plus employé par les compositeurs. Nous envisageons actuellement la création d'une interface qui permettrait une spatialisation de source faisant intervenir des trajectoires elliptiques en cascade en nous inspirant du modèle du système solaire ou des mécanismes horlogers [Figure 26, à gauche]. Chaque source sonore aurait une trajectoire elliptique qui pourrait se raccrocher ou être raccrochée à une autre de façon satellitaire et une vitesse indépendante, afin de créer une spatialisation et une temporalité globale et dynamique.

Le compositeur John Chowning a élargi la spatialisation par mouvements circulaires en faisant intervenir des courbes de Lissajou. Alors qu'un cercle est formé par l'oscillation de

[47] http://www.nintendo.fr/.

deux mêmes fréquences déphasées de π/2, une courbe de Lissajou peut être formée à partir de deux oscillations de fréquences et d'amplitudes différentes [Pottier, 2012]. On pourrait donc imaginer pouvoir intégrer des mouvements similaires au sein d'une nouvelle interface de spatialisation dans la bibliothèque HOA, qui permettrait de faire suivre aux sources des trajectoires s'apparentant à ces courbes [Figure 26, à droite].

Nous pensons continuer à expérimenter d'autres types d'interfaces faisant intervenir des modèles physiques dans les futurs livrables de la bibliothèque HOA pour MaxMSP et de créer les mises en œuvre nécessaires de celles déjà développées et approuvées sur le terrain musical, à destination du logiciel Pure Data ou sous la forme de plugins VST. Les propositions de nouvelles interfaces intégrant des modèles physiques, ou autre gestion algorithmique originale pour la spatialisation, sont quasi-infinies. D'inspirations souvent visuelles, celles-ci devront néanmoins justifier de leur pertinence sur le plan de la perception sonore pour être validées, afin d'échapper à l'écueil que pourrait représenter une prédominance de la vue sur l'audition dans l'élaboration des traitements spatiaux à travers de telles interfaces graphiques.

Nous avons exposé ici une première série d'interfaces de contrôle. Que la gestion s'y fasse de façon graphique ou qu'elle soit assistée de façon algorithmique, celle-ci a toujours pour vocation d'agir sur un encodage ambisonique basé sur une synthèse de sources sonores ponctuelles. Les techniques relatives à l'ambisonie permettent néanmoins d'autres types de synthèses et d'opérations échappant, si l'on peut le dire ainsi, au paradigme d'une vision purement atomique du champ sonore. En effet, le champ sonore est une notion qui pour nous, en tant que compositeur, peut être comprise comme une somme de plusieurs éléments sonores distincts (des sources ponctuelles par exemple), ou bien comme un tout insécable, une masse sonore. Les interfaces que nous étudierons dans la section suivante sont consacrées à des moteurs sonores permettant une gestion plus globale du champ sonore, traitant la masse sonore dans son ensemble.

1.2 MANIPULATION DE CHAMPS SONORES DANS LE DOMAINE DES ONDES PLANES

Nous venons de voir une série d'outils dédiés à la manipulation de la position spatiale de sources sonores ponctuelles au sein d'un plan. Cette opération permet la création d'un champ sonore par encodage d'une ou plusieurs sources sonores dans le domaine des harmoniques circulaires. La décomposition du champ sonore en fonctions spatiales est porteuse d'un nombre infini de possibilités musicales en termes de synthèses ou de transformations des sons et de l'espace, qui sont largement exploitées par les différents modules de la bibliothèque HOA (ex. synthèse de champs diffus, rotation, réverbération)[48]. Il nous serait néanmoins impossible de les aborder toutes ici dans le détail. Aussi, un autre type de représentation du champ sonore, la décomposition en ondes planes, existe et fait apparaître, de même, un certain nombre d'opérations musicales qu'il nous semble intéressant d'approcher ici par l'intermédiaire des interfaces de contrôle originales permettant leur manipulation.

Les ondes planes sont, dans une acception bidimensionnelle de l'espace, assimilables à des lignes parallèles se propageant dans une même direction, que l'on associe souvent acoustiquement aux ondes émises par des sources sonores ponctuelles éloignées. Ainsi, on peut très bien envisager une représentation du champ sonore comme une somme infinie d'ondes planes, réparties autour de l'auditeur et dirigées vers lui [Colafrancesco et al., 2013], [Guillot, 2013]. La décomposition d'un champ sonore en ondes planes, dans la pratique, consiste à discrétiser cette représentation en simulant une prise de son avec une série de microphones virtuels, répartis de manière équidistante sur un cercle autour de l'auditeur situé au centre de ce dispositif imaginaire. Les signaux générés par cette décomposition ont donc été appelés en ce sens « dépendants des microphones virtuels »[49]. L'opération de décomposition en ondes planes pourrait être vulgarisée autrement en disant qu'elle consiste à

[48] Un grand nombre de ces opérations sont réalisables avec un outil particulier de la bibliothèque HOA, à savoir l'objet *hoa.plug~* ; nous aborderons cet outil dans la dernière partie de ce mémoire. Nous renvoyons aussi le lecteur aux *patchs* d'aide relatifs aux autres traitements de la bibliothèque HOA pour MaxMSP, ainsi qu'à [Guillot, 2013] pour le détail de ces opérations.

[49] Pour plus d'informations à ce propos, consulter [Daniel, 2001] ainsi que [Guillot, 2013] et [Colafrancesco et al., 2013] pour son implémentation au sein de la bibliothèque HOA.

diviser le champ sonore en « portions » égales d'espace réparties autour de l'auditeur et sur lesquels des opérations pourraient alors s'exercer de façon indépendante.

Dans la mise en œuvre MaxMSP et Pure Data, l'objet *hoa.projector~* permet d'assurer cette opération de projection dans le domaine des ondes planes. Plusieurs opérations musicales nous sont dès lors accessibles dans ce domaine, comme le filtrage spatial, réalisé en modulant le niveau sonore de chacun des signaux générés par la projection du champ sonore. Nous étudierons donc en premier lieu ce traitement à travers une interface dédiée entièrement à sa manipulation avant de nous intéresser ensuite à d'autres types d'opérations permises par le moteur sonore *hoa.recomposer~*. Ce dernier permet, par opposition à la projection, une opération inverse, à savoir le passage d'une représentation du champ sonore dans le domaine des ondes planes à une représentation dans le domaine des harmoniques circulaires. Cette opération de conversion amènera aussi de nouveaux traitements musicaux, tels que la *distorsion de la perspective*, qui impliqueront dès lors de nouvelles façons de les contrôler au moyen d'une interface de contrôle inédite.

1.2.1 LE FILTRAGE SPATIAL

Le moteur sonore *hoa.space~* a pour fonction simple d'appliquer un gain spécifique à chaque signaux dépendant des microphones virtuels. Utilisé après une étape de projection du champ sonore dans le domaine des ondes planes, cet objet peut permettre de créer une focalisation sur une partie du champ sonore ou d'en retrancher une partie. En ce sens, cette opération peut être assimilée à une sorte de filtrage ; ce filtrage n'étant pas ici d'ordre fréquentiel mais spatial.

Pour illustrer ce traitement nous revenons sur l'installation *Transduction* qui a eu lieu dans le cadre de l'événement *Savante Banlieue 2012*. Lors de cette installation, une caméra Kinect[50], entre autres, captait les mouvements du public situé au centre d'un dispositif de seize haut-parleurs, restituant un champ sonore composé de sources ponctuelles réparties autour de lui [Figure 4]. Le public était alors invité à interagir avec cet environnement sonore

[50] http://www.xbox.com/fr-fr/kinect.

par l'intermédiaire de la caméra qui servait d'interface au contrôle de divers traitements spatiaux comme la rotation, la contraction/dilatation[51] ou le filtrage spatial du champ sonore en transduisant l'énergie du geste en énergie sonore. Le filtrage spatial demeurait total lorsque le public avait les mains baissées ; il pouvait alors « révéler », en pointant une direction à l'aide de sa main gauche, une portion du champ sonore à la manière « d'une lampe torche balayant l'obscurité » [Daniel, 2001]. Ce traitement, ainsi que ce mode d'interaction, ont été largement validés du point de vue musical par la critique du public.

Les applications de ce procédé sont nombreuses et disposent d'un fort potentiel musical. Imaginons par exemple que nous ayons enregistré, à l'aide d'un microphone de type *soundfield*, un concert dans un espace réverbérant, et dans lequel se trouvait des instrumentistes de chaque côté d'un chœur situé en façade. L'utilisateur pourrait alors se servir de cet enregistrement et, grâce à cet outil, focaliser par la suite l'écoute sur le chœur ou sur un seul des instrumentistes, ou encore ne garder que les réflexions de l'espace en filtrant tout l'avant du champ sonore.

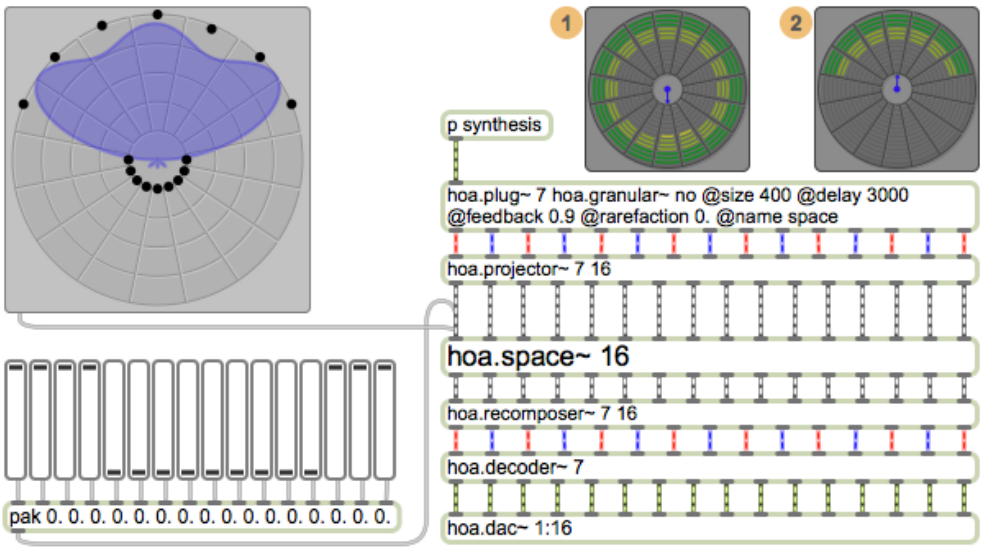

Figure 27 : extrait d'un patch MaxMSP contenant le moteur sonore *hoa.space~* et deux manières de contrôler, l'interface *hoa.space* (en haut), et une liste de *sliders* (en bas). L'arrière d'un champ sonore omnidirectionnel, constitué de grains sonores, est filtré spatialement. L'objet *hoa.meter~*[52] représente les contributions des 16 haut-parleurs pré [1] et post [2] traitement.

[51] Nous revenons sur ce traitement dans la prochaine partie.

[52] Nous détaillons cet objet au chapitre consacré à la représentation des contributions des haut-parleurs.

Ce même procédé peut être appliqué à tout type de champ sonore. Un champ sonore composé de sources sonores virtuelles ayant été encodées au préalable grâce à un outil comme *hoa.map~* tel qu'évoqué précédemment pour l'installation *Transduction*, ou encore à un champ sonore diffus, synthétisé, par exemple, grâce à des techniques granulaires[53] comme le propose la [Figure 27].

Dans les mises en œuvre du moteur sonore *hoa.space~* pour les logiciels MaxMSP et Pure Data, la gestion des gains spécifiques à chaque canaux passe par l'envoi d'un message constitué d'une liste de valeurs comprises entre 0 et 1. Ces valeurs peuvent être transmises par une série de *sliders*, représentant chacun un niveau de gain, comme on peut l'apercevoir sur la partie inférieure de la [Figure 27]. Cette gestion ne nous semblait néanmoins pas très appropriée dans la mesure où elle impliquait d'adapter le nombre de *sliders* au nombre de signaux à contrôler et donc beaucoup d'opérations de *patching*. De plus, ce genre de représentation n'offrait selon nous pas assez de correspondance avec le caractère spatial des données à manipuler ; « l'avant » du champ sonore par exemple étant accessible grâce aux quatre premiers et trois derniers *sliders* [Figure 27].

Il nous fallait donc réfléchir à une meilleure façon d'interagir avec ces données en proposant à l'utilisateur une interface originale, entièrement dédiée à cette opération, qui lui permettrait de contrôler ce moteur sonore tout en lui offrant une représentation graphique plus adaptée du procédé mis en place. Nous avons donc développé l'interface *hoa.space* en réponse à ce besoin.

Cette interface peut être directement reliée au moteur sonore au sein d'un *patch* [Figure 27, partie supérieure]. Elle propose une vue bidimensionnelle d'un espace sur lequel est répartie, de façon circulaire et équidistante, une série de points symbolisant chacun un microphone virtuel ou une « portion » d'espace à filtrer. La distance d'un point au centre du cercle de plus petit rayon représente le niveau de gain d'un microphone pouvant aller de 0 à 1. Cinq cercles concentriques indiquent des niveaux de gain intermédiaires (0.25, 0.5, 0.75).

[53] Nous détaillons l'objet *hoa.plug~* au chapitre lui étant consacré. Voir aussi [Guillot, 2013] et [Roads, 2001].

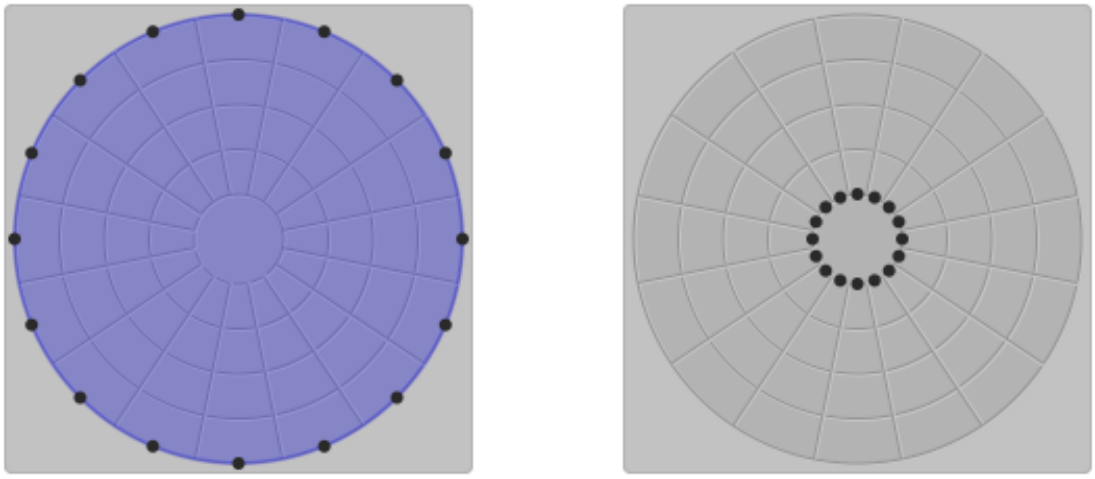

Figure 28 : aucun filtrage (à gauche), filtrage omnidirectionnel (à droite).

Ainsi, quand tous les points se trouvent sur le cercle de plus grand rayon, aucun filtrage n'est effectué, le signal de chaque canal étant alors multiplié par la valeur 1 ; Un filtrage omnidirectionnel est au contraire opéré lorsque l'ensemble des points se trouvent sur le cercle de plus petit rayon [Figure 28].

Les angles des points sont fixes et définis par le nombre total de microphones. L'utilisateur est donc seulement amené à faire varier la distance des points par rapport au centre, à l'aide de la souris. Pour faciliter la prise en main et rendre l'interface d'autant plus intuitive, les coordonnées de la souris sont *mappées* de façon à donner à l'utilisateur la possibilité d'agir globalement sur la forme générée par la disposition des points sur le cercle, lui donnant presque ainsi la sensation physique de modeler ou sculpter un espace sonore audible représenté par la forme transparente bleue.

Bien que le filtrage fréquentiel traditionnel et le type de filtrage sonore mis en place ici n'aient pas grand chose à voir l'un avec l'autre, les interfaces dédiées à chacun des deux traitements semblent partager un certain nombre de similitudes qu'il nous semble important d'expliciter ici à travers les figures suivantes :

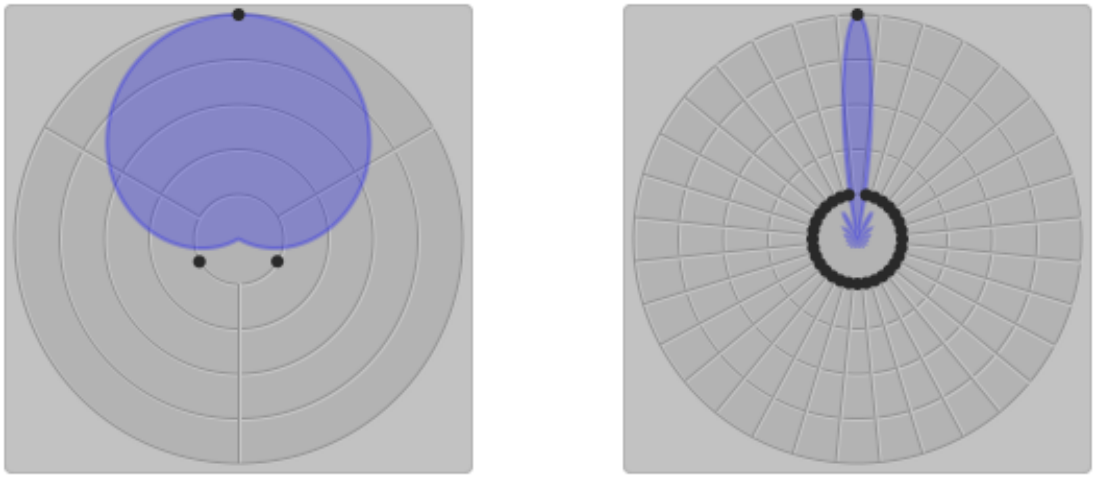

Figure 29 : représentation de la résolution spatiale du filtre. 3 microphones (à gauche), 32 (à droite).

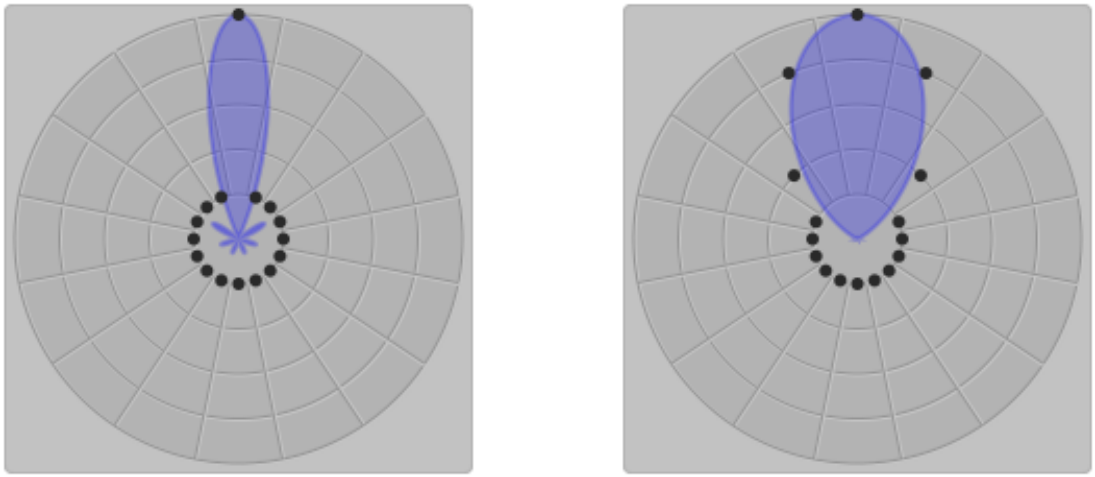

Figure 30 : analogie avec une diminution du facteur Q d'un égaliseur paramétrique.

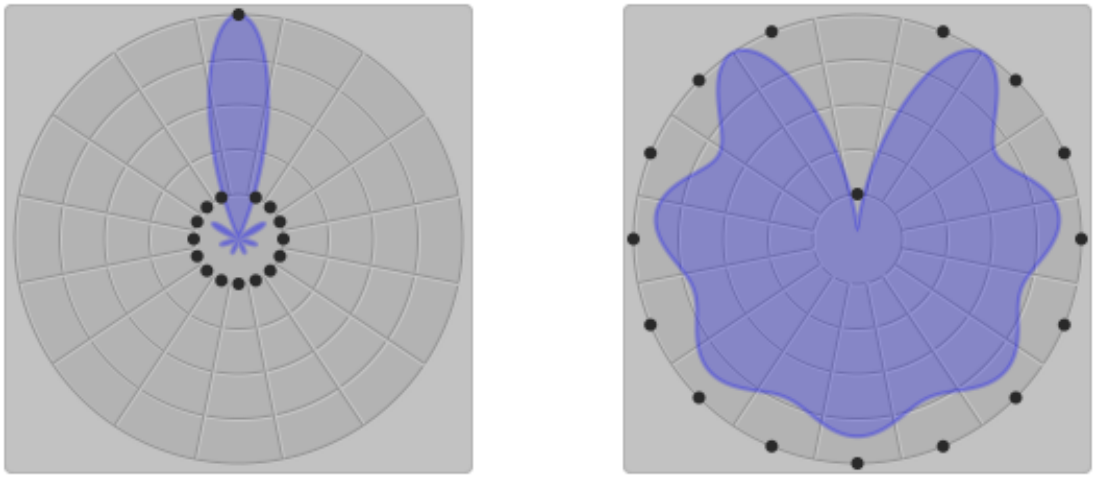

Figure 31 : analogie avec un filtrage de type passe-bande (à gauche), coupe-bande (à droite).

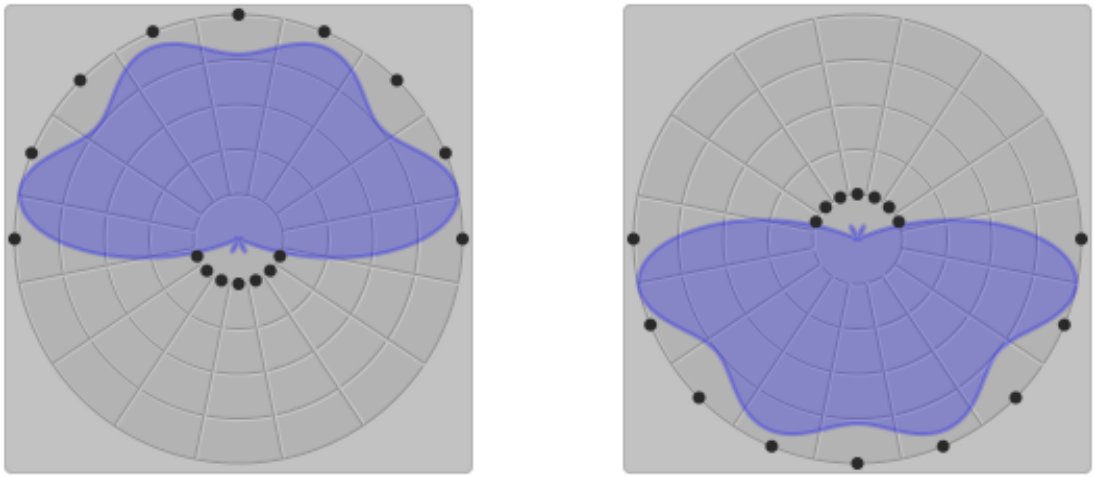

Figure 32 : analogie avec un filtrage de type passe-haut (à gauche), passe-bas (à droite).

Il existe différentes interfaces dédiées aux filtres fréquentiels, parmi celles-ci on trouve des égaliseurs graphiques ou paramétriques. Les égaliseurs graphiques sont constitués d'une série de potentiomètres verticaux contrôlant chacun le gain d'une bande de fréquence spécifique fixe. Ces *faders* sont alors répartis horizontalement et conformément au spectre sonore (fréquences graves à gauche, aiguës à droite). Les égaliseurs graphiques se distinguent notamment par le nombre de bandes qu'ils laissent à l'utilisateur la possibilité de contrôler (5, 10, 15, 32..). L'interface *hoa.space* pourrait donc être qualifiée d'égaliseur graphique dans lequel les *faders*, représentant des bandes d'espace, seraient répartis non-plus horizontalement mais de manière circulaire. Au sein de l'interface *hoa.space*, le nombre de bandes, correspondant au nombre de microphones virtuels, est lui aussi dynamique mais celles-ci ne représentent plus une partie du spectre mais une « portion » d'espace sonore [Figure 29]. La variation du nombre de bandes d'un filtre définit sa qualité fréquentielle ou spatiale suivant le type de filtre. Cette résolution, peut être quantifiée par un facteur de qualité communément appelé *facteur Q*, qui varie en fonction du nombre de bandes. Ainsi, dans notre cas, plus le nombre de microphones virtuels est élevé plus la résolution angulaire du filtre le sera, et plus l'utilisateur sera à même de contrôler avec précision les « portions » d'espace à filtrer. Comme au sein d'un filtre fréquentiel classique, il est possible de simuler une diminution de ce facteur en jouant sur les gains des bandes adjacentes à celle que l'on choisit comme centrale [Figure 30].

D'autres analogies peuvent être faites quant aux types de filtres fréquentiels utilisés traditionnellement. On retrouve en effet, la possibilité d'obtenir par exemple, un filtrage de type « coupe-bande » ou « passe-bande » en agissant sur le gain d'un seul microphone

[Figure 31]. On pourrait rapprocher aussi, d'un point de vue cette fois-ci strictement graphique, la forme d'un filtrage fréquentiel de type « allpass » à celle d'un filtrage spatial laissant passer tout le champ sonore, comme présenté sur la gauche de la [Figure 28].

De la même façon, on pourrait associer un filtre de type « coupe-bas » ou « passe-haut » à une opération que l'on pourrait nommer « coupe-arrière » ou « passe-avant », et inversement, un filtrage de type « passe-bas » ou « coupe-haut » à une opération de type « passe-arrière » ou « coupe-avant » [Figure 32].

Les nombreuses similitudes mises en avant ici, invite donc selon nous à définir de nouveaux termes au vocabulaire de l'espace existant, caractérisant de nouvelles opérations spatiales.

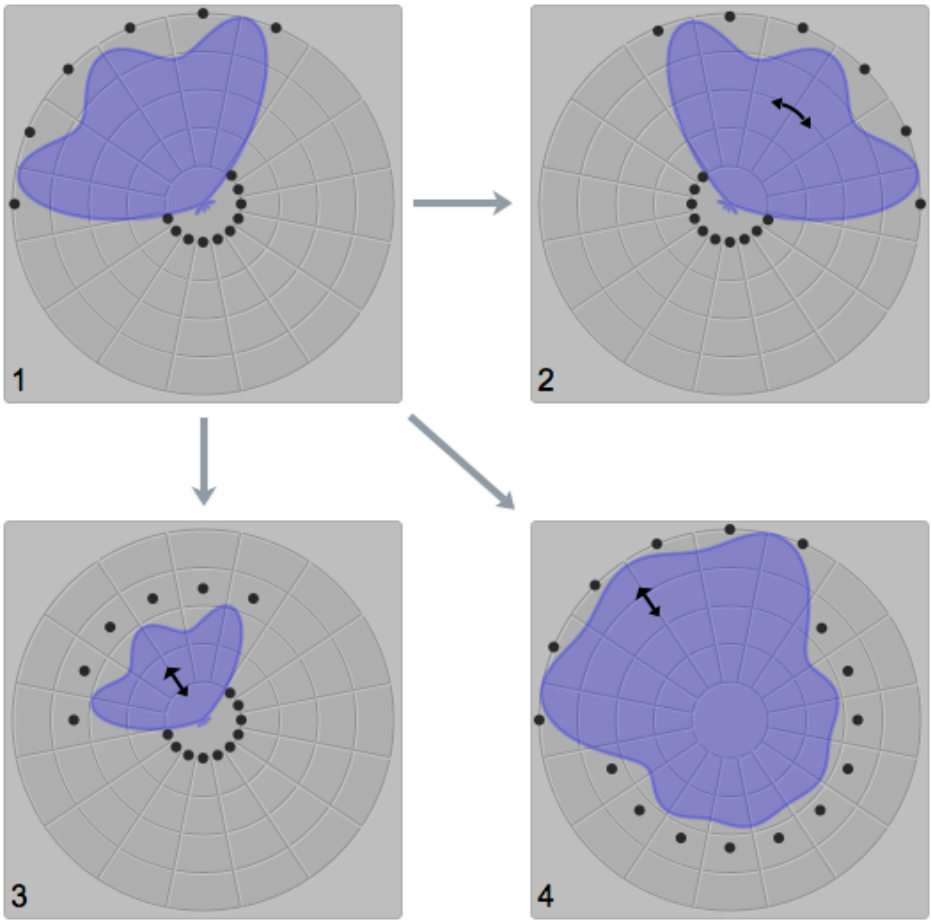

Figure 33 : représentation des interactions permises par la souris au sein de l'interface *hoa.*space pour *MaxMSP*. Filtrage original en [1], forme de filtre obtenu après rotation en [2], après diminution [3] et augmentation du gain général en [4]. Illustration du changement de curseur de la souris, spécifique à chaque mode d'interaction.

Afin d'offrir à l'utilisateur un maximum de contrôle sur le filtrage spatial, nous avons choisi de compléter cette interface en mettant en place un certain nombre d'interactions, destinées à faciliter la maniabilité de cet outil tout en augmentant en nombre et en qualité les opérations musicales permises par celui-ci. Trois modes d'interaction sont donc à présent disponibles par la combinaison de certaines touches du clavier avec les coordonnées de la souris. Le premier est le mode par défaut qui permet de modifier le gain de chaque microphone individuellement au clic ou au *cliqué-glissé* de la souris. Le second mode permet d'exercer une rotation infinie de la forme globale du filtre [Figure 33.2]. Le troisième permet enfin de diminuer [Figure 33.3] ou d'augmenter [Figure 33.4] le gain de tous les microphones à la fois tout en préservant la forme initiale du filtre.

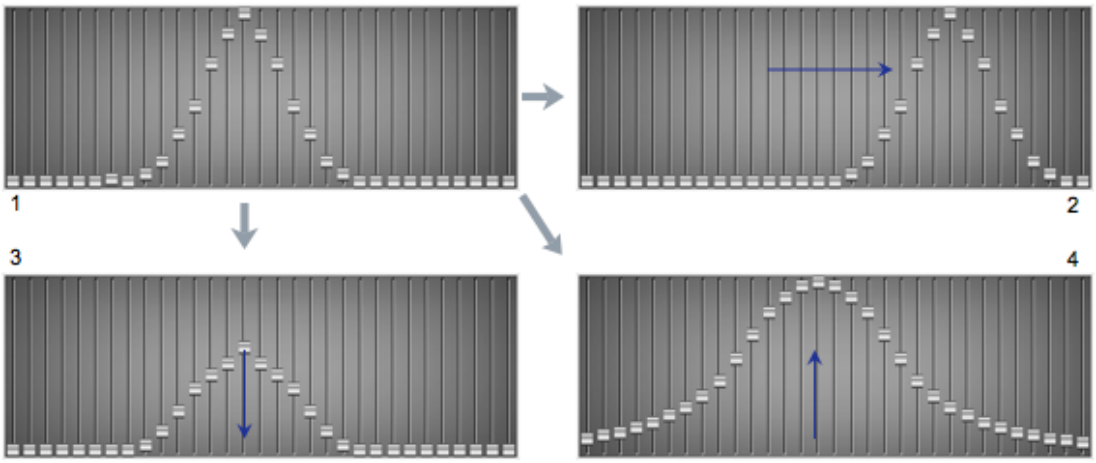

Figure 34 : représentation des interactions permises par la souris au sein de l'interface du plugin *equalize* de la suite *GRM-Tools*. Filtrage original en [1], déplacement horizontal des *faders* en [2], déplacement vertical des *faders* vers le bas [3] ou vers le haut [4].

On retrouve des modes d'interaction similaires au sein du plugin d'égalisation graphique *equalize* des *GRM-Tools*[54] [Figure 34]. En effet, cette suite de plugins, très utilisée dans le milieu de la création électroacoustique, doit non seulement sa popularité aux traitements sonores qu'ils proposent mais aussi à l'aspect très ergonomique de leurs interfaces pensées pour les compositeurs. Ainsi, comme le montre la [Figure 34], les modes d'interaction rejoignent ceux que nous venons d'exposer. Il est possible de déplacer globalement la courbe d'égalisation vers la gauche ou la droite pour opérer un mouvement latéral cyclique vers les fréquences graves ou aiguës [Figure 34.2] ; De même, le gain de l'ensemble des bandes de

[54] GRM-Tools, INA-GRM, http://www.inagrm.com/grmtools.

fréquence peut être géré relativement à un seul *fader* en déplaçant celui-ci vers le haut [Figure 34.3] ou vers le bas [Figure 34.4]. Nous retrouvons encore une analogie, permise ici entre l'opération de balayage fréquentiel s'apparentant à un *flanger* de la [Figure 34.2] et l'opération que l'on pourrait qualifier de balayage spatial à la [Figure 33.2].

Les plugins *GRM-Tools* intègrent tous, depuis la première version, une gestion de *presets*. L'utilisateur a donc la possibilité de sauver l'ensemble des paramètres de l'interface pour les rappeler ensuite ou encore de créer des interpolations entre les différents états. De façon analogue, l'interface *hoa.space* a été rendue compatible avec le système de *preset/pattrstorage* de MaxMSP, comparable à celui des *GRM-Tools*, ce qui la rend très facilement automatisable dans le temps.

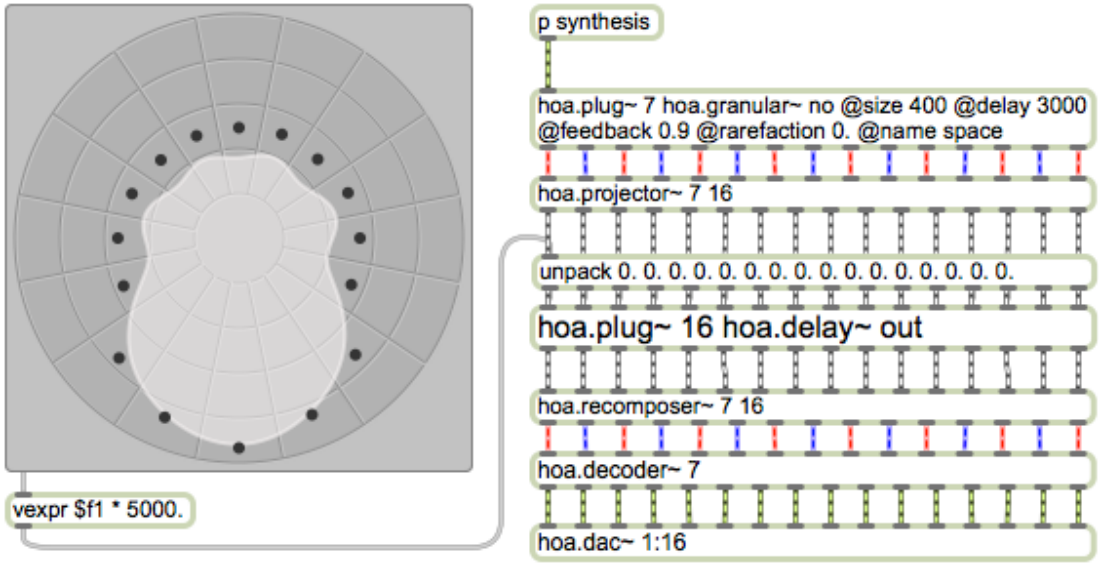

Figure 35 : représentation d'un patch MaxMSP dans lequel est utilisée l'interface *hoa.space* pour contrôler le temps de retard de chaque signal dépendant des microphones virtuels.

Il est important de noter que l'utilisation de l'interface *hoa.space* ne se limite pas au seul cadre du filtrage spatial et peut être généralisée au contrôle d'autres traitements spatiaux. En effet, les données générées en sortie, situées entre 0 et 1 peuvent servir à contrôler d'autres paramètres qu'un gain. La [Figure 35] expose par exemple un cas où l'interface *hoa.space* est utilisée pour gérer la variation des temps de retard de chaque signal dépendant des microphones virtuels. La représentation circulaire des valeurs aide dans ce cas aussi à se figurer plus directement l'incidence de leur manipulation sur l'espace. On pourra ainsi

« lire » cette interface, dans le cas illustré ici, en disant que le temps de retard est plus important à l'arrière du champ sonore qu'à l'avant ; lecture qui aurait été beaucoup moins intuitive si nous avions fait le choix d'employer une interface plus traditionnelle comme un *multislider* au sein MaxMSP.

Cette interface, très accessible et intuitive, a pu être validée sur le plan musical à travers différentes pièces[55] et installations. Bien qu'à un stade assez abouti, la version actuelle pourrait encore bénéficier de quelques améliorations. On pourrait notamment imaginer pouvoir borner les valeurs minimum et maximum des potentiomètres linéaires à des valeurs autres que 0 et 1, pour qu'une mise à l'échelle interne puisse permettre de généraliser plus facilement l'usage de cet objet à destination d'autres traitements spatiaux ; On pourrait de plus envisager de rajouter des indications visuelles donnant des informations sur les valeurs courantes de chacun des potentiomètres ; Ou encore de pouvoir modifier les angles de projection de chacun des microphones pour s'adapter à des configurations de projection différentes[56].

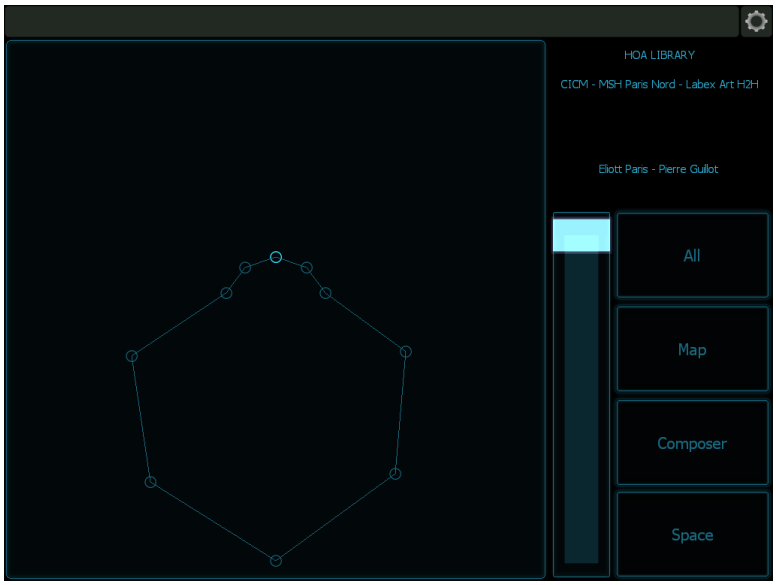

Figure 36 : représentation de la vue *space* de l'interface *Lemur* pour *iPad* présentée au public lors de l'événement *Futur en Seine 2013*.

[55] Nous pensons notamment aux pièces *Evoluciones* de la compositrice Marisa Acuña et *Turdus* du compositeur João Svidzinsky, crées en juin 2013 dans le cadre de l'atelier de composition de l'université Paris-VIII.

[56] Notons que cette dernière fonction devrait, dans ce cas, être aussi présente au sein du moteur sonore *hoa.projector~*.

Un des enjeux de développement futur consiste aussi à étudier les possibilités d'intégration de cette interface pour des supports différents, tels que les tablettes graphiques, afin de tirer parti d'autres modes d'interaction et tenter ainsi de gagner en ergonomie. L'interface que nous avons présentée lors de l'événement *Futur en seine 2013* représente une première piste en ce sens [Figure 36] en permettant de gérer plusieurs microphones en même temps grâce au système *multitouch* de l'iPad à travers l'application *Lemur*.

Nous venons d'exposer ici un traitement musical permis par le passage dans le domaine des ondes planes et une manière de le contrôler de façon simple et ludique grâce à l'interface originale qui lui est dédiée. L'opération inverse, c'est à dire le passage du champ sonore du domaine des ondes planes à celui des harmoniques circulaires est permis par le moteur sonore *hoa.recomposer~*. Cette opération de conversion amène aussi de nouveaux traitements musicaux qui impliquent dès lors de nouvelles façons de les contrôler. C'est ce que nous souhaitons aborder ci-après.

1.2.2 LA DISTORSION DE LA PERSPECTIVE

Le moteur sonore *hoa.recomposer~*, utilisé après une étape de projection du champ sonore, permet de passer du domaine des ondes planes au domaine des harmoniques circulaires par une opération de ré-encodage des signaux dépendants des microphones virtuels. Il peut s'apparenter en ce sens à un moteur sonore composé d'une série de *n* objets *hoa.encoder~* regroupé en un seul, dont le nombre correspondrait à celui des microphones ayant servis à discrétiser le cercle au cours de l'étape de *prise de son virtuelle*.

Dans l'actuelle mise en œuvre de la bibliothèque pour MaxMSP et Pure Data, ce moteur sonore dispose de trois modes nommés respectivement *fixe*, *fisheye* et *free*, appelant à des manipulations et des traitements de l'espace spécifiques.

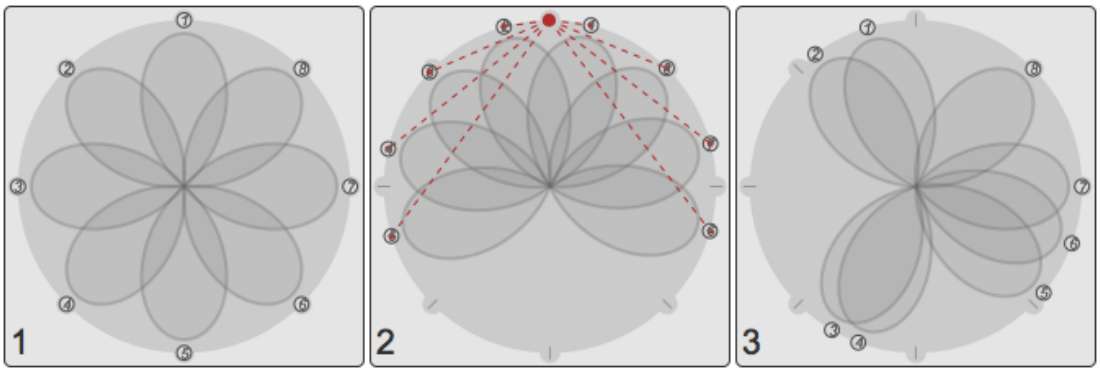

Figure 37 : représentation schématique des trois modes de recomposition disponibles au sein du moteur sonore *hoa.recomposer~* : mode *fixe*, *fisheye* et *free* (de gauche à droite), pour huit microphones virtuels.

Le mode *fixe* a pour simple fonction d'encoder chaque signal dépendant des microphones virtuels selon un angle d'incidence correspondant à l'angle des haut-parleurs d'un système de restitution ambisonique classique pour un nombre de haut-parleurs donné [Figure 37.1]. Il n'a donc d'autres fonctions que celle de permettre l'interopérabilité entre le domaine des ondes planes et des harmoniques circulaires et ne dispose donc d'aucun paramètre « jouable ».

Le mode *fisheye*, appelé ainsi par analogie avec les traitements optiques accessibles grâce aux objectifs du même nom, permet d'opérer ce que Gerzon a appelé une *distorsion de la perspective* [Gerzon, 1992] par contraction ou dilatation de la scène sonore frontale, comme illustré à la [Figure 38].

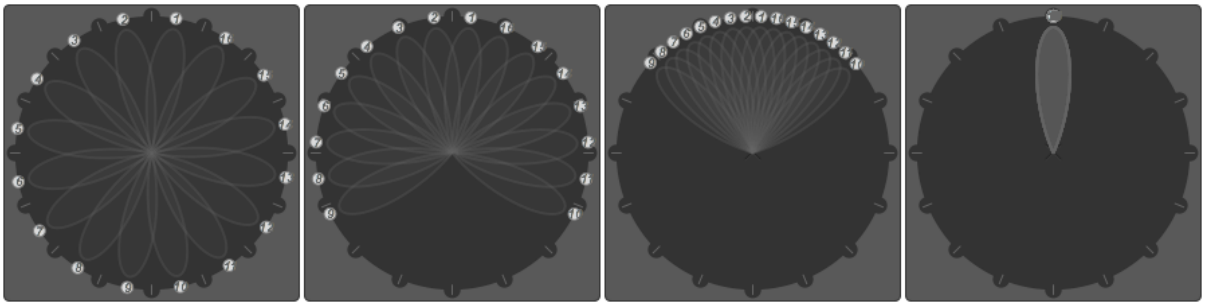

Figure 38 : illustration d'une opération progressive de distorsion de la perspective de type *fisheye*, par contraction (de gauche à droite) ou dilatation (de droite à gauche) de la scène frontale.

Cette distorsion est obtenue par modification automatique des angles d'incidence initiaux des signaux dépendants des microphones virtuels. Dans l'implémentation actuelle de la bibliothèque, cette manipulation a été rendue paramétrable grâce à une variable unique évoluant entre 0 et 1. Ainsi, quand le facteur *fisheye* est égal à 1, aucune distorsion n'est effectuée et les angles d'encodage des signaux dépendants des microphones virtuels demeurent les même qu'avec le mode *fixe* pour un nombre de microphones donné [Figure 37.1]. En revanche, plus celui-ci se rapproche de 0, plus la scène sonore se contractera, jusqu'à produire l'effet d'un champ sonore assimilable à une source sonore ponctuelle encodée selon un angle d'incidence de 0° [Figure 38.4].

Le traitement de type *fisheye* a pu être validé à de nombreuses reprises sur le plan musical lors de plusieurs créations et installations sonores. La compositrice Anne Sèdes, dans le cadre de sa pièce *Immersion*[57] pour violoncelle et électronique en temps réel, se l'est par exemple appropriée en donnant à entendre les variations spatiales produites par le passage d'un champ diffus composé de grains sonores à un champ sonore composé des mêmes éléments mais assimilable cette fois-ci à une source sonore ponctuelle par contraction de la scène sonore frontale. Cette variation a, dans ce cas, été composée de façon à en laisser la gestion par l'instrumentiste grâce à un *mapping* d'amplitude associé au facteur *fisheye* de l'objet *hoa.recomposer~*[58].

Ce même traitement a pu être aussi apprécié lors de l'installation *Transduction*, déjà citée dans les précédents chapitres [Figure 4], cette fois-ci à travers un contrôle de type gestuel du facteur *fisheye*. Le public était ici invité à contracter un champ sonore, composé de sources

[57] Pièce interprétée par le violoncelliste Guilherme Carvalho le 27 novembre 2012 dans le cadre d'un concert-lecture à l'Université Paul Valéry - Montpellier III, et le 13 mai dans le cadre des JIM 2013.

[58] La compositrice décrit avec plus de précision ses manières de faire dans [Colafrancesco et al., 2013].

ponctuelles réparties autour de lui, vers le centre de la scène frontale, par repliement ou déploiement de ses deux bras en direction de la caméra située face à lui. L'effet de ce traitement est illustré par la [Figure 39].

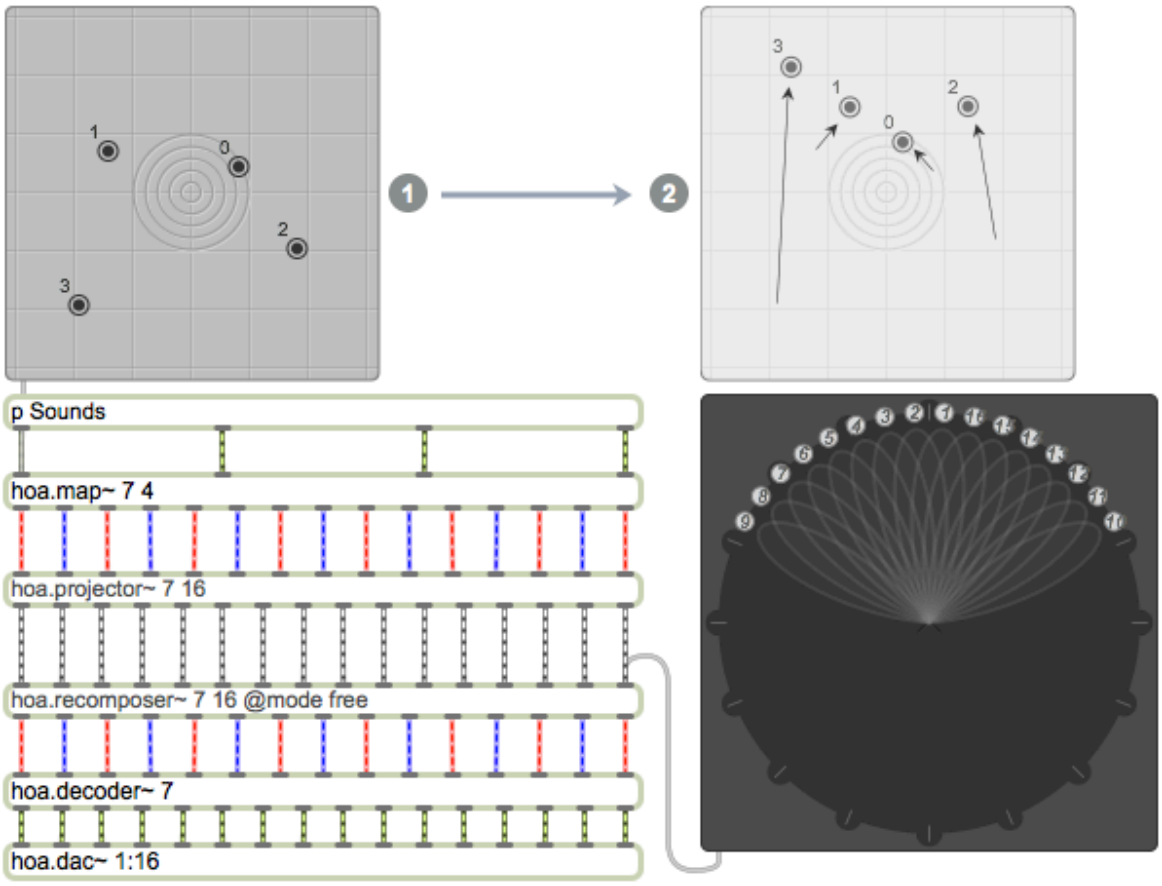

Figure 39 : représentation d'un patch MaxMSP dans lequel un champ sonore, composé de quatre sources encodées grâce à l'objet *hoa.map* [1], subit une contraction ou resserrement en direction d'un angle d'azimut 0° grâce au moteur sonore *hoa.recomposer~* et de son interface graphique associée. L'interface *hoa.map~* située à droite [2] représente la position approximée des sources sonores telles qu'elles pourraient être perçues après traitement.

Le facteur *fisheye*, automatisant les variations d'angle des microphones, constitue donc un moyen particulièrement efficace pour gérer la contraction du champ sonore. Il se révèle aussi très maniable et intuitif pour des auditeurs non-initiés, comme dans le cadre de la présentation de la bibliothèque HOA lors de l'événement *Futur en Seine 2013*, où le public était invité à manipuler celui-ci à l'aide d'un simple potentiomètre linéaire sur tablette tactile. Néanmoins il peut sembler quelque peu limitatif pour un musicien ou un compositeur qui cherche à exercer des traitements autres qu'une contraction du champ sonore vers l'avant.

Anticipant ce besoin, nous avons donc doté le moteur sonore *hoa.recomposer~* d'un dernier mode, appelé *free*, qui permet de reproduire à la fois l'effet des deux précédemment cités (*fixe* et *fisheye*) [Figure 39] mais qui autorise désormais aussi une gestion individuelle de l'angle d'incidence de chaque canal ainsi qu'une variation de leur directivité [Figure 37.3]. L'utilisation de ce mode supporte donc plus de flexibilité et permet d'ouvrir de nouvelles perspectives musicales. En revanche il implique aussi la gestion d'un grand nombre de paramètres, égal au double du nombre total de signaux à encoder (angle et directivité de chacun d'entre eux), ce qui le rend difficilement contrôlable sans le secours d'une interface graphique dédiée. Nous avons donc développé, parallèlement à ce moteur sonore, l'objet graphique *hoa.recomposer* qui répond à ce besoin en proposant une représentation de l'ensemble des canaux tout en autorisant une manipulation plus intuitive de ceux-ci.

Le moteur sonore de ce dernier mode peut être vu comme une série de couple d'objets *hoa.encoder~* (pour l'encodage ambisonique) et *hoa.wider~* (pour la variation de la directivité) gérant chacun la synthèse d'un canal spécifique de la prise de son virtuelle. Or nous avons vu au premier chapitre que ce couple d'objets pouvait être facilement contrôlable grâce à l'interface *hoa.control* dédiée à ce processus. Nous nous sommes donc inspiré de cette dernière, en la sérialisant elle aussi sous la forme d'une interface originale, en l'adaptant aux besoins de ce nouveau contexte.

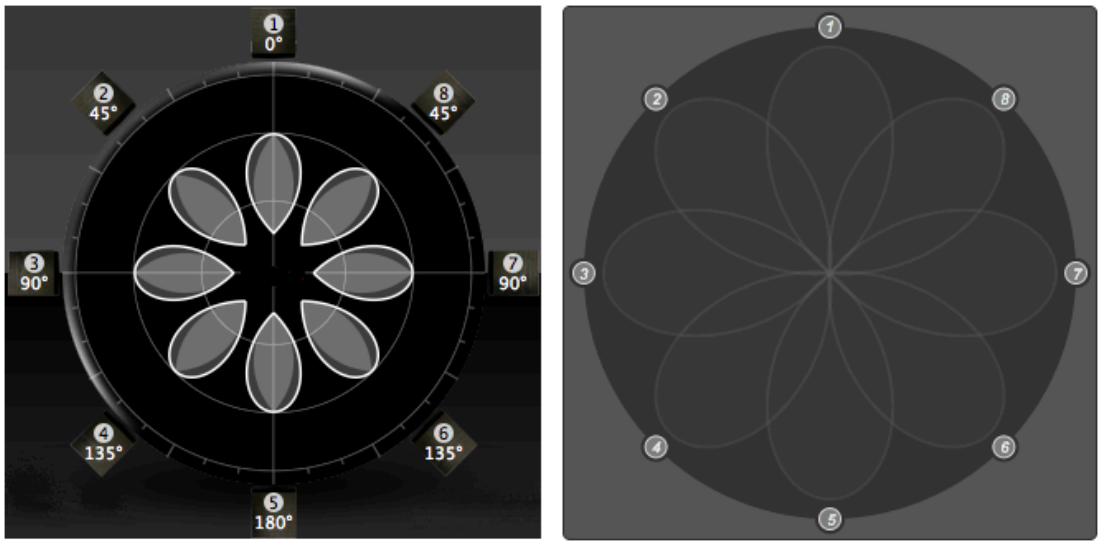

Figure 40 : représentation de l'interface du plugin *Harpex-B* (à gauche) et de l'interface *hoa.recomposer* pour MaxMSP (à droite) présentant huit microphones virtuels dans leurs positions par défaut.

L'aspect graphique de cette nouvelle interface a été inspiré en partie du plugin *Harpex-B*[59] [Harpex, 2012], comme peut en témoigner la [Figure 40]. Ce plugin, crée par la société du même nom, permet à l'inverse de l'objet *hoa.recomposer~*, une opération de décodage d'une prise de son effectuée à partir de leur microphone *soundfield* (restreint à un ordre de décomposition 1) pour différents systèmes de restitution en paramétrant la disposition et le nombre de microphones autour du cercle [Berge et al., 2010].

L'interface *hoa.recomposer* offre elle aussi une vue bidimensionnelle d'un espace ou se trouve répartie sur un cercle une série de sources sonores ponctuelles, correspondant aux différents canaux d'une *prise de son virtuelle*, symbolisés ici par des lobes transparents indiquant leur directivité respective. Les traits proéminents sur le pourtour du cercle indiquent la position optimale des sources pour une recomposition idéale et sans distorsion du champ sonore. La directivité maximale des canaux est fonction de l'ordre de décomposition.

Pour faciliter la manipulation des différents canaux et permettre des interventions globales sur l'ensemble du champ sonore, nous avons mis en place un certain nombre d'interactions rendant possibles les traitements sonores cités plus haut.

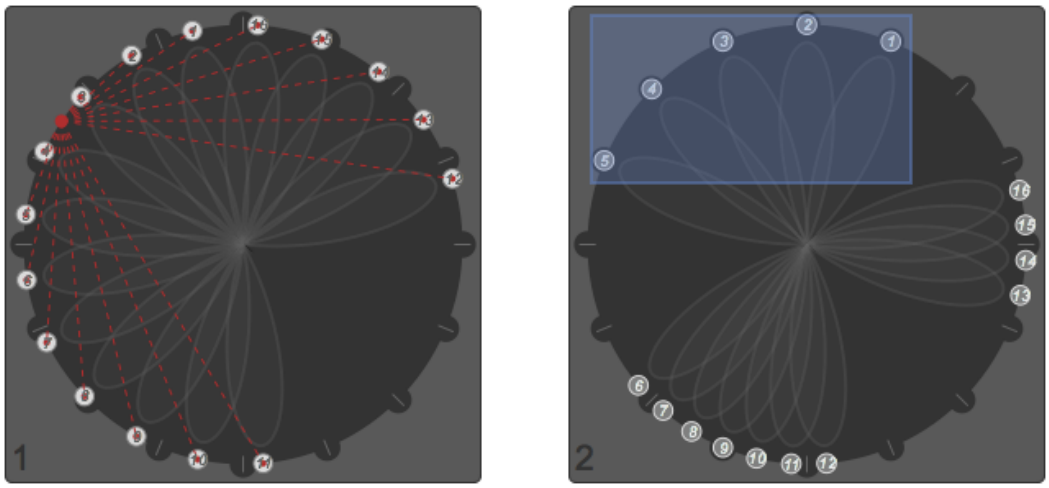

Figure 41 : représentation de la prévisualisation d'une opération de type *fisheye* (à gauche). Représentation d'un rectangle de sélection multiple ainsi qu'un resserrement subit par deux groupes différents de microphones (à droite).

[59] http://www.harpex.net/.

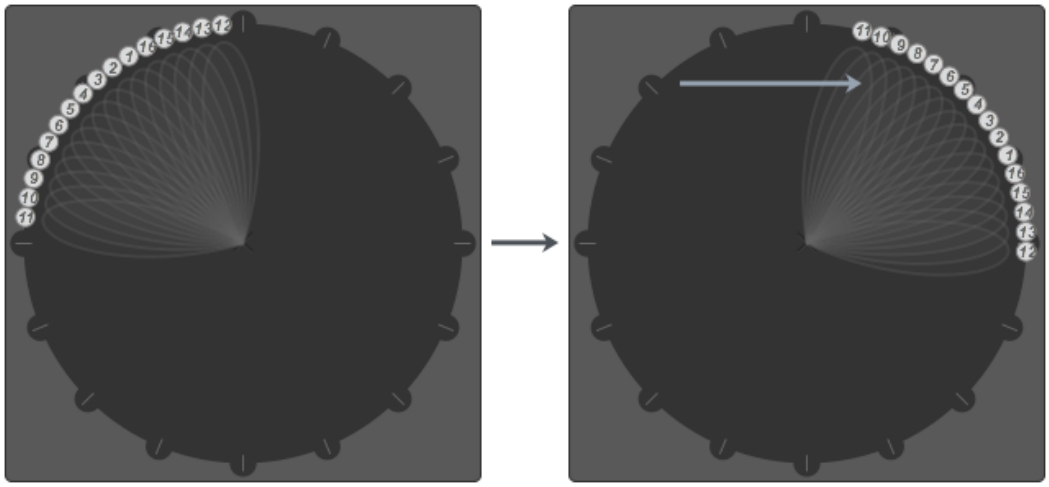

Figure 42 : représentation d'un déplacement rotatif de l'ensemble des canaux vers la droite suite à une opération de resserrement (de gauche à droite).

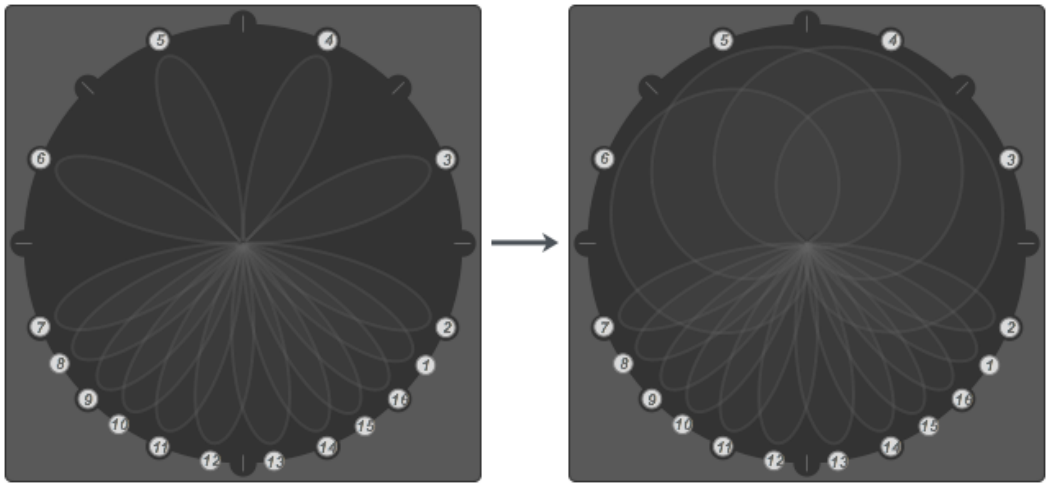

Figure 43 : représentation d'une diminution de la directivité des microphones d'index 3, 4, 5 et 6.

Plusieurs opérations de sélection manuelles ont été mises en place. L'utilisateur peut dès lors exercer une sélection individuelle en cliquant successivement sur les points qu'il souhaite manipuler, sélectionner un ensemble de points par rectangle de sélection [Figure 41.2], ou encore les sélectionner tous simultanément grâce au raccourci clavier *cmd+a*. Une couleur paramétrable indique l'état actuel de sélection de chaque source. Grâce à ces outils de sélection, les interactions possibles sur une source deviennent aussi réalisables sur plusieurs.

A l'aide du mode *free*, l'utilisateur peut désormais exercer une *distorsion de la perspective* de type *fisheye* dans n'importe quelle direction de la scène sonore. Une fois un groupe de source sélectionné, l'utilisateur peut appuyer sur la touche *ctrl* pour prévisualiser la manipulation qu'il est sur le point d'exercer. Des lignes apparaissent alors, reliant chaque point sélectionné au point central de perspective, définissant ainsi ce que l'on pourrait

nommer un *point de fuite* par analogie avec l'acception visuelle de la perspective utilisée en dessin [Figure 41.1]. Une fois ce point défini, l'utilisateur peut donc contracter l'ensemble des canaux en déplaçant la souris dans la direction du centre ou les rétracter en tirant dans la direction opposée. Ce paramètre agit donc à la manière d'un *méta-contrôleur* en contrôlant la rétractation de chacun des canaux de façon globale. L'autre force de cette interface, alliée à ce mode, réside aussi dans le fait qu'elle autorise de multiples *points de fuite*, offrant ainsi plus de flexibilité à la distorsion de la perspective de la scène sonore. L'utilisateur peut donc exercer des traitements de type *fisheye* successivement sur des groupes de canaux différents [Figure 41.2].

Comme au sein de l'interface *hoa.control*, les lobes de directivité des différents canaux font ici office de potentiomètres rotatifs infinis. Les angles de ceux-ci sont donc librement manipulables individuellement pour permettre des distorsions locales de l'espace sonore ou par groupe pour exercer par exemple une rotation de l'ensemble ou partie du champ sonore [Figure 43]. Il peut être aussi intéressant de conserver une répartition équidistante des angles d'un ou plusieurs canaux mais en en changeant ou en en inversant l'ordre pour obtenir des traitement spatiaux et musicaux originaux. Pour cela, l'utilisateur dispose d'une fonction lui permettant d'activer un *magnétisme* en maintenant la touche *cmd* de son clavier enfoncée tout en modifiant l'angle d'un microphone virtuel pour faire en sorte que celui-ci soit *attiré* plus facilement vers l'angle par défaut le plus proche.

La dernière opération permise consiste à faire varier la directivité d'un ou plusieurs microphones. Cette variation s'exerce en cliquant sur le ou les microphones sélectionnés puis en déplaçant la souris, tout en maintenant la touche *shift* du clavier enfoncée, en direction du centre pour en diminuer la directivité ; ou dans la direction opposée pour l'augmenter [Figure 43]. Cette manipulation devrait permettre de créer un effet « zoom » sur le champ sonore en « rapprochant » perceptivement les signaux les moins directifs du centre de la scène sonore circulaire[60].

[60] Cette opération n'a malheureusement pas pu, faute de temps, être testée à l'heure actuelle.

Afin de faciliter l'automation des paramètres d'angle et de directivité des signaux dépendants des microphones virtuels dans le temps et pour ajouter plus de potentiel musical à cet outil, nous avons choisi de rendre l'interface compatible avec le système de *preset/pattrstorage* de MaxMSP.

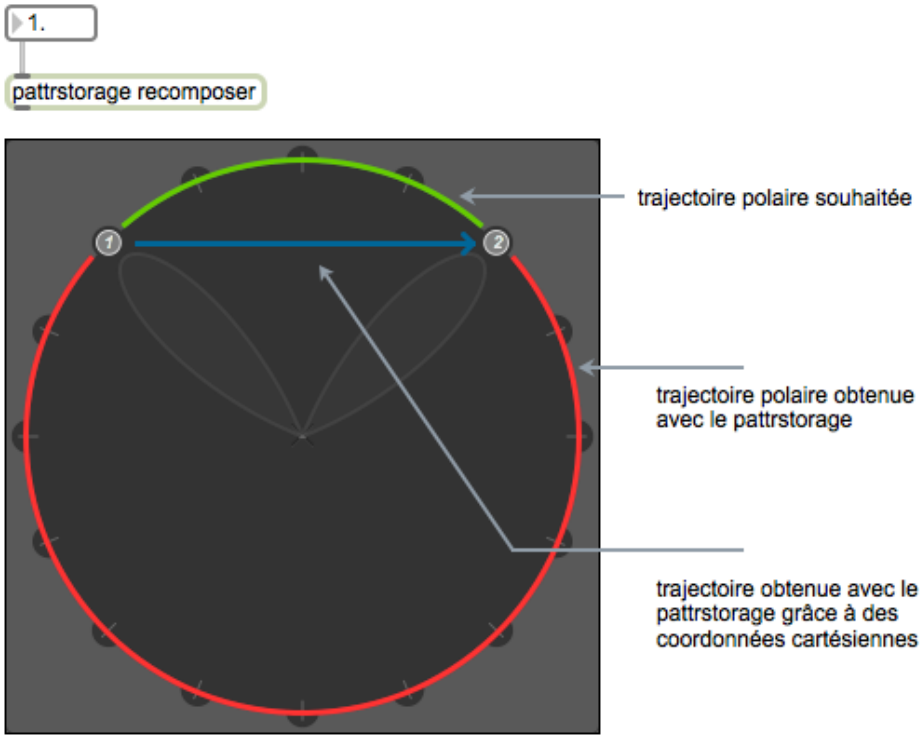

Figure 44 : représentation de la trajectoire effectuée par un microphone selon le type de données fournies au *pattrstorage* pour l'interpolation des données d'angles entre deux positions différentes.

Nous avons dû pour cela nous confronter aux limites de ce système dans son implémentation courante. L'interpolation d'une variable entre deux états de valeur au sein du *pattrstorage* se fait de manière linéaire. Il convient donc parfaitement à des paramètres ayant des bornes minimum et maximum non bouclables comme la variable de directivité des microphones qui évolue entre 0 et 1. Ce système n'est, en revanche, plus aussi pertinent pour des paramètres ayant pour variable des données cycliques comme la phase et, dans notre cas, l'angle d'incidence des signaux exprimés en radian (entre 0 et 2π) ; paramètre dans lequel la valeur minimum 0 est égale à la valeur maximum 2π. Pour illustrer ce problème, servons nous de la [Figure 44]. Dans cet exemple, un *pattrstorage* est utilisé pour créer une interpolation entre un premier état ou l'angle est égal à π/4 (soit approximativement 0.78) et un second ou il est égal à 7*π/4 (soit environ 5.49). Si nous fournissons de telles données au système *pattrstorage*, celui-ci va donc incrémenter la valeur d'angle du premier état jusqu'à atteindre

celle du second, menant ainsi à la création d'une trajectoire comme représenté ici en rouge. Or, pour obtenir la trajectoire souhaitée, représentée en vert, cela nécessiterait un passage à 0 lors de l'interpolation. Deux solutions se proposaient alors à nous ; soit nous devions recréer un système similaire au *pattrstorage*, intégré directement à l'interface (comme nous avons dû le faire pour l'objet *hoa.map*), soit nous formations autrement les données fournies au *pattrstorage*. Nous avons opté pour la seconde solution. Ainsi, les données fournies au *pattrstorage* ne sont plus des données d'angle en radian mais des coordonnées cartésiennes indiquant la position de chaque point permettant à présent d'obtenir la trajectoire représentée en bleue. Nous devons ensuite les réinterpréter en tant que données polaires au sein de notre interface, en forçant une distance par rapport au centre égale, afin d'obtenir la trajectoire équivalente à celle représentée en vert à la [Figure 44].

Le moteur sonore *hoa.recomposer~*, permet donc d'exercer des traitements de l'espace et des opérations musicales innovantes qui ont pu être validées à plusieurs reprises sur le plan de la création électroacoustique. Couplé avec l'interface graphique *hoa.recomposer,* ces opérations deviennent très facilement paramétrables et manipulables en temps réel par le compositeur ou automatisable dans le temps. L'interface *hoa.recomposer~* n'est pour l'instant disponible que pour MaxMSP, les futurs livrables de la bibliothèque devraient remédier à cela en proposant entre autre une version compatible avec Pure Data.

Nous venons de voir dans cette partie une série d'interfaces dédiées au contrôle de la mise en espace du son spécifiques et originales, dédiées aux moteurs sonores ambisoniques de la bibliothèque HOA. Il nous a néanmoins semblé indispensable de nous intéresser aussi au sein de ce projet à la manière de représenter les données spatiales et sonores générées par ces différents moteurs. Nous tenterons donc, dans la partie suivante, de présenter les enjeux d'une telle entreprise ainsi que les propositions logicielles que nous avons pu apporter pour y répondre.

2 DES INTERFACES DE REPRESENTATION GRAPHIQUE DU CHAMP SONORE

Dans la typologie des contrôleurs élaborée par Couturier, les contrôleurs n'offrant pas d'autres interactions physiques que celles liées à la vue sont appelées *indicateurs visuels*, « les éléments graphiques de type *indicateurs visuels* permettent à l'utilisateur d'avoir des informations sur le déroulement de l'action, sur l'état du système, comme la valeur de certains paramètres ou la représentation du son produit » [Couturier, 2004]. Il rajoute que « ces éléments graphiques ne sont pas *interactifs* dans le sens où l'on ne peut pas agir directement sur eux avec un périphérique d'entrée, même s'il existe un lien entre l'état de l'élément et les gestes effectués sur les périphériques d'entrée », ou sur les autres contrôleurs présents à l'écran [Couturier, 2004]. La segmentation que nous opérons entre les interfaces graphiques de contrôle et de représentation du champ sonore s'opère donc en fonction du degré d'interaction que celles-ci autorisent, mais aussi et surtout en fonction du type de données qu'elles sont à même de représenter. En effet, une interface dite de contrôle telle que *hoa.map*, que nous avons présentée plus haut, permet elle aussi d'offrir à l'utilisateur une « représentation » des sources sonores ponctuelles sur un espace en deux dimensions. Cette interface peut être gérée de façon automatique, par un envoi de messages en entrée et ne requérir alors aucune action particulière de la part de l'utilisateur. Or, cette représentation n'est dans ce cadre que purement symbolique dans la mesure où la position des sources à l'écran n'informe en rien sur la qualité du son traité et peut même ne pas se refléter du tout d'un point de vue sonore et spatial.

Les interfaces de représentation du champ sonore sont donc des types d'*indicateurs visuels* particuliers, voués à offrir une réelle relation causale entre les différents aspects du son spatialisé et leur représentation graphique par succession d'instantanés se mettant à jour automatiquement dans le temps. Elles peuvent donc ainsi aider à corroborer la caractérisation d'un champ sonore par l'écoute, à travers la mise en exergue graphique de certains paramètres sonores ou perceptifs tels que l'amplitude, l'intensité, la phase, le contenu spectral, de façon globale ou spécifique à un point ou à une certaine zone de l'espace.

La bibliothèque HOA, se devait donc de se doter d'interfaces performantes permettant à l'utilisateur de se représenter les signaux qu'il manipule en fonction du type de décomposition du champ sonore qui y transite. Ceci afin de mieux contrôler les opérations

exercées sur le champ sonore en en visualisant l'effet à chaque moment de la chaîne de traitement.

Ces interfaces nous ont été très utiles lors de la phase de développement et de test de la bibliothèque HOA, notamment pour valider des nouveaux traitements musicaux que nous mettions en place à des ordres de décomposition beaucoup plus élevés par extrapolation[61]. Elles ont pu permettre de vérifier aussi graphiquement la validité de ces mêmes traitements lorsque nous n'avions pas accès à un système de restitution ou que celui-ci n'était pas approprié. Pour le musicien ou le compositeur, elles représentent de la même manière un moyen efficace lui permettant de prévisualiser l'effet d'un traitement qu'il souhaite mettre en place quand il ne dispose pas du secours de l'écoute. Ces interfaces de représentation nous ont aussi largement servi, lors de l'atelier de programmation présenté aux *JIM 2013*, de l'installation *futur en seine* ou encore au cours des présentations diverses de la bibliothèque que nous avons pu effectuer auprès des étudiants de l'Université Paris-VIII pour visualiser et expliciter l'incidence d'un traitement particulier sur le champ sonore à l'aide de ces descripteurs audio.

Nous étudierons dans un premier temps les solutions que nous proposons au sein de la bibliothèque HOA pour la représenter un champ sonore dans le domaine des harmoniques circulaires, ce qui nous donnera aussi l'occasion de mieux définir la nature de cette décomposition. Nous nous intéresserons dans un second temps à une représentation plus commune au sein des pratiques audionumériques, à savoir la représentation du niveau sonore sous la forme d'une interface de type *Peak Programme Meter* que nous avons adaptée aux besoins spécifiques de l'ambisonie. Enfin, nous exposerons une troisième interface, dédiée elle à une représentation spatio-fréquentielle du champ sonore. Nous proposerons pour chacune d'entre elles une lecture des représentations offertes au travers d'un ou plusieurs exemples concrets mettant en exergue leurs potentiels descriptifs et musicaux.

[61] Le système à seize canaux que nous avions à notre disposition lors de la phase de développement et de test de la bibliothèque ne nous permettant pas de monter à des ordres de décomposition supérieurs à sept.

2.1 REPRESENTATION DES HARMONIQUES CIRCULAIRES

Dans la mise en œuvre de la bibliothèque HOA à destination des environnements de programmation modulaires de type MaxMSP ou Pure Data, la plupart des moteurs sonores sont amenés à traiter, que ce soit en entrée et/ou en sortie, des séries de signaux dépendants des harmoniques circulaires. Chaque série de signaux est représentative d'un état du champ sonore à un endroit et à un moment précis de la chaîne de traitement ambisonique. Il nous paraissait dès lors indispensable de fournir aux musiciens et aux compositeurs, un outil performant leur permettant de se représenter graphiquement le champ sonore dans ce domaine particulier. C'est ce que se propose de faire l'interface *hoa.scope~*.

2.1.1 LA DECOMPOSITION EN HARMONIQUES CIRCULAIRES

L'ambisonie, appliquée à des systèmes de restitution tridimensionnel, est basée sur une décomposition du champ sonore en harmoniques sphériques. Cependant, dans le cadre de la bibliothèque HOA, nous nous limitons pour l'instant à des traitements ambisoniques en deux dimensions, et donc nous basons sur une décomposition du champ sonore en fonctions spatiales non-plus sphériques mais circulaires.

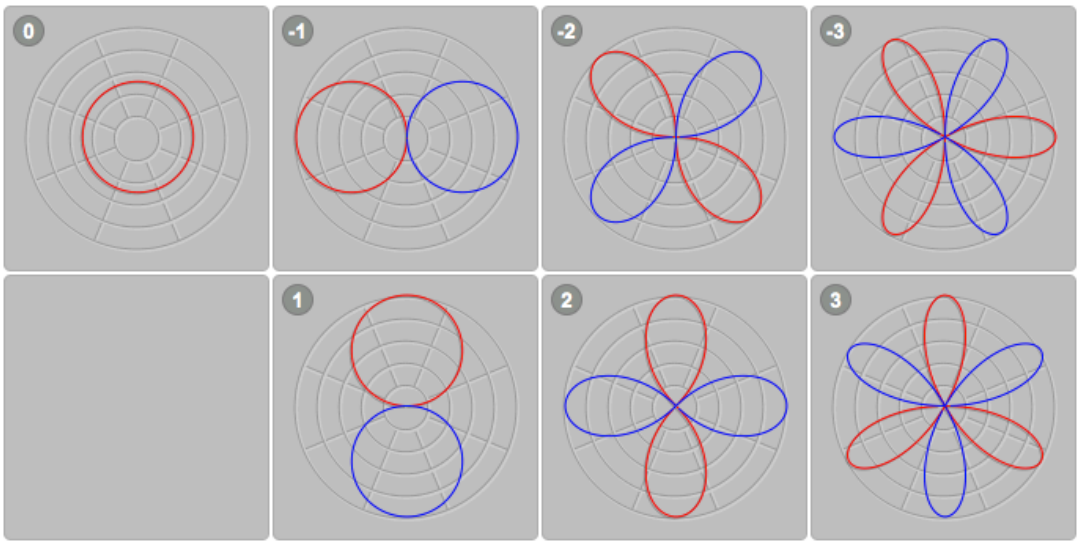

Figure 45 : représentation des sept premières harmoniques circulaires, les lobes positifs sont représentés en rouge, les lobes négatifs en bleu.

La décomposition du champ sonore en harmoniques sphériques ou circulaires partage un certain nombre de similitudes avec la décomposition en série de Fourier, où une fonction

périodique est décomposée en une somme pondérée de fonctions sinusoïdales [Colafrancesco, 2012]. Les harmoniques circulaires sont des fonctions spatiales dépendantes d'un ordre *n* et d'un indice *i* qui prennent en variable un angle α en radian. Mis à part l'ordre 0 qui ne possède qu'une seule harmonique d'indice 0, et que nous considérons alors comme positive, chaque ordre *n* comprend deux harmoniques d'indice respectifs *-n* et *n*, la première étant dite négative, la seconde positive. La [Figure 45] offre une représentation des sept premières harmoniques en fonction de leur indice pour un ordre de décomposition pouvant donc aller jusqu'à trois[62]. L'une des représentations d'un champ sonore possible dans le domaine des harmoniques circulaires, pour un ordre de décomposition donné, consiste donc, de façon analogue à la reconstitution d'une série de Fourier, à représenter la somme pondérée de l'ensemble des fonctions spatiales que cet ordre comprend.

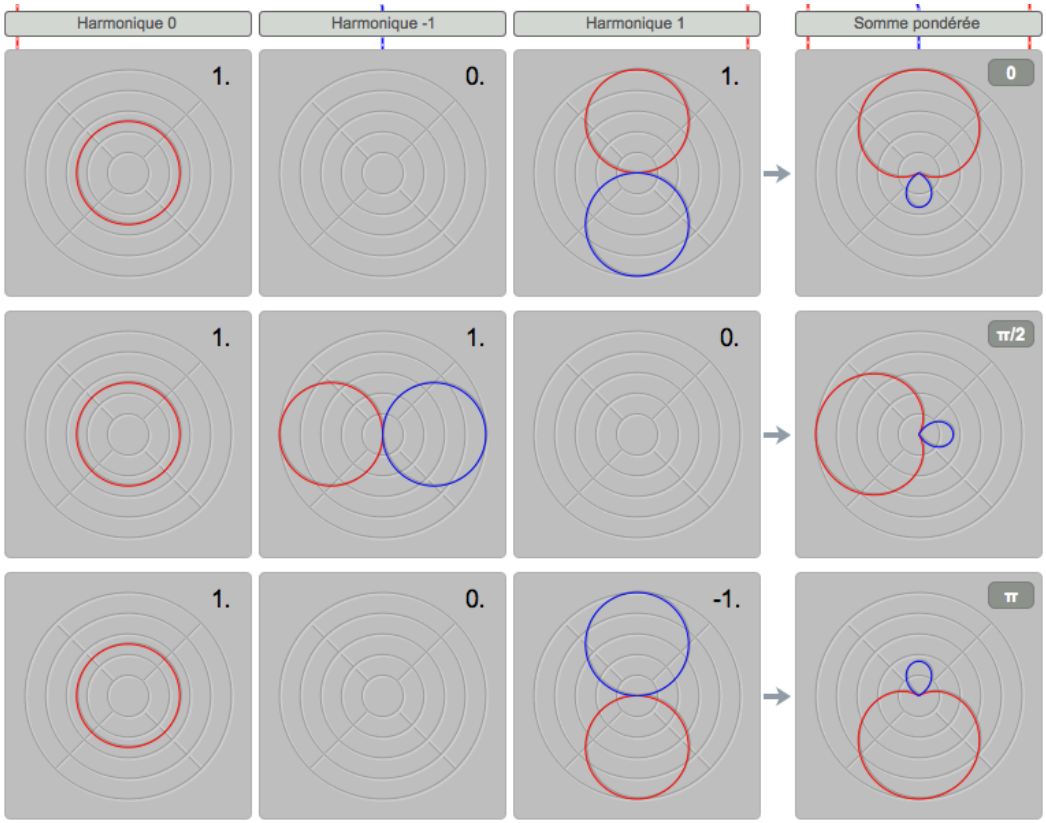

Figure 46 : représentation des 3 harmoniques d'un ordre de décomposition 1 avec leurs valeurs de gains respectives (en noir en haut à droite de chaque visualisations), contribuant chacune à générer les figures de directivité représentées sur la droite, correspondant respectivement à des angles d'incidence de 0, π/2 et π (de haut en bas).

[62] Il est courant, en électronique notamment, de représenter la polarité grâce à la couleur rouge pour désigner un signal positif, et bleu pour un signal négatif. Nous avons choisi de garder ce même code au sein de notre représentation en colorant les fonctions positives en rouge, les fonctions négatives en bleu.

2.1.2 *PRESENTATION ET LECTURE DE LA REPRESENTATION*

L'interface *hoa.scope~* propose donc une vue bidimensionnelle de l'espace servant à la fois à représenter les fonctions harmoniques individuellement [Figure 45], mais aussi un champ sonore complet sous la forme d'une somme pondérée de ses fonctions spatiales individuelles; cette dernière représentation étant la plus communément utilisée. A travers l'exemple donné par la [Figure 46], elle permet donc de mettre en exergue le principe de l'encodage ambisonique, en montrant que le champ sonore généré pour un ordre de décomposition donné (dans ce cas égal à 1), consiste simplement à faire varier l'amplitude de chacune des composantes spatiales individuelle dépendant de cet ordre[63].

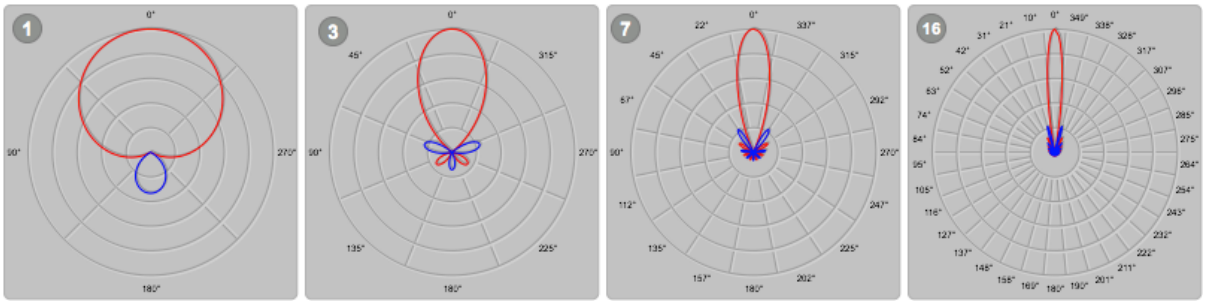

Figure 47 : représentation de la somme pondérée des harmoniques circulaires d'un signal d'amplitude 1 encodé avec un angle d'incidence de 0° pour un ordre de décomposition 1, 3, 7 et 16.

Dans le cadre d'une décomposition de Fourier, la reconstitution est d'autant plus fine que le nombre de composantes fréquentielles augmente. Il en va de même pour une représentation dans le domaine des harmoniques sphériques ou circulaires du champ sonore dans la mesure où, plus nous considérons de composantes spatiales d'ordre élevées, plus le champ sonore peut être représenté en détail et décodé avec plus de précision [Colafrancesco, 2012]. La [Figure 47] met en évidence cette augmentation de la résolution spatiale en fonction de l'ordre de décomposition : le lobe proéminant se resserre à mesure que l'ordre de décomposition augmente, donnant ainsi à la source sonore encodée une résolution angulaire d'autant plus fine.

Le nombre d'entrées de l'objet *hoa.scope~*, fonction de l'ordre de décomposition, correspondra donc au nombre d'harmoniques que celui-ci comprend. Nous pouvons aussi

[63] Nous renvoyons le lecteur à [Guillot, 2013] qui décrit avec précision les fondements mathématiques inhérents à l'encodage ambisonique.

noter au passage que le nombre d'axes et d'angles discrétisant le cercle augmente lui aussi en fonction de l'ordre pour indiquer une résolution spatiale plus importante [Figure 47].

Les contributions des harmoniques circulaires sont normalisées à une amplitude mesurée à 1 afin que celles-ci ne dépassent pas le cercle de plus gros rayon au sein de notre représentation ; il se peut donc que les lobes de plus faible amplitude demeurent donc trop petit sur la représentation pour les apercevoir si la plus grosse contribution dépasse cette amplitude limite. D'autre part, il peut arriver aussi que l'on ait à représenter un champ sonore de faible amplitude. Pour permettre à l'utilisateur d'avoir une représentation correcte de celui-ci dans ce cas spécifique, nous avons intégré à l'objet un paramètre de gain (par défaut égal à 1) qui peut être appliqué à l'ensemble des contributions pour visualiser avec plus de précision le champ sonore, en effectuant une sorte de « zoom » sur la représentation.

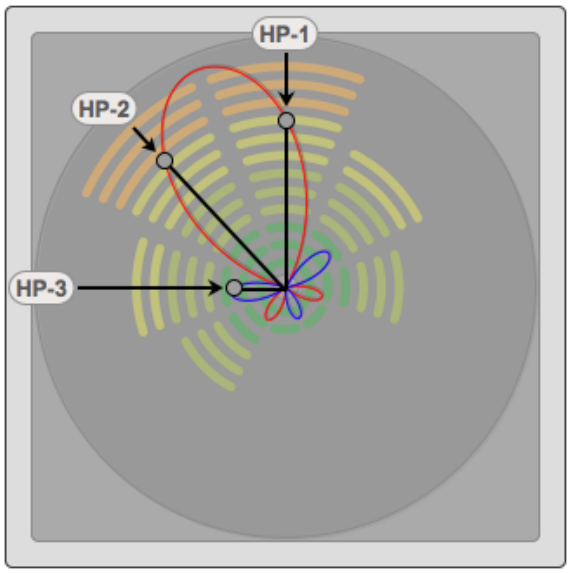

Figure 48 : représentation d'un signal d'amplitude 1 encodé pour un ordre de décomposition 3, avec un angle d'incidence de π/8, au sein de l'interface *hoa.scope~* superposée à l'interface *hoa.meter~*[64]. Les points placés sur les lobes ainsi que les modulomètres circulaires représentent l'amplitude et la phase des signaux dépendant des haut-parleurs physiques après une opération de décodage par projection pour un système de restitution à huit haut-parleurs.

La représentation graphique du champ sonore sous la forme d'une somme de contributions au sein de l'objet *hoa.scope~* est assimilable à une opération de décodage ambisonique par projection sur chacun des degrés du cercle. Comme nous le fait remarquer Guillot, « le processus [de décodage par projection] revient […] à envoyer le signal de la

[64] Cet objet sera discuté en détail dans la partie qui suit.

source sonore dans tous les haut-parleurs et à appliquer à chaque haut-parleur un gain correspondant à la distance entre le centre du cercle et le point de la contribution générale du champ sonore correspondant à l'angle d'incidence du haut-parleur » [Guillot, 2013]. Cette représentation permet donc, comme le montre la [Figure 48], d'obtenir des informations sur l'amplitude et la phase du signal pour n'importe quel angle donné du cercle de restitution ambisonique.

En apprenant à « lire » cette représentation, l'utilisateur peut donc assez aisément se figurer le champ sonore, le caractériser et même définir le type de traitement qu'on a pu lui appliquer. Les contributions des harmoniques circulaires représentées à la [Figure 6] nous ont par exemple permis de mettre en évidence, le traitement de compensation de la distance mis en place au sein du moteur sonore *hoa.map~*, en représentant la baisse d'amplitude de la source quand la distance de celle-ci augmente en dehors du cercle de haut-parleurs et la variation de sa résolution angulaire lorsqu'elle se trouve à l'intérieur grâce à une représentation claire de la somme de chaque contributions.

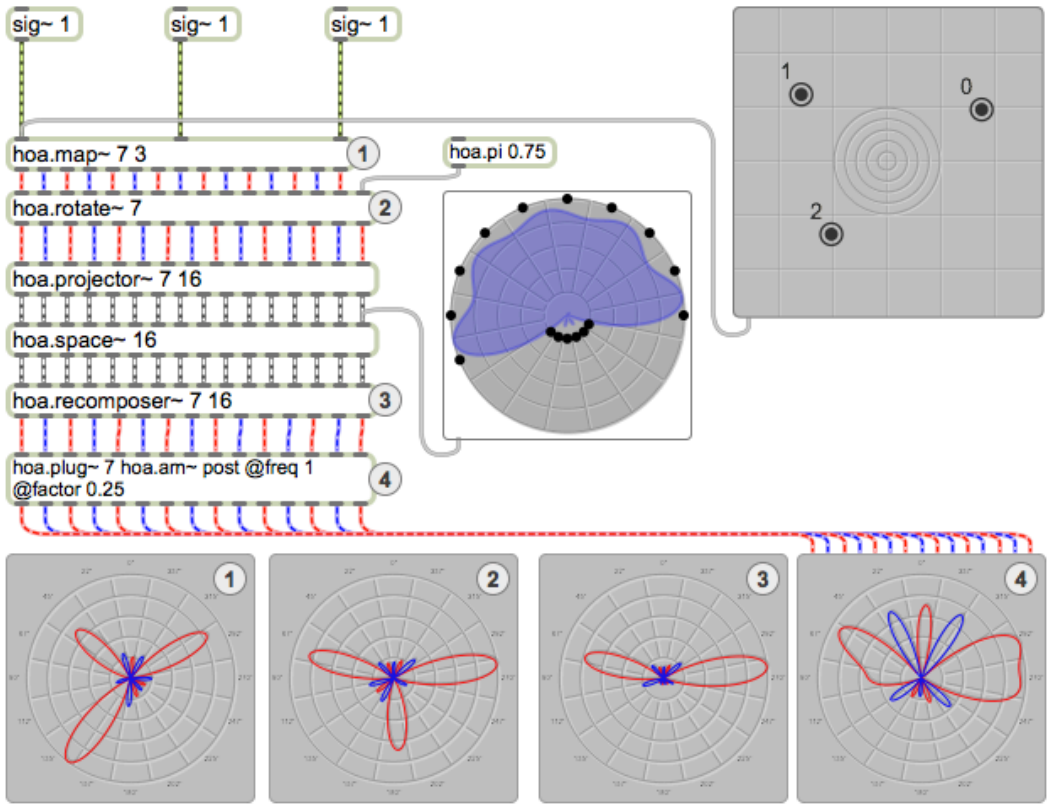

Figure 49 : représentation d'un patch MaxMSP au sein duquel l'objet *hoa.scope~* est utilisé pour représenter le champ sonore à quatre instants différents d'une chaîne de traitement ambisonique. Encodage de trois signaux [1], rotation du champ sonore de 3π/4 [2], recomposition après filtrage spatial [3], modulation d'amplitude des signaux dépendants des harmoniques [4].

Prenons l'exemple donné par la [Figure 49]. Celle-ci représente quatre états du champ sonore à des instants différents de la chaîne de traitement ambisonique à l'aide de l'interface *hoa.scope~*. On distingue ainsi clairement, à l'aide des trois lobes rouges proéminents de la [Figure 49.1], trois sources sonores ponctuelles d'amplitude différente et de phase positive. En se fiant à la répartition des lobes à la [Figure 49.2], on pourra en déduire qu'une rotation a été appliquée à l'ensemble du champ sonore. De même, le fait que le lobe présent à l'arrière du champ sonore ait disparu entre la [Figure 49.2] et la [Figure 49.3] peut témoigner d'un filtrage spatial exercé à cet endroit. La dernière figure dévoile enfin une déformation du champ sonore due à une modulation d'amplitude des différentes harmoniques lui donnant un caractère plus diffus[65]. On voit assez nettement sur la [Figure 49.4] un élargissement des différents lobes dû à la variation d'amplitude de chaque contribution, néanmoins, comme toutes représentations visuelles d'un événement temporel, celle-ci souffre aussi de son caractère instantané. Nous invitons donc le lecteur à se figurer plus nettement ce traitement dans le temps.

Grâce à cette interface originale, le musicien et le compositeur ont donc à leur disposition un outil puissant et particulièrement bien adapté à l'ambisonie leur permettant de se représenter aisément le champ sonore dans le domaine des harmoniques circulaires. Il existe néanmoins d'autres manières de se représenter le champ sonore qui pourront venir compléter celle-ci. Nous nous proposons donc de revenir sur l'une d'entre elles, dédiée à la représentation du niveau de l'ensemble des haut-parleurs du système de restitution.

[65] Nous traitons de l'outil *hoa.plug~* qui permet d'établir ce type de traitement dans la partie lui étant consacrée. Pour plus d'informations sur la modulation d'amplitude appliquée au domaine des harmoniques circulaires ou sur d'autres traitements de synthèse de champs diffus, le lecteur pourra se référer à [Guillot, 2013].

2.2 REPRESENTATION DES CONTRIBUTIONS DES HAUT-PARLEURS

Au cours de l'élaboration de la bibliothèque HOA, nous avons très vite éprouvé le besoin de visualiser l'amplitude correspondant à chacun des haut-parleurs afin de vérifier graphiquement la validité de nos traitements. S'est donc posée la question de la meilleure représentation à adopter afin de rendre compte au mieux de l'amplitude de ces signaux dans le contexte particulier du multicanal, et plus précisément de l'ambisonie dans lequel nous nous placions.

2.2.1 REPRESENTATION CLASSIQUE DU NIVEAU SONORE

Pour représenter le niveau sonore d'un signal audio monophonique, il est courant de recourir à un modulomètre de type *PPM*[66] permettant d'indiquer le niveau de crête de celui-ci. Ce type de descripteur audio, utilisé aussi bien en analogique qu'en numérique, peut revêtir des formes diverses en fonction de leur norme, des niveaux de référence ou de l'affichage utilisé.

Une des représentations courante au sein des interfaces matérielles, consiste à afficher sur une échelle en décibels, verticale ou horizontale, allant de -60dB ou -50dB à 0dB ou +6dB, des diodes de couleurs discrétisant celle-ci. Le niveau du signal audio peut alors être représenté graphiquement grâce à l'allumage ou l'extinction progressive de ces diodes. Celles-ci prennent en général une couleur verte pour indiquer un signal de faible amplitude, jaune et orange pour une amplitude moyenne et forte, et rouge lorsque le signal sature.

Les interfaces logicielles de type *PPM* se calquent en général sur les interfaces matérielles que nous venons d'évoquer à l'instant, en reproduisant graphiquement l'aspect et les comportements de celles-ci. L'échelle de valeurs demeure en décibels et les diodes sont alors représentées par des rectangles de pixels de la même couleur que ces dernières lorsqu'elles sont allumées, et de couleur noire quand ils doivent signifier qu'elles sont éteintes. Il est à noter que les interfaces logicielles, par rapport aux matérielles, offrent souvent l'avantage d'être plus paramétrable (nombre de diodes, couleurs, échelle variable, taille). Au sein de l'environnement de programmation MaxMSP, c'est l'objet *meter~* qui intègre la fonction d'un *PPM* en offrant la possibilité de représenter l'amplitude d'un signal monophonique.

[66] *Peak Programme Meter*.

2.2.2 ADAPTATION DE LA REPRESENTATION AU CONTEXTE AMBISONIQUE

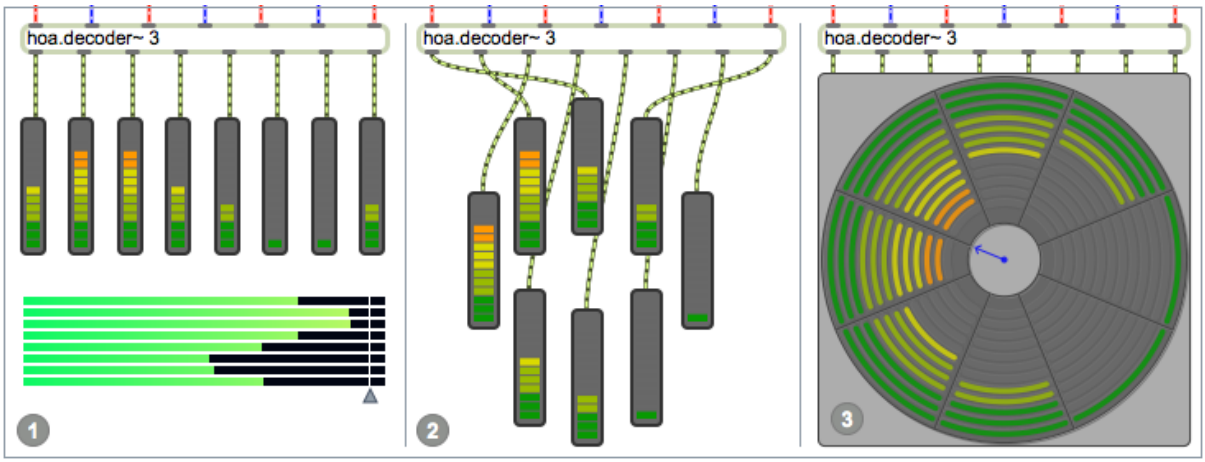

Figure 50 : représentation des contributions de huit haut-parleurs dans un *patch* MaxMSP pour l'encodage d'un signal d'amplitude 1 et un angle d'incidence approximatif de 0.376 radians, pour un ordre de décomposition 3. Trois interfaces graphiques différentes : l'objet *live.gain~* [1, en bas], une série d'objets *meter~* en [1, en haut] et [2], l'interface *hoa.meter~* en [3].

Dans un contexte multicanal, il est courant d'utiliser une série horizontale ou verticale de *meter~* pour représenter les niveaux sonores correspondant aux sorties audio physiques [Figure 54.1]. Néanmoins, si cette représentation fonctionne assez bien dans un contexte stéréo, elle supporte bien souvent assez mal une configuration plus importante de haut-parleurs dans la mesure où elle ne traduit plus la répartition spatiale de ceux-ci, et n'offre donc plus de correspondance avec l'écoute. Pour remédier à cela, les compositeurs ont donc souvent recours à des *parades* graphiques en disposant chaque interface à leur convenance, tel que l'illustre la [Figure 54.2]. Cette répartition graphique offre l'avantage d'être plus cohérente avec le système de restitution utilisé, mais implique aussi un temps de *patching* conséquent qui doit être de surcroît renouvelé dès lors que le contexte de diffusion s'en trouve changé. Partant de ce constat, nous avons donc décidé de développer une interface graphique originale, dédiée à la représentation du champ sonore en ambisonie, sous la forme de contributions de haut-parleurs, capable de traduire graphiquement la qualité circulaire de leur répartition spatiale [Figure 54.3]. Nous parlons de contributions des haut-parleurs dans la mesure où chacun d'entre eux contribue de façon significative à la restitution d'un champ sonore en un point de l'espace situé idéalement au centre du cercle formé par ceux-ci.

L'interface *hoa.meter~* peut donc être vue comme une série de modulomètres de type *PPM*, répartie de manière circulaire et dirigés vers le centre, permettant chacun de représenter le niveau de crête d'un signal audio dont l'amplitude varie entre -1 et 1. Une attention

particulière a été donnée, lors du développement, à faire en sorte que le comportement de chaque modulomètre soit en tous points similaire à celui de l'objet *meter~* natif de MaxMSP, afin que la transition vers cette nouvelle interface se fasse de la manière la plus transparente qui soit pour un utilisateur habitué à cette dernière. En adoptant des stratégies de *reverse engineering* sur l'objet *meter~,* nous avons donc pu reproduire l'ensemble de ses fonctionnalités au sein de notre interface. Les *leds* de chaque modulomètre discrétisent une échelle en décibel variable en fonction du nombre total de *leds* et du nombre de décibels que chacune d'elles est à même de représenter[67]. Ces *leds* sont représentées par des arcs de cercles dont les couleurs varient du vert au orange en fonction de l'intensité du signal. Une *led* de couleur rouge apparaît lorsque le signal reçu par ce modulomètre sature, c'est à dire quand l'amplitude atteint une amplitude absolue supérieure à 1 soit 0dB, pour disparaître après deux secondes si le signal perd en intensité[68].

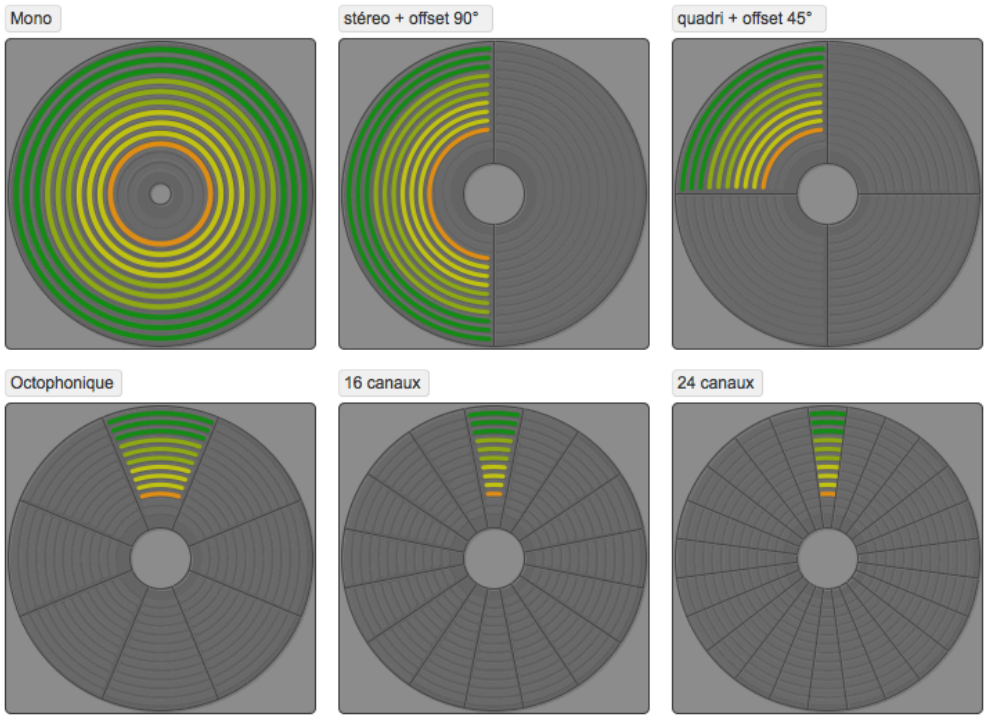

Figure 51 : représentation de l'amplitude d'un signal d'amplitude 0.5 dans le premier canal en fonction du nombre de haut-parleurs et de l'*offset* spécifié : configuration Mono, 2 canaux pour un système stéréo ou binaural, quadriphonique avec un offset de 45°, octophonique, 16 et 24 canaux (de gauche à droite et de haut en bas).

[67] Avec une configuration de 12 *leds* et un nombre de décibel par *led* égal à 3, les modulomètres seront donc à même de représenter des variations d'intensité entre (12 * -3 = -36dB) et 0dB. On peut donc ajuster la précision de l'échelle en jouant sur ces paramètres.

[68] Nous invitons le lecteur à se référer à la documentation des objets *meter~* et *hoa.meter~* disponible dans MaxMSP pour plus d'informations sur le fonctionnement de ces objets.

Nous venons de voir que l'objet *hoa.meter~* se composait de modulomètres similaires par leurs comportements à celui de l'objet *meter~*. Néanmoins, la force de cet objet réside surtout en sa capacité à pouvoir représenter le niveau sonore d'un ensemble de signaux correspondant aux haut-parleurs. Elle dispose donc d'un ensemble de fonctions spécifiques à cet usage que nous discutons à présent.

Il est donc désormais possible de visualiser l'intensité de un à 64 signaux sur une même interface en spécifiant simplement le nombre de canaux [Figure 51]. Par son dynamisme, celle-ci peut donc s'adapter à différents systèmes de restitution où les haut-parleurs sont répartis en cercle autour des auditeurs, et offrir ainsi une topographie sonore plus adéquate du contexte de diffusion.

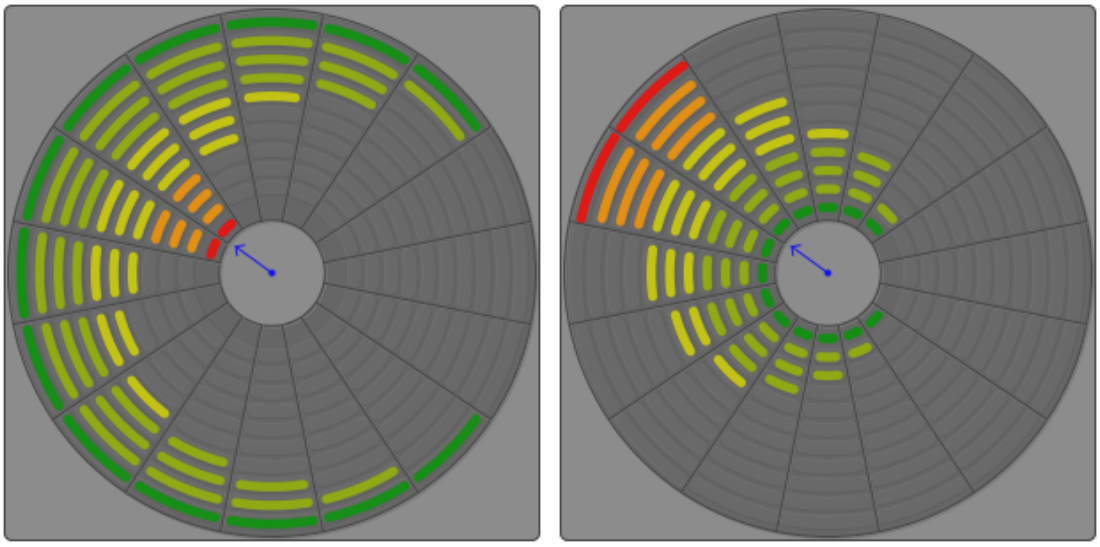

Figure 52 : représentation des contributions de 16 haut-parleurs pour un signal d'amplitude 2 encodé avec un angle d'incidence de π/3. Les modulomètres sont orientés vers le centre (à gauche) et l'extérieur du cercle (à droite).

Cette interface a été l'une des premières à avoir été développée au sein du projet HOA, elle a donc pu bénéficier de nombreuses améliorations en fonctions de nos besoins spécifiques et des demandes formulées par les premiers utilisateurs.

Nous avons par exemple pu laisser le choix de l'orientation des modulomètres au désir de l'utilisateur qui peut alors décider si la direction de l'échelle des décibels doit être centripète [Figure 52.1] ou centrifuge [Figure 52.1]. La première représentation mettant selon nous plus l'accent sur le niveau des haut-parleurs, la seconde sur la perception égocentrée de l'auditeur. Il est aussi possible d'inverser le sens de rotation des canaux afin de s'adapter à des moteurs

sonores pensés pour une répartition horaire ou antihoraire des haut-parleurs. Des adaptations comme celles-ci ont été faites pour faciliter l'usage de cette interface avec d'autres bibliothèques de spatialisation qui ne partagerait pas les mêmes conventions notamment en termes de numérotation des canaux.

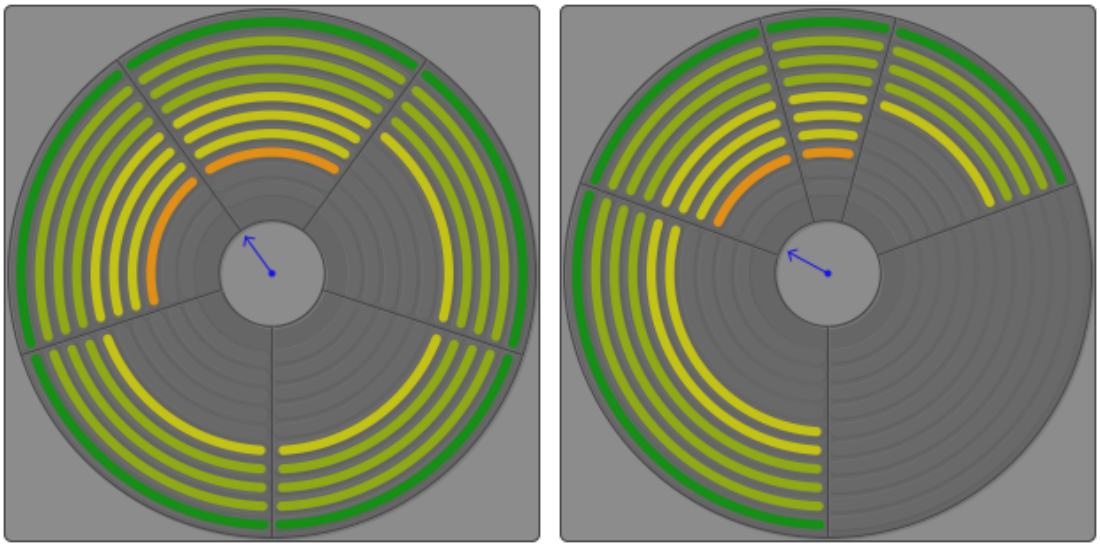

Figure 53 : représentation des contributions des haut-parleurs pour l'encodage d'un même signal d'amplitude 1 avec un angle d'incidence de $\pi/5$ et pour un ordre de décomposition 2. Décodage ambisonique pour 5 haut-parleurs (à gauche), décodage irrégulier pour 5 haut-parleurs en mode *panning* (à droite).

Le développement de cette interface, comme ce fut le cas pour toutes les autres, s'est fait en symbiose avec celui des moteurs sonores associés. Bien que l'ambisonie se limite à un stricte placement des haut-parleurs, les recherches menées dans le cadre du projet HOA ont notamment permis d'élargir les possibilités de décodage vers des systèmes de restitution irréguliers de haut-parleurs (stéréo, 5.1, 7.1 ou autre)[69]. Il a donc fallu refléter ces changements graphiquement au sein de l'interface en laissant à l'utilisateur la possibilité de spécifier chacun des angles des haut-parleurs de l'interface afin de s'adapter à la configuration particulière qu'il utilise [Figure 53].

Cette interface intègre aussi un descripteur audio représentant « la direction moyenne de provenance de l'énergie » [Daniel, 2001], symbolisé par une flèche au centre de l'objet. La divergence de ce vecteur énergie que laisse apparaître l'interface de droite par rapport à celle de gauche [Figure 53] est due aux techniques de décodages particulières que nous effectuons

[69] Le lecteur pourra se référer à ce propos au dernier chapitre où nous proposons un module de configuration dynamique du décodage dans lequel l'interface *hoa.meter~* s'y retrouve intégrée.

pour s'adapter à des configurations irrégulières de haut-parleurs, comme dans cet exemple pour un système de type 5.1[70].

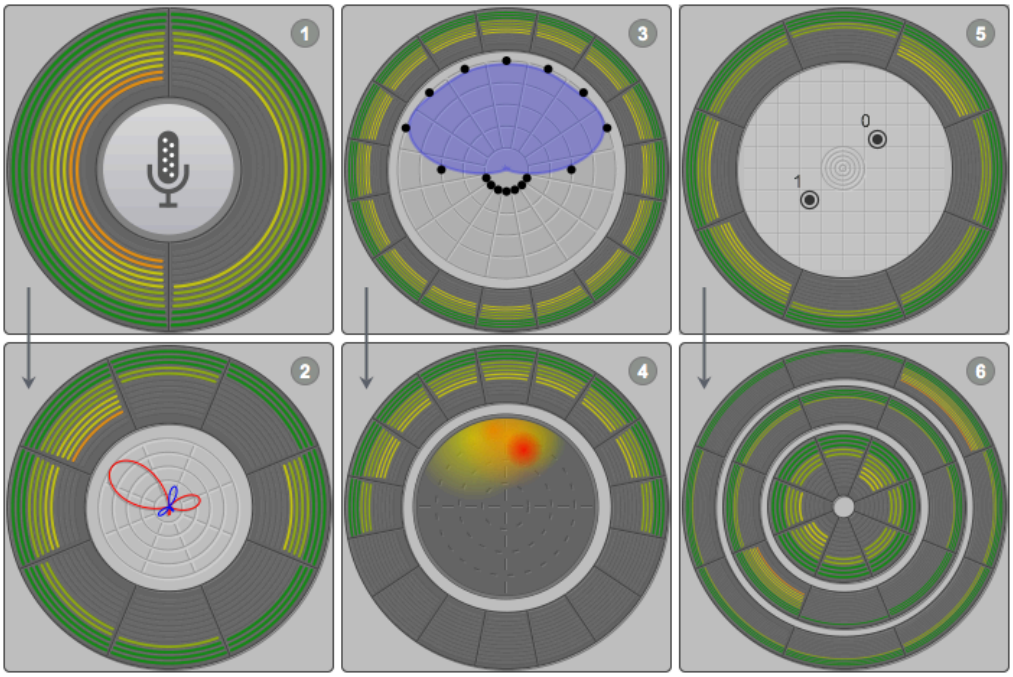

Figure 54 : représentation de l'interface *hoa.meter~* pour MaxMSP, superposée à d'autres interfaces de contrôle ou de visualisation de la bibliothèque HOA.

L'interface *hoa.meter~* a été rendue extrêmement configurable afin que l'utilisateur puisse la personnaliser en fonction de ses besoins spécifiques pour l'adapter et l'intégrer à un maximum de contexte musicaux différents. Une des personnalisations possibles se situe, entre autres, au niveau du rayon du cercle intérieur qui peut être agrandi ou rétréci afin de laisser une place libre au centre, en rétractant le rayon de chaque modulomètre. Cette place libre peut alors être exploitée par l'utilisateur en y plaçant d'autres interfaces de contrôle ou de visualisation pour mettre ainsi en exergue d'autres informations en corrélées aux contributions des haut-parleurs. La configuration visible à la [Figure 54.1] permet par exemple d'insister sur la provenance du signal en représentant le niveau sonore de deux des canaux d'un microphone, à travers la symbolique apportée par l'interface *ezadc~* placée au centre, et une configuration de type stéréo de l'objet *hoa.meter~*. Ces deux signaux, convertis en sources ponctuelles par un encodage dans le domaine des harmoniques circulaire, forment

[70] Nous renvoyons à [Guillot, 2013] pour des informations détaillées sur les techniques de décodages et leurs implémentations au sein de la bibliothèque HOA. Le lecteur peut se référer aussi à l'aide de l'objet *hoa.decoder~*.

un champ sonore que la complémentarité des interfaces *hoa.scope~* et *hoa.meter~* de la [Figure 54.2] nous aide à interpréter. La superposition des interfaces présentées aux [Figure 54.3 et Figure 54.4] met l'accent sur l'incidence d'un traitement ambisonique sur la contribution de chacun des haut-parleurs. La figure supérieure indique le niveau sonore avant un traitement de filtrage spatial paramétré par l'objet *hoa.space~*, la partie inférieure insiste sur la conséquence de ce traitement sur le champ sonore en représentant la perte de niveau des haut-parleurs situés à l'arrière ainsi que sa nouvelle répartition spatio-fréquentielle à travers l'interface *hoa.spectrum~*[71]. Enfin, la dernière série de figures démontre une façon originale de mettre en avant les composantes individuelles d'un champ sonore composite en représentant séparément les contributions des haut-parleurs générées par l'encodage successif de deux sources sonores virtuelles [Figure 54.5 et Figure 54.6].

Ces exemples illustrent l'aspect modulaire de la bibliothèque HOA qui s'exprime aussi par la flexibilité des interfaces graphiques existantes autorisant la création de nouvelles représentations par recontextualisation et personnalisation.

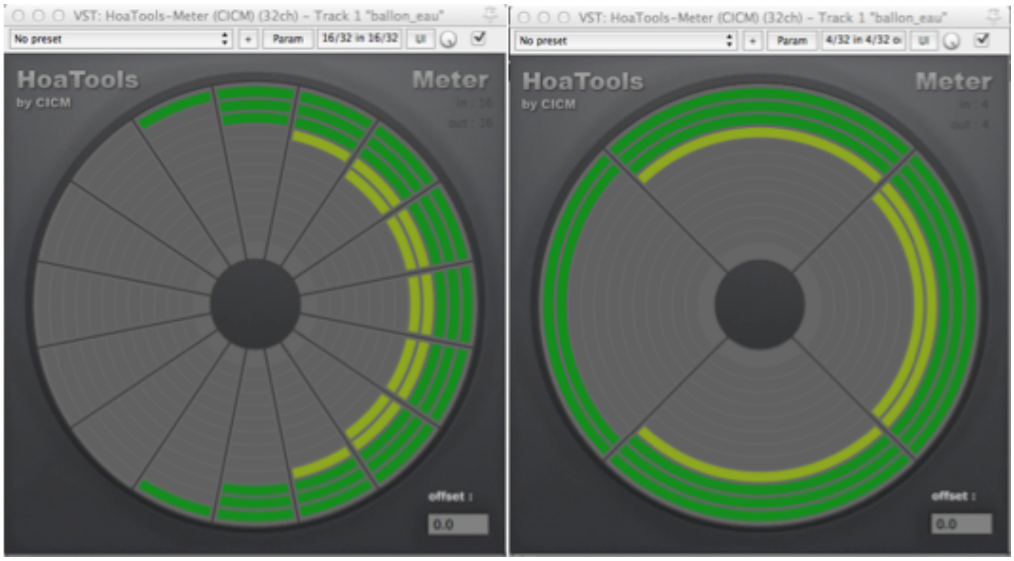

Figure 55 : représentation des contributions de 16 haut-parleurs (à gauche) et 4 haut-parleurs (à droite) au sein du plugin VST HoaMeter pour une source sonore encodée avec un angle d'incidence de $-\pi/2$ et une distance approximativement égale au 2/3 du cercle des haut-parleurs par rapport à l'auditeur situé au centre.

Pour permettre d'utiliser cette interface dans d'autres contextes de création musicale, et notamment au sein de logiciels de type *time-line*, nous avons crée une version VST de

[71] Cette interface est discutée plus en détail dans la partie consacrée à la représentation spatio-fréquentielle du champ sonore.

celle-ci. Chargée au sein d'un logiciel hôte tel que *Reaper*, cette interface vient compléter la représentation habituelle des niveaux d'amplitude en fournissant des informations plus adaptées à un contexte ambisonique ou multicanal où le système de restitution sur haut-parleurs utilisé est réparti en cercle autour de l'auditeur [Figure 55]. Tout comme le plugin *HoaMap* présenté plus haut, celui-ci s'adapte automatiquement à la configuration audio choisie par l'utilisateur en associant le nombre de haut-parleurs représentés au nombre d'entrées/sorties du plugin le précédant dans la chaîne de traitement DSP[72], ou de la piste au sein de laquelle il a été chargé. Notons enfin que l'interface *hoa.meter~* est aussi disponible au sein de la mise en œuvre Pure Data de la bibliothèque HOA[73].

L'interface *hoa.meter~*, particulièrement aboutie, permet donc comme nous avons pu le voir, d'offrir une meilleure représentation des niveaux sonores dépendants des haut-parleurs physique dans le cadre de l'ambisonie. Par les nombreuses améliorations qui lui ont été apportées, elle s'adapte aussi à des configurations ou système de restitution sur haut-parleurs plus particuliers. Nous revenons à présent sur l'interface *hoa.spectrum~* que nous avons pu apercevoir brièvement à la [Figure 54.4].

[72] *Digital Signal Processing*.

[73] Le lecteur pourra se référer à la mise en œuvre Pure Data de la bibliothèque HOA pour en avoir un aperçu.

2.3 REPRESENTATION SPATIO-FREQUENTIELLE DU CHAMP SONORE

Nous venons de voir deux interfaces dédiées à la représentation du champ sonore à travers la présentation des objets *hoa.scope~* et *hoa.meter~*. La première permet d'obtenir une représentation du champ sonore sous la forme d'une somme de fonctions spatiales ; nous fournissant alors des informations sur la phase et l'amplitude des signaux en fonction d'un angle d'incidence donné. La seconde offrant une représentation de l'amplitude des haut-parleurs une fois le champ sonore décodé pour un système de restitution défini. Afin de compléter ces représentations nous avons décidé de développer une nouvelle interface qui puisse s'ajouter à ces dernières afin de décrire avec toujours plus de précision un champ sonore, en fournissant cette fois-ci à l'utilisateur des informations sur son contenu fréquentiel. Cette interface n'est pas aussi aboutie que les deux que nous avons précédemment présentées, mais constitue tout de même un prototype très avancé qui pourra prochainement s'intégrer aux futurs livrables de la bibliothèque HOA pour MaxMSP.

2.3.1 FONCTIONNEMENT DE L'INTERFACE

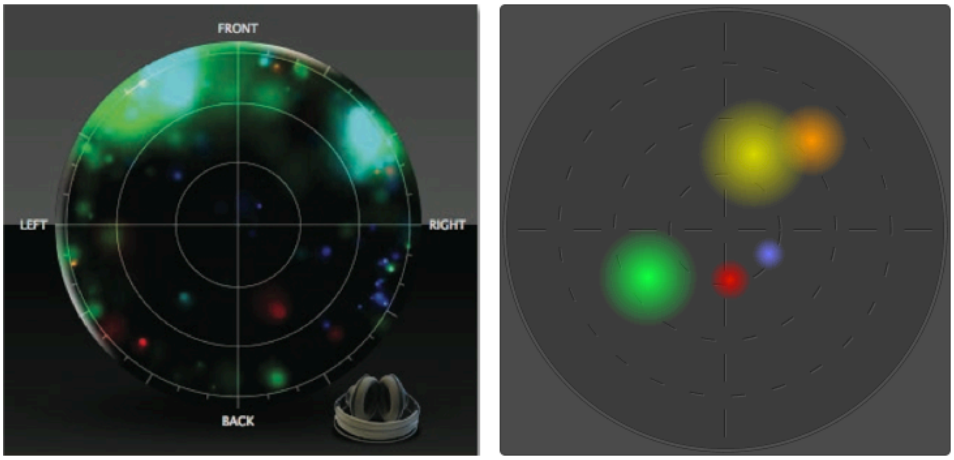

Figure 56 : l'interface de visualisation spatio-fréquentielle du plugin *Harpex-B* représentant un champ sonore caractérisé par une position plutôt frontale et un contenu fréquentiel relativement aigu (à gauche) ; représentation d'un champ sonore à caractère omnidirectionnel et au contenu fréquentiel médium au sein de l'interface *hoa.spectrum~* pour MaxMSP (à droite).

Le plugin *Harpex-B* [Harpex, 2012], que nous avons déjà eu l'occasion d'évoquer dans le contexte de la présentation de l'interface *hoa.recomposer*[74], dispose, en plus de la vue permettant une manipulation des microphones virtuels, d'une vue particulière destinée à

[74] Le lecteur pourra se reporter à la partie traitant de la distorsion de la perspective.

représenter les données spatiales et fréquentielles d'un champ sonore enregistré grâce à leur microphone *soundfield* [Figure 56, à gauche]. Celle-ci nous a donc inspiré la création d'une interface graphique dédiée à la même fonction au sein de la mise en œuvre MaxMSP de la bibliothèque HOA. Cette interface, que nous avons appelée *hoa.spectrum~*, offre donc une vue bidimensionnelle de l'espace qui représente le contenu spatio-fréquentiel d'un champ sonore, décodé pour un nombre de haut-parleurs spécifique, en affichant une série de disques au sein de cet espace, aux couleurs, rayons et positions variables [Figure 56, à droite]. Les couleurs de ces disques correspondent chacune à une bande de fréquence différente du champ sonore. L'amplitude spécifique à une bande de fréquence est représentée par le rayon de son disque associé. Enfin, les coordonnées spatiales de ceux-ci témoignent du point de l'espace où l'amplitude de la bande de fréquence considérée du champ sonore est la plus élevée.

Dans la mise en œuvre actuelle, le nombre de bandes spectrales considérées par l'objet est paramétrable entre 1 et 5. Il peut donc y avoir jusqu'à 5 points représentatifs de la masse spectrale du champ sonore simultanément affichés sur l'interface. L'amplitude de chaque bande de fréquences du champ sonore est obtenue en discrétisant le spectre sonore par filtrages successifs du signal de chaque canal.

	1 bande	2 bandes	3 bandes	4 bandes	5 bandes
Freq-1	2400 Hz	2400 Hz	200 Hz	200 Hz	200 Hz
Freq-2	/	2400 Hz	2400 Hz	600 Hz	300 Hz
Freq-3	/	/	4000 Hz	2400 Hz	600 Hz
Freq-4	/	/	/	4000 Hz	2400 Hz
Freq-5	/	/	/	/	4000 Hz

Table 1 : fréquences de coupure des différents filtres en fonction du nombre total de bandes configuré dans l'objet *hoa.spectrum~*.

Pour une configuration à une seule bande, l'amplitude de celle-ci est obtenue grâce à un filtrage de type *highshelf*. Pour une configuration où le nombre de bandes est supérieur ou égal à deux, l'amplitude de la première bande est obtenue par filtrage de type *lowpass*, la dernière par filtrage de type *highpass*, les filtrages permettant d'obtenir les amplitudes des bandes intermédiaires sont de type *bandpass*. Les fréquences de coupure des différents filtres cités, en fonction du nombre total de bandes considérées, sont indiquées par la [Table 1].

Grâce à cette technique de discrétisation du champ sonore nous obtenons donc une série de valeurs d'amplitude égale au nombre de haut-parleurs factorisé par le nombre de bandes de fréquences, nous permettant ensuite de construire notre représentation graphique. La position spatiale de chaque disque est obtenue en calculant les coordonnées du vecteur énergie correspondant à chacune des bandes de fréquences ; le rayon des disques quant à lui, correspond à une moyenne de l'amplitude de chacune d'entre elles. Enfin, nous leur attribuons des couleurs variant du rouge au bleu, en fonction du nombre total de bandes, pour indiquer des régions spectrales allant respectivement du grave à l'aigu.

2.3.2 LECTURE DE L'INTERFACE

Nous proposons à présent une lecture de la représentation fournie par cette interface à travers une série d'exemples qui pourront témoigner autant de son potentiel musical que de ses limites actuelles[75].

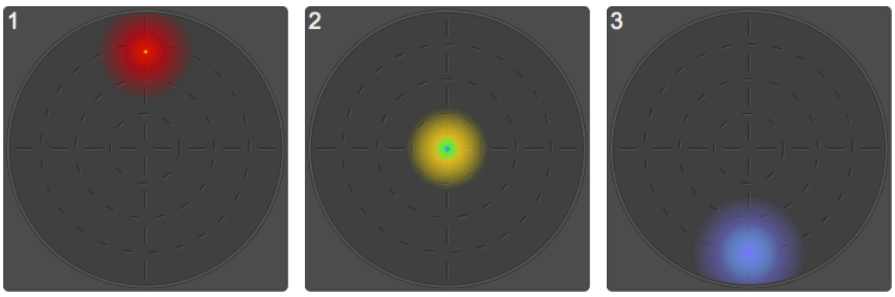

Figure 57 : représentation dans le temps, au sein de l'interface *hoa.spectrum~ pour MaxMSP,* d'une variation fréquentielle du grave à l'aigu obtenue par filtrage d'un bruit blanc avec des filtres *passe-bande* aux fréquences centrales de 50Hz, 400Hz et 6000Hz (de gauche à droite). Déplacement simultané du bruit blanc de l'avant à l'arrière du champ sonore au sein du cercle de haut-parleurs (de gauche à droite).

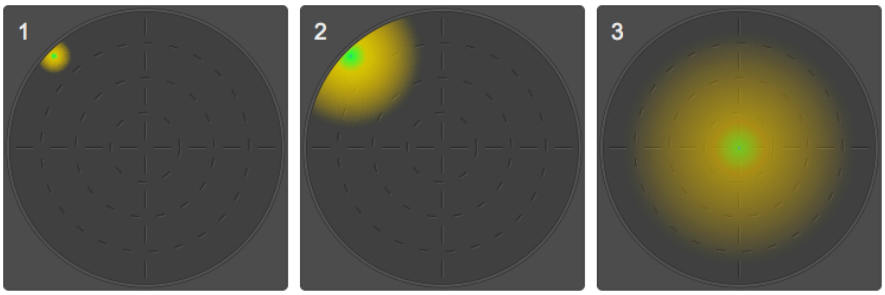

Figure 58 : représentation, au sein de l'interface *hoa.spectrum~ pour MaxMSP,* d'un sinus à 440Hz encodé avec un angle d'incidence de π/4 et une distance de 2, 1, 0 (de gauche à droite).

[75] Nous invitons le lecteur à se faire une meilleure idée de la représentation au sein de la mise en œuvre MaxMSP ; une visualisation dans le temps renseignant toujours plus qu'une série d'instantanés sur la nature du champ sonore étudié.

Dans le cas spécifique où un champ sonore est composé d'une source sonore ponctuelle unique, l'interface *hoa.spectrum~* peut servir à « retrouver », en quelque sorte, la position d'origine de cette source sonore virtuelle tout en donnant à la fois des indications sur son contenu fréquentiel. On perçoit en effet nettement, dans l'exemple donné par la [Figure 57], que nous sommes en présence d'une source sonore ponctuelle dans la mesure où tous les disques sont réunis en un point. À travers la position spatiale successive des disques, on perçoit le déplacement d'avant en arrière que la source sonore subit. La variation de couleur de ces disques trahit quant à elle la variation de sa masse spectrale allant du grave à l'aigu ; la faible variation de leur diamètre semble aussi témoigner de la faible dynamique de cette source. L'exemple de la [Figure 58] semble aussi montrer clairement le contenu spectral médium d'un champ sonore composé d'une unique source ponctuelle ainsi qu'une variation de la distance par rapport au centre. En revanche, nous ne serons pas en mesure de dire, à la seule vue de cette représentation, si la source sonore se rapproche du cercle de haut-parleurs ou si son intensité a simplement augmentée entre la [Figure 58.1] et la [Figure 58.2][76]

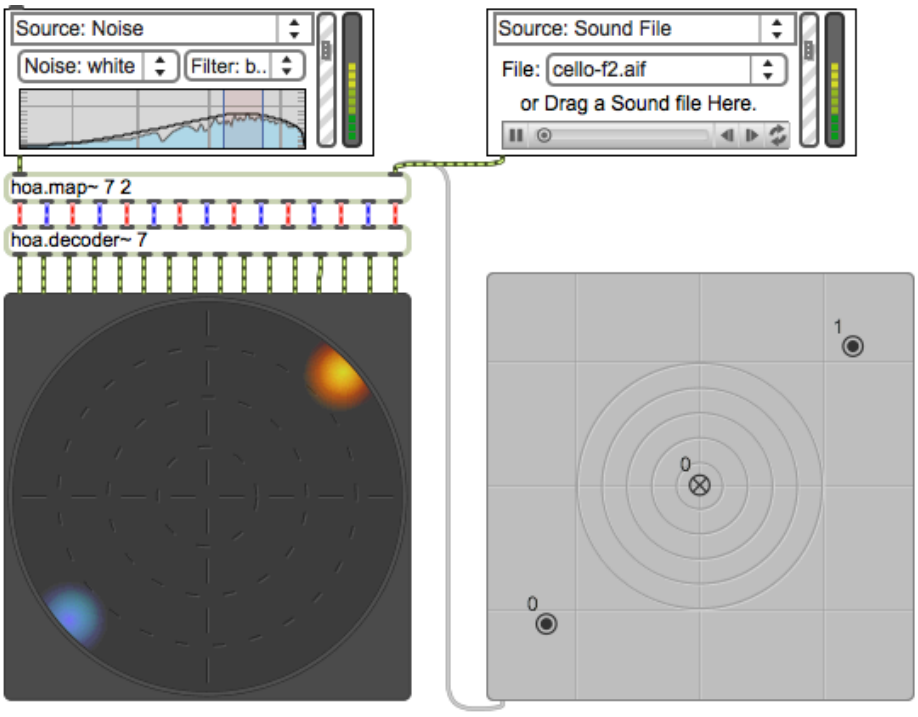

Figure 59 : représentation d'un *patch* MaxMSP dans lequel deux sources sonores virtuelles sont encodées grâce à l'objet *hoa.map~*, selon les coordonnées fournies par l'interface graphique *hoa.map* (en bas, à droite). L'interface *hoa.spectrum~* (en bas, à gauche) représente le contenu fréquentiel et spatial du champ sonore ainsi crée.

[76] Ceci étant dû à la technique que nous employons pour la compensation de la distance explicitée à la partie consacrée au contrôle de coordonnées spatiales. Se référer aussi à [Guillot, 2013].

À travers la représentation offerte par l'interface *hoa.spectrum~* à la [Figure 59], on peut déduire que le champ sonore qui y est représenté se compose de deux sources sonores à caractère plutôt ponctuel, relativement proche du cercle de haut-parleurs (ou très éloignées mais de forte amplitude). La première source qui se situe à un angle environ égal à $3\pi/4$, semble avoir un contenu fréquentiel aigu, tandis que la seconde semble plus évoluer dans un registre médium et se situer autour d'un angle de $-\pi/4$. Notons néanmoins que si plusieurs sources ponctuelles se situent dans un même registre spectral, elles auront tendance à s'indifférencier du point de vue de cette représentation.

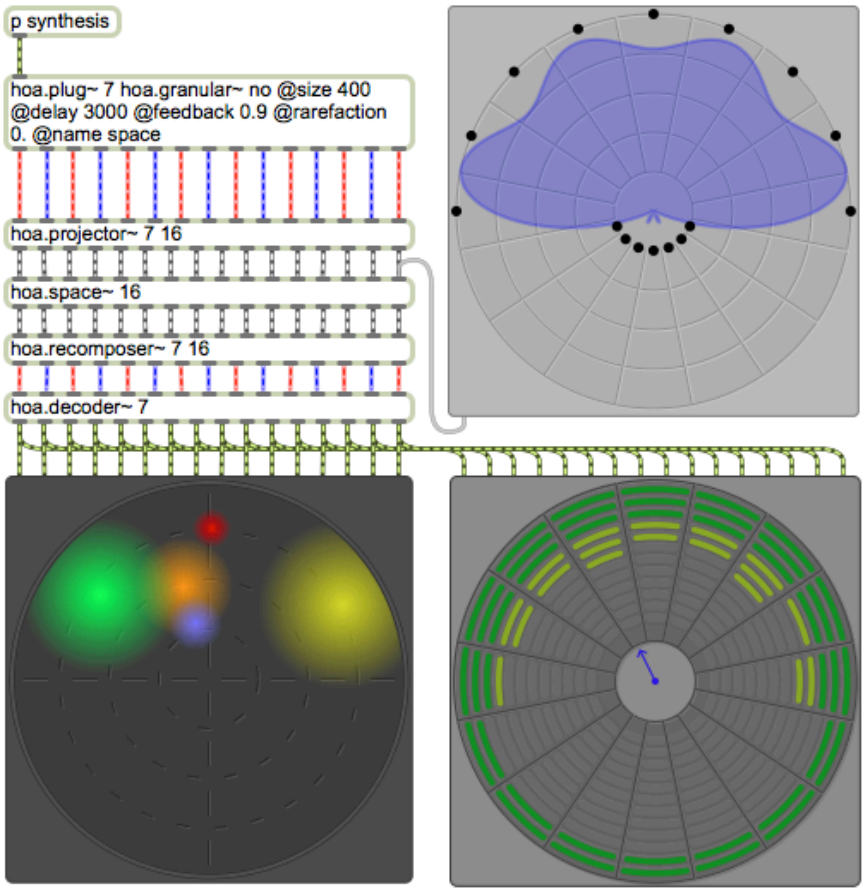

Figure 60 : représentation d'un *patch* MaxMSP dans lequel un champ sonore diffus, composé de grain sonores, est synthétisé grâce à l'objet *hoa.plug~*, puis filtré spatialement avant d'être décodé. L'interface *hoa.meter~* permet de visualiser les contributions de chacun des seize haut-parleurs, tandis que l'interface *hoa.spectrum~* révèle le contenu fréquentiel du champ sonore tout en donnant une indication sur sa localisation.

La [Figure 60] illustre une synthèse de champ sonore diffus dont l'arrière est ensuite filtré spatialement. L'interface *hoa.spectrum~* permet de refléter graphiquement l'incidence de ce traitement en représentant l'ensemble des disques sur la partie supérieure uniquement. Cette représentation met aussi en avant le contenu plutôt medium de ce champ sonore si l'on

se fie à la petitesse des bandes de fréquences extrêmes (disques rouge et bleu) par rapport aux bandes intermédiaires.

Cet objet, de facture encore relativement récente, n'a pu actuellement être testé dans aucune création musicale. Les tests que nous avons pu réaliser par nous-même nous ont néanmoins paru assez probants du point de vue de la représentation graphique corrélée à l'écoute pour en valider l'approche. Il s'agit donc désormais d'étudier de plus près la faisabilité des améliorations futures dont il pourrait bénéficier. Nous pensons notamment qu'une résolution fréquentielle plus importante permettrait de représenter avec plus de fidélité le champ sonore. Il faudrait donc pour cela laisser à l'utilisateur la possibilité d'ajouter plus de bandes, en passant si besoin par une discrétisation du champ sonore grâce à un filtrage de type *FFT*.

L'utilisateur de la bibliothèque HOA dispose donc, à travers la série d'interfaces que nous venons de présenter en détail, d'outils particulièrement performants lui permettant de se représenter le champ sonore en le décrivant de multiples manières. Elles autorisent donc une gestion plus précise des traitements ambisoniques dans la mesure où l'on peut désormais visualiser et analyser finement l'incidence de chaque manipulation sur le champ sonore, et ceci à tous les niveaux de la chaîne de traitement du signal.

Nous revenons à présent sur la série d'utilitaires que nous avons développés dans le cadre de la mise en œuvre MaxMSP de la bibliothèque HOA pour pallier différentes contraintes imposées par l'approche modulaire que nous avons choisi d'adopter.

3 DES INTERFACES POUR LA MISE EN PLACE DES TRAITEMENTS AMBISONIQUES

L'un des principaux enjeux du développement de la bibliothèque HOA consistait à donner aux musiciens et aux compositeurs les outils les plus performants et musicaux qui soient en les concevant de manière à ce qu'ils demeurent toujours pour eux des plus accessibles. Comme nous avons pu le voir à travers les deux précédentes parties, la bibliothèque HOA dispose de moteurs sonores puissants permettant d'effectuer de nombreux traitements de l'espace et du son, et d'interfaces graphiques de contrôle et de représentation du champ sonore augmentant l'interactivité possible entre l'utilisateur et les traitements qu'il met en œuvre. Pour augmenter la modularité de la chaîne de traitement et permettre d'autres interactions, nous avons fait le choix de séparer les modules d'encodage des modules de décodage ambisonique au sein de la mise en œuvre de la bibliothèque HOA à destination des environnements de programmation visuelle tels que Pure Data ou MaxMSP. Ce choix, qui a aussi été fait par d'autres bibliothèque ambisoniques [Wakefield, 2006], a été effectué afin de permettre l'émergence de pratiques nouvelles de mise en espace des sons ; d'une part grâce aux moteurs sonores que nous proposons, mais aussi en laissant les utilisateurs créer leurs propres traitements ambisoniques en autorisant la manipulation directe des signaux dépendant des harmoniques circulaires. Le dynamisme et la modularité de la bibliothèque représentent donc un gage de flexibilité propice à la création sonore. Néanmoins, cela entraîne dans un même temps une complexité accrue de mise en place des processus de traitement pour l'utilisateur, qui se retrouve à devoir travailler avec un nombre de canaux et de haut-parleurs assez important qui, de surcroît, augmente plus l'ordre de décomposition ambisonique est élevé. Cet état de fait a une double conséquence qui pouvait tendre selon nous à freiner quelque peu la créativité de l'utilisateur.

La première conséquence est d'ordre matérielle : en effet, les musiciens ou les compositeurs travaillant chez eux, ne disposent bien souvent que d'un système de restitution sur haut-parleurs réduit, qui ne leur permet pas d'explorer ni de mettre à profit tout le potentiel créatif des techniques ambisoniques. D'un autre côté, les studios ou les salles de concert sont, elles, souvent équipées de systèmes de restitution plus conséquents qui s'adaptent donc mieux à ces techniques. Or, l'enjeu de développement de la bibliothèque HOA est justement de rendre accessibles ses outils de mises en espace des sons à une communauté d'utilisateurs aussi large que possible, incluant donc les pratiques de studio autant que les pratiques relevant du

home studio. Afin de réduire, ou de rendre moins contraignante, la fracture entre le dispositif employé lors du processus de création et de diffusion d'une œuvre spatialisée grâce aux techniques ambisoniques, nous avons donc décidé d'élargir les techniques de décodage disponibles au sein de la bibliothèque HOA afin qu'elles puissent s'adapter à un maximum de configurations standards ou irrégulières (ex. quadriphonie, octophonie, 5.1, stéréo, casque).

La seconde conséquence est plutôt d'ordre ergonomique dans la mesure où le traitement d'une série de canaux, en particulier dans le cadre de l'ambisonie, implique des opérations et manipulations spécifiques qui diffèrent, par leur nombre ou leur nature, de celles employées lors d'un simple traitement monophonique. Pour rendre la mise en place des traitements ambisoniques la plus facile et confortable possible pour l'utilisateur au sein de l'environnement logiciel MaxMSP nous avons donc développé des outils utilitaires permettant de sérialiser au maximum les opérations afin de donner à l'utilisateur l'impression qu'il travaille sur l'ensemble de la chaîne DSP comme il le ferait pour un seul et même signal.

Nous présenterons donc ici une série d'utilitaires, sous la forme d'interfaces graphiques dédiées, d'*externals* ou de *patchs*, qui tentent tous de répondre à leur manière aux limites actuelles en terme de modularité et de dynamisme imposées par les environnements de programmation visuelle de type MaxMSP en améliorant l'ergonomie globale de l'interface logicielle proposée et, par voie de conséquence, « l'expérience utilisateur » générale de la bibliothèque HOA.

3.1 PARALLELISATION ET FACTORISATION DES TRAITEMENTS

Nous verrons à travers un exemple simple, un gain appliqué à une série de signaux, les stratégies de mise en place qui sont à notre disposition au sein de cet environnement logiciel et la ou lesquelles nous avons choisi d'adopter afin de les généraliser à d'autres types d'opérations multicanales. L'exposition de ces stratégies mettront à jour un outil particulièrement adapté aux traitements ambisoniques, *hoa.plug~*, permettant une mise en œuvre aisée de transformations et de synthèses de champs sonores.

3.1.1 GESTION DU GAIN EN AMBISONIE

Un traitement aussi simple qu'une gestion de gain peut donner lieu à des opérations musicales très employées dans la création audionumérique, telles que les *fade-in*, *fade-out* ou encore *crossfades*. Cette opération pourtant simple devient néanmoins vite fastidieuse si l'on doit agir sur le gain de chaque canal individuellement alors que l'on cherche à opérer une variation générale de niveau sur l'ensemble des canaux.

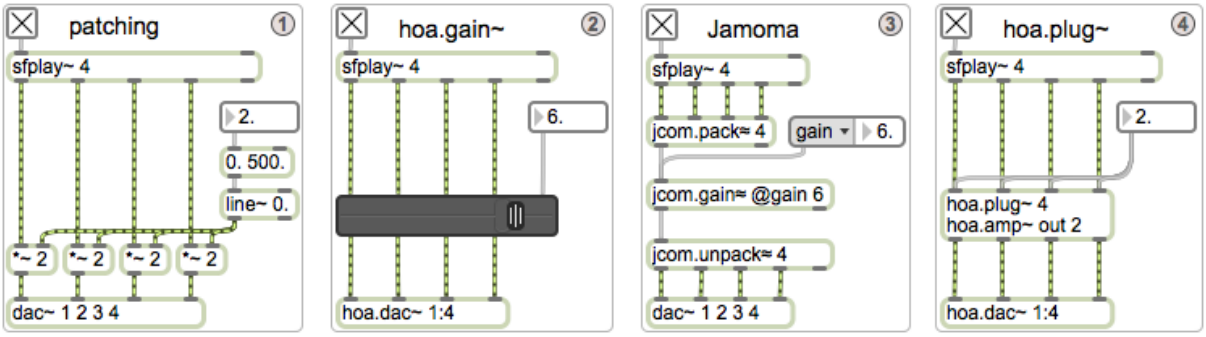

Figure 61 : représentation de quatre stratégies de gestion de gain multicanal pouvant être adoptées au sein de l'environnement de programmation MaxMSP. Gestion individuelle des différents canaux [1], traitement groupé à travers l'interface graphique *hoa.gain~* [2], traitement groupé permis par les modules de traitement multicanaux de la bibliothèque Jamoma [3], traitement groupé grâce à l'objet *hoa.plug~* [4].

Cette opération peut être réalisée de multiples manières au sein d'un environnement de programmation visuel tel que MaxMSP en adoptant des stratégies et outils différents [Figure 61]. Une des manières d'opérer consiste à insérer au sein de la chaîne de traitement une opération de multiplication sur chaque canal de la série de signaux dont on veut modifier le niveau [Figure 61.1]. Cette stratégie, bien qu'étant la plus traditionnellement employée, comporte néanmoins la contrainte de devoir répéter les manipulations de *patching* nécessaires à sa mise en place autant de fois qu'il y a de canaux.

Pour faciliter la mise en place de ce traitement en particulier nous avons donc développé un objet original, dédié spécifiquement à cette opération. L'objet *hoa.gain~* est une interface graphique de type potentiomètre linéaire permettant une gestion simplifiée d'un niveau de gain commun à une série de signaux [Figure 61.2].

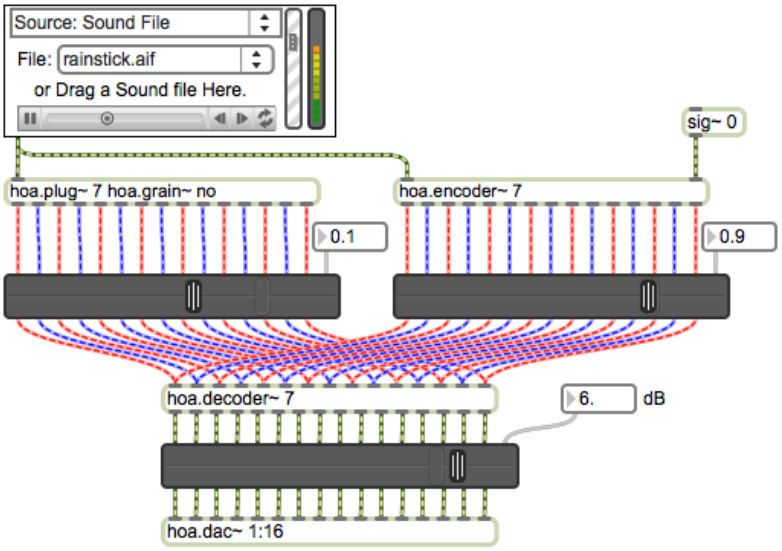

Figure 62 : représentation d'un patch MaxMSP dans lequel un champ sonore est crée par mixage, grâce à l'interface graphique *hoa.gain~*, entre un champ diffus et une source ponctuelle synthétisés à partir d'un même son de bâton de pluie à un ordre de décomposition 7, puis décodé pour un système de 16 haut-parleurs. Le gain de sortie général est ensuite augmenté de 6dB grâce à l'interface *hoa.gain~*.

Il suffit donc désormais à l'utilisateur de spécifier le nombre de canaux sur lesquels il souhaite opérer la variation de gain pour que l'objet adapte dynamiquement le nombre d'entrées et puisse effectuer ce traitement sur autant de canaux que nécessaire. La [Figure 62] met en avant la facilité de mise en place d'un traitement de type *crossfade* opéré dans le domaine des harmoniques circulaires entre deux champs sonores ainsi qu'une variation de gain opérée sur l'ensemble du champ sonore décodé. Cet objet se place à mi-chemin entre les autres interfaces de ce type au sein de MaxMSP tel que l'objet *slider*, *gain~* ou encore *live.gain~* desquels il reprend certaines fonctions ou en ajoute. Il a notamment été doté de multiples fonctions permettant de personnaliser son comportement afin de l'adapter à un maximum de contextes et d'améliorer son ergonomie. L'utilisateur peut par exemple spécifier l'échelle qu'il souhaite utiliser (décibels, amplitude, midi) ainsi que les valeurs minimum et maximum de celle-ci ; ou encore définir une valeur de gain par défaut qui sera chargée lors de l'ouverture du *patch* et lorsqu'il double-cliquera sur l'interface.

Certaines bibliothèques externes ont aussi proposé des solutions ingénieuses pour contourner ce problème en ajoutant plus de dynamisme au processus de traitement multicanal. C'est par exemple le cas du *framework AudioGraph* de la bibliothèque *Jamoma*[77]

[77] http://www.jamoma.org/.

[Place et al., 2010], qui permet de construire une chaîne de traitement dynamique à partir d'un ensemble de modules s'adaptant à une série indéfinie de canaux [Figure 61.3]. Le mode opératoire est assez simple, l'utilisateur est amené à grouper la série de canaux séparés qu'il souhaite traiter afin de générer une corde unique qui va permettre de connecter les différents modules de traitements multicanaux entre eux. Il suffit ensuite à l'utilisateur de dégrouper cette connexion unique, que l'on pourrait apparenter alors à une matrice d'échantillons sonores, afin de retrouver chaque canal individuel. Bien qu'assez pratique et bien pensée, cette méthode présente tout de même quelques inconvénients. En effet, entre le moment où les signaux sont groupés et le moment où ceux-ci sont dégroupés, l'utilisateur ne peut effectuer des traitements qu'à partir des objets compatibles avec ce mode de traitement. L'utilisateur devenant donc complètement dépendant des moteurs sonores actuellement implémentés, ou non, au sein de la bibliothèque en question. Dans le cadre du développement de la bibliothèque HOA, nous nous sommes donc confrontés à la même problématique du multicanal mais avons choisi d'adopter une stratégie quelque peu différente. Nous voulions, en effet, « fermer » un minimum la bibliothèque, c'est à dire l'intégrer de façon à ce que l'ensemble de nos moteurs sonores puisse fonctionner en symbiose avec les objets traditionnels de MaxMSP, ceci afin de contraindre au minimum les possibilités de création musicale. Pour cela, nous avons choisi de laisser apparente et pleinement accessible la série de canaux traitée par la chaîne DSP. L'utilisateur peut donc agir sur chaque canal indépendant, à l'aide des objets de signal traditionnels de MaxMSP, à tous les niveaux de la chaîne de traitement. Pour remédier aux problèmes exposés par la [Figure 61.4], nous avons crée un outil se rapprochant conceptuellement de l'approche adoptée par *Jamoma* mais offrant un degré de liberté plus important. L'objet *hoa.plug~* permet lui de paralléliser un même traitement à un nombre défini mais dynamique de canaux. Il suffit donc à l'utilisateur de créer le traitement qu'il souhaite effectuer comme s'il se situait dans un contexte monophonique, puis de charger ce *patch* au sein de l'objet *hoa.plug~* pour que celui-ci applique ce même traitement à tous les signaux de la série de canaux. Dans l'exemple de la [Figure 61.3], le *patch hoa.amp~*, qui permet une variation lissée d'amplitude sur un signal monophonique, est chargé au sein de l'objet *hoa.plug~* pour effectuer un traitement équivalent à celui produit par les autres méthodes[78].

[78] Le lecteur pourra se référer à l'aide de l'objet *hoa.amp~* au sein de la distribution MaxMSP de la bibliothèque HOA.

Nous voyons assez clairement, à la lumière de la [Figure 61], que la meilleure stratégie à adopter, dans le cas spécifique d'une gestion de gain sur un ensemble de canaux, passe par l'interface *hoa.gain~*, dédiée exclusivement à ce processus, et donc plus appropriée qu'aucune autre. Il convient néanmoins de généraliser cette approche à d'autres opérations qu'une simple multiplication du signal et de l'adapter aux besoins spécifiques du traitement dans le domaine des harmoniques circulaires. C'est ce que nous proposons au lecteur d'étudier dans la partie suivante dédiée à l'objet *hoa.plug~*.

3.1.2 GENERALISATION ET ADAPTATION AUX HARMONIQUES CIRCULAIRES

Effectuer un traitement dans le domaine des harmoniques circulaires, et plus généralement dans le cadre du multicanal, se résume le plus souvent à appliquer ce même traitement en parallèle sur différents canaux. Or, comme nous avons pu le voir avec l'exemple du gain, les spécificités d'un environnement de programmation graphique tel que MaxMSP ne facilitent pas ce type de mise en place. L'utilisateur se retrouve à devoir effectuer un grand nombre de manipulations telles que l'instanciation et la connexion d'objets, qui augmentent relativement à l'ordre de décomposition et à la complexité du traitement à mettre en place. Afin de factoriser ces interventions, nous avons développé l'objet *hoa.plug~*, grâce auquel, un même traitement, crée sous forme de *patch*, peut désormais être appliqué à un ensemble de canaux.

Comme nous l'avons vu dans l'exemple précédent, une variation d'amplitude devient donc très facilement réalisable grâce à une abstraction telle qu'*hoa.amp~* chargé au sein d'un objet *hoa.plug~*. Afin de généraliser cette méthode de mise en place à d'autres types d'opérations, nous fournissons aux utilisateurs une série d'abstractions compatibles avec cet objet, reproduisant le comportement des modules élémentaires de traitement du signal natifs de cet environnement logiciel[79]. Parmi eux, on trouvera notamment des opérateurs arithmétiques de base (ex : multiplication, division, addition, soustraction), des opérateurs conditionnels (ex : égal à, différent de) ou encore des opérations plus complexes comme des filtrages. L'objet *hoa.plug~* nous a aussi permis de proposer par exemple une variation du couple d'objet *send~/receive~* qui permet, une fois chargé au sein de l'objet *hoa.plug~*, d'envoyer et

[79] Le lecteur pourra se reporter à l'onglet *Plug-Operators* du *patch hoa.overview* au sein de la distribution MaxMSP de la bibliothèque HOA pour avoir une vue d'ensemble de ces opérations.

recevoir non-plus un signal monophonique mais un champ sonore tout entier en transmettant l'ensemble de la série de canaux[80].

Outre le fait d'offrir une amélioration de l'ergonomie générale des opérations multicanales, le principal intérêt de l'objet *hoa.plug~* se situe dans le fait qu'il s'adapte parfaitement au contexte ambisonique et plus particulièrement au domaine des harmoniques circulaires.

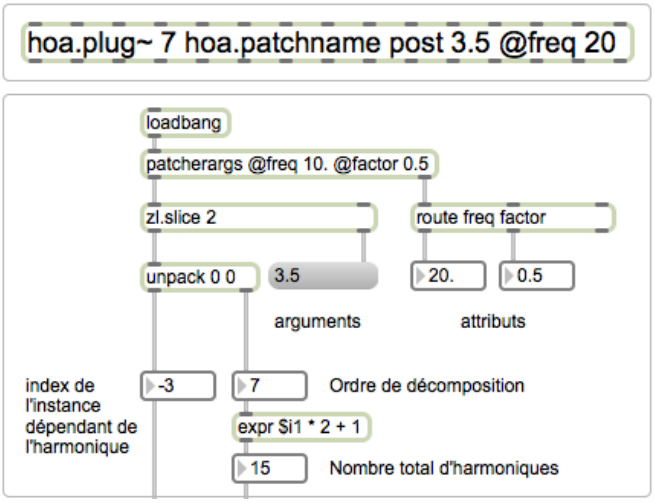

Figure 63 : un exemple d'arguments passés à l'objet *hoa.plug~* avec dans l'ordre (en haut, de gauche à droite): l'ordre de décomposition (ici 7), le nom du patch à charger, le mode (*pre*, *post*, *no* ou *out*), des arguments lambda et attributs nommés. L'encadré du bas représente l'instance n°-3 du traitement et une manière de retrouver les informations passées en argument à l'objet *hoa.plug~* au sein de chaque instance.

L'objet *hoa.plug~*, de façon analogue à l'objet *poly~* de MaxMSP, permet d'instancier *n* fois le même traitement. La différence est qu'ici, le nombre d'instance est dépendant de l'ordre de décomposition ambisonique ; chaque instance correspondant alors à une harmonique circulaire spécifique[81]. En récupérant ces informations au sein de chaque instance grâce à l'objet *patcherargs* [Figure 63], il est donc désormais très facile d'élaborer des traitements ambisoniques originaux, variabilisés selon l'ordre de décomposition général et l'indice spécifique de chaque composante spatiale.

[80] Se reporter à l'aide des abstractions *hoa.send~* et *hoa.receive~*.

[81] Notons que le comportement de l'objet *hoa.plug~* décrit ici diffère quelque peu lorsqu'il est en mode *out*. Nous invitons le lecteur à se référer à l'aide de l'objet pour plus d'information à ce sujet.

Nous avons vu au chapitre consacré à la représentation graphique des harmoniques circulaires que l'angle d'incidence d'une source ponctuelle dépendait des coefficients de pondération appliqués à chacune des composantes spatiales d'un ordre de décomposition donné [Figure 46]. En pondérant d'une autre manière le signal dépendant de chacune des harmoniques suite à une opération d'encodage, on peut donc modifier l'angle d'incidence attribué initialement à la source ou donner une toute autre nature au champ sonore.

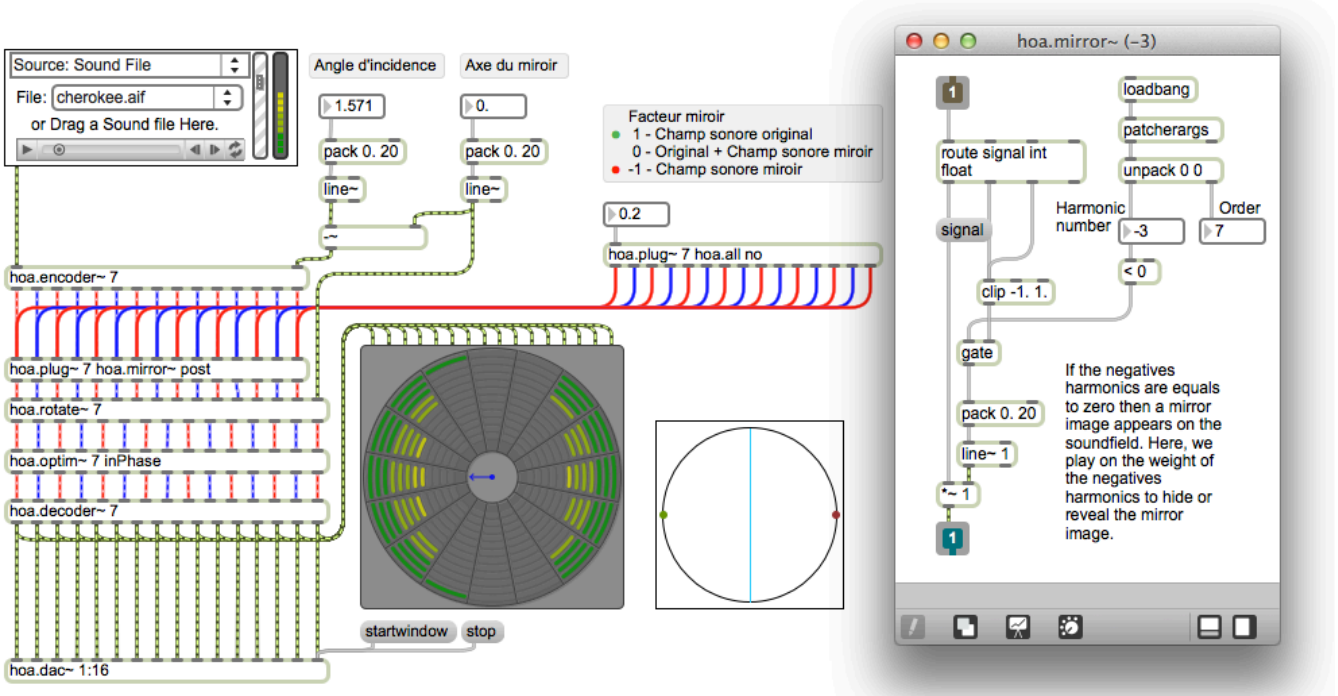

Figure 64 : représentation du traitement miroir à travers l'abstraction *hoa.mirror~* chargée au sein d'un objet *hoa.plug~*. La fenêtre sur la droite représente l'instance correspondant à l'harmonique (-3) du traitement miroir.

Le *patch* présenté à la [Figure 64] permet par exemple de faire apparaître et de contrôler l'amplitude d'une « image miroir », créée par symétrie axiale, d'une source sonore ponctuelle préalablement encodée dans le domaine des harmoniques circulaires. Le contrôle de l'angle de l'axe de symétrie et le degré de variation de la source sonore initiale par rapport à cet axe est permis par la conjonction des objets *hoa.encoder~* et *hoa.rotate~*. Le principe du traitement miroir consiste à modifier la contribution de chacune des harmoniques négatives du champ sonore en leur appliquant une valeur de gain commune, de façon à révéler ou cacher plus ou moins cette « image miroir ». Nous nous trouvons donc dans une configuration de traitement assez proche de l'opération que nous cherchions à effectuer dans la partie précédente consacrée à la gestion d'une valeur de gain commune à une série de signaux ; à la seule différence qu'ici, nous ne voulons pas que ce gain affecte les signaux

dépendants des harmoniques positives du champ sonore. Pour cela, quelques adaptations sont nécessaires à l'abstraction *hoa.amp~*. Le *patch* comporte ici deux objets *hoa.plug~*, le premier (*hoa.all*) sert simplement à transmettre la valeur de gain spécifiée à chacune des instances du second. Le traitement mis en place ensuite dans chacune des instances de ce second objet *hoa.plug~*, à travers l'abstraction *hoa.mirror~*, consiste à récupérer l'index de l'instance ou de l'harmonique grâce à l'objet *patcherarg*, puis d'appliquer un gain de 1 si celui-ci est positif, et le gain spécifié si l'index est négatif. En « jouant » sur ce facteur de gain, il nous est donc possible d'exercer une sorte de *crossfade* entre le champ sonore original et le champ sonore « miroir ».

Nous avons choisi d'illustrer ici le fonctionnement de l'objet *hoa.plug~* à travers le traitement simple mis en place au sein de l'abstraction *hoa.mirror~* pour des raisons principalement didactiques. Il est néanmoins possible de réaliser des traitements aux rendus autrement plus complexes et musicaux en s'inspirant notamment de traitements monophoniques existants appliqués au domaine des harmoniques circulaires. La bibliothèque HOA pour MaxMSP propose un certain nombre de traitements ambisoniques sous la forme d'abstractions, destinées à être chargées directement au sein de l'objet *hoa.plug~*[82]. Ces traitements ont été élaborés à partir des différentes expérimentations que nous avons pu effectuer et valider durant le projet de développement. Parmi ceux-ci, l'utilisateur pourra notamment trouver une adaptation au domaine des harmoniques circulaires d'un traitement granulaire permettant une synthèse de champ diffus (à travers l'abstraction *hoa.grain~* et *hoa.granular~*), un traitement de modulation d'amplitude (*hoa.am~*) ou de décorrélation microtemporelle (*hoa.decorrelation~*) destinés tous deux à donner un caractère plus ou moins diffus au champ sonore[83].

Ces traitements constituent à la fois des outils « clefs en main » pour le musicien qui cherche à opérer rapidement des transformations du champ sonore, mais aussi de très bons exemples pédagogiques à l'intention d'un utilisateur plus avancé cherchant à créer de nouveaux traitements originaux dans le domaine des harmoniques circulaires.

---

[82] Le lecteur pourra se reporter à l'onglet *Plug-Process* du *patch hoa.overview* au sein de la distribution MaxMSP de la bibliothèque HOA pour avoir une vue d'ensemble de ces traitements.

[83] Nous n'avons malheureusement pas le temps de développer plus en détail ici l'ensemble de ces mises en œuvres ; nous renvoyons pour cela le lecteur au mémoire de Pierre Guillot consacré aux traitements musicaux en ambisonie. L'auteur y explique avec précision ces différents traitements, ainsi que les stratégies de mise en œuvre qui ont été adoptées, notamment en termes de *mapping* des différents paramètres en fonction de l'ordre de décomposition et de l'indice de chaque composante spatiale [Guillot, 2013].

Ce module de traitement, représentant pour nous l'un des atouts majeurs de la bibliothèque HOA, a été brièvement présenté lors de l'édition 2013 des JIM [Colafrancesco et al., 2013]. Il a de même pu être expérimenté au cours de plusieurs créations musicales, notamment au cours des diverses installations déjà citées dans ce mémoire pour proposer au public des matières sonores créées par synthèse de champs diffus. La compositrice Anne Sèdes, pour sa pièce *immersion*[84], a utilisé la synthèse de champ sonore par granulation grâce à l'abstraction *hoa.grain~* chargée au sein de l'objet *hoa.plug~*. Cette abstraction lui a permis de synthétiser un champ diffus à partir d'un son de violoncelle produit en temps réel auquel elle a pu ensuite appliquer un traitement de type *fisheye* pour resserrer le champ sonore en un point de la scène sonore et simuler ainsi l'encodage d'une source sonore ponctuelle à partir de ce matériau sonore. Le compositeur Alain Bonardi, pour sa pièce *pianotronics 1*, pour piano et électronique en temps réel, a pu quant à lui s'approprier l'objet *hoa.plug~* en élaborant un traitement original basé sur une série d'*harmonizers*. Chacune des instances du *patch* transposant les sons du piano en fonction de l'ordre de décomposition ambisonique global et de l'indice de l'harmonique dont dépendait chacune des instances de l'objet *hoa.plug~*.

Notons que cet objet n'est pour l'instant pas disponible au sein de la mise en œuvre Pure Data de la bibliothèque HOA. Afin de proposer des traitements aux rendus similaires à ceux que nous proposons dans MaxMSP, nous avons donc développé des *externals* reproduisant les mêmes fonctions (ex : *hoa.granular~, hoa.delay~*). En revanche, il est donc de ce fait encore impossible de créer ses propres traitements grâce à la méthode que nous venons de présenter au sein de cet environnement. Nous réfléchissons actuellement à une solution alternative qui puisse autoriser cette fonctionnalité.

[84] Nous renvoyons le lecteur à la partie de ce mémoire consacrée à la distorsion de la perspective dans laquelle nous évoquons celle-ci. Se référer aussi à [Colafrancesco et al., 2013].

3.2 DYNAMISME ET MODULARITE

Nous aborderons ici les solutions que nous proposons au sein de la bibliothèque HOA pour MaxMSP, afin d'automatiser les opérations redondantes de *patching* telles que la connexion des différents modules de la chaîne de traitement.

Nous reviendrons enfin sur les possibilités de décodages offertes par la bibliothèque HOA à travers un module permettant à l'utilisateur de basculer très facilement d'une configuration à une autre.

3.2.1 CONNECTIONS AUTOMATIQUES ET ORDRES DYNAMIQUES

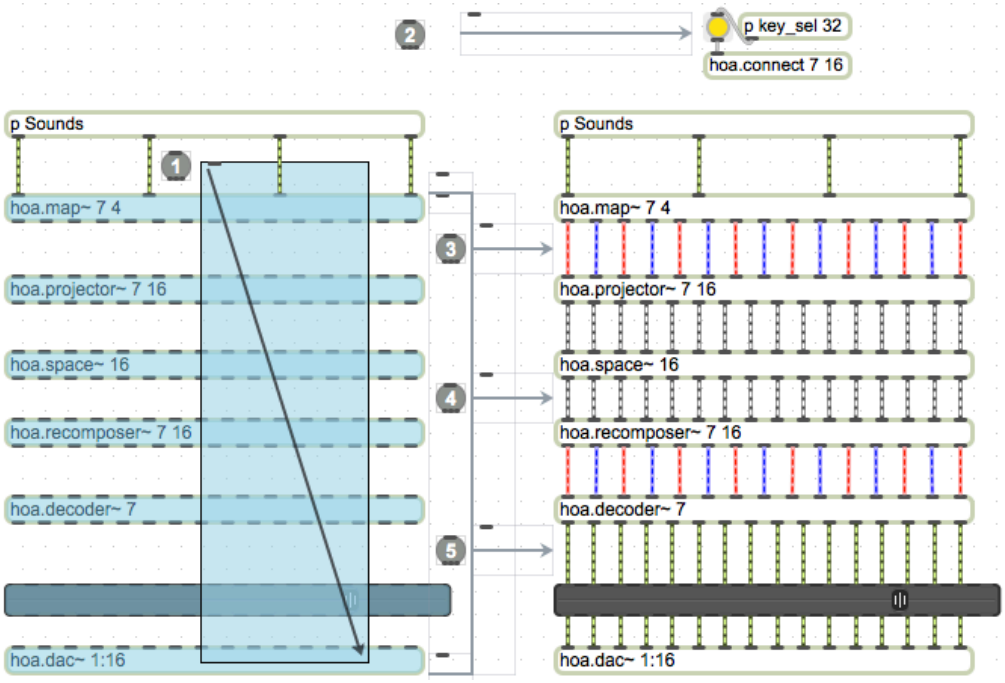

Figure 65 : représentation d'un patch MaxMSP en mode édition illustrant la fonctionnalité et l'action de l'objet *hoa.connect* pour un ordre de décomposition 7 et une configuration à 16 haut-parleurs.

Pour raccorder entre eux un ensemble de modules composant une chaîne de traitement complexe telle que celle présentée à la droite de la [Figure 65], l'utilisateur doit effectuer un total de 94 opérations de raccordement (ce nombre augmentant à des ordres de décomposition plus élevés). Ces opérations représentent un temps de mise en place assez considérable pour un musicien ou un compositeur, lors de la phase de création d'une œuvre, qui pourrait être employé à bien meilleur escient. Nous avons donc cherché à remédier à ce problème en automatisant ce processus. Grâce à l'emploi de l'objet *hoa.connect*, pour connecter entre eux l'ensemble de ces modules, l'utilisateur n'a désormais plus qu'à

sélectionner de haut en bas les objets qu'il souhaite connecter [Figure 65.1], puis envoyer un *bang* à l'objet *hoa.connect*[85] [Figure 65.2]. Le raccordement se fait alors de façon automatique entre tous les objets sélectionnés, grâce à une opération ne nécessitant désormais plus que deux manipulations. La sélection peut se faire soit en sélectionnant chaque objet successivement ou par rectangle de sélection, le raccordement se faisant selon l'ordre dans lequel chaque objet a été sélectionné. Le lecteur aura aussi pu remarquer que des couleurs particulières ont aussi été attribuées aux connexions entre les différents modules de la bibliothèque HOA. Par analogie avec la coloration syntaxique que l'on peut trouver dans la plupart des IDE[86] traditionnels, cette coloration permet d'ajouter une couche d'information supplémentaire à la chaîne de traitement que l'utilisateur crée en s'opérant de manière automatique lors du processus de connexion cité plus-haut. Les couleurs attribuées à chacune des séries de cordes de la chaîne de traitement dépendent de la manière dont on se représente la décomposition du champ sonore y transitant. Ainsi, dans le cadre d'une décomposition du champ sonore en série de composantes spatiales, les cordes prendront alternativement des couleurs rouge puis bleue, signifiant que l'on a à faire à des signaux dépendant des harmoniques, aux indices respectivement positifs et négatifs [Figure 65.3]. Les séries de signaux reliant des modules opérant des transformations du champ sonore dans le domaine des ondes planes sont quant à eux représentés en blanc [Figure 65.4]. Enfin, la couleur des signaux ne dépendant pas des objets HOA, ou n'étant pas caractérisables comme appartenant clairement à un domaine en particulier, reste quant à elle inchangée [Figure 65.4].

Conscient du fait que ces améliorations ergonomiques du *patching* puissent ne pas convenir à tout le monde, toutes ces actions automatiques (raccordement des objets, coloration contextuelle des cordes reliant entre eux les différents modules de la chaîne de traitement) ont été rendues paramétrables, optionnelles et facilement désactivables afin de satisfaire aux usages moins conventionnels ou plus spécifiques.

L'objet *hoa.connect* fait partie de ces utilitaires qui nous étaient indispensables lors de la phase de test et de développement de la bibliothèque, nous épargnant ainsi les opérations redondantes et extrêmement chronophages de raccordement entre les différents modules. Cet outil, apportant une fonction à la fois pratique et pédagogique, a de même pu être largement apprécié des utilisateurs ayant actuellement pris en main la bibliothèque.

---

[85] Cette dernière opération peut être exécutée en associant l'appui sur une touche du clavier à l'envoi du message *bang* (ici la touche n°32 correspondant à la barre d'espace).

[86] *Integrated development environment*. (ex. xcode, visual studio).

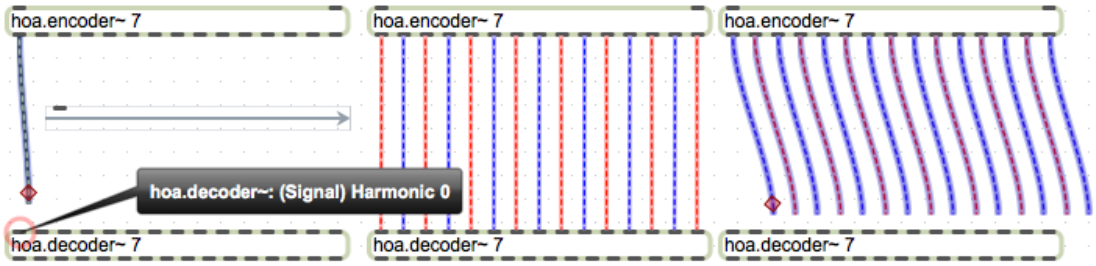

Figure 66 : représentation d'un *patch* MaxMSP en mode édition, illustrant une connexion automatique des entrées/sorties des objets *hoa.encoder~* et *hoa.decoder~* permise par la connexion unique de la première entrée des deux objets (à gauche et au milieu). Illustration de la déconnexion d'une série de cordes permise par la manipulation de la première (à droite).

Plusieurs pistes sont néanmoins encore envisageables pour améliorer l'ergonomie de la bibliothèque HOA pour MaxMSP et rendre la mise en place de traitement encore plus confortable. Nous réfléchissons par exemple actuellement à la connexion/déconnexion automatique de deux modules consécutifs de la chaîne de traitement grâce à la manipulation de la première corde de chaque module comme illustré à la [Figure 66].

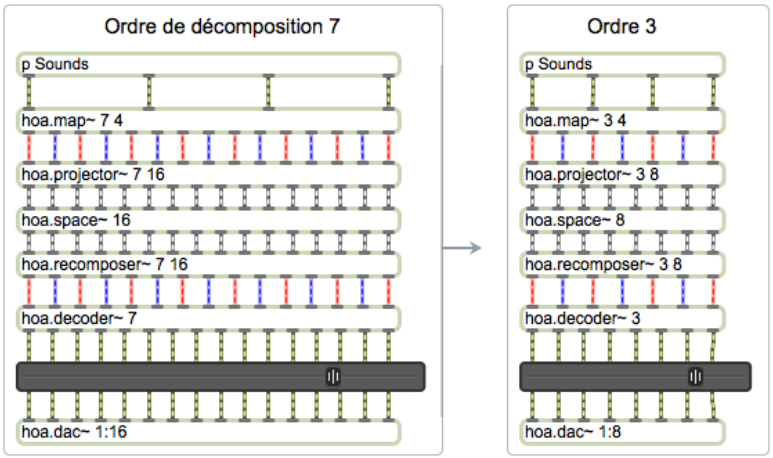

Figure 67 : représentation du passage d'une chaîne de traitement pour un ordre de décomposition de 7 à 3.

Dans la mise en œuvre actuelle de la bibliothèque, l'ordre de décomposition se définit au sein de chaque module individuellement. Une autre piste à explorer se situerait donc aussi dans le fait de proposer à l'utilisateur la possibilité d'adapter dynamiquement et globalement l'ensemble des modules composant une chaîne de traitement ambisonique à un nouvel ordre

de décomposition ambisonique de façon générale[87] [Figure 67]. L'opération consisterait à redéfinir le nombre d'entrées et de sorties nécessaires pour chaque modules puis à les reconnecter entre eux. Toutes les interfaces graphiques dépendantes de l'ordre de décomposition ou du nombre de haut-parleurs ont dès à présent été rendues dynamiques et adaptent désormais leur nombre d'entrées à l'ordre ou au nombre de haut-parleurs spécifié.

Certains moteurs sonores adoptent aussi ce genre de comportements dynamiques. C'est par exemple le cas de l'objet *hoa.decoder~*. Cet objet adapte automatiquement son nombre de sorties en fonction de la configuration choisie, en adaptant parallèlement le nombre d'entrée des objets situés en aval de la chaîne (*hoa.meter~, hoa.gain~*, *hoa.dac~*) si la nouvelle configuration choisie le nécessite. Cette fonction de connexion automatique au sein de cet objet est d'ailleurs à la base de l'élaboration du module de configuration dynamique du décodage que nous présentons dans cette dernière partie.

3.2.2 MODULE DE CONFIGURATION DYNAMIQUE DU DECODAGE

Pour finir d'égrainer cette suite non-exhaustive d'utilitaires mis à la disposition de l'utilisateur au sein de la bibliothèque HOA, nous aborderons une dernière série de module permettant de configurer dynamiquement le décodage d'une chaîne de traitement ambisonique pour un ordre de décomposition donné afin de s'adapter à des systèmes de restitution particuliers.

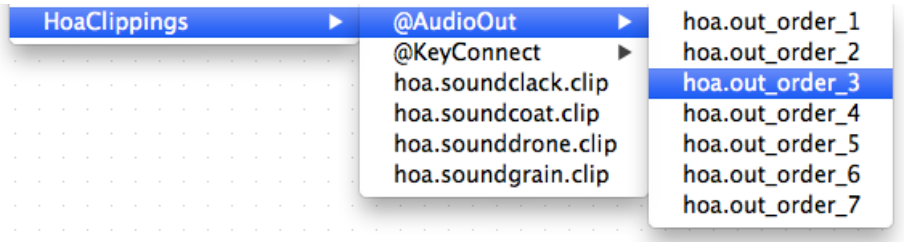

Figure 68 : représentation de l'accès au menu contextuel des *clippings* au sein d'un *patch* MaxMSP.

[87] Cela pourrait néanmoins poser quelques problèmes dans la mesure où certains traitements, par exemple la synthèse de champ diffus par granulation, ne sont viables qu'à des ordres de décomposition élevés.

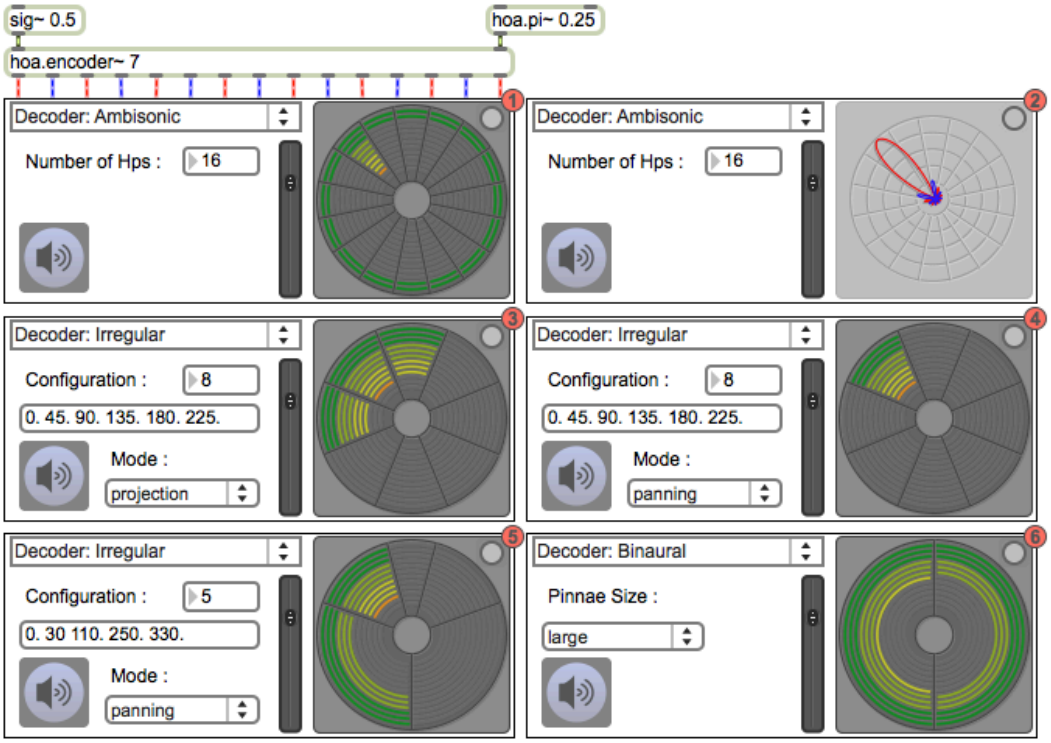

Figure 69 : représentation de quelques configurations disponibles au sein du bpatcher *hoa.helpout* pour MaxMSP à un ordre de décomposition 7. Représentation d'un signal d'amplitude 0.5 encodé pour un angle d'incidence de $\pi/4$.

En ambisonie, le décodage permet de convertir ou restituer un champ sonore précédemment encodé dans le domaine des harmoniques circulaires, en signaux destinés à des haut-parleurs physiques. Le décodage ambisonique consiste à projeter le champ sonore sur une série de haut-parleurs répartis de façon équidistante sur un cercle autour de l'auditeur. Cette technique de diffusion est assez contraignante dans la mesure où elle nécessite, pour un rendu optimal et sans distorsion du champ sonore, un nombre minimum défini de haut-parleurs et un placement précis de ceux-ci. Le nombre de haut-parleurs nécessaire à un décodage ambisonique classique est fonction de l'ordre de décomposition. Pour un ordre n, il est égal à $(2 * n + 1)$. Ainsi, pour un ordre de décomposition 3, un minimum de sept haut-parleurs sera nécessaire pour restituer le champ sonore (il sera néanmoins souvent d'usage d'en avoir huit). Afin de s'adapter à des configurations moins conventionnelles et ainsi permettre à des utilisateurs ne disposant pas toujours d'un système de restitution ambisonique approprié, nous avons choisi d'élargir nos techniques de décodage. Au sein de la mise en œuvre MaxMSP, l'opération de décodage se fait grâce à l'objet *hoa.decoder~*[88]. La bibliothèque HOA permet, par l'intermédiaire de cet objet, de s'adapter à divers systèmes de

[88] Nous invitons le lecteur à se référer à l'aide de cet objet au sein du livrable MaxMSP de la bibliothèque HOA pour des informations détaillées sur son fonctionnement.

restitution et configurations. Nous proposons grâce à lui un module permettant de basculer simplement d'une configuration à une autre pour des ordres de décomposition allant de 1 jusqu'à 7. Ces modules sont facilement accessibles à partir d'un *patch* depuis le menu *clippings* de MaxMSP [Figure 68]. Placées comme module de sortie dans la plupart des *patchs* d'aide de la mise en œuvre MaxMSP (conçus pour la plupart avec un ordre de décomposition de 7), ces interfaces permettent à l'utilisateur d'adapter facilement la chaîne de traitement qui leur est présentée à leurs propres systèmes de restitution. Chacun de ces *bpatchers* contient les objets *hoa.decoder~*, *hoa.gain~*, deux interfaces de représentation du signal, ainsi qu'un objet *hoa.dac~* qui nous permet de proposer un module « tout en un » de sortie. Par défaut, le décodage est un décodage par projection pour un système ambisonique classique [Figure 69.1]. Dans ce mode, le nombre de haut-parleurs ne peut être défini en dessous du minimum requis (par exemple 15 à l'ordre 7), l'utilisateur peut néanmoins augmenter ce nombre s'il a plus de haut-parleurs en sa possession[89]. Si l'utilisateur dispose au contraire d'un système de restitution restreint ne lui permettant pas de monter à des ordres de décomposition élevés, il devient dès lors nécessaire de passer à une configuration de décodage irrégulière afin de s'adapter à un nombre de haut-parleurs inférieur à celui requis normalement pour un ordre de décomposition donné. Deux modes de décodage irréguliers sont proposés au sein de l'objet *hoa.decoder~* et par voie de conséquence dans ce module. Le premier opère un décodage par projection adaptée à la configuration souhaitée [Figure 69.3], le second opère un décodage par *panning* [Figure 69.4] grâce à une adaptation de la technique de « *vector base amplitude panning* » [Pulkki, 2002]. Ces deux techniques permettent donc de s'adapter à priori à n'importe quel système de restitution sur haut-parleurs répartis de façon circulaire autour de l'auditeur. Les [Figure 69.3 et 75.4] exposaient une configuration de type octophonique, mais il est aussi possible de s'adapter à d'autres configurations comme un système 5.1, en spécifiant les angles de chacun des haut-parleurs [Figure 69.5]. Nous proposons enfin un rendu au casque grâce à une technique de restitution binaurale du champ sonore avec une configuration possible du lobe d'oreille [Figure 69.6] [90].

[89] Notons qu'augmenter le nombre de haut-parleurs ne fera que renforcer le rendu du champ sonore, sans pour autant améliorer la résolution spatiale.

[90] Nous ne disposons malheureusement pas du temps nécessaire à une explication plus détaillée de ces différents modes dans ce cadre. Nous renvoyons donc le lecteur à [Guillot, 2013] ainsi qu'à [Colafrancesco et al., 2013], qui décrivent avec précision l'ensemble de ces techniques de décodage particulières ainsi que les stratégies ayant été adoptées pour permettre ces mises en œuvre.

En sus du dynamisme offert par ces techniques de décodages, ce module permet de gérer facilement le niveau général de sortie grâce à l'objet *hoa.gain~*. Il permet enfin d'offrir aussi une représentation des signaux reçus en entrée à travers l'intégration de deux interfaces sur lesquelles l'utilisateur peut basculer à partir du menu contextuel se situant en haut à droite de chacun d'eux. Deux représentations différentes du champ sonore sont pour l'instant permises par l'intermédiaire des interfaces *hoa.meter~* et *hoa.scope~*. La première, comme nous avons pu le voir précédemment, qui permet d'offrir, suite à l'opération de décodage, une représentation des contributions de chacun des haut-parleurs en fonction du système de restitution choisi [Figure 69.1]. La seconde offrant elle une représentation du champ sonore dans le domaine des harmoniques circulaires, avant l'opération de décodage [Figure 69.2]. Afin de fournir une représentation toujours plus détaillée du champ sonore, l'objet *hoa.spectrum~* viendra compléter ces deux représentations au sein de ces interfaces utilitaires lorsque celui-ci aura atteint une version plus définitive dans les futurs livrables de la mise en œuvre MaxMSP de la bibliothèque HOA.

Nous venons de présenter ici une série de modules venant compléter la collection d'interfaces de contrôle et de représentation précédemment exposées au sein de la bibliothèque HOA pour MaxMSP. Ces outils ont pour vocation d'améliorer l'intégration et l'ergonomie générale de la bibliothèque au sein de cet environnement de programmation visuelle et permettent de remédier de façon pertinente aux inconvénients de l'approche modulaire que nous avons choisi d'adopter. Certains de ces objets utilitaires, comme *hoa.connect~* ou *hoa.plug~*, ont été initialement conçus et développés pour nos propres besoins, apparus au cours de l'élaboration de la bibliothèque HOA, afin de nous faciliter le prototypage des différents moteurs sonore au sein de l'environnement logiciel MaxMSP qui nous servait alors de laboratoire expérimental. Ils ont pu permettre, ou du moins faciliter l'élaboration et l'émergence de traitements divers tels que la réverbération ou l'ordre fractionnaire qui n'ont été implémenté que plus tard en langage C++ pour être optimisés.

Cette série d'utilitaire, bien qu'à un stade avancé de développement, reste tout de même perfectible et sujette à de possibles modifications ultérieures. Nous n'excluons donc pas de changer notre façon d'opérer en nous rapprochant par exemple de l'approche adoptée par la bibliothèque *jamoma* en terme de gestion des différents canaux si de nouvelles contraintes le nécessitent. Ce pourrait notamment être le cas si nous décidons de rendre compatible l'ensemble des modules de la bibliothèque avec des systèmes de restitution en trois dimensions ; ceux-ci nécessiteraient un nombre d'entrées et de sorties beaucoup plus important complexifiant d'autant la mise en place de traitements, les objets prenant dès lors beaucoup plus de place au sein d'un *patch*[91].

L'utilité des différents modules exposés ici a pu largement être démontrée lors de la présentation de la bibliothèque que nous avons pu effectuer lors des *JIM 2013*[92]. Cette démonstration se déroulait à la manière d'un atelier participatif au cours duquel le public était invité à prendre en main l'ensemble des outils de la bibliothèque en reproduisant eux-mêmes, sur leur ordinateur personnel, un *patch* MaxMSP que nous développions en temps réel et projetions sur grand écran[93]. Outre l'aspect pédagogique et ludique de la démarche, cet atelier de *live patching* a pu mettre en avant la rapidité de création de traitements ambisoniques permise à la fois par les différentes interfaces graphiques mais aussi par l'ensemble des outils utilitaires que la bibliothèque intègre, tels que l'objet *hoa.connect* ou *hoa.plug~* ou encore le module de configuration dynamique du décodage qui leur a par exemple permis d'obtenir rapidement un rendu du même traitement au casque.

[91] Notons au passage que pour un même ordre de décomposition de 7, un encodeur ambisonique en deux dimensions dispose de 15 harmoniques et comporte donc 15 sorties, un encoder en trois dimensions en nécessiterait 64.

[92] Journées d'informatique musicale, édition 2013, organisées par le centre de recherche en informatique et création musicale à l'Université de Paris-VIII du 13 au 15 mai.

[93] Notons que le public présent aux JIM, à la différence de celui présent lors des autres installations que nous avons pu faire, était un public initié aux pratiques de la composition assistée par ordinateur et aux environnements de programmation visuelle tels que MaxMSP.

Conclusion

La bibliothèque HOA a été mise à la portée du musicien et du compositeur à travers les différentes mises en œuvre que nous avons pu effectuer notamment à destination des environnements de programmation visuelle comme MaxMSP ou Pure Data, mais aussi sous la forme de plugins VST, qui agissent toutes en tant que tel comme interface entre l'utilisateur et les algorithmes de traitement du signal. Ces mises en œuvre intègrent un ensemble d'interfaces graphiques de représentation et d'interaction, facilitant pour l'utilisateur la compréhension des principes sous-jacents à l'ambisonie ainsi que le contrôle, la manipulation et la représentation des champs sonores. Ces interfaces offrent ainsi une valeur ajoutée indéniable aux moteurs sonores eux-mêmes en proposant des modes d'interaction spécifiques augmentant, en nombre et en qualité, les opérations réalisables sur le champ sonore grâce à des représentations symboliques des processus sonores mis en œuvre.

Nous avons abordé en premier lieu la question du contrôle du champ sonore en nous intéressant à une des fonctionnalités qui reste sûrement des plus attendues de la part des musiciens ou compositeurs abordant une nouvelle bibliothèque de mise en espace du son, à savoir le contrôle de sources sonores ponctuelles. Après avoir décrit les spécificités et les principes relatifs à l'encodage ambisonique de sources sonores ponctuelles, nous avons abordé les solutions que nous proposions en terme de gestion précise de coordonnées à travers une interface graphique dédiée à cet effet ; permettant la manipulation d'une série de sources et de groupes ainsi que la création de trajectoires dans l'espace et dans le temps. Nous avons ensuite pu élargir le contrôle des sources sonores à des paramètres de plus haut niveau, assistant algorithmiquement le compositeur dans la mise en espace du son grâce à des interfaces tirant parti de modèles physiques tels que celui des *boids* [Reynolds, 1987] afin d'ajouter un dynamisme inhérent au procédé de spatialisation. La bibliothèque HOA, grâce aux techniques ambisoniques, permet aussi d'exercer des transformations d'ensemble sur le champ sonore. Nous avons étudié à ce propos une série d'interfaces graphiques, dédiées au contrôle de traitements permis par l'interopérabilité entre le domaine des harmoniques

circulaires et des ondes planes, qui facilitent la manipulation d'opérations telles que le filtrage spatial ou encore la *distorsion de la perspective* [Daniel, 2001].

Un autre mode d'interaction possible entre l'utilisateur et la matière sonore mise en espace se situe dans la visualisation et la description du phénomène sonore lui-même. La seconde partie de ce mémoire a donc été consacrée à ces interfaces, permettant d'offrir des représentations graphiques du champ sonore à partir de l'extraction de paramètres sonores physiques ou perceptifs tels que l'amplitude, l'intensité, la phase ou le contenu spectral. Ces interfaces permettent ainsi un meilleur contrôle des traitements ambisoniques dans la mesure où l'utilisateur peut désormais visualiser et analyser finement l'incidence de chaque manipulation qu'il exerce sur le champ sonore, et ceci à tous les niveaux de la chaîne de traitement du signal.

Le compositeur, à travers la mise en œuvre de la bibliothèque à destination des environnements de programmation visuelle, peut mettre en place la chaîne de traitement ambisonique qu'il souhaite ou encore créer toute sorte de traitements encore inouïs dans le domaine des harmoniques circulaires grâce à l'approche modulaire que nous avons choisi d'adopter. Néanmoins, nous avons évoqué le fait que ces environnements ne facilitaient pas ce type de mise en place. Nous avons donc cherché à faciliter la mise en place et la création de traitements ambisoniques en factorisant et en parallélisant un maximum d'opérations de connexions ou d'instanciations d'objets. Le compositeur, facteur de ses propres instruments numériques, a donc désormais à sa disposition une série d'outils utilitaires à vocation à la fois pédagogique et pratique améliorant l'ergonomie de la bibliothèque HOA pour MaxMSP et l'interface générale de cet environnement de programmation visuelle.

L'ensemble des moteurs sonores et interfaces de la bibliothèque a pu être validé, à la fois par la communauté scientifique et sur le plan de la création musicale, à travers différentes installations sonores ou ateliers que nous avons pu présenter au cours du projet. Ces installations nous ont permis de valider l'ergonomie de nos interfaces mais aussi de mettre à jour de nouvelles stratégies de contrôle du champ sonore innovantes à travers l'emploi de contrôleurs gestuels tels que la *Kinect*, la *wii Balance Board* ou les tablettes graphiques. Ces contrôleurs autorisent souvent des opérations moins précises sur le plan spatial ou temporel qu'une simple interface graphique sur ordinateur, mais offrent en parallèle des interactions beaucoup plus complexes et ludiques par leur rapport particulier au

toucher ou au geste. Nous avons présenté ici quelques prototypes d'interfaces graphiques permettant le contrôle de processus sonores de la bibliothèque HOA sur tablette graphique par une exploitation du système *multitouch*. Ces différents modes de contrôle représentent selon nous un vecteur évident de création musicale que nous devrons explorer lors des projets de développement futurs de la bibliothèque HOA.

Les principales perspectives d'évolution futures pour les interfaces de la bibliothèque HOA se situent aussi actuellement dans la résolution des problèmes liés à la gestion des paramètres musicaux dans le temps, et dans l'adaptation aux systèmes de restitution tridimensionnels. Afin de s'adapter à un maximum de configurations accessibles aux musiciens, l'ensemble des traitements sonores disponibles au sein de la bibliothèque HOA est restreint pour l'instant à deux dimensions spatiales. Nous proposons alors des solutions de décodage pour différents systèmes de restitution communs tels que les systèmes ambisoniques, stéréophoniques, un rendu au casque ou pour des systèmes standard tels que le 5.1. L'ajout d'une troisième dimension au sein des moteurs sonores représenterait néanmoins un paramètre intéressant à exploiter sur le plan de la création musicale. Si les techniques ambisoniques permettent de passer aisément d'un système de représentation à un autre, l'adaptation de la plupart des interfaces présentées ici pour prendre en compte une troisième dimension, pose néanmoins de nombreux problèmes que nous devrons solutionner.

La question du temps n'a été traitée que de façon partielle ici, elle reste cependant une dimension cruciale de l'écriture de l'espace. La mise en œuvre sous la forme de plugin VST des traitements et interfaces permet d'apporter une première réponse à la gestion des paramètres dans le temps, déléguant ainsi cette tâche au logiciel hôte par l'intermédiaire des automations. La gestion du temps est en revanche plus complexe à mettre en place au sein des logiciels de programmation visuelle. Des solutions existent néanmoins, notamment en tirant parti de systèmes tels que le *pattrstorage*[94]. L'ensemble de nos interfaces de contrôle, comme nous avons pu le voir, a été rendu compatible avec ce système. Néanmoins il semble ne pas convenir à tous les usages et n'offre pas de représentation des événements spatiaux dans le temps. S'ouvre alors devant nous un chantier ambitieux, déjà entamé par d'autres

[94] Nous renvoyons le lecteur aux *patchs* d'aides respectifs de cette série d'objets (disponible uniquement sous MaxMSP)

projets de recherche[95], destiné à offrir une représentation ainsi qu'un contrôle solide des traitements de l'espace et du son dans le temps.

[95] Nous pensons notamment aux projets *Virage* ou *i-score* : http://i-score.org/.

Bibliographie

[Aengus, 2005] Aengus Martin, *Algorithms for Sonification : Smoke, Boids and the Game of Life*, Thèse de doctorat, Music & Media Technologies, Department of Electronic & Electrical Engineering & School of Music, Trinity College, Dublin, 2005.

[Berge et al., 2010] Svein Berge, Natasha Barrett, *High Angular resolution Planewave expansion*, Proceedings of the 2nd international symposium on Ambisonics and Spherical Acoustics, Paris, France, 2010.

[Colafrancesco, 2012] Julien Colafrancesco, *L'ambisonie d'ordre supérieur et son appropriation par les musiciens : présentation de la bibliothèque Max/Msp Hoa.Lib*, Actes des Journées d'Informatique Musicales 2012, Mons, Belgique, 2012.

[Colafrancesco et al, 2013] Julien Colafrancesco, Pierre Guillot, Eliott Paris, Anne Sedes, Alain Bonardi, *La bibliothèque Hoa, bilan et perspectives*, Actes des Journées d'Informatique Musicales, 2013, Paris, France, 2013.

[Couturier, 2004] Couturier Jean-michel, *Utilisation avancée d'interfaces graphiques dans le contrôle gestuel de processus sonores*, thèse de doctorat, Université Aix-Marseille II, spécialité ATIAM, déc. 2004.

[Daniel, 2001] Jérôme Daniel, *Représentation De Champs Acoustiques, application à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimédia*, Thèse de l'Université Paris 6, 2001.

[Gerzon, 1992] Michael Gerzon, *General Metatheorie Of Auditory Localisation*, AES Preprint, 1992.

[Guillot, 2013] Pierre Guillot, *Les traitements musicaux en ambisonie*, mémoire de Master II, Université Paris-VIII, juin 2013.

[Guillot et al., 2013] Pierre Guillot, Eliott Paris, Manuel Deneu, « *La bibliothèque HOA pour MaxMSP, Pure Data, VST, FAUST* », Revue Francophone d'Informatique Musicale [En ligne], Numéros, n° 3 - automne 2013, à paraître.

[Harpex, 2012] Harpex Ltd, Manuel d'utilisateur du plugin Harpex version 1.2, http://harpex.net.

[Penha, Oliveira, 2012] Rui Penha, João Pedro Oliveira, *Spatium, uma abordagem modular e open source ao software para espacialisação*, Seminário Música Ciência Tecnologia, 2012.

[Place et al., 2010], Timothy Place, Trond Lossius, Nils Peter, *The Jamoma Audio Graph Layer*, Proceedings of the DAFx-10, Graz, Austria, 2010.

[Pottier, 2012] Laurent Pottier, « Le contrôle de la spatialisation », *La spatialisation des musiques électroacoustiques*, sous la direction de Laurent Pottier, Publications de l'université de Saint-Etienne, Saint-Etienne, 2012.

[Reynolds, 1987] Craig W. Reynolds, *Flocks, Herds and Schools : A Distributed Behavioral Model*, reprint of the original publication in the proceedings of SIGGRAPH '87 (Computer Graphics 21(4), July 1987, edited by Maureen C. Stone, page 25-34).

[Reynolds, 1988] Craig W. Reynolds, *Not Bumping Into Things*, notes on "obstacle avoidance" for the course on Physically Based Modeling at SIGGRAPH 88, August 1 through 5 in Atlanta, Georgia. www.red3d.com/cwr/nobump/nobump.html

[Roads, 2001] Curtis Roads, *Microsound*, The MIT Press, Cambridge, Etats-Unis, 2001.

[Schacher, Kocher, 2006] Jan C. Schacher & Philippe Kocher, *Ambisonics Spatialization Tools for Max/MSP*, in : Proceedings of the international Computer Music Conference ICMC, New Orleans, November 2006.

[Schacher, 2010] Jan C. Schacher, *Seven years of ICST ambisonics tools for Max/MSP – a brief report*, in : Proceedings of the 2nd international symposium on Ambisonics and Spherical Acoustics - May 6-7, Paris, France, 2010.

[Thiebaut, 2005] Jean-Baptiste Thiebaut, *A graphical interface for trajectory design and musical purposes*, Actes des Journées d'Informatique Musicales 2005 (JIM 05), Saint Denis, France, 2-4 Juin 2005.

[Vinet, 1999], Hugues Vinet, « Concepts d'interfaces graphiques pour la production musicale et sonore » in *Interfaces homme-machine et création musicale*, sous la direction d'Hugues Vinet et François Delalande, Hermès-Sciences, Paris, 1999.

[Wakefield, 2006], Graham Wakefield, *Third-Order Ambisonic Extension for Max/MSP with Musical Applications*. ICMC, New Orleans, Etats-Unis, 2006.

Annexes

Le lecteur pourra se référer au site internet du projet HOA, sur lequel il trouvera les différentes mises en œuvre de la bibliothèque (pour MaxMSP, Pure Data, VST, FAUST), une série de textes généraux sur l'ambisonie d'ordre supérieur, ainsi qu'un descriptif plus détaillé des installations réalisées : http://www.mshparisnord.fr/hoalibrary/.

L'ensemble des codes sources de la bibliothèque HOA, ainsi que les *patchs* des différentes installations citées sont disponibles sur le répertoire *github* du projet HOA :

https://github.com/cicm/hoalibrary.

Le lecteur pourra aussi trouver les prototypes en cours d'élaboration dans le répertoire de *pre-release* : https://github.com/CICM/HoaLibrary/tree/master/_prerelease

Et de développement : https://github.com/CICM/HoaLibrary/tree/master/Max6/builds

La publication des JIM 2013 : Julien Colafrancesco, Pierre Guillot, Eliott Paris, Anne Sédès, Alain Bonardi, *La bibliothèque HOA, bilan et perspectives*, Actes des journées d'Informatique Musicales.

La publication RFIM : Pierre Guillot, Eliott Paris, Manuel Deneu, *La bibliothèque HOA pour MaxMSP, Pure Data, VST, FAUST*, Revue Francophone d'Informatique Musicale [En ligne], Numéro 3 - automne 2013, à paraître.