# Conception d'icones auditives paramétrées pour les interfaces homme-machine

Stéphane Conversy

Septembre 2000

# Table des matières


**1 Introduction** **6**
1.1 Son et interface . . . . . 6
1.2 Plan . . . . . 8

**2 La perception auditive** **9**
2.1 L'approche écologique de la perception visuelle . . . . . 9
2.1.1 Le système perceptuel visuel humain . . . . . 10
2.1.2 Informations structurées . . . . . 11
2.1.3 Invariants . . . . . 13
2.1.4 *Affordances* . . . . . 14
2.1.5 Théorie de l'acquisition d'informations . . . . . 16
2.2 L'approche écologique de la perception auditive . . . . . 17
2.2.1 Description physique du son . . . . . 17
2.2.2 L'appareillage humain pour la perception auditive . . 20
2.2.3 Flux sonore ambiant . . . . . 21
2.2.4 Informations sonores structurées . . . . . 22
2.2.5 Physique des événements sonores . . . . . 23
2.2.6 Relations entre paramètres physiques et sonores . . . 28
2.2.7 Taxonomie de Gaver . . . . . 29
2.3 Expériences sur l'approche écologique de la perception auditive 29
2.3.1 Identification des sons . . . . . 29
2.3.2 Perception de dimensions écologiques . . . . . 32
2.4 Résumé . . . . . 35

**3 Son et interfaces homme-machine** **37**
3.1 Le son dans l'interface . . . . . 37
3.1.1 Spécificités du son . . . . . 38
3.1.2 Rareté des interfaces auditives . . . . . 40
3.2 Techniques . . . . . 42
3.2.1 Les icones auditives . . . . . 42
3.2.2 Traitements . . . . . 43
3.2.3 Les *Earcons* . . . . . 43
3.2.4 *Auralisation* . . . . . 45

3.2.5 Correspondances entre dimensions . . . . . 46
3.3 Son et tâches de fond . . . . . 47
3.3.1 Audio Aura . . . . . 48
3.3.2 ShareMon . . . . . 49
3.3.3 OutToLunch . . . . . 52
3.3.4 Arkola . . . . . 53
3.3.5 ENO . . . . . 55
3.3.6 EAR . . . . . 56
3.3.7 *Earcons* . . . . . 56
3.4 Evénements et interfaces auditives synchrones . . . . . 57
3.4.1 Sonic Finder . . . . . 57
3.4.2 *Earcons* . . . . . 59
3.4.3 Feedback redondant . . . . . 61
3.4.4 Interfaces pour non-voyants . . . . . 62
3.4.5 Jeux et son 3D . . . . . 63
3.5 Résumé . . . . . 63

**4 Synthèse sonore d'icones auditives paramétrées 65**
4.1 Conception d'icones auditives . . . . . 65
4.1.1 Son numérique . . . . . 66
4.1.2 Sons échantillonnés . . . . . 67
4.1.3 Synthèse . . . . . 69
4.2 Modèles physiques . . . . . 71
4.2.1 Son de corde pincée . . . . . 71
4.2.2 Masses et ressorts . . . . . 72
4.3 Modèles de signaux . . . . . 73
4.4 Objets frappés et frottés de Gaver . . . . . 74
4.5 Synthèse ad-hoc . . . . . 76
4.5.1 Motivations . . . . . 76
4.5.2 Principes . . . . . 79
4.5.3 Mise en œuvre . . . . . 82
4.5.4 Les machines de Gaver . . . . . 83
4.5.5 Le vent . . . . . 84
4.5.6 Les vagues . . . . . 87
4.5.7 La pluie . . . . . 89
4.5.8 Le bouillonnement . . . . . 92
4.5.9 L'ascenseur . . . . . 95
4.6 Résumé . . . . . 97

**5 Sofa 98**
5.1 Systèmes de synthèse . . . . . 98
5.1.1 Caractéristiques d'un système pour icones auditives . 99
5.1.2 Systèmes musicaux . . . . . 101
5.1.3 Systèmes IHM . . . . . 104

5.2 Sofa . . . . . 106
5.2.1 Principes . . . . . 106
5.2.2 Temps réel . . . . . 110
5.2.3 Scripts . . . . . 113
5.2.4 Intégration . . . . . 121
5.3 Exemples d'application . . . . . 123
5.3.1 Exemples d'usages . . . . . 124
5.3.2 Suivi de téléchargement . . . . . 124
5.3.3 Suivi de travaux d'impressions . . . . . 126
5.4 Résumé . . . . . 127

**6 Conclusion** **129**

# Chapitre 1

# Introduction

La thèse présentée dans ce mémoire s'inscrit dans le cadre de l'"interaction homme-machine". L'interaction homme-machine est un domaine multidisciplinaire qui réunit le génie logiciel, la psychologie, le *design* et l'art et qui a pour objet l'étude et le développement de systèmes informatiques appelés à être appréhendés et manipulés par des êtres humains. Concevoir une interface consiste donc à faire appel à des théories sur le comportement humain, à des outils matériels et logiciels, à des méthodes de conception et à des méthodes de validation.

## 1.1 Son et interface

Nous nous servons de cinq sens pour percevoir notre environnement : la vue, l'ouïe, le toucher, le goût et l'odorat. Jusqu'à présent, c'est l'aspect visuel qui été le plus étudié dans les interfaces. D'ailleurs, le mot "interface" en lui-même est presque devenu un diminutif de "interface graphique" tant les ordinateurs d'aujourd'hui ne communiquent que par cet intermédiaire. Les applications utilisant le canal audio commencent cependant à se développer. Les travaux de recherche sur la multi-modalité, et surtout la synthèse et la reconnaissance de la parole commencent à porter leurs fruits, et de récents produits commerciaux utilisent les techniques qui ont été développées dans les laboratioires.

En revanche, l'utilisation du son non-parlé reste très rare. Certes, les ordinateurs personnels produisent quelques sons lorsqu'une erreur survient, mais les interfaces ne vont pas au-delà d'une production simple de un ou deux éléments sonores différents. Pourtant, les informations sonores que nous captons nous permettent d'appréhender l'environnement qui nous entoure. Par exemple, si nous ne voyons pas une voiture s'approcher parce qu'elle est cachée par une maison, entendre le son de son moteur nous incite à ne pas franchir la rue. De plus, par le son du moteur et des crissements de pneu, nous pouvons déterminer à quelle vitesse va la voiture et dans

quelle disposition se trouve le conducteur, ce qui peut nous conduire à nous protéger. Les sons sont donc des vecteurs d'information qui nous permettent de nous comporter et de survivre dans le monde.

Nos travaux s'inscrivent dans le domaine spécifique de l'utilisation de sons non-parlés dans l'interface, dans le sens machine vers utilisateur. De nombreux travaux de recherche ont eu pour but d'étudier comment les sons non-parlés pouvaient être incorporés dans l'interface. Ils ont suivi des théories "classiques" sur la perception des sons [12], et ont utilisé les caractéristiques de bas-niveau des sons comme la fréquence ou l'amplitude. D'autres ont utilisé des motifs musicaux pour transmettre des informations. Pour notre part, nous avons suivi les travaux de Gibson et son approche écologique de la perception [43, 42], et ceci pour deux raisons. Premièrement, cette théorie de la perception nous paraît plus facile à appréhender par les utilisateurs, car elle décrit le son en termes d'éléments d'interêt pour l'être humain vis-à-vis d'un objet sonore, et non en termes musicaux. Deuxièmement, elle conduit à des techniques de sonorisation qui permettent d'utiliser naturellement ces sons dans les interfaces, les icones auditives paramétrées. Les icones auditives sont des sons contrôlables par des dimensions écologiques comme la taille d'un objet ou la force du vent.

Si la théorie de la perception et les principes des icones auditives ont déjà été décrits par Gaver [35], peu d'applications utilisent les icones auditives paramétrées dans leur interface. La première raison tient au fait que peu d'ordinateurs personnels sont livrés en standard avec du matériel de production sonore (carte sonore et haut-parleurs), et que les sons sont généralement considérés comme une gêne pour les utilisateurs. Les concepteurs d'interface sont donc peu enclins à utiliser des sons dans l'interface, et ceci reste vrai pour toute technique sonore. La deuxième raison vient du fait qu'il existe peu d'algorithmes permettant la génération d'icones auditives paramétrées par des attributs de haut-niveau. Seuls les travaux de Gaver ont pavé cette voie pourtant prometteuse [38]. Le premier but de cette thèse est de pallier le manque d'icones auditives paramétrées et d'en proposer de nouvelles. Ainsi, dans le troisième chaptire, nous décrivons plusieurs algorithmes de synthèse en temps réel d'icones auditives qui permettent de transmettre des informations de haut niveau sur des activités continues.

Il existe peu d'outils permettant de créer et d'intégrer des icones auditives paramétrées dans des applications. La disponibilité d'outils de prototypage et d'intégration est nécessaire au développement d'interface. En effet, la conception participative permet aux utilisateurs et aux concepteurs de concevoir ensemble des interfaces. La conception suit un cycle partant de la génération d'idées, passant par une phase de prototypage puis par une phase de tests [56, 61]. Le cycle recommence et par itérations successives on produit une interface qui convient aux utilisateurs, grâce à leurs idées et au savoir-faire des concepteurs. Pour réduire la durée des cycles et pour tester rapidement des idées de conception, il faut disposer d'outils de prototy-

page. Malheureusement, si de tels outils existent dans le domaine graphique comme Tcl/Tk [1] [62], ou dans le domaine du collecticiel comme GroupKit [69], il n'existe pas d'outils similaires pour les icones auditives paramétrées. La deuxième partie de cette thèse décrit un outil qui permet la conception et l'intégration d'icones auditives dans l'interface.

## 1.2 Plan

Dans le premier chapitre, nous décrivons l'approche écologique de la perception. Nous introduisons les principaux concepts de l'approche écologique en abordant son application à la perception visuelle, puis nous abordons la perception auditive en mettant l'accent sur les aspects de compositions de sons complexes avec des sons plus simples. La fin du chapitre décrit des expériences menées par des chercheurs en psychologie qui appuient cette approche et qui l'évaluent.

Au second chapitre, nous présentons les différents travaux de recherche qui concernent l'utilisation du son dans les interfaces. La première partie présente les différents moyens d'utilisation du son, puis nous décrivons les travaux par type d'activité. Tout d'abord, nous abordons le suivi d'activités d'arrière-plan puis la notification événementielle.

Dans le troisième chapitre, nous étudions les différents moyens dont nous disposons pour créer des icones auditives paramétrées. Nous montrons ensuite comment une méthode de synthèse "ad-hoc" permet de produire ces sons, notamment en utilisant le fait que les invariants des sons sont aussi importants que leur spectre.

Enfin, dans le quatrième chapitre, nous introduisons Sofa, un outil de conception et d'intégration d'icones auditives paramétrées. Nous montrons pourquoi les systèmes de synthèse disponibles ne conviennent pas à la conception d'icones auditives, et nous donnons les principales caractéristiques qu'un tel système doit posséder. Sofa permet notamment d'utiliser des scripts pour programmer le comportement des sons et pour les contrôler. La fin du chapitre est consacrée à des exemples d'application, qui illustrent les capacités d'intégration de Sofa dans les applications interactives.

[1] http ://www.scriptics.com

# Chapitre 2

# La perception auditive

Selon le dictionnaire Hachette, la perception est la “représentation d’un objet, construite par la conscience à partir des sensations”. Cette définition n’est pas satisfaisante parce que l’on ne connait pas précisément le vecteur de la représentation, ni ce qu’est exactement la conscience dans ce cas. Ne peut-on percevoir inconsciemment ? Quand je marche dans un bois, ai-je conscience des arbres que j’évite ? Certains concepts sont difficiles à définir, soit parce qu’ils s’expriment à l’aide d’autres concepts eux-même difficiles à définir, soit parce que l’on ne s’entend pas sur la définition de concepts reliés. “être sensible à”, “percevoir”, “avoir conscience de”, sont des concepts dont la définition est capitale pour la compréhension du comportement humain, et donc capitale pour la conception d’une interface homme-machine. En identifiant les éléments mis en jeu lors de l’appréhension par l’Homme de son environnement, nous ne pouvons que mieux les utiliser.

L’objet de ce chapitre est d’analyser la façon dont nous percevons le monde auditif. L’approche utilisée est celle de la perception écologique. James Jerome Gibson est l’inventeur de cette approche, et l’a principalement dévelopée pour la perception visuelle [43]. La première partie du chapitre est donc consacrée à la perception visuelle et introduit les concepts importants de l’approche écologique. La deuxième partie reprend ces concepts pour les appliquer à l’approche auditive. Enfin, la troisième partie décrit quelques expériences qui tentent de valider cette approche et d’en mesurer la valeur, notamment en vue d’une utilisation dans une interface.

## 2.1 L’approche écologique de la perception visuelle

L’approche écologique de la perception est une explication du processus de capture d’informations de l’environnement que l’on doit à James Jerome Gibson, psychologue du XXème siècle [43]. Elle remet en cause à la fois la notion d’informations à percevoir et la définition des systèmes perceptifs humains : les informations à percevoir ne sont pas de simples stimuli mais

un flot d'informations structurées, et les organes de la perception forment un véritable système perceptif actif [42].

Cette section introduit les principales notions de l'approche écologique de la perception par l'intermédiaire de l'analyse de la perception visuelle. Tout d'abord, le système perceptif visuel est analysé et redéfini. Ensuite, les informations disponibles dans le flux optique sont décrites. Les sections suivantes introduisent deux concepts clefs de l'approche écologique que sont les invariants et les *affordances*. Enfin, la dernière partie de la section concerne une théorie de la connaissance proposée par Gibson et construite à partir des concepts introduits précédemment.

### 2.1.1 Le système perceptuel visuel humain

Une des notions clefs de l'approche écologique de la perception consiste à redéfinir ce qu'il y a à percevoir, et les moyens humains pour percevoir. Ces définitions sont fortement liées car l'on ne perçoit évidemment pas ce qui ne peut être perçu par le système perceptuel.

Généralement, lorsque l'on énumère les organes humains entrant en jeu dans le processus de perception visuelle, on se concentre sur l'œil, et plus précisément sur la rétine. La rétine est formée de bâtonnets sensibles à l'intensité et à la couleur de rayons de lumière. Selon les approches traditionnelles de la perception, l'acte de perception ne commence qu'à partir de l'image formée au fond de la rétine, c'est-à-dire d'une surface formée de points de couleur et d'intensités différentes.

Pour Gibson, l'œil n'est qu'une partie d'un système perceptuel qui a changé au cours du temps comme l'explique la théorie darwinnienne de l'évolution : il s'est adapté aux informations les plus pertinentes pour la survie de l'être humain. En fait, plusieurs autres parties du corps humain participent au processus de vision. Notre système perceptuel visuel est constitué de deux yeux mobiles fichés dans une tête. La tête est elle-même mobile et peut tourner selon troix axes au niveau du cou. Enfin un être humain peut se déplacer grâce à ses jambes.

En considérant un tel système, les informations à percevoir ne se résument plus à une image rétinienne. Grâce aux possibilités de mouvement des organes, l'être humain peut agir sur la perception : il peut bouger ses yeux, pencher la tête, avancer. Les informations visuelles deviennent dynamiques, l'image se transforme en flot continu d'images. De plus, une personne peut revenir à une position initiale, fixer un point et revenir à un point précédent, avancer et reculer. Ces mouvements peuvent être commandés par une volonté de percevoir "mieux", c'est-à-dire que le processus perceptif n'est pas seulement un processus passif mais une boucle perception-action, où l'action génère la perception et la perception commande l'action.

### 2.1.2 Informations structurées

L'environnement est un terme qui réfère les objets qui font partie du quotidien des animaux. Il ne s'agit pas d'un espace physique, ni d'objets géométriques situés à des positions dans un monde à trois dimensions. La définition de l'environnement n'a de sens que pour l'animal pour lequel on va définir l'environnement. Ce concept est à rapprocher de celui de "niche écologique", qui se définit en termes d'emplacements, d'objets, et d'événements qui habitent et rythment la vie d'un animal. Le terme "écologique" accolé à celui de "perception" vient du fait que les définitions des concepts mis en jeu sont ceux qui permettent à l'animal de vivre et de se comporter, et qui ont permis à l'animal de s'adapter au cours de son évolution. Pour les êtres humains, l'environnement est structurellement hiérarchisé en montagnes, collines, bosses et rochers. Il existe des objets détachés comme les animaux, les feuilles mortes, les bouts de bois et les cailloux. D'autres sont attachés ou considérés comme tels car trop lourds à déplacer, comme les branches des arbres et les rochers. De plus, il n'existe pas de temps abstrait qui s'écoule de façon uniforme, mais des événements qui rythment la vie de tous les jours. Ces événements sont définis par les changements que les éléments de l'environnement subissent. Les événements sont eux-même hiérarchisés et peuvent spécifier des saisons, le jour et la nuit ou le débit d'une rivière par l'observation de la vitesse d'écoulement. Enfin, les animaux vivent dans un medium qui leur permet de respirer, que ce soit l'air pour les animaux terrestres, ou l'eau pour les animaux marins, voire qui leur permet de voler pour les oiseaux. L'air possède aussi cette propriété qui est de laisser passer la lumière et qui permet donc la vision.

Les objets de l'environnement sont visibles parce qu'ils émettent de la lumière ou parce qu'ils sont illuminés. Les éléments qui émettent de la lumière ne produisent généralement pas d'informations structurées. Seuls ceux qui émettent à faible puissance comme un bougie ou une petite lampe, peuvent servir à leur perception car leurs lumières n'éblouissent pas l'observateur. En revanche, une observation directe prolongée du soleil engendre des dommages irréversibles. De plus, il s'agirait d'une observation inutile car on ne discerne pas d'informations dans la lumière directe. Si la lumière du soleil ne produit pas d'information directement perceptible, elle illumine les objets de l'environnement et les rend visibles. Plus précisément, c'est grâce à l'illumination que l'on discerne les objets, en permettant la perception du contour des surfaces illuminées.

La notion de flux optique ambiant (*ambient optic array*) permet de comprendre en quoi les informations visuelles sont structurées. Le flux optique ambiant en un point est l'ensemble des informations visuelles qui convergent en ce point. Pour un être humain, l'ensemble des points possibles correspond aux positions possibles pour un tel observateur : un point est une position, un emplacement dans l'espace écologique. Quand une personne occupe ef-

fectivement un tel point, le flux optique ambiant se trouve augmenté par les informations concernant cette personne elle-même (bras, jambes, nez...). Le point d'observation fixe n'est mentionné qu'à titre d'explication. Dans un flux optique ambiant, le point d'observation n'est pas fixe, car l'observateur peut se déplacer dans l'environnement.

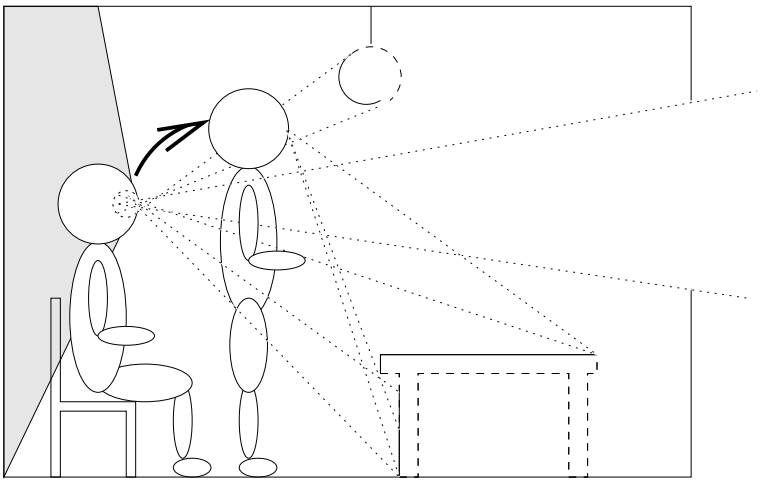

FIG. 2.1 – Le flux optique ambiant. Le flux optique ambiant est composé d'angles visuels formés par les surfaces visibles. Lorsque l'observateur se déplace, les informations de perspective changent, mais les invariants de structure ne changent pas.

Le flux optique ambiant n'est pas un ensemble de rayons lumineux indiscernables les uns des autres. Il est structuré pour l'observateur par des ensembles hiérarchiques, que l'on peut décrire selon le niveau de détail. Ainsi, à l'extérieur, les composants de l'environnement peuvent être hiérarchisés en deux parties comprenant le ciel et le sol. Dans chaque partie, il y a des sous-parties mobiles (nuages et oiseaux pour le ciel, animaux et cascades pour le sol par exemple) et immobiles (soleil pour le ciel, montagnes pour le sol). Les sous-parties peuvent elles-même être subdivisées. Par exemple, une *forêt* est un ensemble d'*arbres* composés d'un *tronc* et de *feuilles*.

Les composants du flux optique ambiant sont les angles visuels des composants de l'environnement (figure 2.1). Ils sont eux-même hiérarchisés : l'angle visuel de la forêt contient les angles des arbres qui la composent. Les angles possèdent cette caractéristique particulière qui stipule que quelle que soit la taille des objets dont ils proviennent, les angles sont petits pour les objets situés à l'horizon. De même, l'angle d'un objet grossit quand l'observateur se rapproche de cet objet, et rapetisse quand il s'en éloigne. La vitesse du changement de taille dépend de la vitesse de rapprochement et de la taille effective de l'objet. Les propriétés des angles visuels sont immuables dans

l'environnement naturel des animaux terrestres. C'est en s'appuyant sur ce type de propriétés qu'il a appris à connaitre et qu'il observe tous les jours qu'un animal perçoit les propriétés des éléments de son environnement. Par exemple elles permettent de discerner ce qui est atteignable rapidement de ce qui ne l'est pas. Le flux optique ambiant ne se résume pas à une disposition d'angles solides. D'autres paramètres, comme la texture, le gradiant, et les ombres en font aussi partie.

Le flux optique ambiant en un point fixe est un cas spécial du flux optique ambiant en un point mobile. Quand un observateur se déplace, les angles visuels grandissent ou diminuent. A chaque déplacement, le flux optique change, tout en conservant certaines propriétés. Il n'y pas de changements soudains de structure : un objet de forme ronde ne passe pas brusquement à une forme carrée par exemple. Ces propriétés sont dites invariantes par rapport au déplacement, ou invariantes par rapport à la perspective. Le fait de se déplacer vise en partie à découvrir les invariants de perspective qui ne sont pas perceptibles si le flux ne change pas. Ce qui change constitue l'information qui spécifie le mouvement. Ce qui ne change pas spécifie la structure de la disposition des objets. Ainsi, une table est perçue en tant que telle lorsque l'on découvre qu'elle est constituée d'une planche horizontale soutenue par des pieds. C'est en modifiant la vue, en changeant le flux optique que l'on perçoit les relations qui lient les pieds à la planche, et que l'on découvre qu'ils la soutiennent (figure 2.1). Bien sûr, il est possible de *reconnaître* une table sans faire de mouvement. En revanche, apprendre à *connaître* une table passe par le mouvement.

L'observateur perçoit également des informations relatives à lui-même. Le côté droit du flux optique de l'œil gauche est obstrué par le coté gauche du nez et vice-versa. De même, le fait de se déplacer est spécifié par le changement de structure du flux ansi que par l'apparation régulière des jambes et des pieds. Grâce aux informations dites proprioceptives, un observateur peut classifier les objets comme étant "à portée" ou "hors de portée" en tendant le bras.

### 2.1.3 Invariants

Pendant le processus de perception, il y a en parallèle une conscience de la persistance et et une conscience du changement. Quand une des dimensions du flux optique change, on perçoit que certaines choses changent quand d'autres ne changent pas. Ce qui ne change pas est dit invariant par rapport au changement de dimension. Les objets de la perception sont à la fois les variants et les invariants ; un animal est conscient de ces deux aspects. Gibson énumère quatre types d'invariants :

**Invariant selon le changement d'illumination.** Le soleil, les lampes peuvent émettre des lumières dans différentes directions, de différentes couleurs et de différentes intensités. La perception des dispositions des surfaces

reste la même quand l'illumination ne subit pas un changement abrupt. Il existe donc des invariants à l'illumination qui permettent cette continuité de perception.

**Invariant selon le changement du point d'observation.** Quand l'observateur se déplace ou bouge sa tête, le flux optique subit des changements de perspective, et des changements plus radicaux correspondant à l'occlusion et à l'apparition de surfaces. La disposition des surfaces en profondeur est spécifiée par l'invariant d'apparation-disparition qui est perçu à l'aide d'un déplacement suivi du retour à la position initiale. De même, la direction d'un déplacement est spécifiée par le point autour duquel les structures changent peu.

**Invariant selon la prise de vue.** Tourner la tête pour parcourir l'environnement conduit à l'apparation et à la disparition de surfaces selon le contour du champ de vision. Ce processus est réversible grâce à un parcours en sens inverse.

**Invariant selon les perturbations locales.** Les perturbations locales concernent les perturbations de la structure géométrique elle-même, comme les mouvements des objets détachés (animaux, pierre qui roule), ou les vaguelettes d'un lac. Percevoir ces changements ne change pas la perception des surfaces en ce qu'elles sont, c'est-à-dire un animal, une pierre, ou un lac.

La réversibilité des actions est importante puisqu'elle permet de distinguer les changements dûs à l'environnement des changements dûs à l'observateur. Si une surface disparaît et apparaît derrière une autre quand il oscille la tête, c'est que les objets sont immobiles et qu'ils sont placés l'un derrière l'autre par rapport à l'observateur. Par conséquent, il peut percevoir les positions relatives des objets en oscillant la tête et en constatant la disparition et l'apparation de surface. De plus, si des surfaces disparaissent et apparaîssent sans action de la part de l'observateur, et donc sans possibilité pour lui de revenir à une situation précédente, c'est qu'il observe un objet en mouvement ou un événement.

### 2.1.4 *Affordances*

Les surfaces qui représentent les artefacts de l'environnement ont une signification pour un animal. La question qui se pose est celle de savoir comment l'on passe de la perception des surfaces à la perception de ce que l'environnement propose ou permet à un observateur. Gibson définit les *affordances* d'un artefact comme les propriétés de cet objet qui peuvent être utiles à un animal. Selon lui, la perception de la composition et de la disposition des surfaces conduit à la perception de leurs *affordances*, et il y a perception directe des propriétés d'un objet relatives à un animal (figure 2.2).

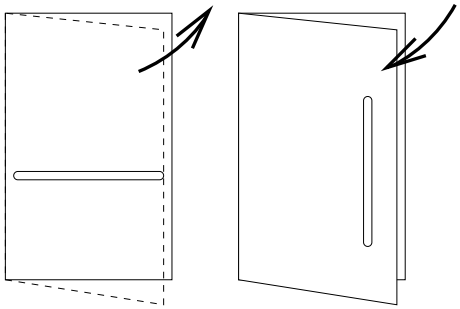

FIG. 2.2 – "Affordance". La porte de gauche a une poignée horizontale et basse, elle doit être poussée. La porte de droite a une poignée verticale, facilement préhensible et loin des gonds : elle "demande" à être tirée [60].

Par example, pour un être humain, un plan d'eau lui permet de se désaltérer et de se laver. Mais il lui "permet" aussi de se noyer. Le même plan d'eau permet au poisson de respirer, et à certains insectes de marcher. Ici, le plan d'eau n'a pas une "valeur" ou une signification objective, sortie de tout contexte. Il n'a de signification que pour un animal donné. En fait, les *affordances* n'existent qu'à travers deux parties : l'observateur et l'observé. Les *affordances* n'existent pas sans l'observateur. Cet aspect des *affordances* est à rapprocher du concept d'indexicalité en ethnométhodologie qui stipule que tout comportement n'a de signification que dans le contexte dans lequel il est observé, et qui ne peut être généralisée [26]. Ici, même si un même objet a plusieurs *affordances*, elles ne le sont qu'en fonction de l'observateur.

Si les *affordances* permettent de percevoir ce que l'on peut faire avec un objet, elles ne classifient pas cet objet. Par exemple, une chaise permet à un être humain de s'asseoir, de se reposer. Mais elle permet aussi à ce même être humain d'atteindre une ampoule grillée et de la changer, ou de casser une vitre en cas de feu. Une chaise n'est pas classée comme un objet qui permet de s'asseoir. Elle permet de s'asseoir comme elle permet d'autres utilisations.

Les *affordances* peuvent être positives ou négatives. Ainsi, si le plan d'eau permet de se désaltérer, il peut aussi permettre la noyade. Enfin, il n'y a pas de différence entre l'environnement naturel et l'environnement façonné par l'homme. Si l'homme a changé l'environnement qui l'entoure, c'est pour mieux s'en servir. Tout ce qui s'applique à l'environnement naturel s'applique à l'environnement "civilisé". Les *affordances* s'appliquent aussi aux éléments contruits par l'homme.

Pour Gibson, les sensations ne sont pas décorées de significations par un processus mental. De même, les objets ne sont pas décorés d'*affordances* par un processus mental. Les *affordances* sont perçues directement, et elles le sont car elles sont spécifiées dans les informations de stimulus. Nous avons

vu que nous percevions les invariants de structure. Les *affordances* peuvent être considérées comme des invariants d'invariants. Par exemple, c'est la combinaison des invariants de la perception proprioceptive et des invariants d'un objet qui permet d'évaluer les propriétés de cet objet relativement à l'observateur. En percevant à la fois un objet et lui-même, par exemple ses mains, l'observateur peut évaluer ce qu'il peut faire de l'objet, par exemple le fait d'être saisi par une main. Nous ne percevons le monde que parce que nous savons ce qu'il y a à percevoir.

Le concept d'*affordances* est dérivé des travaux des Gestaltiens, qui estiment eux aussi que les "valeurs" des objets sont perçues directement [78] (p137). La différence réside dans le fait que pour eux, les *affordances* changent en fonction des besoins de l'observateur. Ainsi, une boîte aux lettres indique à un observateur qu'il peut y poster une lettre. Cette "valeur" de la boîte aux lettres (le fait que l'on peut y poster des lettres) ne serait perçue que si l'observateur a une lettre à poster. Pour Gibson, les *affordances* ne changent pas en fonction des besoins. La boîte aux lettres est toujours considérée comme pouvant recevoir des lettres dans une société qui possède un système postal. A un moment donné, il se peut que cette *affordance* ne soit pas perçue, mais elle peut être perçue comme telle, même si l'observateur n'a rien à poster.

### 2.1.5 Théorie de l'acquisition d'informations

En expliquant sa théorie de l'acquisition d'informations, Gibson redéfini les principaux concepts impliqués dans la perception. Ainsi, la perception est vue comme une activité continue, sans pause, qui nous sert à rester en contact constamment avec le monde. Ce qui est perçu l'est en termes d'adéquation avec l'observateur. Les éléments ne sont pas classés selon leur couleur, leur forme etc. mais selon leurs capacités : "emplacements" plutôt qu'"espaces", "objets attachés et détachés" plutôt qu'objets dans un espace physique sans signification etc. Leurs changements d'états sont perçus comme les événements qui rythment le temps qui passe. Les informations de spécification des objets dont se sert le processus de perception sont directement disponibles dans le flux optique. L'acquisition d'information se fait à l'aide de systèmes perceptuels, dont le comportement peut être décrit selon une boucle perception-action. Ces systèmes perceptuels s'affinent et s'adaptent, optimisent leurs ajustements,. L'observateur distingue ce qui change et ce qui ne change pas : il perçoit les invariants du flot de sensations et ce qui varie avec le flot de sensations.

Le processus d'acquisition ne se sert pas de la mémoire pour fonctionner. L'information n'a pas à être gardée en mémoire, car elle est toujours disponible. La capacité toujours plus grande de capter des informations ne constitue pas les informations en elle-même, et n'aide pas le système perceptuel à fonctionner en faisant en sorte qu'il "s'attende" à percevoir des choses

précises. Seule la pratique et le règlage permettent d'affiner nos systèmes perceptuels : c'est en partie ce qui les distingue des systèmes sensoriels qui eux n'évoluent pas. Enfin, la perception ne se décrit pas en termes de perception passée et de perception présente. Elle ne se termine pas brusquement, comme la sensation peut le faire (en fermant les yeux par exemple), elle est continue et reste présente même si la sensation cesse.

Traditionnellement, les psychologues définissaient à l'aide de sens, nerfs, et entrées les processus mentaux de plus haut-niveau, comme la mémoire, la connaissance, l'inférence, le jugement etc. La redéfinition de l'acte perceptif par Gibson implique une redéfinition de ces concepts. Selon lui, tous ces concepts peuvent s'expliquer par une perception sans sensations. Ainsi, se souvenir c'est percevoir des invariants déjà perçus, mais sans que les informations sensorielles soient disponibles pour les sytèmes perceptuels. Ce qui nous permet de distinguer la perception du réel de la perception de l'imaginaire, c'est le fait que nos sytèmes perceptuels peuvent s'ajuster. Cela permet de tester la réalité, voire de découvrir de nouveaux invariants. Ainsi, faire varier la pupille permet de voir les choses plus ou moins floues, bouger la tête permet de faire apparaître et disparaître des surfaces.

Enfin, Gibson distingue "être au courant de" et "conscient de" (p250 de [43]) :

> the term awareness is used to imply a direct pickup of information, not necessarily to imply consciousness

En fait, la connaissance elle-même peut se définir en termes d'acquisition d'invariants. L'acte de perception du monde, et l'acte de concevoir le monde sont un seul et même acte, qui ne diffèrent que par le degré.

## 2.2 L'approche écologique de la perception auditive

La section précédente a introduit les concepts importants de la perception écologique visuelle. Cette section a pour but d'analyser la perception auditive en reprenant ces concepts et en les adaptant aux spécificités du son. Comme pour la vision, il est nécessaire de décrire le système perceptuel auditif. De même, il s'agit d'analyser la façon dont le son arrive jusqu'à nous, puis de comprendre comment les informations sonores sont structurées. Enfin, cette section analyse la taxonomie de Gaver [39] qui permet de classer les événements produisant du son.

### 2.2.1 Description physique du son

Cette partie est une introduction aux termes et aux concepts utilisés dans les discussions sur le son et l'écoute des sons. Nnous présentons les différentes manières de représenter un son, ainsi que ses caractéristiques principales.

### Représentation du son

Les sons sont des variations de pression qui se propagent dans l'air. Une des manières les plus répandues de représenter un son est de représenter l'amplitude en fonction du temps. On appelle cette représentation une forme d'onde.

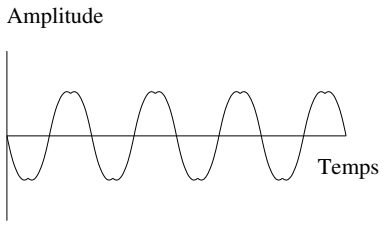


FIG. 2.3 – Une forme d'onde sinusoïdale. Ce graphe représente le son dans le domaine temporel.

Le son montré en figure 2.3 est une sinusoïde. Ce type de son est rarement rencontré dans la nature. Cependant, grâce aux travaux de Fourier [27], on montre que les sons complexes peuvent être exprimés comme la somme de sinusoïdes d'amplitudes, de fréquences et de phases différentes. A l'aide d'une transformée de Fourier, on peut retrouver ces sinusoïdes avec leurs caractéristiques respectives à partir d'une forme d'onde. On obtient un spectre, le deuxième type usuel de représentation d'un son. Le spectre représente l'amplitude en fonction de la fréquence. Ainsi, la forme d'onde de la figure 2.3 peut être représentée par la figure 2.4.

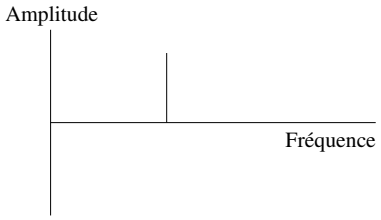


FIG. 2.4 – Le spectre d'une sinusoïde. Ce graphe représente le son dans le domaine fréquentiel.

Il est important de noter que la transformée de Fourier permet de passer du domaine temporel au domaine fréquentiel. L'opération inverse est réalisée avec la transformée de Fourier inverse. De plus, il n'y a pas de perte d'information lors de ces deux opérations. La fréquence est en quelque sorte l' "inverse" du temps : une forme d'onde est une fonction du temps (en $s$) et le spectre est une fonction des fréquences (en $Hz$ ou $s^{-1}$). D'une manière plus

générale, la transformée de Fourier permet de passer d'une représentation en fonction d'une variable à une représentation en fonction de l'inverse de cette variable. Avec une transformée de Fourier, on peut représenter une forme d'onde complexe de façon plus explicite (cf. figure 2.5), car on peut voir la contribution de chaque sinusoïde au son global. Les composantes fréquentielles d'un son sont appelées des partiels.

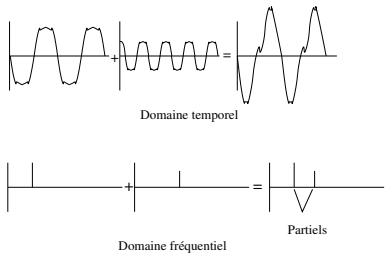


FIG. 2.5 – Un forme d'onde complexe composée de deux partiels. Le domaine des fréquences est plus facile à manipuler.

Si l'on regarde à nouveau la figure 2.5, on remarque que les deux composantes sinusoïdales n'ont ni la même amplitude, ni la même phase. Si l'on changeait la phase de la deuxième composante par exemple, la forme d'onde ne serait plus du tout la même. Pourtant, le son entendu serait identique au précédent, c'est-à-dire que la perception d'un son ne dépend pas des phases de ses composantes.

### Enveloppe d'amplitude

Généralement, un son évolue au cours du temps ne serait-ce que parce qu'il a un début et une fin. Un moyen de caractériser l'évolution de l'amplitude au cours du temps est de tracer l'*enveloppe* d'amplitude, c'est-à-dire le contour de la forme d'onde (cf. figure 2.6). La représentation la plus répandue dans le domaine musical est dite ADSR ("Attack", "Decay", "Sustain", "Release"). Cependant, il n'est pas obligatoire de se limiter à quatre segments, ni même à des segments.

Ce type de représentation peut être adapté au spectre, en définissant l'enveloppe de chaque partiel (figure 2.7). Les évolutions relatives de l'amplitude de chaque partiel permettent de caractériser la "couleur" d'un son, autrement dit son *timbre*. C'est généralement en analysant son timbre que l'on essaie de caractériser ce qui différencie un son d'un autre. Comprendre comment évoluent les partiels permet parfois de synthétiser des sons similaires. Nous reviendrons sur ce point dans les chapitres suivants.

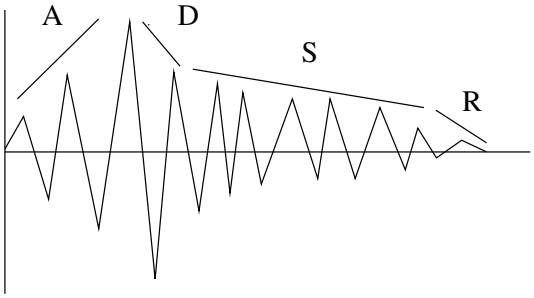


FIG. 2.6 – Une enveloppe ADSR pour une forme d'onde.

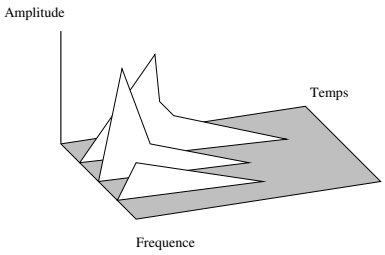


FIG. 2.7 – Un spectre variable au cours du temps.

### 2.2.2 L'appareillage humain pour la perception auditive

Le son est convoyé par les différences de pression de l'air. Les vagues de pression sont principalement captées par les oreilles. Elles entrent dans le canal auditif avant de faire vibrer le tympan, qui sert de cloison entre le milieu extérieur (air) et le milieu interne de l'oreille qui est liquide. En vibrant, le tympan fait vibrer le milieu interne. Ce système est hérité du système des ancêtres des mammifères : amphibiens, il n'y avait pas de différence de milieu entre l'oreille interne et externe, les vibrations passaient directement à l'intérieur de l'animal et faisaient vibrer entièrement l'animal [41].

L'oreille interne est composée de deux systèmes de perception de nature similaire mais de degrés différents. En étant sensible aux mouvements relativement longs et lents, l'un permet de capter les déplacements linéaires et les rotations. C'est ce système qui permet entre autre aux animaux de se tenir en équilibre, et de percevoir leurs propres mouvements. L'oreille participe donc certainement à la qualité de perception d'autres systèmes perceptuels, comme la vision.

L'autre système capte les vibrations rapides, que l'on appelle sons. Il est essentiellement composée du limaçon, un tuyau enroulé en spirale. L'évolution du limaçon différencie les mammifères d'autres espèces, comme les serpents ou les oiseaux. Chez eux, le limaçon est beaucoup moins développé, et est

à l'état d'embryon ches les amphibiens. Cette évolution permet aux mammifères de capter auditivement des informations beaucoup plus riches. A l'intérieur du limaçon se tient une bande hérissée de très petits "poils". Les poils ondulent avec le liquide interne quand il vibre, et créent une signature d'excitation spécifique du spectre sonore instantané. Les poils sont connectés aux fibres nerveuses qui transmettent les influx au cerveau. Ils ne sont pas connectés un à un aux fibres nerveuses : une fibre peut être activée par plusieurs poils, et un poil peut activer plusieurs fibres.

Les oreilles des mammifères sont mobiles et orientables, à des degré divers selon les espèces. Les oreilles sont disposées sur une tête orientable selon trois axes. Chez l'être humain, la tête est elle-même posée sur un corps qui peut se déplacer, se contorsionner au niveau du bassin, et tourner sur lui-même.

### 2.2.3 Flux sonore ambiant

A l'instar de l'étude de la perception visuelle, il convient de définir les informations disponibles pour le système auditif. Les animaux évoluent dans un flot sonore ambiant. Le son est propagé par le medium (l'air). Il est créé pas des vibrations rapides de corps, qui se propagent au medium. Contrairement aux informations visuelles qui proviennent en majeure partie de lumière réverbérée par des corps ou des substances, les informations sonores proviennent en grande partie de sons radiants, qui proviennent directement des objets vibrants. Les informations provenant de sons réverbérés spécifient l'environnement dans lequel se trouve l'observateur et l'objet sonore [39].

Pour les rayons de lumière, le medium *permet* le passage, car il est translucide. Sur terre, l'intensité des rayons lumineux peut être modifiée par le brouillard, ou la pollution. Sous l'eau, par le jeu des diffractions et à causes des particules stagnantes, le medium transforme le flux optique. Pour le son, le medium est le *vecteur* de propagation du son. La qualité du son dépend donc de la qualité du medium. Ainsi, la vitesse de propagation des vagues de pression de l'air a une valeur finie de 300 m/s à la surface de la terre. Cette propriété a un conséquence directe sur la perception du son. Comme la vitesse du son est relativement peu élevée par rapport à la vitesse de déplacement d'objets sur terre, la vitesse de l'objet sonore par rapport au medium a une influence sur le spectre des sons émis. L'oreille a une très bonne acuité au niveau de la différentiation de fréquences très proches, elle est donc sensible aux perturbations engendrées par les déplacements de l'objet sonore : c'est l'effet Doppler, que l'on peut expérimenter lorsqu'un véhicule muni d'une sirène se rapproche ou s'éloigne de l'auditeur.

La propagation des vibrations de l'air est d'ordre mécanique et est donc susceptible de subir des pertes durant le processus. Ce phénomène résulte en une perte d'intensité proportionnelle à la distance, qui est différente selon les fréquences mises en jeu. Ainsi, les fréquences hautes subissent plus de pertes que les fréquences basses, le medium agissant comme un filtre dit

"passe-bas". Les invariants spécifiant la distance sont donc la perte globale d'intensité, et la perte progressive des fréquences les plus hautes.

Le système de perception auditive est capable de localiser une source sonore. Les informations caractérisant la localisation proviennent en partie de la différence d'intensité et du décalage temporel perçus entre les deux oreilles. Gibson doute que cette différence se traduise en différentes sensations [41]. Il considère les deux oreilles comme formant un seul système perceptuel et pense que seule la différence est perçue, et non pas les valeurs à chaque oreille. Quand les informations commencent à être reçues, le système de perception auditive s'ajuste et s'oriente en activant les muscles internes et externes des oreilles et en tournant la tête. Pour se diriger vers une source, il suffit d'orienter la tête pour réduire les différences temporelles et d'intensité, et avancer dans cette direction. Un animal perçoit qu'il se rapproche d'une source quand le son devient plus fort et plus riche en hautes fréquences.

Le système perceptuel s'adapte également pour pouvoir discriminer plusieurs sources sonores. Ainsi, s'il existe plusieurs sources sonores dans l'environnement, un animal est capable de concentrer son attention sur une source particulière. Ce comportement peut s'expliquer en partie par une orientation du système auditif telle que le système synchronise en intensité et en décalage temporel les deux flux sonores pour la source considérée et qu'il désynchronise les autres. Il existe d'autres indices dans le flux sonore qui permettent la localisation. Comme nous le verrons plus loin dans ce chapitre, les échos qui accompagent un son peuvent aider ce processus, ainsi que la "coloration" du son due à la réverbération.

### 2.2.4 Informations sonores structurées

A l'instar de la vision, nous abordons maintenant ce qui est auditivement pertinent d'un point de vue écologique, et comment cela se traduit dans le flux auditif. En fait, on peut décrire une scène sonore comme étant composée d'une ou plusieurs *sources* émettant des sons depuis des *positions* dans un *environnement* [39]. Ainsi, une voiture dispose d'un moteur et de roues. Les parties du moteur qui s'entrechoquent et qui se frictionnent produisent des vagues de pression caractéristiques des éléments mis en jeu et du type d'interaction qui les anime. Ici, les pièces du moteur représentent les sources, c'est-à-dire des objets physiques qui lorsqu'ils sont soumis à une interaction vibrent et émettent des sons. Dans ce cas, les vagues de pression sont le résultat d'interactions sur des sources définies. Tous les sons ne sont pas de cette nature : ainsi, les sons dûs à une explosion ne sont pas dûs à des matériaux vibrant, mais à une action directe sur le medium et à un changement rapide de pression.

Quand un son est émis, il est propagé dans plusieurs direction. Les vagues de pression sont en général des sphères concentriques, dont le centre est la source sonore. En voyageant dans l'environnement, les vagues de pression su-

bissent des changements dûs au medium. Les sons sont ainsi très différents sous l'eau et sur terre. Même sur terre, le fait de traverser une atmosphère humide comme un brouillard dense modifie le son. L'environnement terrestre est généralement couvert d'obstacles sur lesquels le son rebondit ou au travers desquels le son est propagé. Ainsi, dans un environnement montagneux les sons sont accompagnés d'échos qui spécifient la taille des vallées. De même, les endroits clos ont des obstacles qui génèrent des réflexions multiples. Le son arrive à nos oreilles suivi d'un bruit résultant des réflexions structurées que l'on référence sous le terme de "réverbération".

Les sons permettent à leur sources d'être localisées, à la fois en direction et en distance. La direction des ondes sonores permet d'en déterminer le centre, donc la position de la source émettrice. Cependant, le système perceptuel auditif n'a pas l'acuité du système visuel et ne permet pas de déterminer l'emplacement d'une source avec une aussi grande précision. L'atténuation des fréquences les plus hautes est un invariant spécifiant la distance des sources sonores.

Comme pour les objets visuels, les sons sont hiérarchisés. Ainsi, si un piéton écoute une voiture, il entend généralement un bruit de moteur, voire de frottements de roues. Si l'auditeur est mécanicien, il aura tendance à détailler le son en une composition de sons provenant des éléments mécaniques du moteur qu'il connaît. Cette décompostion hiérarchique a une influence sur la perception d'attributs comme la distance. Si un immeuble s'effondre, un observateur pourra le localiser s'il est situé à quelques centaines de mètres. Si l'observateur se trouve par malheur au milieu des bâtiments, il entendra des sons provenant de toutes les directions. Dans le premier cas, c'est le fait que les bruits de murs génèrent un son global qui permet de le localiser. Ces deux exemples mettent en évidence le concept d'*affordances*, ces informations perçues qui dépendent à la fois de l'objet d'intérêt et des intérêts de l'observateur. Le piéton ne traverse pas la route parce qu'il a entendu une voiture. Le mécanicien trouve la panne en écoutant les bruits qui ne lui paraissent pas normaux. L'observateur distant de l'effondrement se verra piqué de curiosité, alors que la victime d'un tremblement de terre cherchera à se protéger.

### 2.2.5 Physique des événements sonores

Les sons spécifient et sont spécifiés par les attributs des sources. Ainsi, des attributs tels que la taille, le poids ou la forme sont des dimensions qui peuvent être directement perçues par le système de perception. La question qui se pose est de savoir comment ces variables ont une influence sur le spectre des sons émis, ou en d'autres termes, quels sont les invariants qui spécifient ces dimensions.

Pour Gaver, on peut classifier les sons selon trois grandes catégories de sources [39]. La première concerne les sons resultant de vibrations de

matériaux plus ou moins solides. Ainsi, un claquement de porte consiste en un choc entre la porte et le montant de la porte. De même, les bruits de moteurs sont essentiellement dûs aux frottements des pièces metalliques qui le composent.

Dans la deuxième catégorie, on trouve les sons engageant des éléments liquides. Parmi ces sons, on trouve ceux composés de sons de gouttes d'eau. Le bruit des chutes d'eau par exemple est la somme d'une multitude de bruits de gouttes d'eau et d'écoulements. Une autre sous-catégorie correspond aux bruits transformés par les cavités temporaires formées par les liquides, comme lorsqu'une bouteille se vide.

Enfin, dans la troisième catégorie, on trouve les sons formés par les explosions. Ces sons naissent de vagues de pression directement introduites par des phénomènes atmosphériques, comme la compression subite du medium.

### Solides en vibration

Quand on frappe un diapason, celui-ci se met à vibrer et émet un son. Le diapason est conçu de façon telle qu'il émet un son composé d'une seule fréquence déterminée, ce qui permet au musicien d'accorder son instrument. Le diapason émet un son car sa vibration est transmise au medium, puis propagée jusqu'à notre système perceptuel. Une grande variété d'événements générant des sons sont de ce type, comme les frottements, les pas d'un être humain, ou du verre qui se brise. En fait, un solide au repos est un système en équilibre. En appliquant une force, le solide se déforme et revient à sa position initiale à cause des forces internes qui spécifient sa rigidité. Quand le solide vibre, des forces de frottement internes et externes entre en jeu, ce qui conduit à une perte d'énergie progressive. Au bout d'un temps dépendant des caractéristiques du solide, le système atteint sa position d'équilibre et ne vibre plus. Le son émis diminue en intensité durant cette période [39].

Plusieurs dimensions écologiques ont une influence sur le spectre sonore résultant d'une vibration. Elles peuvent être divisées en trois catégories : les dimensions spécifiant l'*interaction*, celles spécifiant le *matériau* du solide, et celles spécifiant sa *configuration* [39]. Ainsi, le type d'interaction modifie l'évolution de l'amplitude du son, ainsi que son spectre. Par exemple, après un impact, le son résultant ne sera pas changé par l'interaction (qui est terminée), alors qu'un frottement fournit une quantité continue d'énergie qui se traduit par une évolution constante du spectre. De même, plus la force d'un impact est grande, plus grande est la bande de fréquences couvertes par le spectre. La configuration du solide a elle aussi une influence sur le son résultant. La forme change la signature du spectre, alors que la taille joue sur la fréquence de base. Si le solide n'est pas homogène, ou qu'il comporte des cavités, la disposition des fréquences dans le spectre est différente de celles d'un son provenant d'une masse homogène.

Les attributs énumérés ci-dessus ont soit une influence majoritairement

spectrale, soit une influence majoritairement temporelle. Par exemple, la taille d'un objet détermine en grande partie la fréquence de base de son spectre, et a une influence plus modérée sur la longueur du son. Cependant, il est difficile de classer catégoriquement les dimensions par le type de leur influence. Par exemple, si la longueur d'une barre est surtout spécifiée par son influence sur la fréquence de base, elle contraint également la durée d'un son d'impact. Ainsi, les sons d'impacts sur une barre en métal ont des fréquences hautes qui durent plus longtemps que les fréquences basses. Si la barre est grosse, il y a moins de hautes fréquences, et le son dure moins lontemps.

### Evénements aérodynamiques

La spécificité des sons provenant d'événements aérodynamiques réside dans le fait que la variation de pression du medium est directement introduite par l'événement. Ainsi, quand un ballon explose, le gaz interne sous pression est libéré et se dilate immédiatement et brusquement. C'est l'onde frontale de pression qui est perçue. Il semble que la présence de hautes fréquences caractérise les explosions soudaines et violentes, alors que celle des basses fréquences dépendent de la quantité de gaz concerné [39].

Il existe d'autres événements susceptibles de produire des explosions. Ainsi, le bruit du tonnerre est dû à la soudaine augmentation de température et à la brusque dilatation de l'air situé le long de l'éclair. De même, les explosifs repoussent l'air quand une réaction chimique ou nucléaire a lieu. D'autres événements dont la cause vient de phénomènes aérodynamiques font vibrer des solides : le vent dans les cables et les feuilles par exemple, ou le bruit des pales d'un ventilateur.

### Liquides

Les sons impliquant des liquides peuvent être en partie assimilés à des sons impliquant des solides : ils résultent le plus souvent d'impacts de gouttes sur une surface liquide. Mais le son si caractéristique des gouttes d'eau provient du fait qu'en heurtant la surface, cette dernière se déforme en donnant naissance à des cavités résonnantes. Les cavités changent de forme au cours de l'impact, passant d'une forme de "bol" entouré d'une "couronne" de goutelettes, à une bulle d'air complètement entourée d'eau qui se résorbe rapidement. Un son de goutte est donc un son d'impact coloré par la réverbération du "bol" évoluant au cours du temps, suivi d'une suite d'impacts de goutelettes. Un tel son dépend notamment de la densité du liquide. Lorsqu'il s'agit d'un objet qui tombe dans l'eau, le son produit dépend de la masse et de la forme de l'objet.

**Evénements complexes**

Les événements sonores décrits jusqu'ici peuvent être considérés comme élémentaires. Ils peuvent être inclus dans une séquence composée de multiples événements sonores, et servir de base à un événement de plus haut-niveau. Ainsi, on peut décomposer le bruit d'une porte qui se ferme en des sons successifs tels que le bruit de la poignée qui tourne, le grincement des gonds, l'impact de la porte sur le montant, et le cliquetis du pêne.

Certains sons isolés n'apportent pas les mêmes informations qu'ils apportent quand ils sont entendus accompagnés. Ainsi, un impact simple ne spécifie pas un bruit de pas. Un impact suivi d'autres impacts, avec une période restant dans les limites de la marche humaine sera perçu comme un bruit de pas. De plus, en entendant un bruit de pas, nous ne percevons pas une suite d'impacts, mais bien une seule et même information, celle d'une personne qui marche. Nous percevons en fait un invariant, celui qui stipule qu'un impact suivi d'autres impacts avec une fréquence restant dans les limites de la marche humaine spécifie une marche humaine. Le concept "personne qui marche" est directement perçu, ainsi que les informations qui l'accompagnent : la vitesse est spécifiée par la fréquence des impacts, alors que la santé (ou vulnérabilité pour un prédateur) est spécifiée par la régularité. Si la vitesse est trop grande, ou l'irrégularité trop forte, nous cesserons de percevoir une personne qui marche. Cet exemple montre comment le fait de composer des événements complexes permet d'ajouter des informations aux informations fournies par les événements plus simples. Ainsi, si l'on peut deviner à partir du seul son d'impact sa force ainsi que le type des matériaux mis en jeu, une séquence d'impacts apportent les informations supplémentaires décrites plus haut. On peut se demander d'ailleurs si lors de l'écoute du premier son d'impact, on n'évalue pas d'abord les caractéristiques du son en termes de propriétés propres au son entendu, ces caractéristiques se transformant ensuite en propriétés différentes comme la masse d'un personne dans cet exemple.

La perception de différents sons complexes à partir de même sons de base a été étudiée par Warren et Verbrugge [82] qui ont découvert que la distinction entre un verre qui se casse et un verre qui rebondit est faite par les sujets sur la base d'invariants spécifiques temporels, plutôt que sur les informations du spectre. Bien que le bruit que l'on trouve au début du son du verre qui se brise est absent dans le cas du rebond, enlever cette information du spectre ne change pas la perception de l'événement. Il semble que la seule information nécessaire se situe dans les structures temporelles de plus haut niveau. Pour tester cette hypothèse, les auteurs ont créé des sons composés d'échantillons d'impact sur du verre, ne variant que dans leurs relations temporelles. Dans le cas de l'impact, les sons se succèdent selon une période qui diminue au cours du temps, comme le ferait un verre qui rebondit. Dans le deuxième cas, un ensemble de sons sont joués en même temps, puis sont

répétés à des intervalles irréguliers en reprenant le son de rebond pour chacun des éclats de verre qui se dispersent sur le sol (figure 2.8). Ces ensembles de rebonds sont perçus comme faisant partie d'un seul événement à cause du synchronisme du premier rebond. Cette seule information temporelle est suffisante pour que les sujets distinguent les deux classes de sons. Ainsi, seuls certains invariants sont nécessaires pour spécifier ces sons, dans ce cas, les relations temporelles.

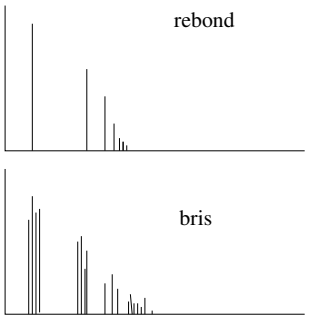


FIG. 2.8 – Relations temporelles entre les échantillons. Pour le rebond, l'amplitude des sons diminue au cours du temps et leur fréquence augmente. Pour le bris, de multiples impacts sont entendus au même moment, puis chacun suit le comportement d'un rebond.

En fait, il est difficile de déterminer la frontière entre événement complexe et événement simple. Par exemple, Gaver décrit le son d'une goutte d'eau comme un impact suivi de petits impacts de goutelettes éjectées par le premier impact [39]. Il s'agit déjà d'une perception d'un "tout" sonore, composée de parties certes difficilement discernables sans analyse fine. Pourtant, dans l'approche écologique de la perception, il n'y a pas de différences fondamentales entre la perception d'un événement difficilement décomposable à priori et la perception d'un événement complexe facilement décomposable, comme peut l'être un bruit de pas.

L'évolution dans la composition des sons peut aussi aider la perception d'événements de plus haut-niveau. Prenons l'exemple du remplissage d'un récipient avec un liquide. Si une personne remplit une bouteille avec de l'eau, elle est peu enclinte à faire attention de ne pas perdre du liquide en continuant à remplir même si la bouteille est pleine, car le prix de l'eau du robinet est peu élevé. En revanche, si cette personne remplit un verre avec du vin de qualité, elle évitera de faire déborder le verre et risquer la perte de liquide, le salissement de la nappe, et la gêne sociale. Elle devra donc diminuer la vitesse de remplissage en inclinant moins la bouteille, et en diminuant le taux avec lequel le liquide emplit le verre. Le filet de liquide sera plus mince, et le son spécifiant le filet sera perçu comme tel. De plus, on évitera certainement un débit trop rapide de peur de perdre du liquide.

Cet aspect de l'action sera spécifié par le fait que l'écoulement du liquide ne subira pas d'à-coups dûs au appels d'air successifs d'une bouteille trop penchée. On voit donc que le changement du son spécifiant le filet d'eau et l'absence d'à-coups dûs aux appels d'air peuvent spécifier le remplissage d'un verre avec un breuvage de qualité. Il n'est pas dit ici que ce sont les seuls invariants mis en jeu : un verre de vin sera aussi spécifié par le son produit, car il forme une cavité de forme et de matériau différents d'une bouteille en plastique.

### 2.2.6 Relations entre paramètres physiques et sonores

Comprendre les processus physiques impliqués dans la génération de sons est utile pour comprendre comment les dimensions écologiques sont captées par un animal. Mais comme le dit Gaver :

> savoir comment les processus physiques d'un événement déterminent le son qu'il produit n'est pas la même chose que savoir comment un son spécifie un événement [39].

Ainsi, il se peut que des caractéristiques physiques qui semblent influencer le comportement physique d'un événement ne soient pas perçues par un être humain. Par exemple, s'il est peut être facile de deviner le matériau d'un objet que l'on vient de frapper en écoutant le son produit, il est sûrement plus difficile d'en deviner la densité ou l'homogénéité. Comme le remarque Gibson :

> ce qui est simple pour la physique ne l'est peut-être pas pour la perception et vice-versa [42].

En d'autres termes, il convient de ne pas penser que tout paramètre de description physique qui semble indispensable au physicien sera perçu par un auditeur. Plusieurs dimensions physiques peuvent être perçues comme une seule dimension écologique, comme le montre les expériences de Carello et al. [16]. Il s'agit donc d'utiliser une *physique écologique*, à base de paramètres utiles pour l'observateur.

En outre, le fait que plusieurs paramètres physiques aient une influence sur les mêmes caractéristiques sonores ne doit pas nous laisser penser qu'il existe un processus mental qui permet de distinguer les changements physiques à la seule écoute de deux sons différents. C'est bien parce que le système perceptuel est capable de déterminer les invariants correspondant au changement d'un paramètre physique, que l'événement perçu est reconnu comme tel. Ainsi, si la taille d'un objet et sa forme ont tous deux une influence sur la disposition des fréquences dans le spectre, l'influence du changement de taille sur le spectre sera différente de celle du changement de forme à la fois en quantité et en qualité. En revanche, il est clair que du fait des influences mutuelles de paramètres physiques différents sur les mêmes attributs sonores, il peut être difficile pour un être humain de discerner les

changements quand ceux-ci sont spécifiques, par exemple si la différence est petite ou si plusieurs paramètres changent en même temps.

### 2.2.7 Taxonomie de Gaver

Reprenant son organisation en objets vibrant, sons aérodynamiques et sons liquides, Gaver l'étend et propose une taxonomie classifiant les sons de la vie courante [39]. Le plan de la figure 2.9 est organisé en trois zones correspondant aux catégories sus-citées. Les sons les plus simples se trouvent à la périphérie des zones alors que les plus complexes se situent au centre. Les zones se recouvrent au centre de la figure et spécifient les sons hybrides, composés de sons de plusieurs catégories.

Les sons les plus simples sont décrits en termes d'attributs physiques des sources et des interactions dont ils proviennent. Les sons plus complexes sont de trois sortes : les compositions temporelles de sons identiques comme la marche, les compositions temporelles de sons différents comme le jeu de bowling, et les sons impliquant des matériaux différents, comme un bateau motorisé naviguant sur un lac.

Gaver reconnaît que sa catégorisation ne mentionne pas tous les facteurs qui peuvent influer sur la perception d'un son [39]. Ainsi, comme vu auparavant, il est possible que certaines caractéristiques décrivant une source ou une interaction ne soient effectivement pas perçues. De même, certaines caractéristiques influent sur la qualité même de la perception : des sujets reconnaissent un objet métallique mais pas forcément une déformation en entendant le son d'une canette compressée.

## 2.3 Expériences sur l'approche écologique de la perception auditive

L'approche écologique est une explication de la manière dont les êtres humains perçoivent le monde afin d'y survivre et de l'utiliser. Après avoir analysé la perception auditive selon cette approche, cette section est consacrée à l'étude d'expériences destinées à valider cette approche ou à la décrire plus en détail. La première partie concerne les expériences visant à analyser comment des êtres humains identifient un son de la vie courante. La deuxième partie s'attache aux expériences dont le but est de comprendre comment les êtres humains perçoivent les dimensions écologiques des événements sonores.

### 2.3.1 Identification des sons

#### Description des sons

Dans une expérience destinée à observer la façon dont des sujets reconnaissent des sons quotidiens, Vanderveer leur a présenté une trentaine de

sons et leur a demandé de les décrire en une seule phrase [80]. Les résultats montrent que la plupart des réponses décrivent le son en terme de sources quand elles sont reconnues, et en termes de paramètres de sensations quand elles ne le sont pas.

Gaver a conduit une expérience similaire [39]. Il a demandé aux sujets de l'expérience de décrire en détail ce qu'ils entendaient, en leur soumettant des sons de la vie courante. Comme Vanderveer, il remarque que les sujets décrivent la plupart des sons en termes d'événements et de sources. Il remarque aussi que des détails sont parfois perçus, comme la différentiation entre un son d'une personne descendant un escalier et celui d'une personne en montant un. Enfin, les sujets ont eu du mal a reconnaître les sons inhabituels comme celui de quelqu'un marchant sur un sol recouvert de feuilles de papier. Si les sujets savent identifier des sons de la vie courante, ils ne les décrivent pas spontanément en précisant les attributs qui les caractérisent. Ceci peut s'expliquer par la nature de l'expérience elle-même, qui ne présentait qu'une seule instance de sons de natures différentes.

#### Effets du contexte

Dans le domaine du langage, on sait que la reconnaissance des mots utilise le contexte dans lequel le mot est présenté, en l'occurence les quelques mots qui l'entourent dans la phrase et le sujet de la conversation [6]. Ballas et Mullin ont conduit une expérience visant à vérifier si le même phénomène se produit lors de la reconnaissance de sons de la vie courante [6]. L'expérience consistait à différencier des paires de sons semblables mais provenant de sources différentes, comme le son d'une personne marchant sur des feuilles et celui d'une mèche qui se consume. Pendant l'expérience, l'un des sons était présenté dans une séquence soit sémantiquement consistante avec le son, soit dans une séquence sémantiquement inconsistante avec le deuxième son de la paire non présenté dans la séquence, ou enfin dans une séquence composée de sons variés sans rapport avec l'un des sons de la paire. L'expérience a montré que si les séquences inconsistantes avaient un effet négatif sur la reconnaissance des sons, les séquences consistantes avaient très peu d'effets positifs par rapport à une tâche d'identification non-contextuelle.

Les séquences de sons ne sont malheureusement pas décrites dans l'article. De même, il n'est pas décrit à quel point les sons possédent des caractéristiques communes, comme par exemple s'ils ont subi des traitements comme une réverbération qui pourrait améliorer l'effet contextuel, en accentuant le fait que les sons font partie d'une même scène. Enfin, il n'est pas dit à quel point les sons sont séparés dans la séquences. Si les sons ont des colorations différentes, et sont ségementés d'une façon perceptible, il est possible que la tâche d'identification commence par celles des sons servant à introduire le contexte, et non celle du son étudié. Conscients que le problème peut venir de la reconnaissance du contexte lui-même, les expérimentateurs ont

ajouté une phrase parlée avant la présentation de la séquence. Encore une fois, l'expérience ne montre pas que le contexte a un effet positif significatif sur la reconnaissance.

### Facteurs communs d'identification

Ballas s'est attaché à déterminer les facteurs qui permettent l'identification de sons brefs de tous les jours [5]. Pour cela, il a conduit cinq expériences, chacune destinée à mesurer l'impact des facteurs acoustiques, écologiques, perceptuels et cognitifs.

La première expérience vise à mesurer l'incertitude causale ("causal uncertainty"), c'est-à-dire l'incertitude quant à déterminer la source potentielle d'un son par rapport au temps d'identification. Les sujets écoutent des son de la vie courante, tentent de les identifier, estiment leur familiarité avec chaque son, et mentionnent d'éventuelles alternatives. L'expérience montre qu'il y a une corrélation entre le temps d'identification et l'incertitude causale, c'est-à-dire que plus il y a de causes possibles, plus le temps d'identification est long. Les sons proposés sont des sons brefs, de durée inférieure à 600 ms.

La deuxième expérience a pour but de trouver une corrélation entre le temps d'identification d'un son et sa fréquence écologique, c'est-à-dire le nombre de fois où l'on peut entendre ce son dans la vie courante et dans un temps défini. Il est demandé à plusieurs sujets de porter un dispositif muni d'une alarme pendant une semaine. Au moment où l'alarme sonne, le sujet doit décrire le premier son entendu. Les moments auxquels l'alarme doit sonner sont consignés préalablement sur une feuille fournie à chaque sujet. Les sons relevés sont ensuite regroupés par similarité. Les résultats montrent qu'il existe une faible correspondance entre le temps d'identification et la fréquence écologique d'occurence. Les sons de signaux en particulier (sonnerie de porte, klaxon de voiture) sont entendus relativement peu fréquemment et possèdent l'incertitude causale la plus faible. Il est cependant intéressant de remarquer que tous les sons relevés concernent des événements mécaniques, et que la majorité des sons proviennent de machines modernes, et notamment d'ordinateurs.

Une autre expérimentation destinée à compléter la première teste la validité d'une cause possible pour un son donné. Avant de présenter un son, on donne à lire une phrase décrivant une action qui produit ce son. Le sujet doit décider si l'action décrite peut effectivement produire le son entendu. Cette expérience doit tester si les causes alternatives proposées par les sujets lors de la première expérience peuvent être acceptées par d'autres sujets, et elle doit permettre de constater une corrélation entre la vitesse de confirmation et les causes les plus probables. Les résultats confirment ceux de la première expérience et montrent que les causes les moins probables conduisent à des temps de réflexion plus longs, alors que les plus probables entrainent un temps de décision plus long lorsqu'elles sont rejetées.

Dans une dernière expérimentation, il a été jugé de l'influence de la "typicalité" d'un son sur le temps de reconnaissance. On présente aux sujets des sons qui ont été définis comme typiques ou atypiques d'une cause dans une expérience précédente. La cause est décrite de façon similaire à celle de l'expérience précédente, c'est-à-dire sous la forme d'une description textuelle, et évaluée en termes de "probabilité faible" et "probabilité forte". On demande aux sujets d'évaluer l'adéquation entre le son entendu et sa cause, c'est-à-dire de dire si le son correspond à la description de la cause. Les résultats montrent que les sons typiques sont traités plus rapidement que les sons non-typiques dans toutes les conditions (pour le rejet comme pour l'acceptation, pour des causes probables et improbables).

Les expériences mesurant l'incertitude causale et la réaction à des sons typiques sont liées. En effet, un son reconnu comme typique ne sera pas associé à d'autres sources potentielles, et aura une incertitude causale faible. D'ailleurs, les expériences montrent que l'incertitude causale a une influence sur la rapidité et la capacité d'identification, ce qui est consistant avec l'expérience sur la typicalité. En résumé, les expériences montrent qu'un son typique a plus de chance d'être reconnu qu'un son qui l'est moins.

### 2.3.2 Perception de dimensions écologiques

#### Perception de la longueur d'un objet

Carello et al. ont mesuré l'acuité du système perceptuel avec laquelle il mesure la perception de la longueur [16]. L'expérience consiste à écouter le bruit d'un bâton tombant sur le sol, et à estimer la longueur de ce bâton en fonction du son émis par l'impact. Deux expériences furent conduites, l'une avec des bâtons de 30 à 120 cm (table 2.1), l'autre avec des bâtons plus courts allant de 10 à 40 cm (table 2.2).

| longueur réelle | longueur perçue |
|---|---|
| 30 | 23.5 |
| 45 | 34.8 |
| 60 | 53.1 |
| 75 | 71.7 |
| 90 | 64.1 |
| 105 | 84.49 |
| 120 | 95.3 |

Tab. 2.1 – Longueurs réelles et longueur perçues pour des bâtons de 30cm à 120cm.

Les résultats montrent que les longueurs estimées sont cohérentes avec les longueurs effectives. Pourtant, la deuxième expérience montre que les lon-

| longueur réelle | longueur perçue |
|---|---|
| 10 | 13.6 |
| 15 | 17.4 |
| 20 | 19.4 |
| 25 | 21.9 |
| 30 | 25 |
| 35 | 25.7 |
| 40 | 26.6 |

TAB. 2.2 – Longueurs réelles et longueur perçues pour des bâtons de 10cm à 40cm. Les longueurs perçues sont plus courtes que les longueurs réelles de 25cm à 40cm

gueurs perçues sont plus courtes que les longueurs effectives pour les bâtons de 25 cm à 40 cm. Les auteurs expliquent ce phénomène par le fait que la longueur en elle-même ne modifie pas le son produit par la vibration de l'objet. Les contraintes sur la vibration sont en fait dues a des facteurs tels que la densité, l'elasticité etc. Ces résultats sont consistants avec ceux d'une expérience précédente [79] qui a mesuré comment des sujets étaient capables d'estimer la longueur d'un bâton en le tenant à l'une de ses extrémités avec une main et en le manipulant. Les forces ressenties haptiquement sont principalement dues à la "tension d'inertie", une mesure qui spécifie la répartition du poids le long de l'objet en fonction du point de rotation. En fait, si l'on compare la longueur perçue auditivement avec les paramètres qui décrivent la tension d'inertie, on s'aperçoit que les longueurs perçues sont cohérentes, même entre 25 et 40 cm :

> la perception compressée des petits bâtons est le résultat de la compression de leur magnitudes compressées comparées à une variable physique importante qui contraint les réactions du bâton lorque celui-ci heurte une surface, et par conséquent le son qu'il produit.

La longueur d'un bâton a donc une influence sur le son produit par l'intermédiaire de la tension d'inertie, mais n'est pas le seul facteur d'influence. Par conséquent, si l'on perçoit cette influence, on ne l'estime pas correctement de façon absolue pour certaines longueurs. Ce résultat rejoint la remarque sur la nécessité d'une physique "écologique", qui tient compte des paramètres importants pour l'être humain. Le fait de connaître les éléments physiques d'un son ne veut pas dire que l'on sait comment un son spécifie ces éléments.

| | réelles | | | perçues | |
|---|---|---|---|---|---|
| | hauteur | largeur | surface | hauteur | largeur |
| carré | 48.2 | 48.2 | 2.3 | 32.3 | 31.0 |
| petit rect | 61.0 | 38.1 | 2.3 | 37.3 | 29.9 |
| rectangle allongé | 91.4 | 25.4 | 2.3 | 44.5 | 25.2 |

TAB. 2.3 – Hauteur et largeur réelles et perçues suivent une "échelle définie"

### Perception de la forme d'un objet

Kunkler-Peck et Turvey ont conduit des expériences sur la perception de la forme d'un objet [47]. La première consistait à examiner si les sujets étaient capable de déterminer si un plateau était de forme carrée ou rectangulaire et si possible d'en estimer la longueur et la largeur en écoutant le son produit par un impact sur le plateau.

Les résultats montrent premièrement que les sujets sont capables de percevoir une hauteur et une longueur et deuxièmement qu'il existe une relation linéaire ($percue = 18.1 + 0.29 \times reelle$) entre les dimensions perçues et réelles (tableau 2.3). En fait, il existe une "échelle définie" ("definite scaling") entre les longueurs perçues et les longueurs réelles. L'échelle définie diffère de l'échelle absolue pour laquelle il existe une parfaite correspondance entre les dimensions réelles et perçues, et l'échelle relative, qui permet d'ordonner les dimensions mais pas de les quantifier. Ici, les dimensions perçues sont inférieures de 25% aux dimensions réelles. Même si les sujets ne perçoivent pas *exactement* les dimensions, le fait que les relations entre dimensions perçues et réelles suivent un échelle définie est remarquable. Les autres expériences montrent que les sujets perçoivent de la même façon les dimensions de plateau de matériaux divers (acier, bois, plexiglass). De plus, les sujets parviennent à discriminer la forme des objets frappés (triangle, carré ou rond) et ce quelque soit le matériau les composant.

Ces expériences montrent que les êtres humains sont capables de discriminer des sons selon des dimensions écologiques, et qu'ils sont capables de les quantifier, même si ce n'est pas de façon absolue. On s'aperçoit aussi que la façon dont sont évaluées ces dimensions sont différentes les unes des autres. Ainsi, l'évaluation de la longueur d'un bâton n'est pas la même en qualité que l'évaluation de la hauteur et la largeur d'un plateau. Selon la façon dont on utilise ces dimensions dans une interface, il faut tenir compte de ces facteurs. Par exemple, on peut associer des sons d'impacts à une sélection de fenêtre dans un environnement graphique, et régler les sons d'impacts sur la largeur et la hauteur de la fenêtre. Ici, les sons accompagnent la sélection graphique d'une fenêtre et peuvent donner des indices qui indiquent par le son les différences de forme. Dans ce cas, l'utilisation de la perception de

ces dimensions semble appropriée. En revanche, si l'interface est destinée à des non-voyants et que l'on veut donner des dimensions absolues, le fait que ces dimensions soient quantifiables ne doit pas nous laisser penser qu'elles seront évaluées justement.

## 2.4 Résumé

L'approche écologique de la perception redéfinit les systèmes perceptuels humains ainsi que les informations disponibles dans l'environnement [43]. Les objets de l'environnement possèdent des invariants de structure que l'on perçoit directement. Ces invariants permettent de percevoir un objet selon des dimensions d'interêt pour un être humain par le biais des *affordances*, comme le fait qu'il puisse être pris dans la main.

L'approche écologique de la perception auditive permet de décrire un son comme provenant d'une *source* située à une certaine *position* dans un *environnement*. Les attributs d'intérêt de l'événement sonore pour un être humain sont spécifiés dans le son par des invariants. Ces derniers sont directement perçus, ainsi que les *affordances* associées. Par exemple, il est possible de percevoir la longueur d'un objet en entendant un son d'impact sur cet objet. Les sons complexes peuvent être construits à partir de sons plus simples, et l'agencement de ces derniers apportent de nouveaux invariants. Ainsi, la fréquence d'occurence de sons d'impacts peut caractériser la vitesse d'une personne qui marche.

Des expériences montrent que les sujets, lorsqu'ils identifient un son, le décrivent en termes de sources et d'interactions [80, 39]. Les sons sont d'autant plus facilement identifiés qu'ils sont typiques de l'événement qu'ils caractérisent [5]. De plus, les sujets sont capables de quantifier les dimensions comme la longueur [16] et de reconnaître le matériau et la forme d'un objet [47] à partir de sons d'impact sur ces objets.

Dans le chapitre suivant, nous verrons que l'approche écologique de la perception auditive est à la base du concept d'*icones auditives paramétrées*, des sons dont les dimensions sont écologiques et qui permettent la notification et le suivi d'activités dans les interfaces homme-machine.

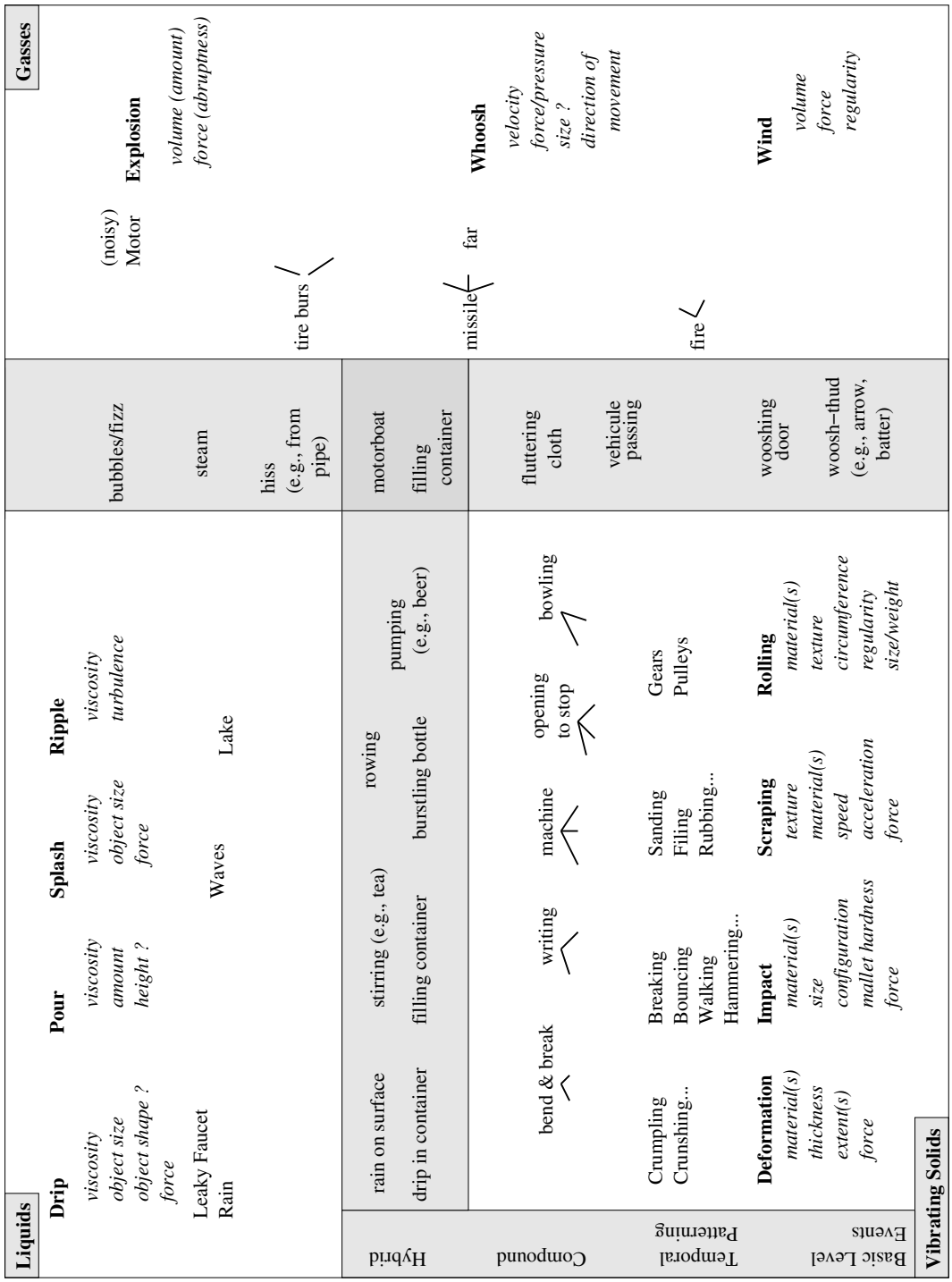


FIG. 2.9 – Taxonomie des sons écologiques de Gaver [39].

# Chapitre 3

# Son et interfaces homme-machine

C'est un lieu commun que de constater que peu d'interfaces homme-machine utilisent le son. Certes, les interfaces de Windows et de MacOS produisent quelques bips, mais il est difficile de parler d'utilisation du son vu la pauvreté des informations délivrées auditivement par ces systèmes. Depuis l'avénement interfaces graphiques et vu l'engouement qu'elles suscitent, on peut se demander pourquoi le canal sonore que nous utilisons activement dans la vie de tous les jour, a été si peu exploité jusqu'ici.

Ce chapitre expose les travaux de recherche effectués dans le domaine de l'utilisation du son dans l'interface. Comme nous le verrons, il est possible d'utiliser le son de diverses manières : en utilisant des sons de tous les jours, des sons musicaux, voire des mélodies. De même, les usages du canal sonore sont variés et chacun d'entre eux utilise un aspect particulier du medium : événements, tâches de fond, collecticiels, "*auralisation*", etc.

La première partie décrit en quoi le canal auditif est spécifique par rapport au canal visuel et en quoi ses caractéristiques sont intéressantes pour l'interface. Ensuite, nous tenterons de déterminer les raisons de sa sous-utilisation dans les interfaces. La deuxième partie énumère les différentes moyens d'utilisation du son. Ensuite, les travaux concernant l'utilisation du son pour le suivi d'activités d'arrière-plan sont énumérés. Enfin, la dernière partie aborde les travaux qui ont utilisé le son de manière synchrone comme "feed-back" des actions de l'utilisateur.

## 3.1 Le son dans l'interface

Le son possède des propriétés qui le rendent intéressant en vue de son utilisation dans une interface. Les avantages et les spécificités du son sont décrites dans la première partie de cette section. La deuxième analyse les raisons qui peuvent expliquer la quasi-absence d'interface sonore dans les

logiciels actuels.

### 3.1.1 Spécificités du son

Nous vivons entourés de sons. Même si nous n'en sommes pas toujours conscients, les sons font partie de l'environnement dans lequel nous évoluons et nous informent constamment des événements et des objets alentour. Souvent, nous remarquons que l'environnement est empli de sons quand l'un d'entre eux nous gêne ou nous agresse. Pourtant, les sons que nous entendons nous aident à nous comporter dans la vie de tous les jours, sans que nous nous plaignons de leur présence. Le son possède des spécificités qui peuvent être utilisées dans l'interface : il donne des informations sur des événements invisibles, il apporte des informations complémentaires à celles captées par la vision et notamment des informations de composition et de contexte, et enfin sa nature est telle que l'on peut en écouter plusieurs à la fois tout en les discriminant.

#### Percevoir des événements distants invisibles

Tout d'abord les sons nous permettent de percevoir des informations qui ne sont pas visibles. Ainsi, quand j'attends l'arrivée de mon amie, je n'ai pas besoin de guetter à ma fenêtre : comme je connais le bruit de sa voiture, je peux vaquer à mes occupations tout en sachant que je percevrai son arrivée. Si je ne suis pas tout à fait sûr qu'il s'agit de sa voiture, je me concentre sur les bruits venant de mon couloir, et j'écoute les sons qui me confirme progressivement qu'il s'agit bien d'elle : gonds de la porte d'entrée de l'immeuble, bruits de pas d'abord lointains puis de plus en plus proches (il semble que l'éloignement d'une personne qui marche est spécifié par la proportion de son réverbéré des bruits de pas par rapport au son direct), bruit de la poignée de porte de mon appartement qui s'ouvre. Cette caractéristique peut être employée dans l'interface pour donner des informations sur des processus en arrière-plan.

#### le son complémentaire à la vision

Parmi les cinq sens, la vue, l'ouïe et l'odorat sont ceux qui nous permettent de capter des informations à distance, sans contact avec les objets ou phénomènes d'intérêts. Si la vision permet de percevoir des caractéristiques d'objets par leur surface, les sons donnent des informations sur la composition des objets. La lumière réverbérée nous donne généralement des informations sur des surfaces illuminées, sur l'aspect extérieur des objets, leur forme, leur texture, etc. Le son donne des informations sur la composition des objets, ou sur leur mécanismes internes. Ils spécifient la forme, la densité des objets dont ils proviennent. Quand nous voyons un tronc d'arbre couché sur le sol, nous percevons que nous pouvons nous assoir dessus, car

les troncs d'arbre suffisament gros sont assez solides pour nous supporter. En revanche, si quelqu'un passant à proximité le frappe à l'aide d'un bâton, nous pouvons percevoir par le son produit que cet arbre est creux et qu'il est donc suceptible de s'effondrer si nous nous asseyons dessus. Par cet exemple on peut voir que la vision et l'ouïe sont complémentaires, et nous aident à prendre des décisions qui n'auraient pas été les mêmes si nous n'avions reçu des informations que par l'intermédiaire d'un seul des deux.

Les sons donnent des indications sur les événements qui leur donnent naissance, comme les impacts, les frottements, les ruissellements. Ils donnent également des informations sur l'environnement par la réverbération, l'écho, et le filtrage dûs aux rebonds et à la traversée du medium. On peut utiliser cet aspect du son pour décharger l'interface graphique d'informations qui l'alourdissent, comme la taille d'un fichier par exemple [35]. L'opacité des objets ne permet pas à la vue de discerner des objets disposés derrière d'autres objets. De plus, le champ de vision de l'être humain est limité au demi-plan situé devant les yeux de l'observateur. Le son contourne les obstacles, directement ou grâce à la réverbération, et passe à travers des obstacles de moyenne dimension comme les murs. Non seulement ceci nous permet d'avoir des informations sur des événements proches de nous, comme quelqu'un marchant dans un couloir attenant, mais cela nous permet aussi de capter des informations sur des événements relativement éloignés. Ainsi, lorsque nous entendons la sirène d'une ambulance au loin, nous savons que nous serons incapable de l'apercevoir, car notre environnement est encombré de constructions et qu'une ambulance roule dans des rues entourées de constructions. Les constructions nous empêchent de voir l'ambulance mais pas de l'entendre. En général, plus l'on s'éloigne d'un observateur, plus son champ de vision est limité par les obstacles. Son "champ auditif" en revanche, est beaucoup moins dépendant de la distance, car les obstacles de l'environnement sont des obstacles aux déplacements et à la vision, mais pas forcément à l'ouïe.

#### flux sonores multiples

Nous entendons des sons provenant de toutes les directions, pas seulement du cône de vision. Le système perceptif est capable de discriminer les sources sonores en les localisant. Quand elles sont proches l'une de l'autre, il est capable sous certaines conditions de les discriminer en percevant les invariants des sons qu'il connaît et en ignorant le reste : ce phénomène est parfois mentionné sous le nom d'"effet de cocktail" ("cocktail party effect"), situation durant laquelle nous sommes capables de nous concentrer sur une des conversations qui se tiennent autour de nous [2]. Cette caractéristique peut être utilisée pour fournir plusieurs flux d'information simultanés.

### 3.1.2 Rareté des interfaces auditives

Les ordinateurs produisent rarement du son fortuitement. Quand ils le font, il s'agit généralement d'un son mécanique que les ingénieurs tentent de supprimer. Ainsi, j'entends actuellement le son du ventilateur refroidissant le processeur de mon ordinateur. Ce son est une gêne car il est peu informatif. En revanche, je distingue parfaitement le son produit par le disque dur quand il se met à fonctionner, indiquant une activité dans le cas d'un processus long, ou le fait que l'état de mon application est sauvegardé automatiquement de temps en temps. Si des spectateurs voient un film durant lequel des personnages tiennent une conversation dans un endroit "visiblement bruyant" (une usine par exemple), si les bruits environnants sont peu ou pas entendus ou si les éclats de voix n'en tiennent pas compte, il y a fortement lieu de croire que les spectateurs ne se sentiront pas impliqués dans l'action. De même, le silence des ordinateurs diminue certainement l'intensité de la sensation d'"engagement direct" lors de leur manipulation par un être humain. L'effet inverse est d'ailleurs utilisé par les réalisateurs de films de science-fiction, qui n'hésitent pas à associer des bruits de réacteurs aux déplacements des vaisseaux spatiaux alors qu'il n'y a pas de sons dans l'espace. Ces effets spéciaux sont destinés à impliquer le public dans l'action, à lui proposer des scènes "réalistes" dans le sens où il a l'habitude d'entendre des bruits de réacteurs quand il voit un avion se déplacer.

Après avoir énuméré les capacités du son à apporter des informations différentes de celles apportées par la vision, il est surprenant de constater que si peu d'applications utilisent ces moyens pour interagir avec un utilisateur. Il existe plusieurs raisons qui expliquent cet état de fait. Premièrement, si historiquement les civilisations transmettent le savoir par la parole et l'écrit, "seuls les écrits restent" et sont étudiés par les générations qui les redécouvrent. C'est la peinture qui témoigne le mieux des scènes de la vie passée comme les scènes de chasse dépeintes sur les murs des grottes préhistoriques ou les vêtements de la renaissance. L'écriture, la peinture voire la sculpture se font sur des supports qui ont existé en même temps que leur invention : les murs, les parchemins, les toiles. Ces supports ont une durée de vie suffisamment grande pour ne pas nécessiter de reproduction fréquente. Le son quand à lui ne possèdait pas de tels supports jusqu'à très récemment. Les musiques étaient transmises pas leur apprentissage. L'invention de l'écriture a permis une certaine forme d'"écriture musicale" que l'on trouve sur les partitions, mais si nous savons comment les populations s'habillaient et peignaient du temps de Léonard de Vinci, nous ne savons pas comment était jouée la Flûte Enchantée de Mozart sous sa direction. L'existence de matériel qui permet la persistance de la vision ainsi que sa manipulation aisée expliquent en partie la prédominance de l'image sur le son.

Quand les civilisations commencèrent à construire des machines, des in-

formations visuelles furent utilisées pour spécifier la sémantique des boutons, leviers ou autre bielles actionnables par un opérateur. Ces informations visuelles trouvèrent leur pendant dans les interfaces graphiques, et les principes d'usage de telles informations ont appliquées à l'informatique [9]. Quand les chercheurs se sont intéressés à des moyens autres que textuels de présenter des informations, ils se sont tournés vers les interfaces visuelles à la fois parce que les sociétés étaient culturellement habituées à utiliser le canal visuel et parce que les dispositifs matériels pouvant afficher des images étaient disponibles. La souris, un des artefacts informatiques les plus communément utilisés, inventée par Doug Engelbart, est fortement couplée à l'information visuelle, car il s'agit dans les interfaces actuelles de déplacer "un curseur" (une information visuelle spécifiant une position) sur un objet lui-même graphique avant d'effectuer une action à l'aide de l'un des boutons. C'est la précision du pouvoir de discrimination du système perceptif qui permet l'utilisation d'interfaces graphiques à manipulation directe. Les travaux de chercheurs comme ceux ayant donné existence au Star de Xerox [1], ou encore ceux de Shneiderman décrivant les principes de la manipulation graphique directe [73, 45] ont entrainé à la fois la conception de nouveaux systèmes et de nouvelles recherches.

De son côté, le son n'a pas bénéficié des mêmes faveurs. Si les recherches sur la synthèse sonore, notamment les travaux de Max Mathews [54], datent de la même époque que celles sur l'infographie [9, 15], les résultats ne donnèrent lieu qu'à une production musicale, pas à des systèmes sonores pour l'interaction. N'ayant ni les mêmes buts (artistiques pour le son, fonctionnels pour l'image), ni les mêmes moyens, la recherche sur les aspects sonores n'a eu que peu d'influence sur les interfaces homme-machine. Encore maintenant, si les ordinateurs de bureau possèdent tous une carte graphique capable d'afficher des primitives graphiques avec des millions de couleurs, la présence d'un circuit de reproduction sonore reste optionnelle.

Enfin, il est souvent considéré parmi les utilisateurs que les sons ne sont que des bruits, et ne peuvent que gêner leur concentration au travail. Les sons qui sont associés aux bruits ne sont effectivement que des *bruits*, pris dans le sens du terme "rapport signal/bruit" : le signal désigne les informations, le bruit désignant l'absence d'information. Si un son ne spécifie pas d'informations, il est un bruit par rapport au signal, et donc un *bruit sonore*. Une anecdote édifiante montre les réticences des utilisateurs vis-a-vis des interfaces sonores si celles-ci sont mal présentées. SonicFinder est un système qui ajoute du son à l'interface du Macintosh [35]. Il améliore la sensation d'immersion dans la métaphore et permet de donner des informations qui ne sont pas présentes visuellement. Apple considérait sérieusement l'éventualité de son incorporation dans le système en standard. Lors d'une présentation à la presse de SonicFinder, le volume sonore a été augmenté pour les besoins de la démonstration. Les journalistes présents lors de la conférence décrièrent le système à cause du caractère intrusif dû au volume

général de l'interface, et Apple a abandonné le projet.

S'il est facile de chasser une vision gênante en fermant les paupières ou en détournant le regard, il n'en est pas de même pour le son. Il peut donc se passer un temps entre le début de la gêne et son élimination, ce qui renforce le sentiment de frustration vis-à-vis de ce médium. Cependant, des moyens existent pour réduire la gêne occasionnée par certains types de sons, comme l'emploi systématique de sons dont les fréquences hautes sont filtrées, ou dont l'attaque n'est pas abrupte.

## 3.2 Techniques

Avant de présenter les travaux de recherche visant à sonoriser les applications interactives, cette section énumère les différentes techniques sonores qui ont été employées dans les interfaces.

### 3.2.1 Les icones auditives

Les icones auditives sont les sons que l'on peut entendre dans la vie courante à l'exclusion des sons musicaux et de la parole [34]. Ils regroupent les sons de phénomènes naturels (vent, rivière, tonnerre), les cris d'animaux, les sons produits par les artefacts humains (machines, voitures, ventilateurs), ceux produits par manipulation (impact, frottement) etc.

Dans le domaine de la recherche sur le son dans l'interface, le terme d'icone auditive fait souvent référence à des enregistrements de sons. Ils sont généralement utilisés pour accompagner des événements, et la seule information donnée par ce type de sons passe par l'identification. Par exemple, entendre le son d'une sonnerie signale qu'une connexion par un utilisateur distant est tentée. Pourtant, le terme d'icone auditive fait référence originellement à des sons de tous les jours qui transmettent non seulement des informations par leur nature, mais aussi par les dimensionss de haut-niveau qui les caractérisent [35]. Elles utilisent les concepts de l'approche écologique que nous avons vus dans le chaptire précédent. Le but de l'utilisation des icones auditives est de profiter des capacités acquises par les êtres humains qui permettent de percevoir naturellement ces dimensions. Si l'interface les utilise, elle est susceptible d'être comprise plus facilement par les utilisateurs en réduisant le "fossé d'évaluation" de Norman[59].

Certains éléments d'interface sont associés naturellement à des sons de tous les jours. Ainsi, Gaver propose d'accompagner le déplacement d'une icone graphique par un son de frottement. La vitesse du déplacement contrôle la vitesse du frottement entendu. En utilisant la même analogie visuellement et auditivement, on augmente la redondance de l'interface et on aide l'apprentissage des utilisateurs ainsi que leur capacité à se souvenir du système. De plus, comme les deux modalités utilisent la même métaphore on augmente le sentiment d'engagement direct [45], ce qui permet d'oublier l'inter-

face et d'avoir le sentiment de travailler dans le monde de l'activité, et non celui de la machine. Cependant, les concepts utilisés par une interface ne possèdent pas toujours une correspondance évidente avec un son de la vie courante. Pourtant, même si le problème de la signification d'un son est posé dans ce cas, on peut toujours le contrôler par des paramètres de haut-niveau compréhensibles par l'utilisateur, de façon à ce que le contrôle soit cohérent avec le concept que l'on veut sonoriser. Ainsi, Gaver propose des sons de machine qui accompagnent une compilation, et qui indiquent la vitesse de compilation par la vitesse perçue de la machine [39].

S'il existe assez peu d'interfaces avec icones auditives autres que celles proposant des échantillons non contrôlables, c'est parce qu'il est difficile de concevoir des icones auditives paramètrables. Ce problème est abordé dans le chapitre suivant.

### 3.2.2 Traitements

L'avénement des interfaces graphiques a entrainé la conception de systèmes qui permettent le partage des ressources matérielles graphiques et proposent un modèle de présentation graphique à deux dimensions à base de fenêtres. Plusieurs extensions ont été proposées pour introduire la modalité sonore dans ces systèmes. Ainsi, Ludwig et al [50] ont implémenté un modèle qui permet entre autres de transformer les sources sonores au moyen de filtres qu'ils appellent "filtears". Différentes techniques employées par les concepteurs sonores de l'industrie du cinéma permettent de manipuler les sons pour les faire ressortir ou pour les atténuer. Ainsi, parmi les techniques qui accentuent un son citons :

- l'auto-animation, qui consiste en une distortion dépendant de la fréquence
- la distorsion
- le "grossissement", en jouant le même son décalé d'un temps très court

et parmi celles qui l'atténuent :

- la distantiation, avec la réverberation et l'écho
- l'assourdissement avec un filtre passe-bas

En manipulant des entités graphiques en trois dimensions à l'aide d'un gant localisable dans un espace à trois dimensions, les sons associés sont transformés pour refléter l'interaction effectuée. Ainsi, pointer une source résulte en une accentuation du son associée à la source. De même, saisir une source atténue le son, comme si elle était confinée dans la main. D'autres traitements ont été proposés, comme la compression pour faire ressortir les sons [3].

### 3.2.3 Les *Earcons*

Blattner, Sumikawa et Greenberg définissent les *earcons*, des motifs musicaux qui servent à donner des informations sur des objets, des opérations

ou des interactions informatiques (fichier, éditer, éditer un fichier) [10]. Selon eux, les icones graphiques sont soit des éléments simples, soit des compositions d'éléments. Les auteurs présentent trois façons de créer des icônes composées. La combinaison consiste à associer des icones. La transformation est une légère modification d'une icone, et permet de représenter des objets avec un état associé. L'héritage permet d'organiser hiérarchiquement les icones pour qu'à chaque niveau de la hiérarchie corresponde une dimension : plus on s'enfonce dans l'arbre, plus la signification d'une icone se spécialise.

Blattner et al. reprennent ces principes pour construire les *earcons*, de petites mélodies de quelques notes appelées "motifs". Elles sont aisément reconnaissables par leur rythme, leur hauteur, et leur volume, et peuvent être combinées, transformées ou héritées. Ainsi, si un mi *crescendo* (dont le volume augmente) correspond à "créer" et si deux blanches, ré et sol correspondent à "fichier" ces deux notes jouées après un mi crescendo correspondent à "créer un fichier".

L'héritage est une peu plus complexe : chaque niveau correspond à un attribut. Tout d'abord, les familles de messages sont distinguées par le rythme (message d'erreurs, fenêtre...). Le deuxième niveau est une mélodie jouée au rythme de la famille à laquelle l'objet appartient. Cette mélodie est jouée avec une sinusoïde, car c'est le timbre le plus neutre. Le troisième niveau correspond au changement de timbre et à une petite transposition. Enfin, la dynamique et le registre interviennent dans le dernier niveau. Ainsi, quand on écoute une *earcon* on entend un rythme, une mélodie, une mélodie avec un timbre différent jouée un peu plus haut, puis cette même mélodie avec une dynamique différente ou jouée dans un autre registre (Figure 3.1)

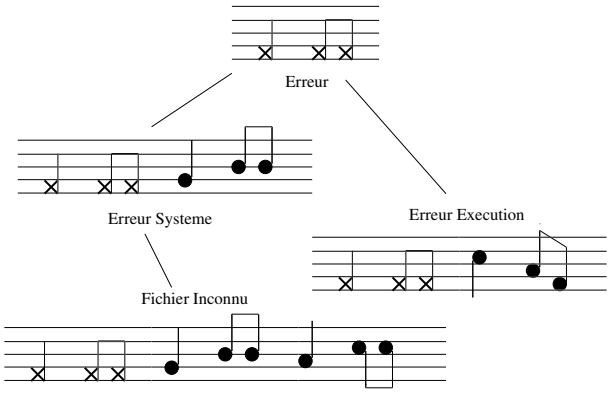


FIG. 3.1 – *Earcons* : un exemple d'héritage.

La transformation permet de " sauter " des étapes dans l'écoute d'une *earcon*. En effet, les utilisateurs experts connaissent au bout d'un certain temps le rythme caractéristique d'un message d'erreur par exemple, et ils

n'ont pas besoin d'entendre ce rythme dans l'*earcon* correspondante. On supprime donc cette étape, l'utilisateur reconnaissant la famille de l'*earcon* au rythme de la mélodie.

L'avantage de cette structure hiérarchique réside dans le fait que l'ajout d'une fonctionnalité ne nécessite pas forcément l'ajout d'une *earcon* entièrement nouvelle : ainsi, si une nouvelle erreur système doit être notifiée par une *earcon*, cette dernière héritera au moins du rythme et de la mélodie de la famille des erreurs système. L'apprentissage de nouvelles *earcons* n'est donc pas forcément difficile.

En revanche, il faut s'entraîner au moins une fois à reconnaître les types d'*earcons*, la famille à laquelle elles se rattachent, etc. Les *earcons* hiérarchiques peuvent donc être classées dans la catégorie "abstraite ", nécessitant un effort de la part de l'utilisateur. Bien que l'apprentissage de nouvelles *earcons* ne soit pas obligatoirement difficile, il se peut que leur nature abstraite rende leur reconnaissance plus difficile si leur nombre est important.

### 3.2.4 *Auralisation*

La visualisation d'informations [15] permet de représenter des données multidimensionnelles sous forme d'attributs graphiques : disposition spatiale, couleurs, formes etc. L'*auralisation* d'informations est le pendant sonore de la visualisation. Les données multidimensionnelles sont projetées sur les dimensions sonores comme le volume, la durée ou la hauteur, voire sur des dimensions musicales comme la texture sonore d'un accord. Le système perceptuel auditif est capable de suivre individuellement les données représentées, ou de reconnaitre des motifs particuliers combinant plusieurs dimensions [11].

Ces techniques de présentation permettent à des étudiants en chimie non-voyants de reconnaître des spectres infra-rouges de diverses substances [63]. Les fréquences des notes entendues correspondent aux fréquences du spectre infra-rouge, leur durée à l'intensité des pics. Les notes sont jouées dans l'ordre décroissant des fréquences pour la première mélodie, dans l'ordre décroissant des intensités pour la deuxième, et enfin toutes ensembles (formant un accord presque toujours dissonant). Ces différentes auditions permettent de retrouver les substances correspondantes. Ainsi, selon Peterson, " L'éthanol sonne comme une danse extra-terrestre. Le polystyrène part d'une fugue sombre, pour arriver à un tintement de cristal... " [63].

Certaines dimensions sonores sont adaptées à des variations des données particulières. Ainsi, la perception de la hauteur est une fonction logarithmique de la fréquence, et convient donc aux données dont la variation suit une fonction logarithmique.

### 3.2.5 Correspondances entre dimensions

La correspondance entre les dimensions du son et les dimensions de l'information à transmettre peut utiliser plusieurs stratégies. La correspondance peut être totalement arbitraire comme l'utilisation de la fréquence des sons pour une dimension quelconque, ou fortement liée à l'événement associé comme la correspondance entre vitesse de déplacement d'une icone graphique et celle perçue dans le son de frottement l'accompagnant.

Gaver énumère deux types de correspondances [35]. Le premier spécifie celles existant entre les données du système et le modèle que l'on souhaite manipuler, comme celui du bureau. Ainsi, à un certain niveau d'abstraction, un fichier est une structure avec un champ qui pointe sur une zone du disque. L'entité du modèle est le *fichier*, et les données référencées par ce modèle sont la structure de données et les octets sur le disque. Cette correspondance est dite "conceptuelle".Le deuxième type de correspondance spécifie la façon dont est représenté le modèle conceptuel, et est appelé correspondance "perceptuelle". C'est grâce aux représentations graphiques, auditives voire tactiles que l'utilisateur peut manipuler les entités du modèle sous-jacent. Cependant, le modèle perceptuel n'est pas complètement consistant. Ainsi, si la métaphore du bureau permet de manipuler des fichiers et des dossiers, observer le contenu d'un dossier se fait par l'intermédiaire d'une "fenêtre", les commandes passent par des menus, et la disquette est éjectée quand elle est posée sur la corbeille...

Gaver définit trois niveaux de correspondance perceptuelle entre sons et informations : symbolique, métaphorique et iconique. Les correspondances symboliques sont à la base de l'*auralisation* de données : les données multi-dimensionnelles sont représentées avec le volume, le timbre, la hauteur des sons. La correspondance iconique est celle qui est la plus facile à comprendre, car la relation entre le son et l'événement est causale. Ainsi, faire glisser une icone produit un son de frottement dont les caractéristiques dépendent de l'interaction et des matériaux mis en jeu. Quand il existe une correspondance immédiate entre un événement de la métaphore employée et un son, il est plus facile pour un utilisateur de reconnaître et d'interpréter le son entendu. Cependant, tous les événements du monde informatique ne produisent pas un son, ou le son correspondant n'est pas assez informatif pour qu'il soit utilisé. Ainsi, la copie de fichier ne génère pas de sons. On pourrait employer le son d'une photocopieuse, mais celui-ci ne possède pas d'information permettant de connaître l'état d'avancement de la copie par exemple. Dans le SonicFinder, un son de bouteille qui se remplit accompagne la copie de fichier [35]. Ici, l'analogie entre les deux processus que sont la copie de fichiers et le remplissage d'une bouteille est métaphorique, alors que le son est iconique.

## 3.3 Son et tâches de fond

Il existe plusieurs types d'applications qui s'exécutent sur un ordinateur. Devant un écran, un utilisateur a son attention concentrée sur une tâche de premier plan, comme un logiciel de dessin ou un traitement de texte. Il peut avoir lancé des applications qui s'exécutent sans manipulation explicite de sa part, comme une compilation ou une impression. Le comportement de ces applications intéresse l'utilisateur puisqu'il les a initiées. Enfin, il existe des processus lancés soit par le système ou considérés comme tel (un serveur web, ou un serveur de fichiers), ou par des utilisateurs distants qui exécutent des processus destinés à profiter des capacités de la machine sur laquelle ils se sont connectés. La classification de ces processus ne se fait pas en fonction d'un éventuel utilisateur "principal" de la machine, mais plutôt en fonction de l'intérêt que porte un utilisateur aux processus d'une machine donnée. L'attention est portée sur son activité principale, mais il a lancé des processus en tâche de fond dont il peut avoir besoin de connaître l'état. S'il n'est peut-être pas nécessaire de connaître à tout moment l'état des autres processus, il peut en avoir besoin, pour savoir si l'un d'entre eux utilise des ressources supérieures à un certain seuil par exemple.

Il existe plusieurs justifications du choix du canal sonore pour surveiller des activités d'arrière-plan. Premièrement, les sons peuvent se fondre dans l'arrière-plan de notre conscience, c'est-à-dire que l'on peut être conscient d'un son sans y apporter une attention explicite. En quelque sorte, nous "oublions" le son. Il revient au premier plan de notre conscience en cas de changement de dimension important, ou si l'auditeur décide d'y porter son attention. Deuxièmement, il est possible pour un utilisateur de se consacrer à une activité concurrente pendant qu'il entend le son produit par celle exécutée en tâche de fond. Enfin, une personne est capable de différencier plusieurs flux sonores parallèles ("effet de cocktail") : il est donc possible de suivre plusieurs activités parallèles en utilisant un flux sonore pour chacune d'entre elles [2].

La notification continue par le son a deux avantages. Premièrement, l'information est disponible immédiatement en se concentrant sur le son. Suivre des activités d'arrière-plan par ce moyen permet donc la prévention plutôt que la réparation : le changement dans un son peut spécifier un signe avant-coureur d'un problème qui s'annonce, et que nous pouvons donc corriger avant qu'il n'arrive. Par exemple, savoir que le chargeur de papier de l'imprimante est presque vide est plus utile qu'être notifié que le chargeur est vide. Deuxièmement, la notification continue minimise les requêtes d'informations explicites par l'utilisateur. L'utilisation des ordinateurs est généralement explicite (mode "pull") : pour obtenir des informations du monde virtuel, il faut s'asseoir devant un écran, et faire des requêtes en manipulant l'interface. En donnant constamment des informations sur les activités qui se déroulent, la notification continue donne non seulement une idée de ce qui

se passe quand l'utilisateur s'y interesse, mais elle donne aussi des informations que l'utilisateur n'a pas demandées et qu'il peut capter de façon opportuniste (mode "push"). Ainsi, entendre le taux d'activité de l'imprimante continûment conduit un utilisateur à s'inquiéter du temps d'attente s'il a un document à imprimer.

Cependant, utiliser des sons continus présente le problème évident de la gêne sonore. Le défi des concepteurs d'interface est donc de fournir des informations continues à l'aide du son sans déranger l'utilisateur. La suite de la section présente des systèmes utilisant une notification sonore continue.

### 3.3.1 Audio Aura

"Audio Aura" est un système qui permet de présenter des informations destinées à être captées de façon opportuniste à l'aide d'indices sonores [57]. Le but de Audio Aura est de donner des informations non requises de manière implicite : il s'agit de créer une interface ambiante qui enrichit le monde physique.

Les informations transmises ne sont pas précises. Les auteurs font la distinction entre une information dont les utilisateurs sont sûrs, et le type d'informations donné par Audio Aura. Les informations demandées explicitement par un utilisateur se doivent d'être perçues, et doivent donc "envahir" la périphérie. En d'autres termes, le fait que ces informations soient mises en avant n'est pas gênant puisque cela a été explicitement demandé. Ici, comme les informations ne sont pas réclamées explicitement, l'utilisateur ne requiert pas qu'elles soient précises mais qu'elles ne soient pas dérangeantes. Pour autant, ce type d'information n'est pas inutile. Nous percevons constamment des objets ou des événements qui nous aident dans notre vie de tous les jours. Ainsi, le fait de passer à proximité d'une imprimante qui fonctionne peut rappeler à un utilisateur qu'il a imprimé un document quelques heures plus tôt et qu'il n'est pas allé le chercher. Il s'agit là d'une utilisation opportuniste d'une information non demandée explicitement.

Ce type d'information peut être collecté lorsque l'on déambule dans les couloirs, pour rendre visite à quelqu'un, pour prendre un café ou aller manger. Audio Aura a été conçu pour être utilisé par un être humain à "activités humaines", c'est-à-dire quelqu'un qui se comporte dans le monde, et donc qui ne reste pas forcément dans son bureau toute la journée. Le système utilise pour cela des badges actifs [44] et des casques sans fils. Un badge actif est un badge qui identifie une personne sans que celle-ci ait besoin de le faire explicitement. Des récepteurs disposés dans l'espace de travail reçoivent des signaux émis par le bagde, et informent le système de l'endroit où la personne se trouve. Il est ainsi possible de donner des informations sonores à une personne à n'importe quel endroit dans le lieu de travail. Les informations sont transmises à l'aide d'un système sans fils et d'un casque audio.

Trois activités ont servi de base à la conception du système : indiquer la quantité de messages électroniques non-lus, savoir depuis quand une personne est partie de son bureau, et avoir une conscience du travail de groupe. Comme les informations sont transmises de façon continue, elle ne doivent pas perturber l'utilisateur et ne doivent donc pas ressembler à des alarmes. Les auteurs ont utilisé des sons réels échantillonnés qui constituent une scène écologique, celle d'un environnement de bord de mer. Ainsi, la quantité de messages contrôle le nombre de cris de mouettes, les messages de personnes ou de groupes connus sont spécifiés par le cri d'oiseaux particulers, tandis que la quantité de vagues correspond à l'activité du groupe.

Une expérience informelle a été conduite avec neuf sujets. Globalement, les utilisateurs ont trouvé les choix sonores adéquats. Ils ont remarqué que les sons restent effectivement à la périphérie, mais qu'il était parfois difficile de se remémorer la signification des sons. Malheureusement, le système était conçu de telle façon que les informations mettaient du temps à être transmises, ce qui entrainait une petite attente d'une dizaine de secondes avant de pouvoir les entendre. Cependant, cette gêne n'est pas très importante pour ce type d'application. Le reproche des utilisateurs est sans doute dû au fait que l'expérience ne visait pas à analyser comment les sujets utilisent ces informations de manière opportuniste, mais à mesurer si les informations sonores sont compréhensibles. Dans ce cadre, des défauts comme celui-ci prennent de l'importance.

### 3.3.2 ShareMon

Cohen constate que les utilisateurs délèguent à l'ordinateur un nombre croissant de tâches qui s'exécutent en tâches de fond pendant qu'ils poursuivent d'autres activités. Il propose de fournir une interface auditive permettant à des utilisateurs d'être conscients de ces activités [19]. Si l'utilisateur lance une ou plusieurs tâches, il peut les oublier et n'aura des informations que s'il s'y intéresse de temps en temps. Si les tâches de fond ont des problèmes d'exécution, il se peut que l'utilisateur ne fasse que constater les problèmes, alors qu'il aurait pu être prévenu si le système l'en avait informé. Par exemple, un utilisateur peut lancer une opération d'impression une heure avant son départ du lieu de travail, et au moment de partir s'apercevoir que le document qu'il voulait lire le soir-même n'a pas pu être imprimé parce qu'il n'y avait plus de papier dans l'imprimante.

L'activité que Cohen se propose de surveiller à l'aide de la modalité sonore est le partage de fichiers. Il est possible sur certains ordinateurs de partager des fichiers avec des utilisateurs distants. Ces derniers peuvent se connecter sur l'ordinateur hôte, lister ou copier des fichiers, et se déconnecter. Cohen conçoit un prototype d'application appelé ShareMon qui donne des notifications à propos de trois événements : la connexion d'un utilisateur distant, sa déconnexion et le temps machine dévolu aux différentes opérations

qu'exécute l'utilisateur distant sur la machine hôte, comme une copie de fichiers.

Plusieurs sortes de notifications sonores ont été étudiées. La première utilise des notifications métaphoriques : "toc-toc" sur une porte quand un utilisateur se connecte, fermeture de porte quand il se déconnecte, et des sons de marche à pied pour informer de l'activité du système. De plus, des raclements de gorge joués de temps en temps rappellent la présence d'un utilisateur quand ce dernier est connecté mais qu'il n'a aucune activité. La vitesse de la marche à pied indique la quantité de temps machine prise par l'application : plus l'activité est grande, plus la marche semble rapide. Le deuxième système de notifications utilise un bruit blanc continu filtré, plus ou moins aigu selon l'activité du système. Il a été remplacé par des bruits de vagues, l'écart entre chaque vague donnant une idée de l'activité. Enfin, un troisième système de notification utilise la synthèse de voix et propose des messages parlés.

Les réactions des utilisateurs concernent tout d'abord la qualité sonore des échantillons utilisés : ils peuvent parfois être confondus avec les sons environnants. Les utilisateurs ont trouvé trop perturbants les sons de raclements de gorge et le bruit filtré. En revanche, le bruit des vagues semble être bien accueilli par les utilisateurs, qui le trouvent soit plaisant à écouter, soit très peu perturbant.

Une première étude sans explication préalable sur la signification des sons utilisés montre comment les sons ont été interprétés. Les sujets étaient censés lire des articles en ligne pendant que ShareMon donnait des informations sur le partage de fichiers. Le "toc-toc" a été compris soit comme une connexion d'un utilisateur distant (signification correcte), soit comme une *tentative* de connexion. Seule un personne devina que le raclement de gorge notifiait de la présence d'un utilisateur inactif. Seul un sujet comprit la signification du son de marche à pied. Globalement, les sujets trouvèrent les sons trop distrayants et trop fréquents. Quelques sujets interprétèrent de façon incorrecte des suites de sons : un sujet conclut qu'un "toc-toc" suivi d'une marche à pied indiquait qu'il s'agissait d'un échec de la connexion et que l'utilisateur distant faisait les cent pas en attendant de pouvoir se connecter. Un autre prit l'association "toc-toc" plus marche à pied pour une connexion et la marche à pied seule pour une déconnexion. Il faut noter ici que ces réactions sont dues pour la plupart à une non-explication volontaire des significations des sons. Une fois leur sémantique expliquée, la plupart des sujets comprirent la signification des messages en cours d'expérience.

Dans une deuxième expérience, le son de "toc-toc" fut augmenté d'un son de gonds grinçant de la porte que l'on ouvre, pour indiquer qu'une connexion avait été établie par un utilisateur inconnu du système, dit "invité". Les utilisateurs "connus" du système voyaient leur connexion spécifiée par un bruit de clefs s'entre-choquant, et d'un bruit d'une clef s'enfonçant et déverouillant une serrure. Le son d'une chaise qui craque rappelait la présence d'utilisa-

teurs connectés, avec un grincement par utilisateur. Enfin, l'ouverture et la fermeture de fichier était accompagnées par un son de tiroir que l'on ouvre et ferme. Les sujets comprirent tous la signification du "toc-toc" suivi du grincement de gonds, mais ne reconnurent pas le bruit de la clef qui tourne dans la serrure. Ils ne furent donc pas capable d'inférer sa signification. De la même façon, le son de la chaise qui craque ne fut pas compris. Enfin, les bruits de tiroirs ne furent pas reconnus. Le bruit de la porte qui se ferme fut parfois mal interprété : comme ce bruit donnait l'impression d'une porte qui claque, il semblait parfois indiquer que quelqu'un s'en allait en colère, ou que l'hôte avait décidé de mettre tout le monde dehors.

Ces expériences montrent la difficulté de concevoir des sons à la fois discrets et informatifs. Elles montrent surtout que l'écoute de séquences de sons peut donner lieu à des interprétations totalement inattendues. Cependant, il faut rappeler que l'un des buts de l'auteur était de voir si les utilisateurs étaient capables d'associer une signification à un son sans explication préalable, objectif qui semble audacieux.

Continuant ses expérimentations avec ShareMon, Cohen utilise des sons provenant de la série de science-fiction "Star Trek" [18]. Ainsi, quand un utilisateur distant se connecte, on entend le crachotement d'une radio suivi du message parlé "Kirk to Entreprise". Quand il se déconnecte, le message devient "Kirk out". L'ouverture d'un fichier est spécifiée par le son du "transporteur" qui se met en route (le "transporteur" est un appareil imaginaire censé transporter des objets en les "dématérialisant"). La fermeture du fichier produit le son de "dématérialisation" des objets transportés. Le rappel de connexion est associé à un son de sonar, que l'on peut entendre sur le "pont" du vaisseau dans la série. La plupart des particiants qui n'étaient pas familiers avec Star Trek ne purent associer de signification aux sons entendus. Les autres comprirent la signification des messages de connexion et déconnexion. Personne ne comprit le son du transporteur se mettant en route.

Les sons de ce type sont appelés "sons de genre" (*genre sounds*) et font appel aux connaissances culturelles des utilisateurs. Ils ne proviennent pas directement d'une cause écologique au sens de Gibson, mais plutôt d'associations répétées avec des événements qui finissent par faire partie de la culture commune. Ainsi, le son du téléphone n'est pas associé à un marteau frappant rapidement une pièce de métal, mais à quelqu'un essayant d'établir une communication. L'un des problèmes de ce genre de sons est qu'il n'est pas possible de concevoir une interface universelle, étant donnée la grande variété des connaissances des personnes, même à l'intérieur d'une société ou d'un pays. Ainsi, les participants non familiers de Star Trek ne purent utiliser le nouveau ShareMon. Les utilisateurs pourraient avoir la possibilité de personnaliser l'interface auditive selon leurs connaissances, en associant des sons différents aux événements. Cependant, étant donnée la difficulté de conception des sons qui doivent à la fois être reconnus et avoir la bonne signi-

fication, les palettes de sons de genre doivent être créées par des spécialistes et dûment testées.

Enfin, Cohen testa une dernier ensemble de sons à connotation musicale [18]. Ainsi, il associa un “ding-dong” à la connexion et un “ding-dong” joué quelques tons en-dessous pour la déconnexion. Le son joué quelques tons en-dessous est un ton “résolvant” l'original : musicalement, il suggère la fermeture de mélodie, sa touche finale en quelque sorte. La fermeture musicale accompagne la fermeture de fichier, qui est l'action de fin associée à l'ouverture. Les sons proposés pour ces deux actions sont les mêmes que pour la connexion/déconnexion, joués à des hauteurs différentes. Bien que les sujets ne furent pas en mesure de donner les significations exactes des sons entendus, ils comprirent que les motifs musicaux avaient un rôle dans la signification du message. Par exemple, certains pensèrent que le motif ding-dong de déconnexion spécifiait une connexion ratée.

### 3.3.3 OutToLunch

Continuant ses explorations de méthodes de suivi d'activités d'arrière-plan, Cohen s'est attaqué à la conscience de groupe [20]. Passant de locaux à base de “cubicle” à des bureaux individuels, il remarque que les sons des touches de clavier et des roulements de souris participaient à l'évaluation inconsciente de l'activité de l'entourage. Il propose de remplacer ces sons par leurs échantillons joués par les machines faisant face aux utilisateurs. Toutes les 30 secondes, l'activité de chaque utilisateur est enregistrée à l'aide du nombre de touches tapées et de la distance parcourue par la souris. Ces informations sont envoyées à un serveur central qui calcule l'activité pour le groupe et renvoie ces informations à chaque machine personnelle. Des sons enregistrés de souris et de clavier sont joués en fonction de l'activité du groupe. Le système choisit au hasard parmi plusieurs instances d'un même son pour éviter la lassitude de la répétition.

Les sujets préfèraient avoir plus de sons de roulement, car ils rendent l'environnement sonore plus agréable à écouter. Ceci peut s'expliquer par le fait que ces sons sont riches en bruit, et que les sons de frappes et de boutons sont moins “saillants” lorsqu'ils sont joués en même temps que le son de roulement. Cependant, les sujets trouvèrent l'ambiance sonore trop distrayante au bout d'un moment. Cohen estime que la raison principale vient du fait que les sons n'étaient pas assez informatifs. Ils étaient plus forts que les sons réels avec une résolution plus faible, toutes les informations directionnelles étaient perdues, et il était impossible de dissocier sa propore activité de celle du groupe avec le système.

En revisant le système, Cohen a fait appel à un musicien professionnel, Michael Brook, qui a conçu un thème musical par personne du groupe, chacun étant composé d'une attaque douce et d'une longue atténuation. L'addition de réverbérations et d'échos à ces caractéristiques leur permettent

d'être mélangés agréablement. Dans la nouvelle version de OutToLunch, une boucle musicale de 13 secondes est jouée avec une guitare continûment (le "drone"). Quand un membre du groupe est actif, le drone disparaît progressivement pour réapparaître en cas d'inactivité du groupe. Les thèmes des utilisateurs sont joués lorsqu'ils sont actifs. La conception des thèmes est telle qu'elle engendre un fond musical plaisant à écouter, même quand plusieurs thèmes sont joués simultanément.

Les sujets trouvèrent la nouvelle interface très plaisante. De plus, ils avaient cette fois des informations différenciant les utilisateurs. Une anecdote racontée par Cohen illustre l'intérêt du son d'arrière-plan, et notamment notre capacité à en tirer des informations de manière opportuniste. Alors qu'il ne comprenait pas un point d'un article qu'il était en train de lire, il reconnut le thème de l'un de ses collègues. Sachant qu'il pouvait l'aider, il lui téléphona :

> Cette interaction sembla aussi simple que si [mon collègue] avait été dans le bureau d'à côté, et que je m'étais levé pour aller lui parler.

Cohen tire des conclusions intéressantes de ce travail. Premièrement, un son continu peut être une base qui rend l'arrivée de nouveaux sons plus plaisante. Ceci est dans doute dû à l'atténuation par les sons déjà présents de l'attaque des sons qui apparaissent. Deuxièmement, le travail d'un concepteur de sons professionnel améliore grandement la qualité de l'interface sonore, et facilite l'acceptation de l'interface par les utilisateurs. Enfin, la variété des interfaces sonores utilisées montrent qu'il existe des moyens originaux pour présenter des informations.

### 3.3.4 Arkola

Gaver, Smith et O'Shea s'intéressent à trois problèmes avec la simulation ARKola [33]. Le premier concerne l'utilisation d'icones auditives dans un environnement complexe. Le deuxième est celui de savoir en quoi l'utilisation du son peut aider la collaboration dans une application multi-utilisateurs. Le troisième porte sur la conception d'icones auditives qui doivent être entendues simultanément pour donner des informations sur plusieurs événements.

ARKola est une simulation d'usine produisant des boissons gazeuses. Elle a été développée à l'aide de SharedARK, qui est un système logiciel modélisant des entités physiques à l'aide de masses et de vélocités [40]. Les utilisateurs travaillent ensemble sur une surface de travail plus grande que la surface de l'écran et peuvent contrôler les objets avec une interface à manipulation directe. ARKola consiste en une usine contenant neuf machines connectées entre elles. Les machines sont des entités de SharedARK, et leur vitesse de travail peut être contrôlées par les utilisateurs. Les machines sont soit des machines fournissant des matières premières (graines,

eau, bouteilles, capsules), soit des machines transformant ou utilisant les matières premières. La tâche des utilisateurs consiste à régler les machines de telle sorte qu'un maximum de bouteilles soient produites. La disposition des machines est suffisamment dispersée pour que l'ensemble des machines ne tienne pas sur un écran.

Lorsque les machines marchent, chacune produit un son différent, comme des bouteilles qui s'entrechoquent, ou une flamme de brûleur pour le chauffe-eau. Les sons spécifiant les machines sont joués toutes les secondes, et ils sont entrecoupés de silence, ce qui laisse plus de chances à des sons produits simultanément de se faire entendre. Il ne s'agit donc pas de sons continus, mais de sons récurrents. Les rythmes des sons reflètent la vitesse de marche de la machine. Si les machines sont arrêtées, elles ne produisent aucun son. Pour éviter les phénomènes de masquage, les sons sont choisis de telle sorte que leurs fréquences occupent des bandes du spectre qui ne se recouvrent pas. Quand des matières premières sont perdues (bouteille, eau), un son est produit (du verre qui se brise, ou un "splachhh" pour l'eau). Ces sons sont produits lorsque les machines ne sont pas réglées harmonieusement, et que les machines fournissant les matières premières sont trop rapides. Les auteurs ont conçu des sons pour que ceux-ci produisent un son global compréhensible lorsqu'ils sont joués simultanément. Ainsi, quand tout se passe bien, l'usine produit un son "global" qui est le résultat de la somme de l'ensemble des sons des machines. De temps en temps, les machines ont un dysfonctionnement que les utilisateurs peuvent réparer.

Les expériences conduites réunissaient deux participants distants pouvant communiquer à l'aide d'une caméra, d'une télévision, d'un micro et d'un haut-parleur. Les sons aident les participants dans leur estimation de la quantité et de la qualité de production. Ainsi, les machines recevant des objets et en produisant d'autres ne produisent un son que lorqu'un objet a été effectivement produit. De plus, les participants estiment si l'usine "tourne" bien, en écoutant le son global des machines. Ils sont donc capable de passer facilement d'un son global à des sons de détails. En revanche, le fait qu'une machine s'arrêtant ne produise plus de sons n'est pas assez alarmant pour que les participants traitent le problème. Certains sons comme le verre qui se brise sont si alarmants que la reflexion laisse parfois place à la panique : les sujets arrêtent la machine en cause sans chercher à voir si le problème ne vient pas d'une machine en aval n'arrivant pas à suivre celle fournissant les bouteilles.

L'autre constatation réside dans le fait que les sons permettent aux utilisateurs de collaborer. Ainsi, ils sont capable de diagnostiquer un problème concernant une machine qui ne se trouve pas dans leur plan de travail, et donc de conseiller son collaborateur pour qu'il le résolve. De plus, l'usage des sons permet à un utilisateur de se déplacer dans l'usine tout en ayant conscience de ce qui se passe sur les machines qu'il est censé surveiller. En fait, c'est la combinaison d'une interface visuelle en premier-plan et d'un

contexte sonore partagé en arrière-plan qui permet aux utilisateurs de surveiller les processus et de collaborer efficacement. Enfin, l'environnement sonore permet aux utilisateurs de se sentir plus immergés dans leur activité.

### 3.3.5 ENO

ENO est un serveur audio conçu pour rendre aisée l'incorporation de sons non-parlés dans les applications sur système Unix [8]. Il permet de partager une ressource physique qui est le système de production sonore d'une machine entre plusieurs applications. Il présente aux applications un espace sonore 2D structuré, dans lequel les concepteurs placent des sources sonores.

L'espace sonore permet de donner l'illusion de la distance d'une source et de son azimuth par rapport au point d'écoute (l'utilisateur). En utilisant un algorithme efficace, ENO filtre les sons en fonction de la distance, simulant ainsi la perte des hautes fréquences des ondes se propageant dans l'air, et règle le volume entre les haut-parleurs gauche et droit de façon différente. ENO ne gère pas un espace en trois dimensions, car les calculs nécessaires à sa simulation sont coûteux et nécessitent l'utilisation d'un casque.

Les sources ainsi positionnées peuvent servir aux regroupements de sons à un endroit particulier de l'espace, pour signifier que les notifications proviennent de la même application par exemple. Le positionnement en profondeur permet aussi de mettre l'accent sur les messages importants en les plaçant près de l'auditeur, les messages moins importants étant situés plus loin.

Plusieurs exemples d'applications ont été conçues avec ENO. "Audio Make" est une version de l'utilitaire Unix "make" augmentée de notifications sonores. Make est une application qui permet de construire un programme en fonction des dépendances de fichiers sources et des dates de modification des fichiers sources et du programmes à générer. Les sons de machine que audio make produit sont contrôlés par les noms des commandes et de leurs arguments et évoluent durant leur exécution. Après quelques exécutions de audio make sur un même ensemble de fichiers, l'utilisateur s'accoutume au son produit et est capable de différencier deux sessions à partir de leurs sons respectifs, par exemple en cas d'erreurs de compilation.

D'autres exemples font appel à de la notification événementielle auditive. "Audio Draw" est un outil de dessin avec feedback sonore pour les opérations de sélection, de déplacement et de redimensionnement. Ainsi, un son de frottement est associé au déplacement d'un objet à la souris, la vitesse du frottement étant associée à celle du déplacement. Une version partagée de Audio Draw permet au son d'être un support à la collaboration. Enfin, "Audio Twm" est un gestionnaire de fenêtres avec notification sonore. Des sons d'impacts et de frottements sont associés aux manipulations des fenêtres. De plus, les sons sont spatialisés en fonction de la disposition des

fenêtres, ce qui rend la scène visuelle consistante avec la scène sonore.

### 3.3.6 EAR

EAR ("Environmental Audio Reminder") est un système de notification sonore qui aide des personnes partageant le même lieu de travail à se tenir informées des événements les concernant [37]. Alors que ARKola offrait un support à une collaboration synchrone dans un logiciel partagé, EAR est plutôt orienté vers la collaboration asynchrone. En effet, EAR est basé sur Khronika, un système de gestion d'événements partagés [49]. Khronika permet à des utilisateurs de s'abonner à des types d'événements, la notification se faisant non pas au moment où un événement est annoncé, mais à un moment approprié au type d'événement en question. Ainsi, une fois qu'un séminaire a été annoncé, le moment le plus opportun pour prévenir de son déroulement se trouve 10 minutes avant le début. EAR permet d'utiliser des icones auditives pour notifier les événements que Khronika génère. Ainsi, le début d'un séminaire est annoncé par des brouhahas qui augmentent progressivement en nombre de voix et en volume, comme lorsque les spectateurs arrivent et attendent la prise de parole de l'orateur. Les haut-parleurs diffusant les informations peuvent se trouver dans les bureaux ou dans les parties communes, ce qui donne le sentiment d'appartenir à une communauté.

Une large variété de sons a été utilisée. Ainsi, l'arrivée d'un mail est notifiée par le son d'une pile de papier tombant sur le sol. La connexion à un nœud du mediaspace [51] est accompagnée par le bruit de quelqu'un qui frappe à la porte, et la déconnexion par celui d'une porte qui se ferme. Les sons utilisés sont des échantillons filtrés pour les rendre agréables à écouter dans un contexte d'espace partagé. Les attaques sont douces, et les hautes fréquences ont été éliminées, ce qui leur permet de s'intégrer dans l'environnement de travail. Le système a fonctionné tel quel pendant plusieurs années, preuve de son caractère peu dérangeant.

### 3.3.7 *Earcons*

Crease a développé une *earcon* complexe destinée à apporter une retour d'information sonore à une barre de progression [24]. Une barre de progression permet de surveiller graphiquement l'évolution d'une tâche dont on connaît l'ampleur. Selon Conn, les barres de progression graphiques se doivent de respecter certaines règles [21]. Elles doivent montrer que la tâche a effectivement démarré, ce qui est déjà accompli, ce qu'il reste à faire, et le fait que la tâche est toujours en train de s'effectuer. De plus, elles doivent notifier les erreurs lors de l'accomplissement, ainsi que la complétion de la tâche. Enfin il faut donner une information sur l'ampleur estimée de la tâche. Crease conçoit une *earcon* à l'aide de plusieurs sons musicaux. Un son de guitare basse spécifie la hauteur "cible" que la hauteur d'un son d'orgue

doit atteindre. Le son d'orgue spécifie ce qui est accompli. Un son de piano joué plus ou moins fréquemment indique la progression de la tâche. Enfin, la fin de la tâche est marqué par trois accords. Toutes les informations citées par Conn ne sont pas notifiées à l'aide de sons : il manque les exceptions, ansi que la durée estimée de la tâche. Avec une évaluation formelle, Crease montre que les sujets sont capables de décrire le déroulement d'une tâche consistant à télécharger des fichiers en écoutant le son produit. Cependant, l'auteur admet qu'il est difficile de suivre plusieurs tâches concurrentes, et que suivre une tâche de longue durée avec ce genre de notification peut être ennuyeux.

## 3.4 Evénements et interfaces auditives synchrones

### 3.4.1 Sonic Finder

Finder est une application du système MacOS qui permet de représenter sous forme graphique des fichiers, des applications et des dossiers. En manipulant directement les objets graphiques à l'aide d'une souris, un utilisateur peut déplacer les fichers d'un répertoire à un autre, les copier, les détruire ou encore les sélectionner en vue d'une opération ultérieure.

SonicFinder est une extension du Finder qui associe des sons aux opérations ci-dessus [35]. Ainsi, sélectionner un fichier ou un dossier est accompagné par un bruit d'impact. Le faire glisser vers une autre position produit un son de frottement. Ouvrir une fenêtre engendre un son qui ressemble à un "woooo-sh", alors que le déplacement à l'aide d'ascenceurs produit des clics. Les attributs de haut-niveau des sons reflètent les attributs des objets du Finder mis en jeu dans l'interaction. Ainsi, quand un objet est selectionné, son type est représenté par le matériau de la source produisant le son : métal pour une application, bois pour un document etc. La taille de l'objet est spécifiée par le son, les gros fichiers ayant des fréquences fondamentales plus basses que les petits.

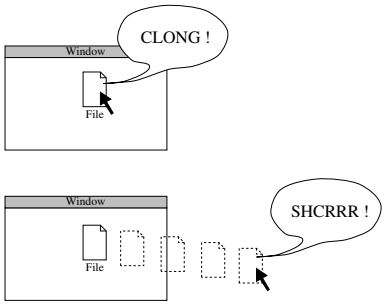


FIG. 3.2 – Sonic Finder : Sélection et déplacement d'un fichier.

Une interaction typique lors de l'utilisation du Finder est le déplacement de fichiers dans un dossier ou dans la corbeille. L'utilisateur selectionne tout d'abord le fichier qu'il va déplacer. En cliquant sur l'icone le représentant, un son d'impact est produit, dont le matériau et la taille reflètent le type et la taille du fichier. Pour une application, le type de matériau est le bois. Le type de matériau associé à un disque est métallique. Comme les disques sont des conteneurs, les sons donnent l'effet d'un objet creux, comme s'ils pouvaient être remplis par quelque chose. Plus il y a de place dans le disque, plus l'objet sonne creux. Dans le Finder original, l'information de taille n'est pas visible ; elle doit être explicitement demandée par l'utilisateur et est généralement affichée dans une boîte de dialogue. En revanche, l'interface auditive donne directement une information de taille.

Ensuite, l'utilisateur fait glisser l'icone, ce qui a pour effet de produire un son de frottements, comme lorsque l'on déplace un objet sur la surface d'un vrai bureau. Le son de frottement est lié au son d'impact dans le sens où il donne l'impression de provenir de la même source. Ainsi, la taille du fichier est spécifiée dans le son du frottement, notamment par la quantité de hautes fréquences. De plus, la surface sur laquelle l'icone glisse change l'aspect du son : au-dessus d'une fenêtre, le son de frottement est joué à une fréquence plus basse d'une octave qu'au-dessus du bureau. Cette information de contexte sert à préciser si le fait de poser l'icone résultera en une opération de copie ou de simple déplacement.

Quand l'utilisateur lâche l'icone déplacée sur l'icone d'un conteneur, le frottement s'arrête et un son d'impact provenant de la cible est produit. A la fin d'une telle interaction, il peut être difficile de trouver la cible, soit parce que celle-ci est partiellement cachée, soit parce qu'elle est de petite taille. Le fait d'amener une icone sur une autre peut conduire à un déplacement simple si le curseur n'est pas placé effectivement sur la cible. Grâce à l'apport du son qui confirme la sélection de la cible, le risque d'erreur est moins grand. Si l'icone est relachée au-dessus de la corbeille, un son de rebuts qui s'entrechoquent est produit. Lorque l'on vide la corbeille, un son de vide-ordure confirme l'effacement des fichiers.

Sonic Finder utilise plusieurs sons même pour une interaction aussi simple que l'effacement d'un fichier. Les sons correspondent au événements qui leur donnent naissance, et sont contrôlés par les dimensions effectives des objets mis en jeu. Cet exemple montre comment le système apporte des informations qui ne sont pas fournies par l'interface graphique (comme la taille des fichiers), ou qui le sont de manière moins évidente (comme le son des conteneurs).

La progression de la copie de fichiers est indiquée par une barre de progression graphique. SonicFinder complète ce retour graphique par le son d'une bouteille qui se remplit. Le son devient de plus en plus aigu au fur et à mesure de la progression de la copie. Bien qu'il semble que l'information sonore soit complètement redondante avec l'information graphique, elle ap-

porte un avantage certain : il n'est plus nécessaire que l'utilisateur ait son attention fixée sur la barre de progression pour connaître l'état de la copie. Il peut donc faire autre chose tout en surveillant l'avancement de la copie en l'écoutant.

Gaver estime qu'il est difficile de quantifier l'apport exact des retours d'informations sonores redondants avec l'information graphique. Pourtant, ils augmentent la sensation d'interagir avec les concepts du système ainsi que le sentiment d'engagement direct, et donc la satisfaction de l'utilisateur. Le fait que certains utilisateurs aient gardé SonicFinder après les évaluations informelles semble indiquer que le système leur apporte satisfaction. D'ailleurs, les utilisateurs se plaignent lorqu'ils retrouvent le Finder original :

> utiliser des Finders silencieux est comparable à porter des bouchons dans les oreilles dans la vie courante [35].

Une des limitations de SonicFinder provenait du fait que les ressources mémoires étaient limitées, par conséquent le nombre et la qualité des échantillons l'etaient aussi. De plus, il n'était pas possible de faire subir des traitments aux échantillons, comme des filtres ou de la réverbération.

### 3.4.2 *Earcons*

Breswter se propose de sonoriser des widgets graphiques avec des *earcons* afin de pallier les erreurs potentielles durant leur manipulation [13]. Il développe pour cela un méthode d'analyse des interfaces qui permet de déceler les erreurs potentielles.

Les erreurs d'interprétation peuvent avoir plusieurs causes. Il est possible que certaines informations ne soient pas disponibles, soit parce qu'elles sont cachées, soit parce que l'écran est trop petit pour que les concepteurs se permettent de les afficher. D'autres informations sont indisponibles à un moment donné. Ainsi, la taille des fichiers n'est souvent visible qu'au moyen d'une boîte de dialogue. Au contraire, il se peut qu'il y ait trop d'informations affichées sur l'écran, et qu'il soit difficile d'en percevoir une parmi les autres.

Selon Dix [29], pour réduire le nombre d'erreurs d'interprétation et de manipulation, il faut que l'interface permette l'observabilité et la prédictabilité. Si l'état de l'application est oublié ou que son changement n'a pas été perçu, alors sa manipulation peut engendrer des erreurs. Par exemple, éditer un fichier qui se trouve sur un volume protégé en écriture résulte en une erreur lorsqu'une sauvegarde est tentée. Si rien n'est prévu dans l'interface pour donner ce genre d'informations, des erreurs peuvent être commises.

Brewster étend une technique d'analyse d'interface de Dix et al. [30] en proposant une technique d'analyse par événements, états et modes. Un "événement" est un phénomène de courte durée qui arrive à un moment défini. Il peut correspondre à une entrée de l'utilisateur (souris, clavier)

comme à une action du système (arrivée d'un mail). Si un événement n'est pas perçu, des erreurs sont suceptibles d'être commises. Un événement peut passer inaperçu parce que l'utilisateur n'était pas dans de bonnes conditions pour le recevoir (absence par exemple) ou parce que le signal physique annonçant l'événement n'est pas approprié. Un "état" est un phénomène qui a une valeur persistante (comme une jauge de température). L'ensemble des états correspond à l'état d'une application. Si ces valeurs ne sont pas disponibles, l'utilisateur peut faire des erreurs. Ainsi, si un disque est plein et qu'il n'y a pas d'informations le spécifiant, l'utilisateur peut essayer d'y glisser un nouveau fichier, ce qui conduira à une erreur. Enfin, les "modes" sont des états du système pour lesquels les interactions ne sont pas toutes disponibles ou sont interprétées différemment. Par exemple, il existe des éditeurs qui ont un mode "édition" pour lequel appuyer sur la touche 'd' insère le caractère 'd', et un mode "commande" ou l'appui sur la même touche supprime un mot. Les modes peuvent être de deux types : spatial ou temporel. Les modes spatiaux correspondent à des zones différentes sur l'écran pour lesquelles les mêmes actions ont des sémantiques différentes. Ils sont moins susceptibles d'engendrer des erreurs, car les zones sont généralement visible.

En analysant les interactions, Brewster détermine les défauts de leurs implémentations par rapport à leur forme générique. Par exemple, une touche "caps-lock" permet de passer d'un mode majuscule au mode minuscule. Les débuts et fins de mode sont spécifiés par deux événements générés par l'appui sur la touche. Le mode est spécifié soit de façon mécanique (la touche reste enfoncée) parfois accompagné par un indicateur lumineux, soit de façon visuelle en regardant les caractères à l'écran. Les deux événements peuvent ne pas avoir été perçus par l'utilisateur, à cause d'un appui fortuit sur la touche par exemple. Brewster propose d'associer un son quand la touche est enfoncée. Pour spécifier le mode, il propose d'associer des bips à chaque appui de touche générant un caractère. Cette technique a l'avantage de ne pas utiliser un son continu qui peut être ennuyeux à la longue.

Une autre exemple vient d'un problème de manipulation des barres de défilement. Sur certains systèmes, la taille de l'ascenceur de la barre n'est pas proportionnelle à la taille de la zone du document affichée. Losque l'on clique dans les zones censées faire défiler le document d'une page, il est possible qu'après le clic le curseur se trouve dans la zone effectuant l'action inverse. Cliquer rapidement au même endroit peut donc donner l'impression que le document défile, alors que l'on alterne entre deux pages. Cette erreur d'appréciation est due au fait que l'utilisateur ne regarde pas l'instrument qu'il manipule, et que la vitesse de défilement ne permet pas de discriminer les pages successives. Brewster propose d'associer un son grave lorsque le document défile vers le bas et un son aigu lorsqu'il défile vers le haut. Les expériences montrent que les sujets font moins d'erreurs avec une barre de défilement auditive. De plus ils trouvent le son associé peu intrusif. Brewster propose d'autre widgets augmentées avec des *earcons*, comme des boutons,

des fenêtres, ou des boîtes à outils. La méthode présentée est une des seules méthodes qui analyse systématiquement les défauts que peut présenter une interface, et qui donne des indications sur les types de sons qui permettent de les pallier.

### 3.4.3 Feedback redondant

Les interfaces à base de métaphore de bureau comme Windows et MacOS sont généralement silencieuses. Seul un son d'avertissement est joué lorsqu'une opération interdite est tentée, ou lorsqu'une erreur est détectée. Ce son peut-être choisi dans une palette par défaut allant du simple "bip" à la goutte d'eau, en passant par le cri du canard. Cependant, MacOS 8 permet aux utilisateurs de choisir une interface avec un nombre important de manipulations graphiques accompagnées par un son. Ainsi, lorsque l'on déroule un menu, passer d'une entrée du menu à l'autre génère un son de cliquetis très bref. De même, appuyer sur un bouton produit le son d'un petit mécanisme que l'on déclenche. Sélectionner une icone produit un son d'impact, à l'instar de SonicFinder.

Non seulement ces sons sont complètement redondants avec la modalité graphique, mais en plus ils ne sont pas paramètrables. Il s'agit d'échantillons, joués à la même fréquence d'échantillonage (ni étirés, ni raccourcis) et sans filtres ni traitements. Ils ne fournissent donc pas d'informations supplémentaires telles que la taille et le type des fichiers, ce que SonicFinder peut apporter. Pourtant, plus on utilise ce système, plus il semble que les sons font partie intégrante de l'interface. A chaque manipulation, on attend un son. Le fait de ne pas l'entendre permet d'identifier une erreur de manipulation. De plus, bien que ces sons soient généralement très brefs, l'association en séquence de sons multiples génère une nouvelle dimension et aide à la manipulation de l'interface. Ainsi, dérouler un menu puis choisir la quatrième entrée dans le menu produit cinq sons : le son du déclenchement du menu, et ceux correspondant au défilement des entrées dans le menu. Le fait de ne pas entendre une séquence suffisament longue peut aider l'utilisateur à détecter qu'il a déclenché une mauvaise commande.

Ce phénomène a été utilisé par Maury dans un système qui permet de choisir une entrée dans un menu sans pointage[52]. L'utilisateur clique sur l'en-tête du menu, le menu se déroule et la sélection passe automatiquement d'une entrée à l'autre selon un rythme régulier. Lorsque la sélection s'est déplacée sur l'entrée souhaitée, l'utilisateur relâche le bouton et déclenche la commande associée. L'interêt de cette technique réside dans le fait que la tâche de pointage n'est pas à faire par l'utilisateur à l'aide d'une souris, mais par le système lui-même. Les expériences montrent que les sélections à l'aide de menus rythmiques sont plus rapides que les sélections avec la technique classique. Parallèlement au retour visuel par vidéo inverse, un son peut être émis à chaque changement d'entrée. Les sons accompagnant les changements

réguliers d'entrée diminuent les taux d'erreurs et semblent indispensables à la perception du rythme et à l'anticipation du relâchement.

L'utilisation constante de MacOS 8 pose toutefois le problème de la nuisance des sons. D'une part, les sons tels que les "tics" des menus ont suffisament d'énergie dans les hautes fréquences pour qu'à la longue on se fatigue de leur écoute. De plus, en travaillant dans un bureau partagé par plusieurs personnes, l'utilisation d'une telle interface se trouve infortuitement partagée : les co-locataires du bureau profitent malheureusement des sons joués par chacun.

### 3.4.4 Interfaces pour non-voyants

Les interfaces graphiques ont facilité l'usage des machines personnelles pour une majorité de la population. Malheureusement, les non-voyants n'ont évidemment pas profité de cette évolution. On peut même dire qu'ils ont perdu en accessibilité car les premières interfaces étaient textuelles, et donc facilement appréhendées par les non-voyants à l'aide de systèmes de synthèse vocale.

Mynatt et Edwards ont proposé un système nommé Mercator [58], qui permet de sonoriser les applications existantes qui utilisent le serveur X Window [72]. En surveillant les échanges entre les applications et le serveur, et en utilisant des programmes qui décrivent les composants graphiques utilisés par une application, Mercator est capable de construire un arbre représentant la hiérarchie des widgets. Le but du système est de faciliter la navigation dans cet arbre à l'aide d'icones auditives. Ainsi, un son est associé au passage sur un composant. Par exemple, le passage sur un bouton produit un clic, changer de fenêtre un impact sur du verre, et un champ de saisie un son de machine à écrire. A l'aide des filtres évoqués plus haut, des transformations ajoutent de la sémantique aux messages convoyés. L'atténuation d'un clic spécifie un bouton désactivé par example. Une réverbération permet de donner une information sur la taille d'un composant particulier.

Tous les composants graphiques ne sont pas sonorisés. Ainsi, certains composants n'ont qu'une signification spatiale, de disposition, et ne sont pas manipulables par un utilisateur voyant. Il est inutile de les sonoriser. Les premières ébauches employaient des sons courts, qui furent rallongés par la suite pour faciliter leur reconnaissance. Bien que les tests montrent qu'il est aisé pour des utilisateurs non-voyants de naviguer dans la hiérarchie de composants graphiques des applications existantes, les informations spatiales sont perdues. Or, ces informations permettent le regroupement, la focalisation ou le survol rapide qui, si leur impact sur la fluidité de l'interaction sont rarement mesurés, permettent de faciliter la navigation. De plus, les préceptes de la manipulation directe, qui consistent à faire manipuler des objets graphiques avec des interactions comme le tirer-relâcher ("drag'n'drop"), sont rarement implémentés dans les boîtes à outils.

Les concepteurs doivent généralement coder eux-mêmes ces interactions. Il semble donc difficile d'interpréter ces interactions uniquement en surveillant le traffic entre l'application et le système graphique, et de les sonoriser automatiquement.

### 3.4.5 Jeux et son 3D

A l'instar du cinéma, les logiciels de jeux utilisent des effets spéciaux pour immerger le joueur dans l'univers simulé. Ce n'est qu'avec l'arrivée de matériels de reproduction abordables que les concepteurs de jeux ont intégrer des sons indispensables au succès des joueurs. Ainsi, il est plus facile de jouer à un "tuer-les-tous" ("kill'm all") en réseau lorsque l'on peut situer un ennemi par les bruits de pas qu'il produit. Une anecdote illustre commment le réalisme peut aller à revers des joueurs. Une société qui commercialisaient des cartes sonores proposait un microprocesseur capable de simuler les réverbérations et les atténuations de son par les obstacles en utilisant un algorithme de lancer de rayon. [1] Les sons ainsi produits étaient si réalistes que les possesseurs de ces cartes étaient désavantagés par rapport au possesseur de cartes de moins bonne qualité. En effet, pour une même scène, les sons moins réalistes des cartes bon marché permettaient aux joueurs de situer leurs adversaires malgré les obstacles, et donc d'avoir des informations supplémentaires leur permettant de gagner.

Les jeux peuvent utiliser des matériels spécialisés qui tentent de reproduire des effets 3D. Ces matériels utilisent des filtres HRTF [67] ("head-related transfer functions") qui simulent les transformations que notre corps fait subir aux sons avant d'arriver à l'intérieur de nos oreilles. Pour obtenir l'effet souhaité l'usage de casques est indispensable, ce qui limite l'utilisation de tels systèmes dans une interface.

Il est remarquable que les joueurs, contrairement aux utilisateurs, ne ressentent pas la fatigue due à la répétition de sons échantillonés toujours semblables. Sans doute est-ce dû au fait que les bandes sonores des jeux sont conçues par des professionnels qui savent fabriquer des ambiances appropriées aux différents types de jeux. Ainsi, les jeux de stratégie (qui demandent plus de réflexion) proposent des sons plus discrets que les jeux d'actions, où la bande sonore participe à l'excitation et l'immersion du joueur.

## 3.5 Résumé

Dans ce chapitre, nous avons abordé les différentes techniques de sonorisation des interfaces homme-machines. Les icones auditives sont des sons

[1] Aureal, http ://www.aureal.com, défunte maintenant. Dans la même veine, un fabricant de cartes vidéo proposait un pilote logiciel permettant de configurer OpenGL pour que toutes les primitives graphiques deviennent transparentes, ce qui donne un avantage certain pour les jeux en réseau.

de la vie courante qui peuvent être contrôlés par des dimensions écologiques [35]. Les *earcons* sont de courtes mélodies qui suivent une structure hiérarchique permettant leur identification et leur paramètrage [10]. Les traitements permettent de transformer un son pour l'accentuer, l'atténuer, ou le réverbérer [50].

La modalité sonore est adaptée au suivi d'activités d'arrière-plan. Nous avons vu comment des systèmes comme Audio Aura [57] ou OutToLunch permettent l'utilisation opportuniste d'informations sonores. Dans des expériences comme ShareMon [19] ou ARKola [33], des sujets sont capables de suivre une activité d'arrière-plan constituée de plusieurs flux sonores simultanés. Enfin, les expériences comme ARKola ou les systèmes comme EAR [37] montrent comment le son peut aider les utilisateurs à collaborer et à avoir une conscience de groupe.

Le son peut aussi être utilisé comme "feed-back" des actions de l'utilisateur. Nous avons vu des exemples de travaux qui abordent chacun un point particulier des interfaces auditives. Ainsi, SonicFinder, en proposant une interface sonore liée de façon causale aux actions de l'utilisateur, illustre les capacités du son à améliorer le sentiment d'engagement direct, et à donner des informations qui complètent celles fournies par l'interface graphique [35]. La nécessité d'une approche plus formelle de l'apport du son dans l'interface a conduit Brewster à utiliser une technique qui permet de déceler les défauts des interfaces existantes et de pallier ces derniers avec des *Earcons* [13]. Enfin, Mercator est un exemple de système qui permet d'intégrer de façon systématique les sons dans l'interface, en sonorisant les applications existantes sans les modifier [58].

Dans les chapitres suivant nous nous appuyons sur ces travaux et sur les principes de l'approche écologique pour présenter notre approche de synthèse d'icones auditives paramétrées et un outil permettant leur intégration dans les applications interactives.

# Chapitre 4

# Synthèse sonore d'icones auditives paramétrées

L'approche que nous avons choisie pour apporter des informations à l'aide du son est celle des icones auditives. Dans la littérature, le terme "icone auditive" fait souvent référence à des échantillons de sons de la vie courante, comme un moteur de voiture, ou une sonnette de porte. Dans une interface auditive, on attend des utilisateurs qu'ils reconnaissent ces échantillons et qu'ils y associent un événement ou un processus informatique. Le seul rôle des échantillons est d'être reconnus et distingués des autres. Pourtant, les icones auditives prennent tout leur intérêt lorsqu'elles sont *paramétrées* par des attributs de la source et de l'interaction. Les icones auditives paramétrées permettent non seulement la classification de l'événement asssocié par la reconnaissance du son, mais elles permettent en plus de transmettre des informations à l'aide des paramètres de haut-niveau qui les spécifient.

Ce chapitre présente les différents moyens de concevoir des icones auditives. La première partie est une discussion sur les mérites comparés de la synthèse sonore et de la reproduction de sons enregistrés. Ensuite, les méthodes utilisées par les musiciens pour la synthèse de son sont présentées. Nous introduisons ensuite une technique de synthèse non pas évaluée en terme de qualité sonore, mais en terme de qualité écologique. Des exemples de sons que nous avons conçus sont ensuite décrits.

## 4.1 Conception d'icones auditives

Pour un concepteur d'interface, une icone auditive doit être contrôlable par ses paramètres de haut-niveau. Ainsi, il doit pouvoir contrôler la *vitesse* d'une machine, et non le nombre de cycles par seconde perçu lors de l'écoute du son de machine. En paramètrant l'icone auditive par des attributs de haut-niveau, la tâche du concepteur qui consiste à faire correspondre des données informatiques avec des données perceptibles sera plus

aisée : il n'aura pas besoin de trouver en premier lieu les paramètres du son qui spécifient la vitesse d'une machine produisant du bruit, puis de trouver les valeurs de ces paramètres qui correspondent le mieux aux valeurs des données à transmettre.

### 4.1.1 Son numérique

Comme tout signal analogique, le son peut être représenté numériquement en échantillonnant à une période régulière l'amplitude des vagues de pression en un point [54]. La fréquence à laquelle s'effectue ces mesures est la fréquence d'échantillonnage. On appelle *échantillon* une valeur instantanée de pression mesurée à un temps particulier. Par extension, on appelle aussi un son enregistré par ces moyens un échantillon. Pour lever l'ambiguïté, nous emploierons le terme d'"échantillon" ou d'"octet" pour parler de la représentation numérique d'une valeur instantanée de pression et le terme de "son échantillonné" pour un son enregistré.

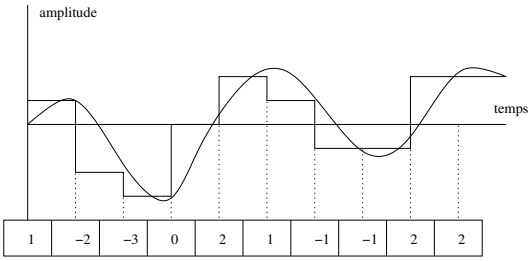


FIG. 4.1 – Audio numérique. L'amplitude de pression est échantillonnée au cours du temps

Toute fréquence inférieure à la moitié de la fréquence d'échantillonnage (fréquence de Nyquist) est captée par le processus d'acquisition. Sachant que l'être humain n'est pas sensible à des fréquences supérieures à 20 KHz, une fréquence d'échantillonnage de 40 KHz est suffisante pour représenter les sons audibles. D'ailleurs, la fréquence d'échantillonnage des disques compacts est de 44,1 KHz. Plus la fréquence d'échantillonnage est élevée, plus la quantité d'information à traiter est élevée : on préfèrera une fréquence d'échantillonnage moins élevée pour les algorithmes sonores afin de diminuer la quantité de calculs. En pratique, les sons de la vie courante ont rarement des fréquences supérieures à 15 KHz, ce qui nous permet de travailler à 32 KHz.

Le nombre de valeurs permises pour un échantillon est appelé *résolution*. Plus ce nombre est grand, meilleure sera la qualité du son. Elle est souvent exprimée en nombre de bits, celle du disque compact étant de 16 bits. Si la taille des sons augmente linéairement avec la résolution, les temps de calculs sont indépendants de la résolution. On peut donc se permettre de

travailler avec une qualité "disque compact" qui est suffisante pour nombre d'applications. En fait, les microprocesseurs d'aujourd'hui disposent souvent d'une unité de calcul en virgule flottante plus rapide que l'unité de calcul sur entiers. Les algorithmes de synthèse travaillent pour la plupart avec des nombres en virgule flottante. Le résultat est ensuite converti en entiers sur 16 bits généralement utilisée par le matériel audio.

Comme les sons sont représentés numériquement, on peut utiliser les échantillons comme entrées d'algorithmes chargés de les transformer. Par exemple, on peut appliquer un filtre passe-bas, qui consiste à générer un échantillon en faisant la somme d'un échantillon de départ et de l'échantillon qui le précède immédiatement :

$$y_n = x_n + x_{n-1}$$

Cet algorithme est dit "filtre passe-bas" car en "lissant" le son, il a tendance à atténuer les hautes fréquences du son d'entrée. Les algorithmes qui prennent des échantillons en entrée et donnent des échantillons en sortie peuvent tous être considérés comme des filtres. On parle de "synthèse" lorsqu'un algorithme ne prend pas d'échantillons en entrée, et qu'il en genère uniquement en fonction de paramètres de contrôle. Les algorithmes de transformation peuvent être combinés pour en faire de plus complexes. Ainsi, on peut simuler une réverbération avec une combinaison de filtres mis en parallèle et en séquence [55]. Plus généralement, toutes les techniques issues du traitement du signal sont applicables [27, 54].

Le problème qui se pose est celui des moyens de *production* d'icones auditives. Le mot "production" est utilisé à dessein, le terme le plus couramment usité étant celui de "reproduction", terme que je trouve trop fortement lié à celui d'"échantillon". En effet, lorsque l'on écoute un son joué par un ordinateur, il s'agit généralement d'une copie d'un son ayant été enregistré, il est donc reproduit pour l'auditeur. Le terme "production" implique que le son n'existe pas encore et qu'il va falloir le créer. En fait, il existe deux moyens de produire des icones auditives. Soit, comme évoqué plus haut, l'on dispose de sons échantillonnés produits par des événements réels, soit l'on utilise des algorithmes qui génèrent les sons à partir de paramètres de contrôle.

### 4.1.2 Sons échantillonnés

Les sons échantillonnés ont un avantage principal, si l'on ne considère que le problème de la reconnaissance, qui est celui du réalisme. En effet, il n'est pas possible de produire des sons qui ne soient plus proches de sons réels, puisque ce sont des sons réels enregistrés. Pourtant, comme on le verra plus loin dans ce chapitre, si le réalisme aide la reconnaissance, le réalisme n'est pas synonyme de reconnaissance. Le deuxième avantage des sons échantillonnés réside dans le fait qu'ils ne nécessitent pas une grande

puissance de la part de la machine qui les joue, car le processus de reproduction consiste à lire une suite de données dans la mémoire, et à les transmettre au matériel de reproduction sonore.

En revanche, les sons échantillonnés ont un défaut majeur qui vient de la difficulté de les contrôler par des paramètres de haut-niveau. En effet, une fois enregistrés, les sons ne sont plus qu'une suite d'octets non structurés, sur lesquels il est difficile d'appliquer des transformations telles qu'elles spécifient un changement dans la structure de la source ou de l'interaction. En termes écologiques, il est difficile de jouer avec les invariants de structure dans les sons échantillonnés, car les invariants qui concernent les octets ne constituent pas la totalité des invariants qui spécifient une source et une interaction. Ainsi, il est possible à partir de sons échantillonnés de manipuler les octets de façon telle que le son résultant spécifie que la source sonore se trouve dans un endroit clos. A l'aide d'une combinaison de filtres qui simulent une réverbération, les échantillons de base sont transformés en d'autres échantillons qui produisent un écho et une diffusion aléatoire des fréquences. De même, faire passer des échantillons dans un filtre passe-bas permet de simuler l'atténuation par un obstacle, comme un mur par exemple. En fait, il semble que les seules manipulations contrôlables par des attributs de haut-niveau soient des manipulations qui spécifient l'*environnement* dans lequel se trouve la source sonore, pas la source elle-même. On peut en effet contrôler la fréquence de coupure du filtre passe-bas d'atténuation par un paramètre d'"épaisseur" de l'obstacle simulé. De même, il est possible de spécifier les dimensions d'une pièce dans laquelle se trouve la source, et de contrôler de façon adéquate chacun des paramètres des filtres de la réverbération. Certains "filtears" constituent un exemple de ces transformations d'échantillons [50]. Ainsi, dans un système de réalité virtuelle, le fait de saisir un objet à l'aide d'un DataGlove a pour effet d'atténuer le son produit par cet objet, comme si la main gênait la propagation des ondes sonores [50].

Il existe une transformation de sons échantillonnés qui peut avoir pour effet de manipuler la "taille" d'une source sonore, si une telle dimension a une signification pour celle-ci. Un des paramètres des sons échantillonnés est la fréquence à laquelle ont été relevés les niveaux de pression de l'air : c'est la fréquence d'échantillonnage, exprimée en Hertz (Hz). Lors de la reproduction du son, il faut que le matériel de reproduction soit configuré pour diffuser les échantillons à la même fréquence pour obtenir le même son à l'écoute. En jouant avec cette dernière fréquence, il est possible d'étirer le son en longueur, ou de l'écourter, ce qui a pour effet respectivement de déplacer les fréquences du spectre vers le bas ou vers le haut, et donc de simuler la taille apparente de la source. En effet, dans la nature plus les sources sont grosses, plus le son qu'elles produisent contient des fréquences graves. Il s'agit d'un invariant que l'on retrouve dans la plupart des sources sonores pour lesquelles la taille a une signification. On peut de même simuler

la force ou la puissance d'une interaction en jouant avec le volume. Ainsi, le volume d'un son d'impact est d'autant plus fort que la force de l'impact est grande. Cependant, il ne s'agit pas du seul invariant spécifiant la force de l'interaction. Ainsi, un impact fort sur un objet a tendance à accentuer les hautes fréquences du son produit [39].

Il existe une solution au problème de contrôle d'icone auditive par échantillons qui consiste à utiliser une multitude de sons échantillonnés correspondant aux sons émis par une source dont les paramètres sont changés légèrement. Ainsi, pour un son d'impact, il suffit d'enregistrer successivement les sons émis par des impacts dont la force est modifiée à chaque enregistrement, du plus faible au plus fort. Le fait de jouer un son d'impact d'une force particulière consistera à choisir le son échantillonné correspondant dans la base constituée précédemment. Cependant cette méthode, si elle peut fonctionner pour un voire deux paramètres, nécessite un grand nombre de sons échantillonnés, comparable aux nombres de variations moyen par dimensions élevé à une puissance égale au nombre de dimensions ($n = v^d$), ce qui peut se révéler rédhibitoire. Pour réduire ce nombre, on peut réduire le nombre de variations par dimensions. Cependant, si une interface nécessite une représentation continue d'une dimension, l'utilisation de tels sons produira des palliers audibles. Par exemple, la vitesse de déplacement lors d'un frottement varie continûment, et ne doit pas donner l'impression de "sauter" d'une valeur à l'autre. Néanmoins, toutes les dimensions ne doivent pas forcément être continues. Ainsi, la taille d'un objet réel changeant rarement en cours d'interaction, il n'est peut-être pas nécessaire de proposer une large variété de sons échantillonnés variant selon cette dimension.

Un autre désavantage lié à l'utilisation des échantillons réside dans le fait qu'ils sont rarement en nombre suffisant pour éviter le sentiment de lassitude dû à la répétition. Ainsi, si un son quasiment instrumental d'impact sur un objet produit pratiquement le même son lorsque l'on ne change pas l'interaction ni la source, les sons de deux vagues de même taille sont différents. Comme l'un des facteurs de nuisance des interfaces sonores provient du fait que les sons ne sont pas assez variés, pouvoir produire des sons similaires mais légèrement différents est important.

### 4.1.3 Synthèse

Synthétiser un son consiste à utiliser des algorithmes qui produisent la suite d'échantillons représentant le son produit [54]. Par exemple, on peut faire appel à la fonction *sinus*, en l'évaluant à des intervalles réguliers qui dépendent de la fréquence souhaitée. Le son produit ressemblera à celui entendu lorsque l'on décroche un téléphone. Ici, l'algorithme prend une valeur en entrée, la fréquence, et produit des échantillons. D'autres algorithmes traitent, ou filtrent ces échantillons, comme celui simulant la réverbération [55].

Les sons échantillonnés ne nécessitent qu'une puissance machine limitée car il suffit de passer les échantillons précédemment enregistrés au matériel de reproduction pour les entendre. Dans le cas de la synthèse, comme il s'agit d'algorithmes, il faut les exécuter afin qu'ils produisent les échantillons à envoyer au matériel. Ce surcroît de calculs a un coût non négligeable qui dépend du type d'algorithme utilisé. Or, l'utilisation envisagée est le rendu de sons qui dépendent soit des actions de l'utilisateur, soit des événements informatiques, et qui ne sont donc pas prévisibles. De plus, il faut que ces algorithmes soient contrôlés fréquemment, afin de refléter les changements des interactions ou des événements le plus rapidement possible. Ces deux caractéristiques que sont l'imprévisibilité et la fréquence élevée de contrôle impliquent que la génération d'échantillons doit se faire en continu, ou en "temps réel". Le principe est le suivant : il s'agit de produire un nombre suffisamment petit d'échantillons pour qu'il n'y ait pas de décalage entre les changements dans l'interaction et les changement perçus dans le son, comme par exemple le son de frottement qui accompagne la souris. Ce nombre d'échantillons correspond à une durée, c'est-à-dire qu'ils représentent un morceau du son à jouer. Si le temps nécessaire au calcul des échantillons par les algorithmes est supérieur à la durée du morceau de son produit, le matériel arrête de jouer des échantillons calculés, ce qui résulte en des clics fort désagréables à entendre. Comme le temps de calcul est contraint, il faut que les algorithmes soient efficaces en temps. Certains types de synthèse très gourmands en calculs sont donc exclus d'une utilisation dans ce contexte.

Cependant, la synthèse permet de produire des sons d'une grande variété, puisque les sons produits dépendent d'algorithmes. Il suffit de changer les paramètres d'entrée des algorithmes pour produire un son différent. Cette caractérisitique a trois avantages. Premièrement, la place mémoire est constante en fonction du nombre de sons. Rappelons que le nombre de sons dépend du nombre de dimensions de l'icone auditive ainsi que du nombre de valeurs possibles pour chaque dimension. Le nombre de sons échantillonnés quant à lui est linéaire en fonction du nombre de sons. Deuxièmement, il permet une variation continue des valeurs de dimensions, ce que ne garantissaient pas les sons échantillonnés. Troisièmement, comme les sons produits le sont en temps réel, ils ne sont pas statiques, et offrent plus de possibilités pour produire des sons variés. Il sont donc susceptibles de ne pas engendrer de sentiment de monotonie.

Le principal problème de la production de sons par synthèse reste la conception des algorithmes. L'objectif principal est de produire des sons identifiables et paramètrables. La première difficulté consiste à trouver des algorithmes qui produisent un son identifiable. Différentes techniques permettent d'analyser les sons ou les sources, d'en retirer les caractérisiques et de les utiliser pour synthétiser un son. Bien que les deux problèmes cités semblent distincts, le problème du contrôle peut se résoudre en passant par le même type d'analyse et en regardant les comportements du son en fonc-

tion des évolutions des paramètres. Les sections suivantes présentent les différentes techniques de synthèses existantes. Elles ont été conçues dans un but musical, pour reproduire des sons d'instruments existant ou pour en produire de totalement nouveaux [28].

## 4.2 Modèles physiques

La première méthode qui permet de concevoir des algorithmes de synthèse est celle partant de l'analyse de la source et de l'interaction [68]. En comprenant les mécanismes internes et externes de l'événement qui produit un son, on est plus à même de les reproduire et donc d'espérer obtenir le même son. Par exemple, on peut décomposer le violon en ses parties, et étudier chacune d'entre elles. On peut ainsi analyser comment elles se comportent lorsqu'on applique une force, comment on peut les décrire en terme d'élasticité ou de texure, et comprendre comment leurs paramètres influencent les vibrations émises lors d'une interaction. De même, on peut analyser les comportements de ces parties lorsqu'elles sont réunies pour former un violon, et appliquer diverses interactions comme le frottement sur les cordes avec un archet à crin de cheval, ou le pincement de l'une des cordes.

Un fois cette analyse effectuée, il faut concevoir un modèle qui simule les comportements physiques de l'instrument étudié, et expérimenter ses réactions sous la contrainte de simulations d'interactions étudiées. Si les réactions sont semblables, les matériaux vibrent de la même façon, les cavités ont le même rôle d'amplification, et l'air virtuel entourant l'instrument entre en vibration de façon semblable à l'air réel, ce qui produit le même son.

### 4.2.1 Son de corde pincée

La synthèse par guide d'ondes est un exemple de synthèse par modèle physique [76, 77]. Elle consiste à simuler la propagation des ondes de vibration du solide excité par une interaction. Ainsi, lorsque l'on pince une corde, celle-ci a une forme triangulaire, dont les sommets sont les attaches à l'instrument et l'endroit tenu par les doigts. Lorsque l'on relâche la corde, la corde tend à rejoindre sa position d'équilibre, ce qui génère deux ondes se propageant le long de la corde. Elles rebondissent quand elles atteignent les extrémités de la corde et se propagent en sens inverse. En rebondissant, une certaine quantité d'énergie est perdue. De même, durant la propagation de l'énergie est perdue de façon différente en fonction de la fréquence.

L'algorithme utilisé pour simuler la propagation des ondes utilise une ligne à retard. La corde est découpée en petits intervalles implémentés sous la forme d'un tableau. En échantillonnant les positions de la corde correspondant aux intervalles, on initialise les valeurs du tableau quand la corde est pincée. L'algorithme consiste à partir de la case du tableau correspondant à l'endroit où est pincée la corde, et à propager la valeur de cette case

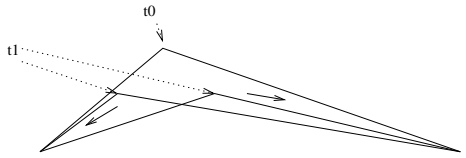


FIG. 4.2 – Synthèse par guide d'ondes. Au temps t0, la corde est pincée et relâchée. Au temps t1, deux ondes de vibration se propagent le long de la corde dans des sens opposés.

dans les cases voisines. En fait, on peut utiliser deux lignes à retard pour implémenter cet algorithme, chacune simulant la propagation d'une des deux ondes. Une ligne à retard est très efficace en temps de calcul, car elle consiste à déplacer un pointeur de case dans un tableau. En additionnant les valeurs des deux lignes à retard, on obtient la suite d'échantillons qui forment le son de corde pincée.

Cet exemple est typique des analyses de source, car les concepteurs sont partis de la mécanique de l'événement générant le son. En revanche, l'originalité réside dans la technique d'implémentation efficace qui n'est pas caractéristique de ce type de synthèse. Ceci est dû en partie à l'ingéniosité des concepteurs, mais aussi à la relative simplicité physique des événements mis en jeu. Dans les exemples qui suivent, on verra que d'autres sons naissent d'une mécanique plus complexe et donc des techniques de reproduction plus lourdes.

### 4.2.2 Masses et ressorts

Un autre moyen de modéliser les comportements physiques d'instruments est de les décrire sous forme d'un système à base de masses et de ressorts [68]. Les vibrations des corps des instruments peuvent être simulées en les considérant comme un ensemble de masses reliées à leurs voisines par des ressorts. La densité des masses, leur disposition, ainsi que l'élasticité des ressorts sont les paramètres qui définissent le comportement des corps lors qu'ils sont excités. L'excitation en un point du corps est propagée aux masses adjacentes par le déplacement des masses sous l'action d'une force. Les masses et les ressorts tendent à revenir à une position d'équilibre et génèrent par conséquent des ondulations qui sont transformées en son.

Cordis-Anima est un système qui permet de construire ces modèles et de simuler leur réaction graphiquement [14]. L'avantage de cette technique réside dans le fait que la modélisation permet non seulement la simulation graphique, mais aussi la simulation sonore et la simulation haptique par extension. Cependant, ces techniques ne s'appliquent que si l'on est capable de définir et modéliser un objet et une interaction, ce qui n'est pas toujours possible pour les événements sonores.

## 4.3 Modèles de signaux

Quand un modèle physique a été développé, sa qualité est évaluée par les jugements subjectifs d'auditeurs, mais aussi grâce à des techniques plus formelles consistant à comparer les spectres des sons simulés à ceux des sons réels. L'analyse des spectres se fait grâce à des transformées de Fourier successives sur toute la durée du son. On obtient ainsi un spectre évoluant au cours du temps. La validation du modèle se fait donc en partie grâce à l'analyse des sons virtuels et réels, et pas seulement en comparant les comportements physiques du modèle par rapport à l'instrument réel. Plutôt que de manipuler les sons échantillonnés comme évoqué dans la première partie de ce chapitre, des chercheurs ont essayé de manipuler le *spectre* de sons réels pour simuler les comportements des sons en fonction des paramètres de la source et de l'interaction.

Ainsi, Serra propose de décomposer les sons en deux parties, l'une à base de composants déterministes, et l'autre à base de composants stochastiques [75]. La partie déterministe correspond aux modes de vibration des instruments, et peut être simulée à l'aide de sinusoïdes dont la fréquence et l'amplitude sont contrôlées par des fonctions. La partie stochastique correspond à la partie du signal qui ressemble à du bruit, la partie de l'énergie qui n'a pas été transformée par le système en vibrations stationnaires. Elle peut être simulée par un bruit blanc passant dans un filtre dont le contrôle dépend du comportement du bruit résiduel. Par exemple, la partie déterministe correspond aux vibrations du corps des instruments à vent, alors que la partie stochastique correspond aux turbulences des flux d'air injectés par l'instrumentaliste. La séparation des deux composantes nécessite des règlages fins des outils d'analyse et dépend des sons étudiés. Lorqu'elle est réalisée, on peut manipuler séparément les deux composantes de façon significative musicalement. Par exemple, une transposition ne nécessite qu'une transposition des sinusoïdes. De même, augmenter la durée d'un son consiste à jouer plus longtemps les sinusoïdes et le bruit. Il est difficile d'appliquer la fonction équivalente sur des sons échantillonnés sans déformer de façon significative le son original.

Cependant, même après une telle analyse, il est difficile de contrôler l'évolution des sinusoïdes ou du bruit en fonction des paramètres physiques de la source ou de l'interaction. Bien que le bruit reste généralement du bruit après transformation, les oscillations quand elles sont transformées résultent en un signal composé d'oscillations et de bruit qui ne sont pas spécifiques du son réel voulu [74].

Ce type de synthèse a aussi été utilisé pour sonoriser des environnements de réalité virtuelle. Des sons échantillonnés non-musicaux ont été analysés à l'aide d'ondelettes [53]. Les analyses à base d'ondelettes diffèrent des analyses à base de transformée de Fourier par la résolution d'analyse qui est dépendante de la fréquence pour les ondelettes. Selon les auteurs, l'analyse

par ondelettes est meilleure pour les signaux stochastiques, que l'on trouve fréquemment dans la nature. Une fois les coefficients de décomposition par ondelette obtenus, leur manipulation permet de changer le son analysé. Par exemple, en décalant les fréquences, on peut changer la force perçue d'un son de pluie, en passant d'une pluie fine à une averse très forte. En manipulant les coefficients de façon plus ad-hoc, un son d'orage peut se transformer en un son d'applaudissements. Cependant, il semble que les auteurs n'aient pas réussi à aller au-delà de la manipulation ad-hoc. Pour autant, ils ont réussi à produire des sons pouvant être identifiés comme provenant de sources réelles, même si le son de départ n'avait rien à voir avec une telle source. Les auteurs n'indiquent pas s'ils ont réussi à contrôler des sons avec plus d'un seul paramètre. Pratiquement tous les sons cités sont contrôlables avec un seul paramètre, ce qui repésente déjà une évolution sur les sons échantillonnés.

## 4.4 Objets frappés et frottés de Gaver

Les analyses de spectre peuvent servir à déterminer les influences qu'ont les paramètres de la source et de l'interaction sur le son. Comme évoqué auparavant, en faisant varier une dimension de l'événement et en analysant le son produit, on peut se servir des informations obtenues pour synthétiser le son, en construisant un modèle sonore simplifié contrôlé par les dimensions de l'événement. La validation du modèle se fait en écoutant le résultat. Cependant, il est difficile d'extraire les invariants (car il s'agit de cela) des masses de données récoltées après analyse sans avoir une connaissance minime des phénomènes mécaniques en jeu. Comprendre la mécanique des événements permet non seulement de déterminer les dimensions des événements, mais aussi de déterminer les informations acoustiques qui spécifient ces dimensions.

Gaver a conduit des expériences pour étudier les informations acoustiques qui spécifient la longueur et le matériau de barres frappées à l'aide d'un marteau [34, 38]. En enregistrant le son de barres de diverses longueur et de matériau allant du bois au métal, il a construit un modèle de la mécanique des impacts. Celui-ci a été construit à l'aide de mesures empiriques des amplitudes de résonnance et d'amortissement. Ainsi, certains matériaux ont des fonctions d'amortissement qui dépendent de la fréquence. Le bois produit des sons qui s'atténuent très rapidement, les basses fréquences durant plus longtemps que les hautes. Le métal produit des sons qui durent longtemps, les basses fréquences disparaissant plus vite que les hautes. De plus, les spectres des sons de métal ont des pics de fréquences étroits, alors que ceux du bois occupent une plus large bande. La longueur a une influence sur les hauteurs des fréquences entendues, les barres les plus courtes ayant les plus hautes fréquences.

Gaver propose un algorithme qui consiste à additionner les sons de plu-

sieurs sinusoïdes, dont chacune dispose d'une fonction d'amplitude et de fréquence [38, 36] :

$$G(t) = \Sigma_n \Theta_n e^{-\delta_n t} cos(\omega_n t)$$

$\Theta_n$ est l'amplitude initiale et dépend principalement de la force de l'impact. $e^{-\delta_n t}$ est la fonction d'atténutation, qui dépend du matériau. $cos(\omega_n t)$ synthétise la sinusoïde dont la fréquence est $\omega_n$ et est inversement proportionnelle à la taille de l'objet. La forme de l'objet détermine la disposition dans le spectre. Ainsi, si la fréquence de base est $w_1$, les fréquences d'une corde sont des harmoniques de cette fréquence : $w_n = nw_1$. Si l'objet est une barre, elles suivent l'équation $w_n = (2n+1)^2/9$. Un algorithme similaire décrit par Van Den Doel et Pai permet de spécifier l'endroit où un objet a été frappé [31].

Avec ces sons d'impact, il est possible de fabriquer des sons de plus haut-niveau, comme un rebond ou le bris d'un objet, en reprenant les techniques de Warren et Verbrugge [82]. En produisant des sons d'impacts successifs avec des intervalles de plus en plus courts, et des forces de moins en moins grandes car la hauteur de chaque rebond diminue avec le temps, on obtient un son qui contient suffisamment d'invariants pour qu'il soit identifié comme un rebond. Un objet qui se brise est un impact suivit de plusieurs impacts simultanés avec des objets de plus petite taille.

En fait, les pics de résonnance que l'on trouve dans ces sons correspondent aux modes de vibrations des objets [28]. Les dispositions des fréquences dépendent de la forme et de la taille de l'objet. Il est possible de modéliser un objet en utilisant une banque de filtres passe-bande, dont les centres de fréquence sont disposés de la même façon que celle citée dans le précédent paragraphe. De plus, la bande passante de chaque pic dépend du matériau utilisé, ce qui n'était pas simulable avec une banque d'oscillateurs sinusoïdaux. Quand les objets sont frappés, l'impact est caractérisé par une impulsion d'énergie, qui est ensuite passé au travers de la banque de filtres (figure 4.3). Quand l'impulsion est arrêtée, l'énergie est absorbée naturellement par chacun des filtres et simule l'atténutation subie par le signal. Si l'on fait passer un bruit filtré dans la banque de filtre, on obtient un son de frottement.

La séparation entre un modèle qui décrit la source et un modèle qui décrit l'interaction a été utilisée par les musiciens pour simuler la voix chantée [28] : l'interaction consiste en un flux d'air qui est filtré par les cordes vocales et la bouche du chanteur. De la même façon l'objet frappé est modèlisé avec une banque de filtres, et différentes interactions sont simulées par différentes entrées filtrées par la banque. Les interactions sont contrôlées elles aussi par des attributs de haut-niveau. L'amplitude de l'impulsion spécifie la force de l'impact, la largeur de bande du filtre du frottement correspond à la surface, alors que son centre spécifie la vitesse du frottement. En séparant modèle et interaction, on rend aisée la conception d'interfaces comme SonicFinder en

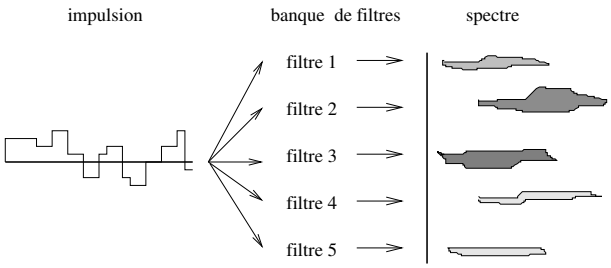


FIG. 4.3 – Sons d'objets de Gaver [36]. L'interaction est modélisée par l'impulsion et la source par la banque de filtres qui accentuent certaines fréquences et en atténuent d'autres.

associant un objet sonore particulier à un objet graphique et en associant des interactions sonores à des interactions graphiques comme la sélection et le déplacement.

## 4.5 Synthèse ad-hoc

L'objet de ce chapitre est d'introduire une technique de synthèse qui a pour but de produire des imitations de *sons*, plus que des imitations de *modèle*. En premier lieu, les motivations qui nous ont conduit à utiliser ce type de synthèse sont présentées. Nous exposons ensuite les principes de la synthèse, c'est-à-dire les techniques qui permettent de concevoir des imitations de sons de tous les jours et leur utilisation. Ensuite, nous exposons des exemples de sons qui ont été conçus avec ce principe. Nous commençons par analyser un son de machine conçu par Gaver selon notre approche [36], puis nous présentons les sons que nous avons conçus.

### 4.5.1 Motivations

Dans les sections précédentes, nous avons vu qu'il était possible de concevoir des algorithmes de synthèse à partir de l'analyse des sources et des interactions qui donnent naissance au son. Cependant, ces techniques ne peuvent être mises en œuvre qui si l'on peut repoduire ces interactions dans des conditions de type "laboratoire", c'est-à-dire qui permettent une analyse systématique en faisant varier les sources et les interactions selon les dimensions d'intérêt. Ainsi, il a été possible d'analyser des impacts en faisant varier la taille, la forme ou le matériau d'un objet. Bien que les sons qui ont pu être synthétisés à partir de cette analyse ne soient pas "musicaux", dans le sens où ils sont peu utilisés par des musiciens (sauf peut-être par les auteurs de musique concrète), ils sont de "type" musical. En effet, comme pour les instruments musicaux, ils peuvent être décomposés en terme

de source et d'interaction, et peuvent donc être étudiés relativement facilement. Cela ne veut pas dire que les phénomènes mécaniques sont simples à comprendre. De plus, les algorithmes obtenus sont contrôlables non pas en termes de paramètres musicaux, mais en termes de paramètres écologiques. Cependant, tout comme les instruments de musique, ils sont faciles à étudier matériellement, car on peut faire varier systématiquement les dimensions de la source et de l'interaction et les étudier dans des conditions de laboratoire.

Malheureusement, les phénomènes sonores non-musicaux ne présentent pas tous ces caractéristiques. Certains phénomènes naturels sont difficilement reproductibles et contrôlables dans des conditions de laboratoire. Par exemple, le son du tonnerre ne peut être entendu et étudié que dans un environnement naturel. Même si l'on peut essayer de comprendre comment il est produit, il est difficile de l'étudier précisément et de le faire varier selon une dimension de force par exemple. Le craquement du tonnerre est dû à la dilatation très rapide de l'air chauffé par un éclair. Cependant, le son du tonnerre écologique, c'est-à-dire tel qu'il est entendu par les animaux, est surtout spécifié par un craquement suivi d'une réverbération caractéristique des structures montagneuses environnantes. En se basant sur ces constatations, il est possible de créer un son fortement réverbéré qui ressemble à celui du tonnerre. Si l'on part effectivement d'une description physique de l'événement tel qu'il est perçu, la conception du son est plutôt *ad-hoc*, puisque l'on ne part pas d'analyse systématique de sons enregistrés.

Même si dans le processus de conception on souhaite analyser une banque de sons enregistrés, se pose le problème du choix des sons à analyser. Par exemple, si l'on souhaite synthétiser un son de vent, il faut savoir le type de son de vent que l'on veut imiter. Or, il existe plusieurs sons qui viennent du phénomène aérodynamique "vent". Ainsi, le bruissement des feuilles agité par l'air peut être catégorisé comme un son de vent. De même, les hurlements de l'air spécifient une tempête dans un environnement polaire, et peuvent être classifiés comme "vent". En fait, analyser un son réel peut conduire à un algorithme qui synthétise à un son "réaliste", mais pas forcément à un son reconnaissable facilement par un auditeur. Ainsi, si l'on demande à une personne se trouvant sous un arbre de dire ce qu'elle entend, elle peut répondre qu'elle entend le "vent" parce qu'elle entend le bruissement des feuilles et qu'elle sent la brise sur son corps. En revanche, le même bruit présenté dans un environnement clos ou par un ordinateur dans un bureau ne conduira pas forcément à une identification aussi facile. Cohen par exemple a enregistré des sons de roulement de souris. Bien que les sons soient réalistes, ils n'ont pas été reconnus par les utilisateurs [20].

Certains phénomènes sont difficiles à simuler même en ayant compris leur mécanique. Ainsi, si l'on peut expliquer le bruit des vagues en termes d'impacts de goutte d'eau, de ruisselement, de bulles d'écume qui éclatent, d'impacts sur une plage, et de réverbération dus aux rouleaux cylindriques qui se font et se défont, cela ne nous avance guère sur la manière de simuler

ce qui se passe. Certes, il est certainement possible de mélanger des milliers de sons de gouttes, de les réverbérer avec la simulation d'un lieu clos à dimensions variables, et de simuler les roulements des galets ou des grains de sable, mais les fonctions de contrôle seraient si nombreuses et les calculs si gourmands qu'il est difficile d'imaginer pouvoir les utiliser en temps réel dans une interface.

Enfin, tous les événements sonores ne fournissent pas d'informations à l'aide de vibrations de pression. Des invariants de plus haut niveau suffisent souvent à fournir des informations intéressantes, et ne nécessitent pas des sons de grande qualité. Ainsi, le bruit des pas peut être simulé par une succession d'impacts. Des informations comme la vitesse de la course ou sa régularité ne sont pas fournies par les sons en eux-mêmes, mais par les caractéristiques de la séquence de sons. Si l'on dispose d'un son d'impact suffisamment bien conçu pour qu'une fois inclus dans un son de marche il permette de reconnaître le son de marche, alors on pourra fournir des informations de haut-niveau par des invariants qui ne concernent pas ce son en lui-même. Ce type de phénomènes a été utilisé par Warren et Verbrugge pour synthétiser des sons de rebonds et de bris d'objets en utilisant un même son d'impact [82].

En fait, la conception d'icones auditives ne doit pas se faire avec pour but des sons réalistes. Les icones auditives doivent être avant tout identifiables avec des dimensions facilement évaluables. Le fait que le spectre des sons synthétisés ne soit pas exactement le même que celui des sons réels n'est pas un problème si les sons sont utilisés dans une interface. En revanche, il faut que les invariants soient suffisamment représentés pour que les sons soient utilisables en tant qu'icones auditives, et qu'ils profitent des capacités de notre système perceptif habitué à les reconnaître et à les interpréter. La question principale devient alors : quels sont les invariants nécessaires qui permettent de reconnaître un son, et quels sont ceux qui spécifient une dimension ? Se poser cette question, c'est déjà admettre que certains invariants sont "moins" utiles que d'autres dans la perception du son. Si l'on prend l'exemple des sons d'impact de Gaver, on constate que les deux techniques de synthèses utilisées différent entre autres par la présence d'un invariant de plus dans la technique à base de filtre [38]. En effet, avec un objet en bois, Gaver a constaté que les pics de fréquences étaient moins définis que ceux associés à un objet en métal. Pourtant, la première technique de synthèse à base d'oscillateurs, qui ne prenait pas cette dimension accoustique en compte, produisait néanmoins des sons suffisamment spécifiques pour qu'un auditeur détermine le matériau de l'objet. Ceci est dû au fait qu'un autre invariant spécifie le matériau, celui qui stipule que les fréquences hautes disparaissent plus vite que les basses pour un son de bois, alors que c'est l'inverse pour un son métallique. Même si le spectre du son synthétisé n'est pas le même que celui du son réel, il contient suffisamment d'invariants pour que le son soit correctement perçu.

De plus, il faut garder à l'esprit que les icones auditives sont destinées à être utilisées dans une interface. Or, parmi toutes les informations disponibles à un moment donné pour un utilisateur, que ce soit par le biais d'icones auditives, mais aussi par le graphisme, provenant de plusieurs processus ou même les informations de voisinage, il y a fort à parier que des invariants subtils seront difficilement perçus. Bien sûr, ces invariants ne sont pas inutiles, car il peut exister des situations pour lesquelles les invariants seront perçus, et même quelque fois sans prise de conscience de la part de l'utilisateur. Ainsi, on peut imaginer que dans une interface graphique à base de fenêtres, on associe un son d'impact quand on clique dans une fenêtre (à la "atwm" dans ENO [8]). Le son d'impact sonne d'autant plus gros que la fenêtre est grande. Rien n'empêche un concepteur d'apporter plus d'invariants dans le son pour spécifier la forme de la fenêtre, notamment son allongement. Nous avons vu dans le premier chapitre que des auditeurs sont capable de percevoir une telle information. Cependant, cette expérience est de type "laboratoire", c'est-à-dire que l'attention de l'auditeur est focalisée sur le son, et que sa tâche consiste justement à déterminer les dimensions de l'objet frappé. Dans une interface homme-machine, l'activité de l'être humain est toute autre, et elle se déroule en parallèle avec d'autres activités dont l'utilisateur perçoit les événements. Gageons que des informations subtiles telles que celles spécifiant les dimensions de la fenêtre ne sont pas fondamentales. En revanche, il se peut que ces informations soient effectivement utilisées, parce que la fenêtre dans laquelle l'utilisateur a laché une icone ne sonne pas comme la fenêtre qu'il avait repérée visuellement quelques secondes auparavant devrait sonner. Comme Gaver le remarque, de tels processus de perception sont difficiles à évaluer [35]. Il semble difficile de concevoir une expérience destinée à mesurer l'apport d'une telle information qui pourrait pourtant fluidifier l'interaction comme le font les sons du SonicFinder. La technique de synthèse ad-hoc peut ne pas reproduire ces invariants sans que cela soit gênant en vue d'une utilisation dans une interface, comme l'exemple des impacts de Gaver le montre [38].

### 4.5.2 Principes

En résumé, bien que les invariants subtils puissent être utiles dans l'interaction, il suffit de synthétiser les invariants les plus importants pour que le son résultant soit utilisable en tant qu'icone auditive. Quels sont les invariants les plus importants pour un son donné ? Pour le graphisme, Gibson a proposé que l'acte de dessiner n'est pas de copier l'"image rétinale", mais que c'est un processus de capture des invariants [43](p279). Dessiner c'est :

> marquer la surface [de dessin] de manière à montrer les invariants
> et à enregistrer une conscience [de ces invariants].

En fait, pour tout enfant ou tout piètre dessinateur comme l'auteur de ces lignes, dessiner n'est pas recopier l'image rétinale, mais tracer ce qui spécifie le concept à dessiner, en d'autres termes ses invariants. Faire un croquis, un dessin sommaire, c'est dessiner les invariants qui permettent l'identification et l'évaluation selon ses dimensions pertinentes. Faire une caricature humoristique, c'est accentuer les traits qui caractèrisent le mieux un personnage, c'est accentuer les invariants qui permettent de le reconnaître à coup sûr. Les dessinateurs de bandes dessinées ou de dessins animés utilisent les caricatures, soit à but humoristique, soit pour accentuer un trait particulier d'un personnage. La méchante sorcière se voit affublée d'une vilaine verrue, alors que Blanche-Neige a la peau pâle des jeunes filles innocentes et pures. Dans les interfaces graphiques, les icones les plus utilisées parce que les plus reconnaissables sont des dessins schématiques ou des croquis qui ne sont pas réalistes. Elles capturent les principales caractéristiques du concept dont elles sont la représentation visuelle. Ainsi, un document dans le système MacOS est un rectangle blanc orné d'un triangle, imitant ainsi une feuille cornée. D'ailleurs, l'emploi de photographies d'objets réels s'ils existent pour représenter les objets de la métaphore est rare. Si un carré gris surmonté d'un onglet spécifie un dossier, sa photographie n'apportera rien de plus, et parfois ne sera pas reconnue.

S'il est pertinent d'utiliser des caricatures pour des icones graphiques, les mêmes raisons nous conduisent à penser que l'emploi de caricatures sonores pour les icones auditives l'est également. Employer des caricatures sonores peut aider l'identification, alors que l'accentuation des traits caractéristiques peut aider l'évaluation des dimensions d'interêt. Même si les sons n'ont pas le même spectre que ceux de sons réels, ils seront de bons candidats à une utilisation dans l'interface, car ils transmettront le même type d'informations par des dimensions similaires. Dans une expérience portant sur la reconnaissance de cris d'animaux, Laas et al. ont d'ailleurs trouvé que les sons imités par des humains étaient mieux reconnus que les cris d'animaux réels [48]. La question qui se pose maintenant est de savoir comment trouver les invariants les plus caractéristiques des sons que l'on veut imiter. Dans le domaine graphique, on pourrait demander à un panel de personnes de dessiner grossièrement un objet, et de prendre les traits (dans le sens anglo-saxon du mot) communs à tous les dessins et en faire une icone. Le fait de prendre les traits communs implique que les icones dessinées ainsi ont de plus fortes chances d'être reconnues par le plus grand nombre d'utilisateurs. Ballas a conduit des expériences dont le but était de déterminer les facteurs influençant les auditeurs lors de tâches d'identification de sons de courte durée de la vie courante [4]. L'un d'elles consistait à déterminer si les sujets s'attendaient à une instance particulière d'un son qu'on allait leur présenter, ou en d'autres termes s'il existe des sons plus typiques que d'autres pour un événement écologique particulier. En particulier, les expérimentateurs ont demandé aux sujets d'imiter des sons afin de comparer les invariants

qu'ils auraient dû reproduire même grossièrement, comme ils l'auraient fait pour des croquis. Certains sons, comme les sons de signaux (sonnette de porte, téléphone) furent imités de la même façon par les sujets. D'autres furent séparés en deux catégories, comme les klaxons de voiture ressemblant parfois à un "bip" et parfois à un "ôôônnn". Certains sons qui paraissent pourtant être familiers comme ceux d'une goutte d'eau ne furent pas imités de la même façon. Les résultats mitigés de cette expérience sont peut-être dûs au faible nombre de sons proposés et au fait qu'il n'est pas facile voire gênant d'imiter des sons. Cependant, le fait que les participants ne donnent pas tous le même son typique pour un événement particulier ne veut pas dire qu'il n'existe pas de son typique pour un tel événement. L'exemple de la goutte d'eau semble paradoxal car les sons imités par les sujets étaient d'une grande variété. Mais un son particulier de goutte d'eau est peut-être identifiable par une majorité de la population.

Les techniques de synthèse basées sur l'analyse de la source conviennent parfaitement aux sons de type "instrumentaux". La technique de synthèse que nous proposons a pour but de produire des imitations de sons, plus que des imitations de modèle. Elle se base sur des techniques efficaces de production et de transformation de sons qui permettent de concevoir des sons de façon "ad-hoc", c'est-à-dire qui n'utilisent pas un modèle physique ou formel de la source sonore. Ici, le but est d'arriver à imiter un son, à reproduire ses invariants, sans que l'algorithme ne reproduise nécessairement les processus mécaniques impliqués dans la production du son. Ces techniques sont comparables à celles des musiciens électroniques qui souhaitaient utiliser des algorithmes efficaces permettant d'imiter un instrument réel. Elles ont été développées jusqu'à ce que les machines aient une puissance suffisante pour pouvoir supporter les calculs plus complexes à base de modèles physiques. En fait, ces techniques ont été peu à peu abandonnées, car le but des musiciens était de produire des sons dont le spectre était *exactement* celui d'un son réel. Bien qu'efficaces et produisant des sons pouvant être identifiés, ces techniques ne produisaient pas de sons de qualité musicale suffisante. Contrairement aux musiciens, nous ne sommes pas intéressés par des sons qui ont le *spectre* des sons d'événements réels, mais par des sons qui ont les *invariants* des sons d'événements réels.

Si les musiciens tentent de reproduire les comportements des instruments, certaines découvertes fortuites ont été employées avec succès avant d'être formalisées. Ainsi, Karplus et Strong ont développé un algorithme produisant un son de corde pincée qui consiste à injecter un bruit blanc dans une ligne à retard et à filtrer cette ligne avec un filtre passe-bas [46]. Bien que cette technique soit complètement ad-hoc, elle a été redécouverte sous la forme des guides d'onde vus plus haut dans ce chapitre. En fait, le processus de conception des guides d'onde est plus formel, car il part d'une explication à base d'équation de propagation d'onde. L'algorithme qui résulte est pratiquement le même que celui de Karplus et Strong, ce qui permet d'expliquer

les surprenants bons résultats de ce dernier. Cet exemple montre que si ces techniques ne sont pas formelles par définition, le "design" [1] a une grande importance quant à la qualité des résultats obtenus. De même, si Gaver a utilisé une analyse des sources sonores pour les sons d'impacts et de frottements, la modèlisation a plutôt suivi une conception ad-hoc. Il s'est en effet aperçu que la manipulation de la largeur de bande du filtre servant à imiter l'interaction donne une sensation de vitesse de frottement. Ce n'est pas en analysant l'interaction qu'il a découvert cette correspondance, mais en manipulant les paramètres et en écoutant les résultats.

### 4.5.3 Mise en œuvre

Créer des sons de manière ad-hoc consiste à écouter un son de tous les jours, en relever les dimensions écologiques et les invariants qui permettent l'identification et l'évaluation des dimensions, et à concevoir un algorithme qui permet de les simuler de façon simple et efficace.

Un tel algorithme est composé de deux parties [54]. Tout d'abord, il s'agit de choisir des générateurs de sons parmi la panoplie dont nous disposons. Par exemple, on peut utiliser un bruit blanc pour les sons à base de souffles ou de frottements. De même, on peut utiliser la synthèse FM [17] pour les sons avec des partiels définis. Enfin, de simples sinusoïdes dûment contrôlées par des enveloppes permettent de simuler une variété de sons simplifiés, comme des cliquetis par exemple. On peut aussi utiliser des oscillateurs plus complexe, voire des sons échantillonnés.

Ensuite, on peut transformer le son produit par les générateurs pour le modeler et le faire ressembler au son recherché. Pour cela, on peut utiliser des filtres de différentes variétés comme les filtres passe-bas ou les filtres passe-bande, en jouant avec les paramètres qui les contrôlent comme la fréquence de coupure ou la largeur de bande. On dispose aussi d'algorithmes comme le "sample-and-hold", qui permet de rendre le son plus bruité. Enfin, des effets peuvent être construits à l'aide de lignes à retard et de filtres spécifiques comme les filtres en peigne. Une fois les sons de base définis, on peut les arranger temporellement pour créer un son de plus haut-niveau. Ils peuvent être joués en séquence en parallèle, ou suivre une disposition temporelle comme celle simulant le rebond.

La suite de ce chapitre est consacrée à l'étude d'algorithmes de synthèse ad-hoc en vue d'une utilisation des sons en tant qu'icones auditives. Les figures associées représentent les réseaux de modules (spécifiés par des boîtes) qui permettent de produire le son décrit. Les connexions entre sorties (cercles sous les boîtes) et entrées (cercles au-dessus des boîtes) d'algorithmes sont

[1] à prendre dans le sens de "conception faisant appel à l'habileté du concepteur plus qu'à des techniques formelles", ou dans la définition du dictionnaire Hachette en ligne : " Mode de création industrielle qui vise à concevoir des objets dont la forme soit à la fois fonctionnelle et esthétique" (http ://www.francophonie.hachette-livre.fr/).

spécifiées par des segments droites. Le détail du formalisme graphique est donné au chapitre suivant, section 5.2.1.

### 4.5.4 Les machines de Gaver

Gaver a conçu un algorithme qui permet de synthétiser un son de "machine". La conception a suivi une démarche ad-hoc [36]. En effet, analyser tous les phénomènes mécaniques impliqués dans la génération du son d'une machine (moteur de voiture, tour...) et les resynthétiser semble difficile. Aussi, Gaver part d'un ensemble de dimensions et de caractéristiques qu'il estime pouvoir trouver dans un son de machine : la taille de la machine, qui influence sur les fréquences du son produit ; le fait que généralement une machine est composée de parties rotatives, ce qui donne un aspect cyclique au son ; et enfin la quantité de travail qu'est en train d'effectuer la machine.

L'algorithme consiste à utiliser une technique de synthèse appelée modulation de fréquence (frequency modulation synthesis, ou FM) [17]. Il consiste à jouer un son dont la fréquence (appellée "porteuse") et à la moduler par une autre fréquence (la "modulation"). Son intérêt réside dans le fait que lorsque la fréquence de modulation est suffisamment grande par rapport à la fréquence de la porteuse, il apparaît d'autres fréquences centrées autour de la porteuse. L'algorithme est peu coûteux en terme de calculs et donne à peu de frais un son résultant avec un spectre très riche, et permet à un concepteur de travailler avec un "matériau" qu'il peut "sculpter". Le nombre de composants apparaissant autour de la porteuse dépend de l'amplitude de la modulation. Plus le nombre de composants est grand, plus le son est riche et complexe. Dans le cas d'un son de machine, la richesse du son donne l'impression d'une plus grande activité de sa part. Ainsi, en faisant correspondre la porteuse à la taille de la machine, l'amplitude de la modulation au travail effectué, et en modulant cette amplitude par une autre sinusoïde pour donner une impression de cycle et de vitesse, il est possible de synthétiser un son identifiable et dont les dimensions sont contrôlables et évaluables.

Le son résultant est un bruit cyclique à une certaine hauteur dont les pulsations correspondent à la vitesse de la machine virtuelle. Quand le travail est faible, le son est plus discret, et quand il augmente il devient plus riche, plus "gros". Evidemment, le son produit ne correspond à aucune machine existante. Pourtant il est reconnu comme pouvant provenir d'une machine car il possède la même signature cyclique caractéristique de ces artefacts. De même, il se comporte et se décrit selon les mêmes dimensions, ce qui permet l'identification et l'utilisation dans une interface. En reprenant les invariants d'un son de machine sans forcément reprendre le son d'une machine existante, on peut utiliser les capacités des êtres humains à les reconnaître et à les évaluer. De plus, l'utilisation d'un algorithme efficace permet l'utilisation de ce son comme icone auditive.

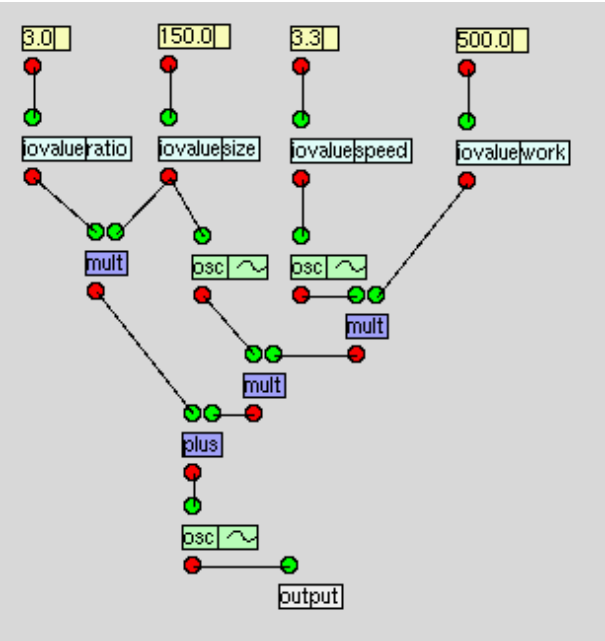


FIG. 4.4 – Le son de machine de Gaver [36]. Il est synthétisé avec un algorithme de moduilation de fréquence, dont l'amplitude de modulation est elle-même modulée par une sinusoïde.

### 4.5.5 Le vent

L'un des buts de cette thèse était de concevoir des sons pouvant être utilisés pour surveiller des activités d'arrière-plan. A cette fin, il faut que le son ne soit ni obstrusif ni ennuyeux à écouter, car il est destiné à être joué continûment, même à faible volume. Gaver a conçu des sons de machine qui sont continus et peuvent donc être utilisés pour suivre des tâches de fond. Nous avons étudié l'utilisation des sons d'environnement naturel comme le vent et les vagues pour trois raisons [22]. Premièrement, comme nous l'avons vu dans la description de l'approche écologique de la perception, ces sons sont connus des êtres humains et seront donc plus faciles à identifier. De plus, ils possèdent des dimensions que chacun a appris à évaluer, et permettent donc la transmission d'information. Deuxièmement, comme ils appartiennent à l'environnement naturel des êtres humains, s'ils sont joués de façon adéquate, c'est-à-dire de manière douce et à un niveau raisonnable, ils sont susceptibles de ne pas être dérangeant pour l'utilisateur. Troisièmement, la nature de ces sons, qui contiennent peu de hautes fréquences et qui ont un comportement tel qu'il est facile d'éviter les motifs cycliques ou répétés et donc ennuyeux à la longue, laisse à penser qu'ils pourront être utilisés de manière continue, sur une durée relativement longue. En effet, un son de vent ne se reproduit

pas au cours du temps, et on n'entend pas deux fois exactement la même vague au bord de la plage.

Un son typique accompagnant le vent ressemble à un souffle qui monte et descend en hauteur continûment et indéfinement. Plus le vent est fort, et plus le souffle change brusquement de direction. De plus, plus le vent est fort, plus hautes seront les hauteurs atteintes, et plus le vent aura tendance à siffler ("fffuuuîîîî") (figure 4.5). Le souffle peut être simulé avec un bruit blanc, qui est synthétisé à l'aide d'une fonction qui tire une valeur au hasard pour chaque échantillon (figure 4.7).

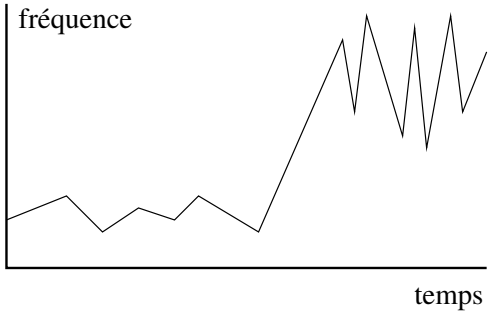


FIG. 4.5 – Evolution de la fréquence centrale du filtre passe-bande en fonction de la force du vent. Lorsque le vent est fort (partie droite) les intervalles de temps entre les changements de fréquences sont plus courts que lorsque le vent est faible. De même, les fréquences sont plus élevées lorsque le vent est fort.

Le bruit blanc donne un "pchhhh" que l'on peut comparer au son que donne une télévision lorsqu'elle n'est pas réglée sur un canal. Il s'agit en fait d'un son contenant toutes les fréquences du spectre, comme la lumière blanche qui contient toutes les couleurs visibles. Pour lui donner une certaine hauteur, il suffit de faire passer ce bruit blanc au travers d'un filtre dit "passe-bande". Ce filtre a pour but d'atténuer les fréquences qui ne sont pas proches de sa fréquence centrale. La distance à la fréquence centrale correspond à la largeur de bande du filtre. Plus la largeur de bande est grande, plus le filtre laisse passer de fréquences et moins il a une "hauteur" définie. Quand la largeur de bande diminue, le filtre ne laisse passer que quelques fréquences ce qui donne un sifflement lorsqu'il est appliqué à un bruit blanc.

L'algorithme du filtre passe-bande que j'ai utilisé peut être décrit comme suit [54] :

$$y_n = G \times (x_n - Rx_n - 2) + b_1 y_{n-1} + b_2 y_{n-2}$$

avec :

$$R = e^{-\pi \frac{B}{S}}$$

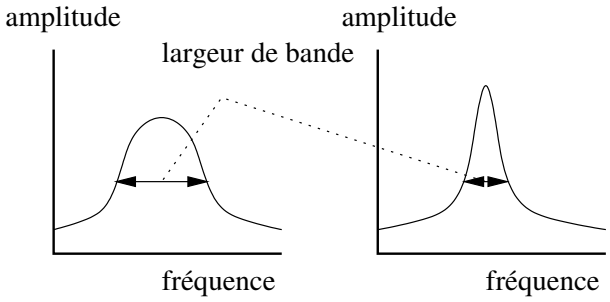


FIG. 4.6 – Filtre passe-bande. Lorsque la largeur de bande est petite, le filtre conduit à des sons dont la hauteur est plus définie.

$$
\begin{aligned}
G &= 1 - R \\
b_1 &= 2Rcos(2\pi\frac{f_c}{S}) \\
b_2 &= -R^2
\end{aligned}
$$

$y_n$ est l'échantillon généré. $x_{n-t}$ et $y_{n-t}$ représentent respectivement les valeurs des échantillons d'entrée et de sortie retardées de $t$ échantillons. $S$ est la fréquence d'échantillonnage, $f_c$ la fréquence centrale du filtre et $B$ sa largeur de bande.

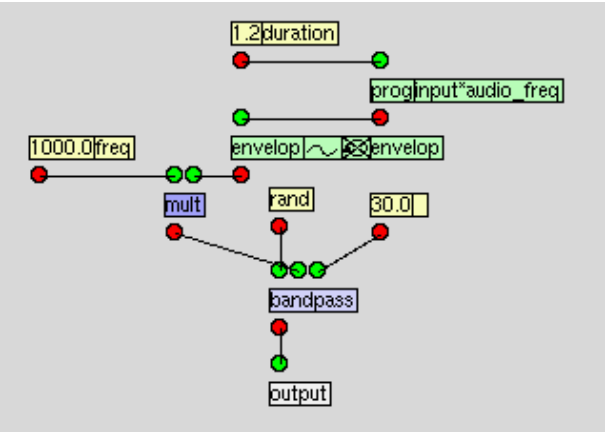


FIG. 4.7 – Le "patch" du son de vent. Un filtre passe-bande filtre un bruit blanc.

En manipulant la fréquence centrale du filtre, on peut donner l'impression que le bruit "monte" et "descend" en hauteur. Pour simuler le bruit du vent, il suffit de tirer aléatoirement deux valeurs. L'une correspond à la hauteur cible que doit atteindre la fréquence centrale du filtre. L'autre correspond à la durée nécessaire à la fréquence pour atteindre cette cible

| force | 0 | 1 |
|---|---|---|
| minfreq | 200 | 300 |
| maxfreq | 300 | 500 |
| mindur | 0.5 | 1 |
| maxdur | 1 | 3 |
| larg. de bande | 60 | 30 |

TAB. 4.1 – Domaines de valeurs des paramètres du filtre pour le son de vent. Les valeurs intérmédiaires doivent être interpolées linéairement par rapport à la force du vent.

linéairement. Plus la force du vent est grande, plus le domaine de valeurs dans lequel est tirée la fréquence cible a des bornes élevées. La fréquence de changement de direction est proportionnelle à la force du vent, donc la durée entre deux changements est inversement proportionnelle à la force. Plus la force est grande, plus petites sont les bornes du domaine de valeur dans lequel les durées sont tirées aléatoirement. Le tableau 4.1 précise les valeurs minimum et maximum que peuvent prendre les paramètres. Ainsi, les fréquences doivent être prises dans le domaine $[minfreq, maxfreq]$, et les durées dans le domaine $[mindur, maxdur]$. Le paramètre "force" du vent est normalisé, et doit donc être compris entre 0 et 1. Les valeurs intermédiaires sont interpolées selon la droite reliant les valeurs minimales et les valeurs maximales. Enfin, il est possible de manipuler la largeur de bande en fonction de la force : plus la force est grande, plus le son a tendance à siffler, c'est-à-dire que la largeur de bande sera plus petite (de 30 à 60 Hz). Le matériau sonore ainsi que le comportement du son forment les invariants qui spécifient le bruit du vent.

### 4.5.6 Les vagues

Nous avons vu que le bruit des vagues est un mélange de gouttes d'eau, de ruisselements, de frottements sur le sable et de réverbération dû à l'encloisonnement formé par la partie tubulaire de la vague. Synthétiser les sons de tous ces phénomènes serait prohibitif. Nous avons conçu un algorithme qui synthétise le son d'une vague à partir de trois paramètres qui spécifient l'événement. Le premier est la taille de la vague, qui influe sur la longueur du son. Le deuxième est la forme de la vague, selon qu'elle s'écrase sur elle-même en arrivant sur la plage ou qu'elle s'étale en longueur. La forme a une influence sur la fin du son : plus elle s'étale, plus elle entraine de sable ou de cailloux avec elle, et plus elle fait de bruit de frottements. Enfin, le troisième paramètre indique le type de plage sur laquelle les vagues se déroulent, en influant sur la "couleur" du son.

Le son de vague a été construit à partir de la description suivante. Tout

d'abord la vague s'écroule et gronde en propageant de la mousse puis s'étale sur le sable plus ou moins loin selon sa forme. A l'instar du bruit du vent, on peut simuler le bruit d'une vague en partant d'un bruit blanc filtré par un filtre passe-bande (figure 4.8). En contrôlant l'amplitude du son, la largeur de bande et la fréquence centrale du filtre par des enveloppes en fonction du temps, on peut imiter le comportement du son. Ainsi, l'amplitude augmente quand la vague s'écroule avant de diminuer puis d'augmenter à nouveau au moment ou elle s'étale sur le sable. Elle finit en douceur en tendant vers zéro. Dans la première phase, les fréquences sont plutôt basses car la vague roule sur elle-même. Pour simuler l'étalement sur la plage, il suffit d'augmenter la fréquence centrale du filtre et d'augmenter la part de bruit dans le son en élargissant la bande passante (figure 4.9).

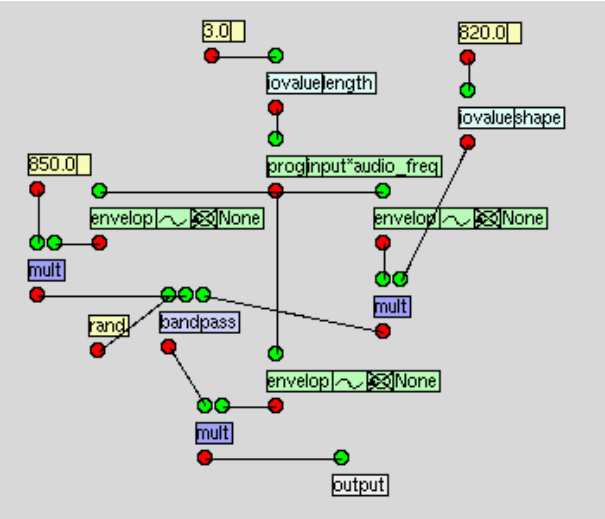


FIG. 4.8 – Le patch du son de vague. Il est similaire au patch du vent (figure 4.7), l'amplitude, la bande passante et la fréquence centrale sont contrôlées par des enveloppes

La simulation d'un rivage consiste en une séquence de sons de vagues. La période entre chaque son dépend de l'état de la mer, agitée ou calme. Cependant, rien n'empêche de simuler le son de la mer avec plusieurs vagues jouées en parallèle. Ainsi, une mer agitée est spécifiée par plusieurs grosses vagues plus ou moins distantes, alors que le son d'une mer calme est composée d'un seul flux de vaguelettes. Enfin, il est possible de rendre le son de la mer totalement continu en gardant les valeurs finales des paramètres du filtre et en augmentant l'amplitude du son jusqu'à la prochaine vague. Ce comportement simule les plages à forte pente où le retour de l'eau dans la mer entraîne les pierres de la même façon que suite à l'étalement sur la plage.

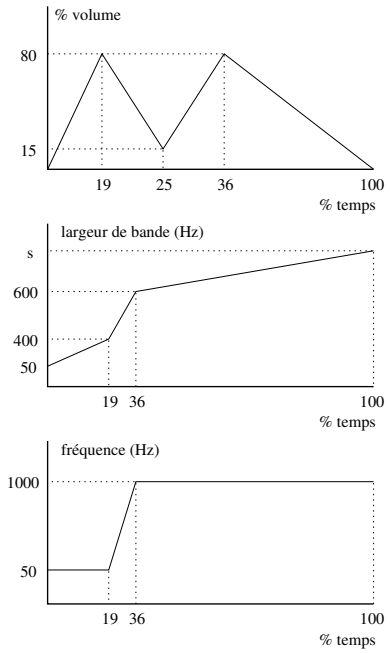


FIG. 4.9 – Les enveloppes d'amplitude, de largeur de bande, et de fréquence centrale contrôlant le filtre passe-bande du son de vague. Pour l'enveloppe de largeur de bande, le paramètre $s$ est contrôlé par la forme de la vague et sa valeur maximum est de 3000 Hz.

### 4.5.7 La pluie

Après le vent et les vagues, un autre phénomène naturel que nous avons étudié est la pluie. La principale difficulté vient de la synthèse de son de gouttes d'eau qui doit être identifiable facilement, car c'est le son qui contribue le plus au processus d'identification d'un son de pluie. En mélangeant une séquence de sons de goutte avec un fond à base de bruit blanc, il serait possible de donner une indication de force de la pluie, en augmentant la contribution du bruit blanc dans le son final. En jouant avec les gouttes d'eau, on peut indiquer le type de matériau ou d'objet sur lesquelles elles ruissellent.

En fait, le son d'une goutte d'eau dépend essentiellement de l'objet sur lequel elle tombe. Si l'objet est petit, le son de la goutte d'eau contient des fréquences élevées. Si le matériau est dur, la goutte d'eau produit un son avec une attaque très courte, voire abrupte. Le son que nous avons conçu pour simuler un son de goutte d'eau utilise une sinusoïde de courte durée (0,2 s) dont la fréquence et l'amplitude sont contrôlées par des enveloppes (figure 4.11). Le son obtenu n'est pas complètement convaincant. La difficulté vient du fait que lorsque l'on change la taille de l'objet, c'est la valeur de la

fréquence qui a une influence sur la taille perçue. Cependant, les enveloppes n'ont pas la même forme selon la fréquence choisie. Si elles gardent la même forme, le son ne ressemblera plus à une goutte d'eau pour certaines valeurs de fréquence. Les enveloppes sont présentées dans la figure 4.11. Le matériau est simulé par l'attaque en fréquence : plus le matériau est dur, plus $m$ est petit avec $m$ dans $[0,02s, 0,2s]$. Quand l'objet est gros (fréquence de 500 Hz), $a2$ est à 91 % de la fréquence de la sinusoïde et $a1$ à 84 %. Quand l'objet est petit (fréquence de 3000 Hz), $a2$ est à 67 % et $a1$ à 43 %.

Nous avons essayé d'interpoler les valeurs des points de l'enveloppe entre plusieurs enveloppes de référence mais le résultat n'est pas satisfaisant. Cependant, lorsque le son de goutte est inclus dans une séquence typique d'un filet de gouttes et lorsque cette séquence est associée à un bruit blanc, il peut être reconnu comme un son de pluie. En utilisant plusieurs filets de gouttes, et en contrôlant la fréquence d'occurrence, on peut donner des informations sur la force de la pluie. Les filets de gouttes peuvent être discriminés par la taille de l'objet sur lequel elles tombent, et par leur disposition dans l'espace.

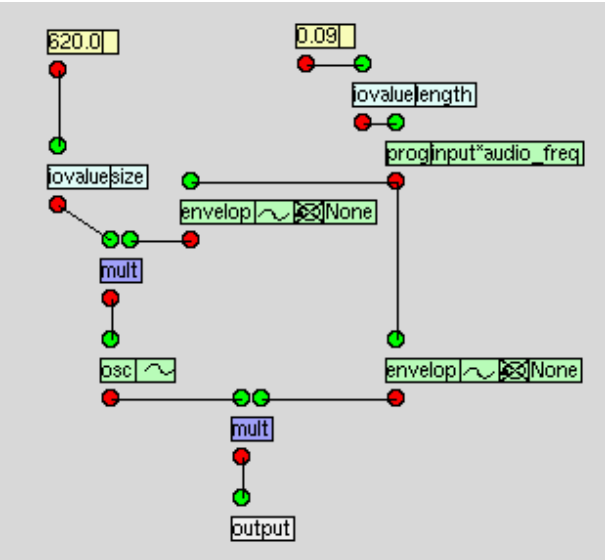


FIG. 4.10 – Le "patch" pour le son de goutte et le son de bulle. Il consiste en une sinusoïde dont la fréquence et le volume suivent des enveloppes.

J'ai trouvé une imitation du son de gouttes très convaincante fournie avec un logiciel de synthèse sonore appelé Virtual Waves. Virtual Waves permet de construire des sons échantillonnés à partir d'algorithmes de synthèse. Il ne travaille pas en temps réel, et les sons échantillonnés sont destinés à être utilisés dans des synthétiseurs dont les seuls paramètres de contrôle sont la hauteur et l'amplitude. Le son de pluie n'est pas paramètré par des attributs de haut-niveau. Il est construit à l'aide de formes d'onde en

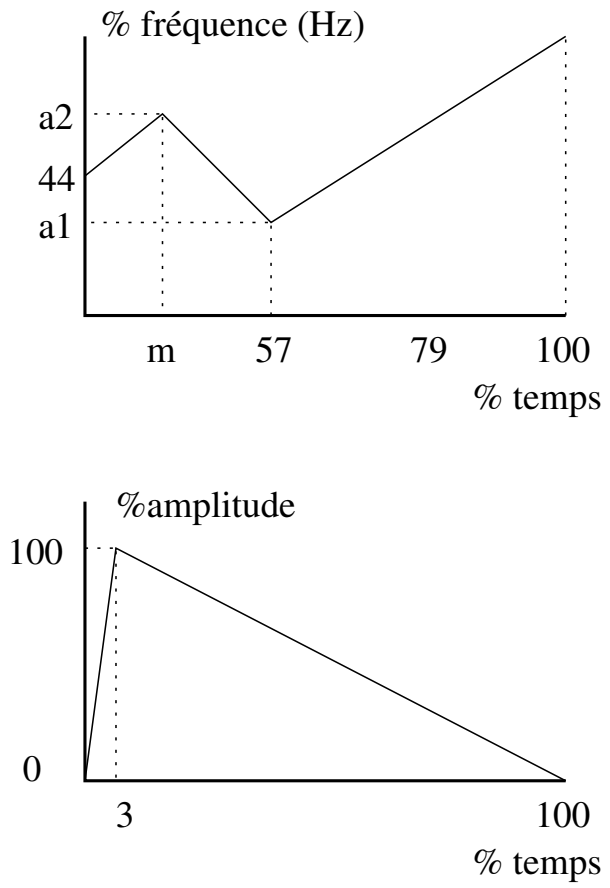


FIG. 4.11 – Les enveloppes de fréquence et d'amplitude pour le son de goutte.

dent de scie qui contrôlent le volume des gouttes d'eau, et d'un filtre passe-bande résonnant au travers duquel passe un bruit blanc filtré (figure 4.12). L'effet de goutte d'eau est produit grâce à un ensemble de paramètres précis du filtre, et est typique du genre d'effets imprévisibles qui peuvent surgir de telles transformations sonores. J'ai essayé de programmer ce son avec d'autres système de synthèse sans succès. En fait, j'ai découvert que l'effet de goutte d'eau n'était obtenu qu'avec le filtre particulier de Virtual Waves. J'ai essayé d'utiliser diverses implémentations d'un tel filtre, sans réussir à obtenir le même son.

Les effets obtenus avec la synthèse ad-hoc sont donc très dépendants des algorithmes utilisés et des paramètres associés à ces algorithmes. Ce phénomène a deux conséquences. Tout d'abord, la conception d'algorithmes est d'autant plus ad-hoc que ces effets ne sont généralement pas prévisibles même lorsque l'on connaît le but d'un algorithme. Cela implique que pour utiliser de façon efficace ces techniques, il faut disposer d'un "designer",

une personne à mi-chemin entre le technicien et l'artiste, qui saura comment obtenir un son et qui connaîtra les différents effets que peuvent donner certains algorithmes. Ensuite, lorsqu'un concepteur décrit un algorithme ad-hoc qui permet de synthétiser une icone auditive, il doit donner toutes les informations qui permettent de le reproduire. Il doit non seulement donner les paramètres réels utilisés par les algorithmes, mais aussi le pseudo-code des algorithmes utilisés. Comme il peut exister plusieurs implémentations connues sous un même nom générique ("un filtre résonnant") et que ces implémentations peuvent donner des résultats différents, ces informations sont nécessaires à la reproductibilité.

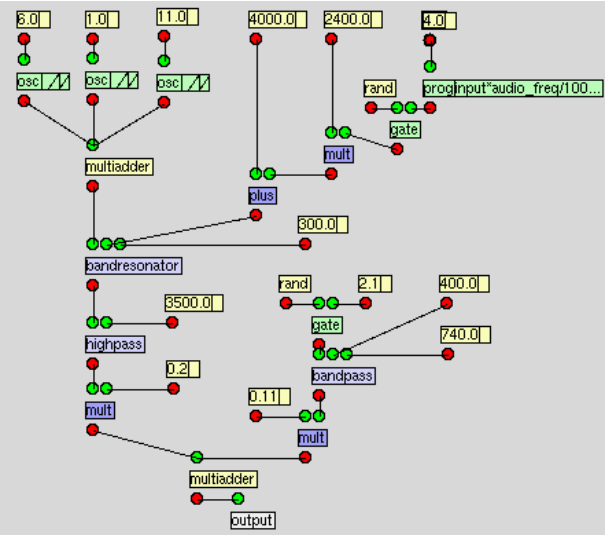


FIG. 4.12 – Le patch du son de pluie de Virtual Waves.

### 4.5.8 Le bouillonnement

Toujours à la recherche de nouveaux sons, en écoutant celui fait par un bouillonnement je touvai celui-ci discret et toutefois informatif quant à l'état du liquide et des diverses transformations subies par celui-ci. Lors de mes périgrinations culinaires je décidai un jour de concocter une "polenta", un met italo-savoyard dont la base est la semoule de maïs. La recette consiste à cuire la semoule de maïs avec de l'eau bouillante de la bonne façon c'est-à-dire ni trop, ni trop peu, et surtout avec la *bonne* quantité d'eau. La première partie consiste donc à faire chauffer de l'eau. Celle-ci commence à produire des bulles au bout d'un certain temps avant d'être complètement en ébullition, indiquant que l'on peut y incorporer la semoule. Tout l'art de la recette consiste à doser la quantité d'eau pour qu'elle ne soit pas entièrement bue par la semoule trop rapidement, et à doser la force de l'échauffement

pour ne pas griller la semoule sans la cuire. Or, quand la température est trop élevée, ou qu'il y a trop peu d'eau, la semoule boit entièrement l'eau et forme une masse compacte dont la vapeur a du mal à se dégager. il faut alors plus de pression pour que les bulles de vapeur s'échappent du mélange eau-semoule, les bulles arrivant finalement à s'éjecter sont donc plus grosses, et *sonnent* plus grosses. Ainsi, par la taille des bulles dont on entend le son, on peut deviner à distance ce qui se passe : plus les bulles sont grosses, plus le mélange-eau semoule est compact. Les grosses bulles ne sont pas synonymes d'échec. Tout dépend du moment où elles apparaissent durant la cuisson. Lorsque l'on entend de grosses bulles à très court terme, la recette est ratée. Lorsqu'elles arrivent au bout "du temps qui convient", la recette est terminée.

Pour imiter un son de bouillonement liquide, j'ai d'abord conçu une caricature de son de bulle. Il s'agit du même patch que celui du son de goutte, une sinusoïde dont l'amplitude et la fréquence sont controlées par des enveloppes (figure 4.10). L'amplitude sert à contrôler le début et la fin du son, et la fréquence permet d'imiter la fin du son qui se termine plus haut que le début ("bouuîîp"). En jouant sur la longueur et la fréquence de base, on peut simuler des caractéristiques de taille et de viscosité. Plus la taille est grande, plus la fréquence est basse. De même, plus la viscosité est grande, plus la longueur de la bulle est grande (figure 4.13). Nous employons des valeurs comprises dans le domaine [200 Hz, 1200 Hz] pour les fréquences et des valeurs dans le domaine [0,1s, 0,3s] pour les durées. Le son résultant ressemble au son échantillon "drop" proposé dans le système MacOS. Rappelons que cet échantillon n'est pas contrôlable par des dimensions de haut-niveau, comme l'est le son proposé ici. Ce dernier pourrait servir à spécifier des niveaux ou des types d'alarmes différents, ce que ne propose pas MacOS.

Le son de bouillonnement consiste à jouer des sons de bulle de tailles variées, avec une fréquence dépendant de l'activité de l'eau (figure 4.14) et comprise dans l'intervalle [1 Hz, 10 Hz]. En tirant des valeurs de taille au hasard pour chaque bulle, on obtient un son typique de bouillonnement. Afin de spécifier plus d'informations, on peut synthétiser deux "flux" de bulles. L'un est destiné aux bulles de grosse taille, l'autre aux bulles de plus petite taille. En faisant varier la contribution de chaque flux dans le son final, on peut donner une idée de la composition du liquide, ou de l'état de la combustion. De même, on retrouve le paramètre de viscosité de chaque bulle dans le son de bouillonnement : la viscosité du bouillonnement contraint la viscosité de chaque bulle.

A ce processus sonore continu, on peut associer un son plus événementiel qui indique une urgence. Ainsi, en utilisant un bruit blanc filtré, on simule le débordement du liquide et le grésillement de celui-ci sur la plaque chauffante ou le feu. L'un des problèmes que les concepteurs d'interface ont avec le son réside dans le fait qu'il est difficile de quantifier absolument une dimension. Si

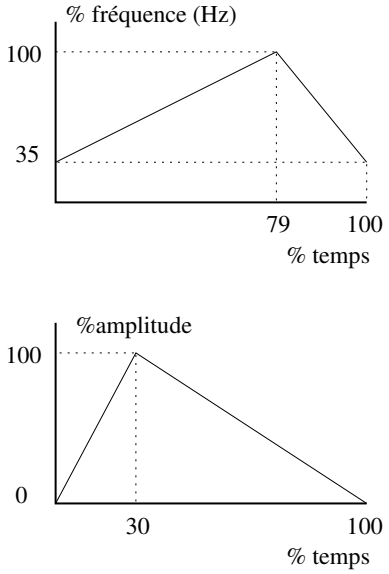


FIG. 4.13 – Les enveloppes de fréquence et d'amplitude pour le son de bulle. Les fréquences vont de 200 Hz à 1200 Hz et les durées de 0,1 s à 0,3 s.

une majorité de la population est capable de comparer deux sons ne différant que par la *valeur absolue* d'une de leur dimension, une minorité est capable d'évaluer la valeur d'une dimension pour un son entendu. Il est donc difficile de proposer des sons qui contiennent des informations absolues. Ainsi, on pourrait concevoir une alarme dont l'urgence est reflétée par la fréquence : plus la fréquence est haute, plus grande est l'urgence.

Il existe des situations qui ne nécessitent pas que les utilisateurs règlent les problèmes immédiatement. Ils peuvent donc ordonner ces problèmes afin de pallier les plus urgents. En revanche, certains problèmes nécessitent une intervention immédiate. Si une alarme à intervention immédiate est spécifiée par une valeur de fréquence particulière, elle est susceptible d'être ignorée par l'utilisateur si celui-ci l'évalue mal. On peut pallier ce problème en associant un son de référence qui indique la valeur maximale d'une fréquence par exemple. La tâche de l'utilisateur consiste à comparer les deux valeurs entendues et à décider s'il doit intervenir immédiatement. Grâce à un avertissement sonore comme le grésillement qui spécifie un problème effectif de l'événement réel (l'eau déborde) dont il faut s'occuper immédiatement (en retirant la casserole du feu, puis en baissant la puissance de la chaleur), on peut spécifier naturellement une limite à ne pas franchir. Quand celle-ci est atteinte, le son est naturellement et suffisament alarmant pour que l'on se charge du problème. L'occurence de ce son est un invariant qui spécifie un état d'urgence pour l'activité qu'est la cuisson.

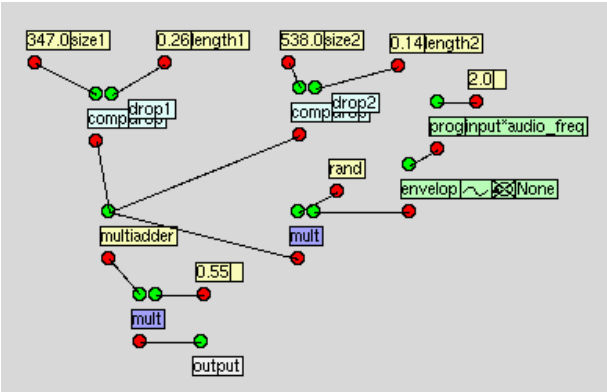


FIG. 4.14 – Le “patch” du bouillonnement. Il s’agit d’un ensemble de flux de bulles (au nombre de 2 dans cet exemple) auquel on peut ajouter un grésillement. Ce dernier est simulé par un bruit dont l’amplitude suit une enveloppe.

### 4.5.9 L’ascenseur

Comme nous sommes plus intéressés par les attributs de haut-niveau d’un son que par le son en lui-même, il est peut-être possible d’évaluer des dimensions écologiques dans des sons qui ne sont pas naturels. Ainsi, nous avons utilisé un effet accoustique, le son de Sheppard-Risset [7], pour suivre la qualité d’une connexion. Le son de Sheppard-Risset est un son qui donne l’impression de monter ou descendre en hauteur indéfiniment [66]. Il est un peu le pendant auditif de l’escalier sans fin du dessinateur M.C. Escher.

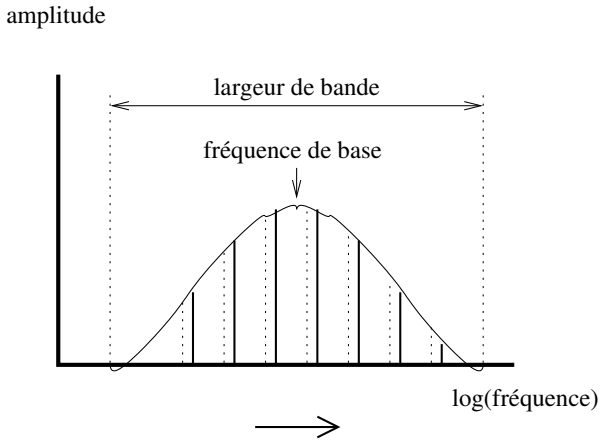


FIG. 4.15 – Composition d’un ton de Sheppard-Risset : pour un son qui “ monte ”, les partiels glissent en suivant la courbe en cloche.

Les sons de Sheppard-Risset sont fait de partiels dont l'amplitude est bornée par une courbe gaussienne (en cloche). Au cours du temps, les fréquences de chaque partiels sont décalées, et leur amplitude est modifiée pour suivre la courbe en cloche. Lorsque le son "est en mouvement", de nouveaux partiels entrent dans la courbe à une extrémité alors que les partiels qui atteignent l'autre extrémité disparaissent (figure 4.15). Ainsi, la fréquence globale du son est constante, mais est perçue comme montant ou descendant indéfiniment. Les sons de Sheppard-Risset sont contrôlés par quatre paramètres :

- la fréquence de base, qui détermine la fréquence centrale de la courbe en cloche ;
- la largeur de bande, qui est le rapport entre la plus haute et la plus basse fréquence de l'enveloppe ;
- la vitesse, qui représente la quantité de décalage des fréquences par unité de temps ;
- la densité, qui est le rapport entre partiels successifs. Pour une bande passante donnée, la densité détermine le nombre de partiels.

Nous avons déterminé empiriquement les valeurs des paramètres nécessaires pour maintenir l'illusion. Six partiels et une densité de 1,2 donnent des résultats satisfaisants. Plus de partiels donnent un son plus riche et plus intrusif. Avec ces valeurs, la largeur de bande optimale est définie par un rapport de 3, soit 1,5 octave. Nous avons trouvé qu'une approximation de la courbe en cloche par une enveloppe triangulaire ne résultait pas en une perte de qualité sonore, ce qui permet d'utiliser un algorithme plus simple.

Nous avons donc développé une icone auditive reprenant cet effet. La vitesse du mouvement peut être changée et peut être mise à zéro, ce qui a pour effet de rendre le son "immobile". On peut évidemment changer la direction du mouvement. Varier la hauteur de la fréquence moyenne donne un impression de taille, alors qu'il n'y a pas vraiment de source naturelle pour un tel son. En fait, nous avons appelé ce son "ascenseur" car il évoque celui d'un ascenseur réel et qu'il peut avoir pour dimension la "taille", la "vitesse" et la "direction". Pourtant, les invariants qui spécifient la direction de marche d'un ascenseur sont très subtils et n'ont rien à voir avec celui de l'icone auditive. Cet exemple montre que les sons ne doivent pas être nécessairement naturels tant qu'ils fournissent des informations utiles par des dimensions écologiques [23]. L'absence de réalisme que nous avons défini comme n'étant pas importante pour une icone auditive est ici portée à son paroxysme, puisqu'il n'existe pas d'événement écologique réel qui produit un tel son. C'est l'utilisation d'invariants écologiques qui permet à des sons abstraits d'avoir des dimensions écologiques, et donc susceptibles d'utiliser les capacités naturelles d'interprétation de notre système perceptif.

Ces résultats sont consistants avec une expérience destinée à analyser les préférences des utilisateurs quant à la correspondance entre données auralisées et dimensions sonores [81]. La tâche consiste à écouter un ensemble de

données avec pour dimensions la température, la pression, la vélocité et la taille. Les dimensions sonores utilisées sont soit la hauteur, soit le tempo. Le but de l'expérience est de voir si les sujets préfèrent une correspondance à polarité positive, comme par exemple une plus grande hauteur pour une plus grande température, ou une correspondance à polarité négative, comme une plus grande hauteur pour une plus petite vitesse. Les résultats indiquent que seules quelques préférences se retrouvent chez la majorité des sujets. Les auteurs concluent que si le signe de certaines polarités peut se deviner (comme la fréquence qui augmente avec la température), d'autres ne peuvent être déterminées qu'expérimentalement. Les auteurs donnent ici l'exemple du tempo qui diminue lorsque que la taille augmente. Si l'on utilise une approche écologique de l'analyse des résultats, cette correspondance semble cohérente avec ce que l'on observe dans la vie courante, c'est-à-dire que les objets lourds ont tendance à se déplacer moins vite. Pour ce cas, les sujets ont utilisé un invariant qui caractérise un objet lourd. De même, une majorité de sujets préfère une polarité négative entre fréquence et taille. Encore une fois, ils ont utilisé un invariant qui stipule que les objets plus gros sonnent à une hauteur plus basse. Les participants ont donc utilisé des dimensions écologiques dans des sons qui ne sont pas associés à des événements réels.

## 4.6 Résumé

Dans ce chapitre, le problème de la conception d'icones auditives a été abordé. Il existe deux moyens de jouer un son avec un système informatique : les échantillons et la synthèse. La synthèse est plus appropriée pour les icones auditives parce qu'elle offre plus de contrôle à condition qu'elle ne nécessite pas de grandes quantités de calculs.

La synthèse de sons peut faire appel à des techniques de modèlisation physique des événements sonores. Cependant, ces techniques nécessitent pour la plupart une grande puissance de calcul et ne sont appropriées qu'aux sons pouvant être analysés facilement. Les techniques à base de modèles de signaux ne sont pas assez facilement contrôlables selon des dimensions écologiques.

Nous avons proposé d'utiliser une méthode de synthèse *ad-hoc*, arguant du fait qu'en tant que concepteurs d'interfaces, nous sommes plus intéressés par la reproduction des *invariants* d'un son que par la reproduction de son *spectre*. Nous avons utilisé des algorithmes efficaces de synthèse pour produire en temps réel des sons de vent, de vagues, de pluie, de bouillonnement, et d'ascenseur qui sont tous contrôlables par des dimensions écologiques.

Dans le chapitre suivant, nous présentons un outil qui permet de concevoir et d'intégrer les icones auditives paramétrées dans les applications interactives.

# Chapitre 5

# Sofa

Après avoir décrit comment des sons synthétisés de manière “ad-hoc” pouvaient apporter des informations par des dimensions écologiques, ce chapitre a pour but d'introduire un outil qui permet de les créer et de les intégrer dans les applications interactives. Tout d'abord, les systèmes de synthèse existant sont évalués en fonction de critères indispensables à la génération et au contrôle d'icones auditives. Ensuite, nous introduisons Sofa, un outil que nous avons développé et qui facilite le prototypage et l'intégration d'icones auditives.

## 5.1 Systèmes de synthèse

Pour pouvoir incorporer des icones auditives dans les applications, il faut pouvoir disposer d'un système qui permette de les créer et de les utiliser. Lorsque les sons sont utilisés et joués, il faut qu'ils soient contrôlables et synthétisables en temps réel. De plus, comme le processus de conception des interfaces est itératif, il est souhaitable que le fossé entre la création des icones auditives et leur utilisation soit le moins large possible. D'autre part, lorsque nous avons commencé à concevoir des icones auditives, l'un de nos buts était d'utiliser un système porté sur différentes plateformes. En effet, nous comptions faire nos premières expérimentations au sein de l'équipe dans laquelle nous travaillons, dont les membres disposent d'une grande variété de machines. A l'époque où ces travaux ont commencé, il existait plusieurs systèmes de création de sons synthétisés qui chacun reprenait une des caractéristiques énoncées plus haut. Malheureusement, pour diverses raisons que nous allons évoquer, aucun ne nous convenait parfaitement. La première partie de ce chapitre est consacrée à l'étude des systèmes existant. Il faut noter que la majorité d'entre eux est consacrée à la production musicale et ne sont pas destinés à être utilisés dans une interface.

### 5.1.1 Caractéristiques d'un système pour icones auditives

Comme nous l'avons vu, il faut que le système de synthèse fonctionne en temps réel. Généralement, les sytèmes temps réel sont développés dans un but musical. Pendant leur utilisation, il est rare que d'autres applications fonctionnent en parallèle sur la même machine. Or, le but de la sonification par icone auditive n'est pas le son en lui-même, mais l'utilisation d'une interface auditive par des applications interactives. Ces applications ont des tâches différentes de la production sonore, et prennent donc du temps machine pour leurs propres calculs. De plus, l'une des orientations de cette thèse est le suivi d'activités d'arrière-plan par le son, ce qui signifie qu'il y a plusieurs activités qui tournent en parallèle. Il faut donc se soucier à la fois d'avoir suffisamment et régulièrement des ressources pour le système de synthèse, mais aussi ne pas les monopoliser au détriment des autres applications. Les contraintes temps-réel sont donc encore plus fortes que pour les systèmes musicaux, car ces derniers sont généralement les seuls processus qui s'exécutent sur une machine.

Pour respecter ces contraintes, il faut que les algorithmes à la disposition du concepteur soient programmés de manière efficace. Evidemment, l'efficacité des algorithmes dépend de la qualité de leur conception, et offre peu de possibilité de configuration de la part d'un utilisateur. En revanche, un concepteur de sons peut "aider" le système à synthétiser un son en contrôlant le graphe de connexions. Ainsi, si à un moment particulier de la génération, le concepteur sait qu'une partie du graphe n'a plus d'influence sur le résultat final, il peut couper le lien qui connecte cette partie au reste du graphe, et éviter ainsi de faire calculer par le système des échantillons inutiles. De même, pour certains sons qui se finissent sans interaction extérieure (comme les sons d'impacts), il peut être utile de les arrêter pour éviter des calculs inutiles. La possibilité d'avoir un contrôle du graphe permet donc de contrôler le montant de travail à effectuer par le système, et de minimiser la puissance de calcul nécessaire à la génération d'un son.

Une interface utilisateur graphique est nécessaire à la conception d'un graphe d'algorithme. Cependant, la majorité des éditeurs de sons ne prévoit de contrôle qu'à travers l'interface graphique ou par des événements MIDI. Pourtant, il est parfois difficile ou peu naturel de programmer un algorithme de façon graphique plutôt que textuelle. Beaucoup de concepteurs d'interface sont aussi des programmeurs. Un langage visuel est pratique pour des utilisateurs non-informaticiens, mais inadapté pour des programmeurs. L'emploi de programmes textuels pour le contrôle est parfois mieux adapté à certains algorithmes, surtout lorqu'un son est composé de plusieurs sons, ou que les dimensions de contrôle doivent être transformées pour respecter les contraintes de contrôle par attributs de haut-niveau. Les icones auditives devant respecter ces deux contraintes, un outil permettant leur conception se doit donc de proposer des moyens de contrôle variés, et la meilleure façon

de ne pas être limité est d'utiliser un langage de programmation.

La conception des interfaces homme-machine se fait de manière itérative [56]. Les utilisateurs et les concepteurs participent à des réunions de travail communes et conçoivent l'interface ensemble. Des prototypes sont construits et testés, et le cycle recommence. Si les outils de conception et de prototypage sont difficiles à mettre en œuvre, ou ne produisent pas des résultats facilement retouchables, il est difficile de profiter pleinement de cette méthode de conception, car les cycles sont plus longs, moins nombreux, et donc moins d'idées sont testées. De plus, des idées d'interface peuvent ne pas être testées si elles impliquent une phase de prototypage trop complexe, et sont donc éliminées par les concepteurs. Enfin, le réglage dans le contexte d'utilisation permet de réduire le fossé entre conception et utilisation. Un outil de conception et de prototypage doit donc donner des résultats utilisables et retouchables facilement.

Enfin, pour permettre la reproductibilité, l'extensibilité, la portabilité et la disponibilité, il faut que le système ait des sources libres [1]. La reproductibilité permet la diffusion des résultats de recherche. Nous avons vu dans le chapitre précédent comment l'implantation d'un filtre résonnant a une influence sur le son produit. Si le code source du filtre avait été disponible, nous aurions pu reproduire le son de pluie tel qu'il avait été fourni. Par ailleurs, en développant notre propre algorithme de filtre résonnant à partir de celui de CSound, nous avons eu des difficultés à obtenir les résultats de CSound avec des paramètres similaires. C'est en observant le code source et en constatant que CSound contraignait la valeur des échantillons dans le domaine de valeur des entiers signés sur 16 bits que nous avons pu comprendre pourquoi le filtre de CSound donnait des sons différents du notre. Ces deux exemples, s'ils ne dispensent pas de décrire précisément les algorithmes utilisés, montrent comment la disponibilité du code source rend possible la reproductibilité. Enfin, l'extensibilité d'un système permet d'ajouter de nouvelles fonctionnalités. Il convient de noter que la disponibilité du source permet l'extensibilité. La disponibilité et la portabilité permettent la diffusion et le test d'interfaces à plus grande échelle.

En résumé, un système permettant la création et l'utilisation d'icones auditives

- doit fonctionner en temps réel,
- doit pouvoir être contrôlé par une interface graphique et des programmes textuels,
- doit permettre une transition aisée entre conception et utilisation,
- doit être extensible, portable, et permettre le déploiement comme le sont les projets "open source".

[1] voir à ce propos http ://www.opensource.org, et notamment l'article en ligne de Eric S. Raymond, "The Cathedral and the Bazaar", http ://www.tuxedo.org/ esr/writings/cathedral-bazaar/

### 5.1.2 Systèmes musicaux

Les systèmes dits “musicaux” sont les systèmes de synthèse qui ont été conçus dans un but musical. Cela ne signifie pas qu’ils ne sont pas utilisables en tant que système de conception d’icones auditives, mais leurs capacités d’intégrations et de déploiement dans des applications interactives sont limitées. La première classification que l’on peut faire lorsque l’on compare des systèmes de synthèse est celle des systèmes qui fonctionnent en temps réel ou en temps différé. Le calcul en temps réel pour une application a deux aspects. Premièrement, travailler en temps réel signifie que les calculs nécessaires à la production d’échantillons ne prennent pas plus de temps que la durée du son produit. Deuxièmement, en calculant les échantillons en direct, les algorithmes peuvent être contrôlés par un utilisateur ou une application cliente, et produire des sons dépendant de ces contrôles.

Rappelons que l’utilisation des icones auditives ne peut se faire qu’avec un système en temps réel. En effet, il n’est pas possible de connaître à l’avance l’évolution des valeurs des paramètres d’une icone auditive, il faut donc que la synthèse réagisse immédiatement aux changements de ces paramètres. On ne peut donc pas précalculer des sons et se contenter de les jouer comme on le ferait avec des sons échantillonnés. De plus, comme nous l’avons dit dans le chapitre précédent, il est prohibitif de calculer tous les échantillons possibles.

#### CSound

Pour un système de synthèse, travailler en temps réel dépend de plusieurs éléments. Premièrement, il faut que le système soit prévu pour le faire, c’est-à-dire qu’il ne soit pas un simple compilateur de description de sons produisant des échantillons que l’on ne peut plus contrôler. C’est le cas des systèmes dit “Music N” (N allant de I à V) développés par Max Mathews. On peut nommer parmi les plus connus des dérivés de Music N le système CSound, surtout utilisé dans un but musical [54]. CSound est véritablement un compilateur de sons, prenant en entrée deux fichiers de description textuelle et donnant un son allant d’une note à une véritable composition musicale. Le premier des deux fichiers concerne l’“orchestre” (.orc), et décrit comment synthétiser les sons des instruments de la séquence sonore finale. Le deuxième est une sorte de “partition” (“score”,.sco), et décrit l’enchaînement des événements musicaux, en donnant les moments auxquels les notes sont joués, l’instrument par lequel elles sont jouées ainsi que les paramètres contrôlant l’instrument, comme le volume ou la hauteur.

Dans l’ochestre suivant :

```
instr 1
a1  oscil 10000, p4, 1
out a1
endin
```

on définit un instrument composé d'un oscillateur dont l'amplitude est de 10000. La fréquence dépend du paramètre "p4" que l'on précisera dans le fichier "partition". L'oscillateur utilise une table d'échantillons dont le numéro est 1.

On peut jouer une note de cet instrument avec une partition comme celle-ci :

```
f1 0 4096 10 1 ; utilise GEN10 pour calculer une forme d'onde sinusoidale
;ins    strt    dur       freq (p4)
i1       0       4        440
i1       4.5       7        220
e                  ; indique la fin de la partition
```

Tout d'abord, un table d'onde dont le numéro est 1 (f1) est créée et remplie par la fonction de générations d'échantillons numéro 10, correspondant à une sinusoïde. La table a 4096 échantillons, et sera utilisé par l'instrument que nous avons vu plus haut. Ensuite, la deuxième partie décrit véritablement la partition. Ici, elle se compose d'une première note, jouée par l'instrument numéro 1, commençant au temps 0 et durant 4 secondes. Le paramètre de fréquence est passée par l'intermédiaire du quatrième paramètre. La première note a une hauteur de 440 Hz, la deuxième une hauteur de 220 Hz.

CSound dispose de très nombreux générateurs et opérateurs, qui ont été développés par les auteurs et par les utilisateurs du système. En revanche, il n'a pas été prévu dès le départ pour être utilisé en temps réel, c'est-à-dire sans fichier de partition mais en répondant à des événements de l'utilisateur ou d'applications clientes. Des extensions ont été dévelopées pour permettre un contrôle avec des événements MIDI, mais ce contrôle reste limité à ces type d'événements et ne permet pas notamment de charger ou modifier un instrument pendant l'exécution.

Si CSound n'est pas adapté au temps réel, le fait que ses sources soient libres, qu'il soit porté sur toutes les plateformes connues, qu'il dispose de nombreux algorithmes, qu'il soit soutenu par une communauté vivante et nombreuse en fait un outil remarquable de description et de diffusion de sons, et donc d'icones auditives. Premièrement, il suffit de disposer de deux descriptions textuelles et de les compiler pour écouter une icone auditive. Deuxièmement, sa disponibilité et son code source libre en font un système de description universel, et donc permettent la reproduction. Malheureusement, la syntaxe des fichiers de description est peu intuitive. Le projet SAOL, qui implante la norme MPEG 4 de compression sonore, essaie de pallier ce problème en proposant un langage actualisé [2].

[2] voir par exemple http ://www.cs.berkeley.edu/ lazzaro/sa/index.html

### Quasimodo

Il existe une nouvelle implantation de CSound appelée Quasimodo [3], dont le but est de prendre les algorithmes ("opcodes") de CSound et de les utiliser dans une application graphique et en temps réel. Nous avons participé au développement de Quasimodo, et développé un système qui permettait l'intégration d'icones auditives dans les applications à l'aide des algorithmes de CSound. Cependant, le modèle sous-jacent de Quasimodo n'etait pas satisfaisant pour le développement d'algorithmes simples. Paul Barton-Davis, l'auteur principal de Quasimodo, voulait simuler des "racks" de modules physiques que les musiciens connectent entre eux, et qui ont chacun une fonction spécifique (synthétiseur, réverbération, boîte à effets...) et qui surtout ont chacun une seule sortie. Les algorithmes de CSound, appelés "opcodes" peuvent générer plusieurs sons en même temps : ils sont polyphoniques. Or Quasimodo est organisé en "modules" logiciels qui gèrent un patch d'opcodes. Seuls les modules peuvent être connectés entre eux, et chaque module gère l'ensemble des voies que génère le patch. Les sorties des modules sont la somme (ou le mixage) de l'ensemble de ces voies. Il n'est pas possible de construire un patch global à partir des modules, dont une instance indépendante serait construite pour chaque voix. Par example, il ne serait pas possible de connecter un module simulant un son d'impact de Gaver à un module simulant l'atténuation par un obstacle, et obtenir deux sons simultanés avec des valeurs différentes des paramètre de l'impact et de l'atténuation. De plus, Quasimodo est difficile à installer car il requiert plusieurs librairies externes en cours de développement, ce qui rend toute modification difficile à diffuser.

### MAX

Un système de synthèse se compose de deux parties. L'une s'occupe de la synthèse proprement dite, et se charge de gérer et de faire fonctionner les algorithmes. L'autre partie sert à la construction et au contrôle des algorithmes. Elle se présente soit sous forme textuelle comme dans CSound, soit sous la forme d'une interface graphique. Par exemple, le système MAX permet de construire graphiquement le réseau connectant les entrées et les sorties des algorithmes produisant un son [64]. L'auteur de MAX classifie son programme comme un "environnement de programmation graphique", car il permet de construire un algorithme à l'aide d'une interface graphique, et sans description textuelle. Le réseau composant le son est appelé "patch" en référence aux modules analogiques utilisés auparavant par les musiciens électroniques, et qui étaient reliés par des câbles ("patch"). L'originalité de MAX réside dans le fait que l'on peut spécifier à la fois le chemin des données sonores et le chemin des événements (début/fin, contrôle) dans le

[3] http ://www.quasimodo.org

même document.

MAX est devenu au cours du temps une interface contrôlant un système temps réel. Pendant ou après la construction du son, il est possible de contrôler en temps réel les paramètres de chaque algorithme, et de changer ainsi le son produit. Les premières versions contrôlaient un système qui se trouvait sur une autre machine spécialisée dans le traitement de signal. Aujourd'hui la dernière incarnation de MAX, jMax, fonctionne sur des machines de bureau. jMax se compose d'un système temps réel de génération sonore appelé FTS [65], et d'une interface écrite en Java qui communique avec FTS au moyen de sockets ou de mémoire partagée [25]. jMax est le plus proche des systèmes qui réunit les caractérisiques mentionnées plus haut. Cependant, le fait qu'il ne fut disponible que tardivement et le fait qu'il disposait de peu de documentation m'ont conduit à développer mon propre système.

### 5.1.3 Systèmes IHM

Les systèmes de synthèse mentionnés jusqu'ici sont destinés à être utilisés dans un but musical. Il existe des outils qui permettent l'intégration de sons dans les applications interactives. S'ils permettent soit la création, soit l'utilisation, ils ne proposent pas toutes les caractéristiques enumérées plus haut.

#### Kyma

Kyma est un environnement de conception sonore écrit en Smalltalk [71]. Il nécessite du matériel spécialisé pour effectuer de la synthèse en temps réel et n'est disponible que sur Macintosh. Il se présente sous la même forme que MAX, c'est-à-dire une interface graphique permettant de connecter des modules et de les contrôler avec des widgets ou des événements MIDI. Kyma va plus loin que MAX en ce qui concerne les algorithmes de génération de sons. Kyma utilise du matériel externe composé de DSP spécialisés dans le calcul intensif. Les patch sont convertis en microcode exécutable par un de ces DSP. Le microcode est modifié en temps réel quand une des parties du graphe composant le son ne contribue plus au son final. Ceci n'est possible que si l'on peut connaître à l'avance la durée d'un son.

Kyma a été utilisé pour auraliser des données [70]. Lorsqu'un concepteur utilise ce type d'interface pour présenter des données multidimensionnelles, il se trouve confrontés à des problèmes récurrents. Kyma fournit plusieurs outils qui permettent d'auraliser les données facilement :

- le décalage, qui interprète un flux de données sous forme d'onde acoustique, éventuellement en décalant les fréquences pour qu'elles soient audibles. Le décalage est aussi utilisé pour produire des flux de données servant à contrôler d'autres paramètre ;

- la correspondance, qui permet à des flux de données de contrôler des paramètres du son comme la fréquence, la fréquence de base, l'amplitude, des filtres, etc ;
- les combinaisons, qui combinent les flux sonores en les additionnant, en les soustrayant, ou en les multipliant ;
- les comparaisons, qui jouent deux signaux différents à gauche et à droite et permettent de les comparer ;
- les marqueurs, qui permettent à des sons d'être joués quand une valeur particulière d'un des flux de données est atteinte ;
- les histogrammes, qui représentent des classes de données par des fréquences, et leur magnitude par leur amplitude. Le résultat sonne comme un timbre qui varie dans le temps, ou comme un accord.

Ces techniques généralisent les techniques qui ont été précédemment abordées par les chercheurs en auralisation. Le système peut proposer d'autres stratégies, comme la modulation d'amplitude qui permet de comparer deux flux très proches.

Malheureusement, Kyma nécessite du matériel spécialisé pour pouvoir produire des sons. Utiliser du matériel spécialisé a trois avantages. Premièrement, comme le matériel est prévu pour faire de la synthèse, il est souvent plus efficace que les machines de bureau pour ce genre d'activité. Deuxièmement, les algorithmes de synthèse n'utilisent plus de temps de calcul sur la machine cliente. Troisièmement, les contraintes temps réel dues au partage des ressources par les processus ne sont plus de mise. En revanche, utiliser du matériel spécialisé limite les interfaces à des prototypes ne sortant pas du cadre du laboratoire, le déploiement est fortement limité. Il convient de noter cependant que de plus en plus, les machines de type PC peuvent être livrées avec des puces sonores, qui proposent généralement une panoplie d'instruments échantillonnés et peu contrôlables mais aussi des algorithmes "connus" comme la réverbération par example. Utiliser ces algorithmes permet de limiter les ressources nécessaires au système de synthèse.

### ENO

ENO est un système client/serveur qui propose des icones auditives paramétrées et synthétisées en temps réel à des développeurs d'interface [8]. Les sons sont décrits avec des objets sur lesquels on applique des interactions. Les objets sont placés dans un plan horizontal à deux dimensions face à l'utilisateur. De plus, on peut définir un groupe d'objets pour que leur sons soient produits séquentiellement, simultanément ou qu'ils se comportent comme un objet qui rebondit. Par exemple, en combinant des groupes parallèles et des groupes "à rebonds", il est possible de créer des sons d'objets qui se brisent.

ENO n'est pas un outil de conception d'icones auditives. Il ne permet pas de construire des algorithmes qui produisent des sons. Cependant, avec la

notion de “groupe”, ENO propose des comportements temporels prédéfinis. Malheureusement, il n’est pas possible de définir facilement des comportements particuliers. Il faut pour cela jouer avec des *timers* du système client, ce qui est une contrainte pour le programmeur et qui peut être difficile si l’on veut faire de la composition précise de sons. L’alternative consiste à ajouter ce comportement dans ENO, en modifiant ses sources, ce qui augmente la durée du cycle conception/test. De plus ENO est limité à des sons prédéfinis, en inclure de nouveaux passe là aussi par la modification des sources.

ENO permet en revanche l’intégration aisée d’icones auditives dans les applications interactives. En effet, une librairie écrite en C et surtout une extension de Tk permet de développer facilement des prototypes d’application utilisant les icones auditives proposées. De plus, le modèle structuré suivant l’approche écologique de la perception à base d’objets, d’interactions sur ces objets, de groupes à héritage d’attributs et de positionnement 2D permet au concepteur d’interface d’utiliser intuitivement les sons.

## 5.2 Sofa

Constatant que les systèmes diponibles n’avaient pas toutes les caractéristiques dont nous voulions disposer, nous avons créé notre propre outil de conception et d’intégration d’icones auditives. La première partie de cette section concerne les principes utilisés par cet outil. La deuxième partie développe les moyens que cet outil met en œuvre pour synthétiser des sons en temps réel. Ensuite, nous décrivons comment les sons peuvent être construits et contrôlés à l’aide de scripts. La quatrième partie décrit comment les sons conçus avec l’outil peuvent être intégrés dans les applications. Enfin, nous abordons les performances du système dans la dernière partie.

### 5.2.1 Principes

Sofa [4] a été développé pour être un outil de conception et d’intégration d’icones auditives à base d’algorithmes de synthèse ad-hoc. Sofa ne sert pas uniquement à la conception des icones, ni uniquement à l’utilisation des icones. En permettant la conception *et* l’intégration, le but est de fournir un outil qui facilite le parcours du cycle conception-réglage-test. Lorsque des concepteurs d’interfaces décident d’intégrer des icones auditives dans une application, ils ne disposent pas forcément des types d’icones auxquelles ils ou les utilisateurs ont pensé. Comme il n’existe que peu d’icones auditives paramétrées, il y a de fortes chances pour qu’ils soient contraints de concevoir leurs propres icones auditives, et de les tester immédiatement dans un contexte d’utilisation. Durant la phase de prototypage, il faut qu’un outil permette la création et le règlage rapide d’icones auditives. De même, il

[4] acronyme de *SOund FActory*

| gens | filtres | ops | divers |
|---|---|---|---|
| valeur | passe-bas | plus | enveloppe |
| bruit | passe-haut | mult | delay |
| rtsin | passe-bande | multiadd | barrière |
| osc | rejette-bande | | composé |
| sinemap | resonnant | | prog |
| | | | entrée/sortie |
| | | | ladspa |
| | | | test niveau |

TAB. 5.1 – Les modules de Sofa.

est nécessaire que l'intégration de ces icones se fasse de la manière la plus transparente possible. S'il est difficile d'intégrer les icones auditives, ou en d'autres termes, d'utiliser ce qui a été créé par l'outil de conception, il y a de fortes chances que des idées issues des phases de brainstorming et de prototypages ne soient pas testées par manque de temps ou de volonté.

A l'instar de MAX, Sofa inclut un éditeur graphique de graphe de signal. Il permet de connecter les entrées et les sorties de "boîtes" qui symbolisent un traitement sonore. Bien que n'étant pas aussi complet que peuvent l'être CSound et jMax, Sofa propose une large variété d'algorithmes. Les algorithmes nécessitant une trop grande puissance de calcul (transformée de Fourier inverse, synthèse granulaire...) n'ont pas été intégrés. Lorsque le besoin s'en fera sentir et que la puissance des machines le permettra, ces algorithmes seront intégrés.

Les algorithmes disponibles sont (tableau 5.1) :

**des générateurs :** Ce sont des algorithmes qui génèrent les échantillons destinés à être envoyés au matériel de production sonore. Parmi les générateurs, on trouve des ocillateurs avec des formes d'onde variées (sinusoïde, triangle, scie, fichier d'échantillons), une fonction aléatoire pour produire du bruit blanc, un module "rtsin" qui produit une sinusoïde sans table d'échantillons, et le module "sinemap" qui permet d'appeler récursivement une fonction sinus. Un module "value" permet de préciser la valeur d'une constante que l'on peut connecter aux entrées d'autres modules. On peut changer la valeur de la constante par un champ de texte ou s'aider par manipulation directe d'un potentiomètre linéaire ("scale") (figure 5.1).

**des filtres :** passe-bas, passe-haut, passe-bande, rejeteur de bande, résonnant, pour transformer un signal.

**des opérateurs arithmétiques :** l'addition, la multiplication, l'addition multiple.

**et divers algorithmes :** un générateur d'enveloppes à base de segments

de droites et son éditeur associé, un “sample-and-hold”, une ligne à retard, un module de composition, un module d’expression arithmétique. Un module permet d’utiliser les *plug-ins* respectant la norme *ladspa* [5], Sofa peut donc être étendu avec des modules tiers, comme par example une très bonne réverbération [6].

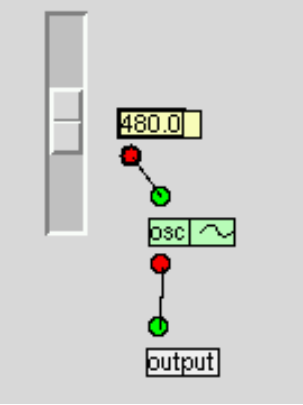


FIG. 5.1 – Exemple de patch. Il est composé d’un oscillateur sinusoïdal dont la fréquence est contrôlée par une widget “scale”

Les algorithmes de génération et transformation sont codés en C++. Chacun dispose de plusieurs entrées et sorties. Les sorties sont matérialisées par des tampons (“buffers”) appartenant à une instance d’un algorithme. L’interface utilisateur permet de connecter les entrées d’un module aux sorties d’un autre module, c’est-à-dire que les modules “recevant” un signal utilisent le contenu du buffer de sortie correspondant. Lorsqu’un son est joué, une liste de modules à exécuter est générée à partir du graphe de modules connectés. A chaque cycle d’exécution, la liste est parcourue et chaque algorithme est exécuté. En partant du module de sortie, l’algorithme de génération de la liste produit une liste ordonnée correctement, c’est-à-dire que les modules les plus en aval sont exécutés à chaque cycle avant les modules les plus en amont. Lorsqu’il y a un cycle dans le graphe, il y a un décalage d’un cycle d’exécution inévitable pour les entrées des modules concernés. La liste de modules ne contient pas de modules “valeur”, car ceux-ci sont modifiés explicitement par l’utilisateur. Il est donc inutile de les recalculer à chaque cycle.

En dehors des modules de génération et de transformation de signal, sont proposés des modules qui permettent la conception et l’utilisation de sous-modules, la définition de paramètres externes, et l’optimisation de calculs. Il

[5] linux audio developer software plug-in architecture, http ://www.ladspa.org
[6] freeverb, http ://www.dreampoint.co.uk

est ainsi possible de définir des réseaux qui peuvent être réutilisés en tant que "sous-sons" par des sons plus complexes. Ainsi, il est fréquent d'utiliser un filtre en peigne dans certains algorithmes, que l'on peut concevoirà l'aide des algorithmes de Sofa. Cet algorithme a besoin d'un signal d'entrée à filtrer, ainsi que deux paramètres contrôlant un gain et la durée d'un délai. Pour pouvoir accéder aux trois entrées de ce filtre quand il est utilisé dans un son, il faut pouvoir définir lesquels parmi les différents modules qui le composent sont des entrées de ce filtre. En insérant un module "iovalue", on spécifie que tout les modules qui sont en amont par rapport à ce module ne font pas partie du son final. La partie du réseau qui est en amont du module "iovalue" sert à tester le patch, et n'est pas prise en compte lorsque le patch est utilisé comme "sous-son". Cette partie est généralement composée d'un module "value" ou d'un oscillateur. Par exemple, dans la figure 4.8, les modules "value" situés au-dessus des modules "iovalues" "length" et "shape" ne sont présents que lorsque l'on édite le patch du son de vague. De plus, avec un module "iovalue", on spécifie le nom de l'entrée ainsi définie. Les noms sont généralement significatifs de la fonction de ces paramètres. Dans l'exemple d'une icone auditive, on pourra stipuler la longueur d'une vague par exemple (figure 4.8). Une fois défini un son, si celui-ci a des paramètres d'entrée extérieure, on peut l'utiliser en tant que sous-son dans un son plus complexe. En utilisant le module "compound", et après avoir précisé le nom du sous-module désiré, les entrées correspondant aux "iovalue" du sous-son apparaissent (figure 5.2).

FIG. 5.2 – Son composé. Le son d'une vague est contrôlable par ses paramètres de longueur et de forme (le texte associé à une entrée n'apparaît que lorsque le curseur la survole).

D'autre part, si l'on peut utiliser les modules d'opérations arithmétiques pour construire une expression, dès que celle dernière devient complexe, il est plus adéquat d'utiliser son expression textuelle. Les opérations arithmétiques fournies peuvent être utilisées pour le signal audio, c'est-à-dire que l'on peut additionner deux sinusoïdes par exemple. Cependant, ce type d'opération est utilisé aussi pour des signaux de contrôle. Par exemple, si l'on veut produire une sinusoïde dont la fréquence est définie par l'expression $freq = infreq*2+440$ (infreq est le paramètre d'entrée), il serait dommage d'utiliser

plusieurs opérations sur le signal audio si l'on sait que le paramètre est contrôlé par l'utilisateur et donc changé peu fréquemment. En utilisant un module "prog", on peut définir des expressions qui prennent un nombre d'entrées quelconque et produisent une valeur de sortie quand l'une de ces entrées changent. Ainsi, le système de synthèse n'utilise qu'un seul module pour produire la valeur finale, et ne la calcule que si l'utilisateur change une entrée.

Pour définir une expression, il suffit d'en donner la description textuelle. Le système analyse le texte et produit autant d'entrées qu'il y a de symboles lexicaux qu'il ne connaît pas. Par exemple, pour produire des valeurs de fréquences en fonction d'un index et d'une fréquence de base (comme pour le spectre des objets frappés de Gaver), on écrira $base_frequency * pow((2 * index + 1), 2)/9$ ce qui donnera une module avec les deux entrées $base_frequency$ et $index$. Il existe une variable réservée qui est la fréquence d'échantillonage ($audio_frequency$) : on peut ainsi obtenir le nombre d'échantillons en fonction d'une durée en seconde en multipliant la durée par $audio_frequency$. Cette expression est souvent utilisée lorsque l'on veut préciser la durée d'un délai (figure 5.3).

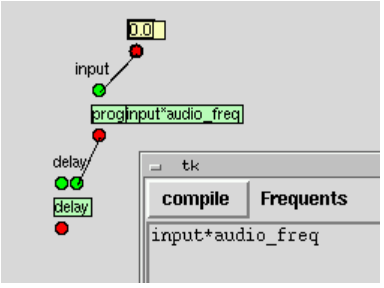


FIG. 5.3 – Exemple de programme et de l'éditeur. Ce programme permet de contrôler la durée du délai avec une mesure en millisecondes plutôt qu'avec un nombre d'échantillons. Les entrées du programme correspondent aux variables non reconnues par l'analyseur (ici, "input").

### 5.2.2 Temps réel

Synthétiser un son consiste à produire des échantillons pour une certaine durée et les passer au matériel de reproduction sonore. Quand le matériel a terminé de jouer les échantillons, il faut en produire de nouveaux et les passer au matériel. Si l'on utilise un algorithme simple qui consiste à utiliser un seul tampon d'échantillons, il faut produire des échantillons dans un temps inférieur à la durée correspondant à un seul échantillon. Cette contrainte étant trop forte, on utilise l'algorithme du double tampon : pendant qu'un tampon est traité par le matériel, l'autre est rempli par le système logiciel.

Quand le matériel a fini de jouer un tampon, il prévient le système logiciel qui lui passe le tampon qu'il vient de calculer, et lance la synthèse sur le tampon inutilisé. Pour synthétiser un tampon, le système dispose maintenant d'une durée équivalente à celle de la longueur du tampon.

Cependant, une fois calculé un tampon n'est plus modifiable. Cela signifie que les modifications effectuée par un système extérieur, que ce soit un utilisateur ou un programme client, ne seront audibles que pour les tampons qui n'ont pas encore été synthétisés. Il existe donc une *latence* entre le moment ou un événement de contrôle a été généré et le moment ou l'effet de ce contrôle est suceptible d'être entendu. Cette latence est égale au maximum à deux fois la longueur d'un tampon. Plus la longueur d'un tampon est petite, plus vite les modifications seront entendues. Si la latence est trop grande, un utilisateur peut percevoir un décalage entre ses actions et ce qu'il entendra. La latence de contrôle est primordiale pour un système destiné à la musique, où la performance de l'instrumentaliste est aussi importante que le son lui-même, et où la virtuosité de l'artiste ne peut souffrir les retards même minimes d'un système logiciel. Si elle est moins importante dans les applications interactives, surtout si l'on veut suivre des activités d'arrière-plan, elle reste indispensable à la sensation de manipuler un système réel lorsque le son est utilisé comme feedback des actions. Ainsi, un utilisateur percevra une latence de 100 ms entre le mouvement d'une icone et un son de frottement, ce qui diminuera la sensation d'implication dans la métaphore.

On peut penser qu'il suffit de diminuer la taille d'un tampon pour éviter ce genre de problème. Cependant, la longueur du tampon définit aussi la période à laquelle le système logiciel de synthèse doit pouvoir calculer le son. Comme les icones auditives font partie d'une interface d'autres logiciels, cela signifie que plusieurs programmes tournent sur la même machine. Un système multitâche comme Unix gère une liste de programmes et leur accorde tour à tour un montant de temps durant lequel ils peuvent s'exécuter ("switch"). Si le logiciel de synthèse n'a pas la main assez vite, c'est-à-dire que la période entre deux sélections par le système d'exploitation est plus grande que la longueur du tampon, le matériel de production sonore aura consommé le tampon précédent sans que le logiciel de synthèse n'ait eu le temps de calculer les prochains échantillons. Il s'agit ici d'un autre type de latence, introduit par les systèmes d'exploitation multitâches. Sur un système comme Windows NT ou Linux, les pics de latence sont de l'ordre de 100 ms. Sur un système comme MacOS 9, les pics observés sont de l'ordre de 10 ms. Il extiste un patch pour Linux qui permet de descendre à des latences de l'ordre de 3 ms. Le patch est en cours d'intégration dans les sources officielles du noyau.

Il existe donc deux types de latence, la latence de contrôle, et la latence de "switch". La latence de contrôle est bornée par le bas par la latence de switch. Il faut donc faire en sorte que le système d'exploitation donne la main au système de synthèse plus souvent qu'aux autres processus pour diminuer

cette latence. Sous Linux, Sofa utilise un *thread* séparé pour synthétiser les sons. Généralement, les *thread*s sont utilisées quand il n'y a pas de choix dans les priorités à accorder aux différentes tâches d'un processus. Dans Sofa, il y a deux tâches : l'une consiste à gérer les actions de l'utilisateur ou d'un programme client, l'autre à calculer les algorithmes de synthèse. La tâche de calcul est prioritaire car l'absence de clics est primordiale pour un système sonore. En accordant un *thread* pour la tâche de calcul, on est sûr qu'elle sera appelée quand le matériel aura besoin d'un nouveau tampon. De plus, toujours sur Linux, on peut configurer un *thread* pourqu'il soit prioritaire par rapport à tous les autres *thread*s gérés par le système. Ainsi, si Sofa a les droits d'exécution nécessaire (avec le bit suid mis à un par exemple), le *thread* de calcul est lancé avec les attributs SCHED_FIFO et une priorité maximale. Ainsi, dès que le matériel a fini de traiter un tampon, c'est le *thread* de synthèse qui sera exécuté parmi tous ceux existants.

Evidemment, si les calculs à exécuter sont trop importants, il est possible que ces calculs prennent plus de temps que la durée d'un tampon. Pour éviter ce phénomène, Sofa propose le module "prog" vu plus haut qui limite le nombre de calculs d'expressions arithmétiques. De plus, il est possible de faire passer la sortie d'un son dans un module qui teste le niveau sonore du son produit, et l'arrête si celui-ci est insuffisant. Ce module permet de gérer des sons comme les impacts de Gaver, dont on ne connaît pas la fin à priori mais dont on sait qu'ils produisent du silence au bout d'un temps indéfini. Ainsi, si plusieurs sons d'impacts sont lancés, ils seront arrêtés automatiquement dès qu'ils ne produiront plus des échantillons audibles. Enfin, grâce à des algorithmes de scripts qui peuvent contrôler le réseau de modules, un concepteur peut explicitement couper une branche de réseau inutile. Cette caractéristique est décrite dans la section suivante.

Généralement, pour des raisons de performances, les systèmes de synthèse proposent deux "qualités" de calcul pour les algorithmes. Ceux qui ont une forte variation dans le temps susceptible d'être perçue sont calculés avec une fréquence égale à la fréquence d'échantillonnage souhaitée. D'autres, comme les sinusoïdes à faible fréquence (LFO, "low frequency oscillator" par exemple) ne varient pas suffisamment dans le temps pour que l'on puisse entendre une différence si leur sortie était calculée à une fréquence d'échantillonnage plus petite. En calculant moins fréquemment ces algorithmes, on ne perd pas en qualité, mais on gagne en performance. Les algorithmes concernés sont des algorithmes qui en contrôlent d'autres, comme une enveloppe d'amplitude par exemple, et sont calculés à une fréquence dite de "contrôle". Cette caractéristique n'a pas encore été implantée dans Sofa. En revanche, le module "prog" appartient à une autre classe de contrôle, celui qui appartient à l'utilisateur ou au client. Les calculs de cette classe n'ont pas à être exécutés régulièrement comme peuvent l'être les algorithmes calculés à la fréquence d'échantillonnage ou à la fréquence de contrôle. Ils ne sont exécutés qu'en cas de changement de la part de l'utilisateur ou du

client, ce qui arrive à une fréquence encore plus faible que celle de contrôle.

Sofa n'utilise pas de système automatique de gestion de ressources. Plusieurs stratégies ont été envisagées dans ENO : le cache de sons calculés et la synthèse adaptive, qui consiste à synthétiser les sons à une fréquence d'échantillonage réduite [8]. Remarquons que le cache ne marche qu'avec des sons se terminant et n'est pas généralisable à des sons continus. La synthèse adaptive peut s'aider de la localisation des sons : les sons les plus éloignés sont moins sensibles à une perte de qualité que les sons les plus proches. D'autres systèmes proposent de dégrader la qualité des algorithmes mis en jeu, mais ils semblent difficilement adaptable aux algorithmes utilisés ici [32].

### 5.2.3 Scripts

Comme nous l'avons vu dans le chapitre précédent, les icones auditives sont contrôlées par des paramètres de haut-niveau, qui correspondent à des paramètres décrivant la source ou l'interaction. Par exemple, le son du vent est contrôlé par un attribut "force". Après avoir défini un son, il convient de pouvoir le contrôler par ces paramètres plutôt que par les valeurs des modules "valeur" faisant partie du réseau de modules. Généralement, un paramètre a une influence sur tout ou partie de l'ensemble des valeurs effectivement contrôlables. Parfois, à un paramètre correspond exactement une valeur, et la fonction transformant la valeur du paramètre en valeur de l'algorithme est suffisamment simple pour être exprimable sous forme d'expression arithmétique soit à l'aide du module "prog", soit avec une combinaison de modules arithmétiques. Cependant, il se peut qu'un paramètre de haut-niveau ait une influence sur plusieurs paramètres de l'algorithme, et que cette influence ne soit pas aisément exprimable par les même moyens. Ainsi, le matériau d'un objet a une influence à la fois sur la façon dont les fréquences s'atténuent au cours du temps et sur la largeur de bande des filtres passe-bande. Ces dépendances sont plus facilement exprimable sous forme de programme textuel, décrit avec un langage de programmation. De plus, il se peut qu'un paramètre de haut-niveau ait une influence indirecte sur les paramètres de l'algorithme. Ainsi, la force du vent a une influence sur les *domaines* de valeur des fréquences centrales du filtre passe-bande, ce qui est également difficilement exprimable sous forme visuelle.

Afin de faciliter la description des fonctions transformant les paramètres de haut-niveau en paramètres des algorithmes de synthèse sonore, et afin de permettre de décrire des comportements non-triviaux du son, Sofa permet de manipuler les modules et leurs connexions à l'aide d'un langage de script. Ainsi, on peut associer une classe à un patch qui permet de contrôler le comportement du son décrit par le patch. La classe est une sous-classe d'une classe prédéfinie "SofaSound", commune à tous les sons définis avec Sofa. Lorsqu'un son est créé, un objet de la classe correspondante est créé et contient les informations sur le patch lui correspondant. La classe So-

faSound a des méthodes d'initialisation, de finalisation, et des méthodes "start" et "stop" qui sont appelées lorsqu'un son est démarré ou arrêté. En associant une classe à un son, il est possible de redéfinir ces méthodes afin de programmer les comportements propres à ce son.

### Python

Le langage de script utilisé dans Sofa est Python [7] Il dispose d'un ensemble de structures de données comme les chaînes de caratères, les listes et les tableaux associatifs. Python est un langage qui permet la programmation "orientée objet" et utilise les concepts de classes, méthodes et instances. Cependant, le langage n'est pas typé : lors d'un appel de méthode ou d'une référence d'un champ d'un objet, il effectue d'abord une recherche pour voir s'il existe une telle méthode ou un tel attribut pour l'objet considéré. Lorsque la méthode ou l'attribut n'existe pas, une exception est levée. Python n'est pas réellement un langage interprété. Il est d'abord compilé en "byte-codes" qui sont ensuite interprétés par une machine virtuelle. Un nombre important de modules variés étendent le langage. Parmi ceux-ci, le module Tkinter permet de construire des interface à l'aide de Tk, la boîte à outils d'interfaces graphiques de Tcl/Tk [8]. Python est écrit en C et est disponible sur de nombreuses plateformes (Unix, MacOS, Windows, BeOS...). Il peut être étendu avec des modules écrits en C.

Les langages de script, s'ils sont plus lents que les langages compilés, sont surtout utilisés pour le "code glue", c'est-à-dire le code qui fait interagir des modules orthogonaux comme la gestion d'une base de donnée, une interface graphique, et un serveur Web par exemple. Le code qui a besoin de la vitesse d'exécution d'un langage compilé peut être écrit en C et être utilisé sous forme de module d'extension. En fait, l'utilisation d'un langage de script comme Python a trois avantages [62]. Premièrement, le fait qu'il ne soit pas typé n'en fait pas un langage "jouet", mais permet le développement rapide de prototypes, ce qui reste l'objectif du concepteur d'interfaces. Deuxièmement, un langage de script permet généralement la modification du programme pendant le contexte d'utilisation, c'est-à-dire sans "relancer" le programme principal. Cette caractéristique est utilisée par Sofa lors de la conception d'une icone auditive et son association avec une classe : lorsque le code source est modifié, il suffit de spécifier à Sofa qu'il est nécessaire de l'évaluer à nouveau. Dans le cas de Python, il se trouve que la recompilation du code source en byte-code est rapide. Son utilisation laisse donc l'interface fluide, c'est-à-dire que l'on n'hésite pas à modifier le corps d'une classe d'une icone auditive pour la tester. Troisièmement, Python est introspectif, et permet d'analyser facilement un code source Python. Ainsi, il est relativement facile d'extraire les variables d'une expression d'un module

[7] http ://www.python.org
[8] http ://dev.scriptics.com

"prog" et d'en faire des entrées de ce module, ou d'extraire les méthodes ajoutées à une classe d'icone auditive pour en faire une version "client". Nous avons choisi Python parce qu'il avait des constructions et les extensions que l'on souhaite trouver dans les langages d'aujourd'hui et qu'il était rapide. Tcl a été écarté à cause de sa syntaxe, et le fait qu'il ne permettait pas la programmation par objets. Les programmes Java nécessitent d'être compilés, et s'exécutent trop lentement sur les machines à notre disposition.

### Comportement de haut-niveau

Nous avons vu que le son du vent est un bruit blanc filtré par un filtre passe-bande. Pour simuler le souffle qui monte et qui descend, il faut configurer l'enveloppe qui contrôle la fréquence centrale du filtre. L'enveloppe est une ligne droite à pente positive ou négative selon que le souffle monte ou qu'il descend. Quand le son arrive à la fin de l'enveloppe, il faut inverser le sens de l'enveloppe. Pour cela, les objets de Sofa disposent d'un ordonnanceur ("scheduler") qui permet d'exécuter des actions à des moments choisis dans le futur. Quand le son du vent est démarré, il suffit d'ordonnancer une action qui change le sens de l'enveloppe au moment où celle-ci se termine. Pour changer l'enveloppe, il faut appeler les méthodes de l'objet correspondant au module "envelop" de ce son. Il faut donc pouvoir trouver cet objet à partir du réseau contenu dans l'objet son. Ceci est rendu possible par le fait que les modules du réseau peuvent être nommés par le concepteur dans l'interace graphique. Ainsi, en nommant le module d'enveloppe "sweep", on peut retrouver l'objet correspondant parmi tous les modules nommés en demandant l'objet nommé "sweep". Il suffit, une fois cet objet référencé, d'appeler les bonnes méthodes pour changer sa configuration. Le son du vent est donc défini par un appel à l'ordonnanceur pour changer de temps en temps la direction du souffle, et par une référence à un module enveloppe qui permet de changer la forme de l'enveloppe :

```
class wind(sofasound.Sound) :
    def __init__(self, models) :
        # initialisation de la classe parente
        sofasound.Sound.__init__(self, models)
        # 'models' est un dictionnaire
        # de modules indexé par le nom des modules
        self.envelop=self.models['sweep'] # récupère l'objet 'sweep'
        self.ms=2000 # période du changement de sens
    def start(self) :
        # ordonnancer une action
        self.after_id = self.scheduler.after(self.ms, self.generate_strength)
    def generate_strength(self) :
        ...
```

La période de changement de sens doit être changée à chaque fois que l'on change le sens du souffle. Nous avons vu que la période ainsi que les

domaines de fréquences parcourus par le souffle dépendent de la force du vent. Il faut donc tirer des valeurs au hasard qui correspondent à la force du vent à chaque fois que le souffle change de direction :

```
def compute_next(self) :
    # tirer les valeurs au hasard
    self.nextdur = whrandom.uniform(self.min_dur,self.max_dur)
    nextfreq = whrandom.uniform(self.min_freq,self.max_freq)
    # reconfigurer l'envelope
    self.envelop.move_point(0, 0, self.nextfreq)
    self.envelop.move_point(1, 1.0, nextfreq)
    self.duration.set_value(self.nextdur)
    self.nextfreq=nextfreq
```

Enfin, pour pouvoir contrôler le vent par sa force, et non par des paramètres de l'algorithme comme la fréquence de coupure, il faut ajouter une méthode de contrôle pour l'objet son du vent (méthode "set_strength") :

```
...
def set_strength(self, v) :
    self.strength = v
    self.compute_state(self.strength)
def compute_state(self, s) :
    self.min_dur = 0.5 + (1-s) * 0.5 # 0.5 ¡ mindur ¡ 1
    self.max_dur = 1 + (1-s) * 2 # 1 ¡ maxdur ¡ 3
    self.min_freq = .2 + s*.1 # 200 ¡ min_freq ¡ 300
    self.max_freq = .3 + s*.2 # 300 ¡ max_freq ¡ 500
...
```

Comme les paramètres du filtre sont calculés périodiquement, il n'est pas utile de les recalculer à chaque fois que l'on change la force du vent. On ne change donc que les domaines de valeurs possibles pour la prochaine durée et la prochaine fréquence (méthode "compute_state"). Ces valeurs sont respectivement inversement proportionnelle et proportionnelle à la force du vent.

L'utilisation de scripts facilite la conception d'icones auditives pour deux raisons. Premièrement, avoir un classe avec chaque type de sons permet d'écrire des méthodes de contrôle de haut-niveau typiques des icones auditives, comme la méthode "set_strength" du son du vent. Deuxièmement, l'utilisation d'algorithmes permet de contrôler le comportement du son, ce qui est difficile voire impossible lorsque l'on utilise un langage visuel.

Rappelons qu'avec ENO, il existe un moyen de spécifier les occurences temporelles d'événements sonores, en les incluant dans un "groupe" qui les joue séquentiellement, parallèlement, ou similairement à un rebond. Il s'agit de routines temporelles prédéfinies. Avec l'exemple du son du vent, on voit que l'aspect temporel des icones auditives peut être plus complexe. Afin de ne pas limiter le concepteur lorsque celui-ci doit manipuler les aspects temporels des icones auditives, il faut donner un outil plus général qu'un ensemble de comportements prédéfinis. C'est l'ordonnanceur qui joue ce rôle

dans Sofa. La présence d'un ordonnanceur permet de décrire des comportements temporels particuliers qui peuvent se trouver dans un seul son, comme celui du vent. Cependant, il est possible de l'utiliser pour fournir une librairie de comportements prédéfinis, comme le séquencement régulier, ou le séquencement à la manière d'un rebond. Ainsi, la classe "bounce" permet d'exéctuer des actions à des temps simulant les rebonds d'un objet :

```
class bounce :
  def __init__(self, scheduler, action, initial_height, elasticity=.5, max_bounce_count=5) :
   ..
```

L'initialisation d'un objet "bounce" prend en paramètre l'ordonnanceur, l'action à exécuter à chaque rebond et la hauteur initiale. Des paramètres optionnels permettent de spécifier l'élasticité et le nombre maximum de rebonds. On peut l'utiliser ainsi :

```
import sofasound
from Tkinter import *
import bounce
class testbound(sofasound.Sound) :
    def __init__(self, models) :
        sofasound.Sound.__init__(self, models)
        self.envelop=self.models['envelop']
        self.duration=self.models['duration']
        # création d'un objet bounce avec un ordonnanceur,
        # l'action à appeler à chaque rebond,
        # la hauteur initiale, et le nombre maximum de rebonds
        # on peut aussi préciser l'élasticité
        self.bounce = bounce.bounce(self.scheduler, self.action,
            .4, max_bounce_count=5)
    def start(self) :
        self.envelop.reset()
        # départ du rebond
        self.bounce.start()
    def action(self, num_bounce) :
        # à chaque rebond, l'enveloppe recommence au début
        self.envelop.reset()
```

La conception d'icone auditive se fait par étapes et raffinements successifs. Elle contient une partie "design", mêlant technique, sens artistique et préférences personnelles. Pour tester facilement des règlages particuliers, le concepteur doit utiliser des outils graphiques comme les échelles de valeur pour affiner certains paramètres. Ainsi, pouvoir tester différentes valeurs limites du domaines de fréquence pour le son du vent en évitant de changer le texte du programme à chaque itération est pratique. Comme Python inclut les widgets de Tk, Sofa permet aux objets de son de les utiliser lorsque Tk est effectivement présent dans le contexte d'exécution. Ainsi, après la construction de la partie objet de la classe parente "sofasound", une champ

membre de l'objet contient un point d'accès à la hiérarchie de widgets Tk. Il est donc possible de disposer de toutes les widgets Tk pour contrôler un son, notamment les échelles de valeur et les boutons. Ainsi, pour tester le son du vent à des forces différentes, on peut introduire une widget qui contrôlera la force :

```
class wind(sofasound.Sound) :
    def __init__(self, models) :
        ...
        if(self.master) : # si Tk est présent, faire une widget de test
            self.top = Toplevel(master)
            self.strength=.5
                self.scale = Scale(self.top, from_=0, to=1, resolution=.01,
command=self.set_strength_by_scale)
            self.scale.pack()
    def set_strength_by_scale(self, v) :
        self.set_strength(string.atof(v))
    ...
```

Tk n'est pas toujours accessible par les objets-sons. En fait, il est possible de dire que les widgets ne doivent pas s'afficher, ce qui est indispensable lors de l'utilisation d'un son par une application cliente.

### Réorganisation du réseau

Si le nommage des modules et les programmes de script permettent de contrôler la façon dont se comporte le son, l'accès au réseau définissant le patch permet aussi de le réorganiser. La réorganisation du patch permet l'ajout et le retrait de sons ou d'effets sonores. Cette caractéristique est utilisée par le son du bouillonnement. Le son de bouillonnement est un flux de bulles dont les paramètres sont la viscosité et la grosseur. En jouant sur la période d'occurence des bulles dans le temps, on peut simuler une activité plus ou moins grande. En contraignant les paramètres des flux à des domaines de valeurs dont les intersections deux-à-deux sont vides, on peut surveiller plusieurs flux en même temps, chacun ayant une activité et une viscosité propre. Cependant, il est inutile de créer des flux inutilisés par défaut. Un concepteur peut vouloir ajouter et retirer des flux à sa guise. La méthode qui implante ce comportement ajoute au réseau existant une sous-partie qui n'est pas créée interactivement, mais par l'intermédiaire du script :

```
...
    def add_stream(self, minsize, maxsize) :
        bubble = self.add_compound("bubble.sofa")
        vsize = self.add_value(username="size", value=minsize)
        vviscosity = self.add_value(username="viscosity")
        self.connect(vsize.output, bubble.get_input("size"))
        self.connect(vsize.output, bubble.get_input("viscosity"))
     conn = self.connect(bubble.output, self.multiadder.get_input("input"))
```

```
        number = self.new_stream_number()
        activity = .5
        self.bubblestreams[number] =
          bubblebtream(bubble, vsize, vviscosity, activity, minsize, maxsize)
    def set_activity(self, streamnumber, value) :
        self.bubblestreams[streamnumber].activity=value
...
```

Si l'un des flux n'est plus utilisé, il peut être simplement retiré, ou voir sa connexion au réseau global détruite. Dans les deux cas, la réorganisation résulte en un patch demandant moins de puissance de calcul. Il faut bien entendu sauvegarder dans une variable l'objet représentant la connexion pour être capable de la détruire.

Le son de bouillonnement peut-être accompagné d'un son de grésillement synonyme d'alerte ou de dépassement de valeur limite. Un tel son peut être synthétisé comme nous l'avons vu avec un bruit blanc dont l'amplitude suit une enveloppe tendant vers zéro. Il suffit pour cela de rétablir la connexion entre cette partie du réseau et le réseau global.

Le son de grésillement peut être considéré comme un son supplémentaire au son de bouillonnement. Il pourrait donc être directement déclenché par l'application cliente. Cependant, les icones auditives doivent être manipulées par des attributs de haut-niveau. Ici, l'apparition du grésillement fait partie intégrante de l'icone auditive. Déclencher un grésillement doit donc être le résultat d'un appel de méthode sur un son de bouillonnement ("bouillonnement-¿grésille()"), et non un son créé, déclenché et géré à part par l'application cliente.

Enfin, pouvoir configurer le réseau permet d'ajouter des enveloppes associées à des paramètres que l'on peut quantifier. Ceci permet de simuler une transition progressive d'une valeur vers une autre. Ainsi, configurer la vitesse de la machine peut se faire en appelant une méthode "set_speed(value)" d'un son de machine. Or, dans la vie réelle, il est rare qu'une machine passe brusquement d'une vitesse à une autre. En proposant une valeur temporelle supplémentaire à la méthode "set_speed(value, delay=1000)", on peut préciser le temps que mettra la vitesse à atteindre la valeur spécifiée dans l'appel de méthode (1000 ms dans l'exemple). Pour mettre en œuvre ce comportement, on peut créer une enveloppe, la configurer selon les valeurs passées en paramètre et la valeur précédente de la vitesse, et la connecter au module "iovalue" "vitesse". Le module "iovalue" sert donc non-seulement de paramètre d'entrée quand le son est utilisé dans un son composé, mais aussi de module de jonction qui aiguille le signal vers tous les modules qui se servent de ce paramètre. De même, il est possible de simuler le démarrage d'une machine en utilisant l'enveloppe pour la vitesse, et une autre enveloppe pour l'amplitude du son résultant. En démarrant, la machine est silencieuse et devient plus bruyante quand elle se met en route. Enfin, en faisant bou-

cler l'enveloppe de vitesse de façon à ce qu'elle donne l'impression que la machine redémarre, on peut simuler un incident, une panne de la machine par une méthode "break_it" par exemple.

La simulation de comportement est une des clefs de la conception d'icones auditives. En effet, l'approche écologique permet de considérer les spectres comme vecteurs d'information, mais les comportements des sons ont un rôle aussi important à jouer dans la transmission d'informations. En donnant accès à un ordonnanceur et en permettant la référence de modules ainsi que la réorganisation du patch, Sofa rend possible ces comportements, et étend donc le spectre d'icones auditives concevables. De plus, ces comportements sont adaptables à d'autres sons. Par exemple, un ascenseur, comme une machine, peut être configuré avec une enveloppe pour simuler une accélération ou pour simuler un incident. En utilisant les capacités du langage de script pour réifier ces comportements, il est possible de les appliquer à icones auditives différentes pour lesquelles ces comportements ont une signification. Ainsi, la classe "bound" vue plus haut est une réification du comportement de rebond.

### Scripts et performance

Sofa lui-même est écrit en Python, et utilise Tkinter pour son interface graphique. Rappelons que les langages de script ne sont pas trop lents s'ils sont utilisés comme "code-glue". Ainsi, il serait inapproprié de calculer des algorithmes de synthèse à l'aide d'un programme écrit en Python. Nous avons donc écrit des algorithmes en C++, que Sofa utilise lors de la définition et de l'utilisation des icones auditives. Lorsque l'on construit un son, les algorithmes C++ sont connectés entre eux. Le processus de synthèse n'utilise que ce réseau d'algorithmes C++. Les programmes en Python représentent 8000 lignes de code, et les algorithmes en C++ 3000 lignes de code.

Les scripts décrivant les icones auditives ne sont exécutés que lorsqu'une méthode d'un objet est invoquée. Seules l'interface utilisateur optionnelle comme l'échelle qui permet de contrôler la force du vent ou les applications clientes appellent ces méthodes. Lorsqu'elles sont appelées, il s'agit de méthodes qui permettent de contrôler un algorithme, elles ne nécessitent donc pas d'être exécutées très rapidement. Lorsque certaines icones auditives nécessitent un contrôle rapide pour du feedback comme le son de frottement de Gaver par exemple, il est possible que les algorithmes de scripts soient trop lents. Dans ce cas, il peut être nécessaire de coder directement l'algorithme en C++, et l'ajouter à la liste des algorithmes de Sofa.

Le nombre de calculs nécessaires à chaque icone auditive dépend de plusieurs paramètres. Premièrement, la façon dont sont codés les algorithmes a une forte influence, surtout que les calculs à l'intérieur de la boucle pour un tampon sont appelés à chaque échantillon. Le nombre d'opérations élémentaires donne une idée des performances d'un algorithme, mais n'est pas le seul

critère influant sur les performances. Par exemple, il est possible que l'utilisation d'une table, notamment pour un oscillateur, produise des défauts de page dans le cache de premier niveau du processeur qui dégradent les performances. Souvent, les tables utilisées à but musical sont d'une taille de 8 Ko. Sachant que bon nombre de processeurs se trouvant dans les ordinateurs de bureau ont un cache de 128 Ko, on peut envisager un nombre maximum de 16 tables si on veut éviter les défauts de cache, et ceci sans compter les données des autres processus. Pour notre application, nous avons réduit la taille des tables à 1024 octets, taille que nous estimons suffisante pour les sons que nous utilisons. Un autre facteur influant sur la rapidité d'exécution tient aux options de compilation. Par exemple, il est non seulement possible de configurer le compilateur avec des degrés d'optimisation divers, mais il est aussi possible de spécifier au compilateur que les boucles doivent être déroulées, ou qu'il peut considérer que les calculs effectués sur des variables n'auront pas de conséquence sur les autres variables qu'il manipule ("anti-aliasing"). Sur des *Silicon Graphics*, ce type d'optimisation peut faire gagner un facteur trois en performance. Enfin, certains microprocesseurs génèrent des exceptions lorsque les calculs qu'ils effectuent peuvent engendrer des erreurs de précision. Ces exceptions sont rattrapées par le système et peuvent écrouler les performances si on ne configure pas le microprocesseur pour qu'il ne les génére pas. Ce genre d'erreurs de precision est fréquent lorsque l'on travaille avec des filtres récursifs. De plus, nous avons implanté une technique de conception d'algorithme en C++ en utilisant les *templates* et les capacités de *mise en ligne* d'appels de fonction pour permettre à l'optimiseur du compilateur d'optimiser sur les calculs combinés d'algorithmes, plutôt que sur les calculs séparés. Cependant, les résultats que nous avons obtenus avec les algorithmes que nous avons conçus ne sont pas suffisamment probants pour justifier une utilisation systématique.

La complexité du graphe d'algorithmes influe aussi sur les performances. Plus il y a d'algorithmes à exécuter, et plus une icone auditive a besoin de ressources. Cependant, les graphes des icones auditives que nous avons décrites n'incluent pas un nombre élevé d'algorithmes, et sont tous à base d'enveloppe, d'oscillateur, de bruit et d'opérations arithmétiques simples. A titre d'exemple, 3 bouillonnements simultanés et un son de vent prennent en moyenne 11 % du CPU à une fréquence d'échantillonnage de 32 Khz sur un processeur de type Intel Celeron à 400 MHz avec 128 Ko de cache et sur un système Linux 2.2.

### 5.2.4 Intégration

Jusqu'ici, Sofa a été présenté comme un outil de conception d'icones auditives. Sofa crée implicitement des classes de son dont les méthodes peuvent être redéfinies par l'utilisateur, ou auxquelles l'utilisateur peut ajouter des méthodes. Les classes de son gérées par Sofa ne peuvent pas être utilisées

directement dans d'autres applications, car leur création et leur comportement dépend fortement du contexte d'exécution fourni par Sofa. Cependant, comme nous l'avons vu, les icones auditives sont contrôlées par des paramètres de haut-niveau, et le processus de conception tend à réifier ce contrôle par des méthodes de haut-niveau comme la méthode "set_strength" du vent. L'interface de la classe correspondant à une icone auditive est l'interface adéquate pour l'application cliente. Il faut donc que les classes qui permettent l'utilisation des icones auditives reprennent cette interface. Par conséquent, Sofa procède à une analyse de la classe du son conçu et génère une classe possèdant la même interface. C'est cette nouvelle classe qui peut être utilisée dans une application cliente (figure 5.4).

La classe générée par Sofa est écrite en Python, mais nous pouvons générer une classe dans tout langage qui permet d'écrire des applications réseau, comme C++ par exemple. Nous avons implanté un serveur, appelé ENO2, qui inclut le contexte d'exécution des classes de son de Sofa. Quand un son est créé par une application cliente avec la classe générée par Sofa, un message est envoyé au serveur qui crée l'icone auditive correspondante en utilisant la classe son de Sofa. Le corps des méthodes de la classe cliente envoie des messages au serveur qui appelle les méthodes correspondantes sur l'objet qu'il gère.

On comprend ici l'intérêt de gérer les scripts de comportement au niveau du serveur. Tout d'abord, cela permet d'écrire des clients moins complexes. En effet, si les comportements étaient gérés par le client, en utilisant un serveur minimum dont la tâche consisterait à jouer des sons simples, le client devrait gérer lui-même un ordonnanceur. Cela reviendrait à obliger le programmeur de l'application cliente à appeler une fonction à des temps précis pour que l'ordonnanceur puisse faire son office. Ensuite, comme les comportements sont décrits par des algorithmes en Python, cela obligerait l'application cliente à utiliser un interprète Python. De plus, les algos de comportement seraient répandus dans les applications clientes, et non partagés au niveau du serveur. Enfin, en produisant des classes qui ne font qu'envoyer des messages à un serveur, il est plus facile de générer des interfaces pour plusieurs langages, et les interfaces étant très légères ne modifient pas les performances de l'application cliente. L'autre intérêt de l'utilisation d'un serveur réside dans le fait que l'on peut suivre des applications distantes en produisant les sons sur une machine locale, à l'instar du système X Window [72].

Le modèle hiérarchique de ENO a été repris. On peut créer des groupes dans lequel on place des sons. Les groupes et les sons peuvent être positionnés dans un espace 2D, et leurs gains peuvent être règlés. Si les groupes permettent l'héritage d'attributs, ils ne permettent plus le comportement automatique comme la parallèlisation ou le séquencement. Ces comportements doivent être décrits dans des scripts. En revanche, on peut associer des transformations de son à un groupe en reprenant les *filtears* [50]. Ainsi, on peut

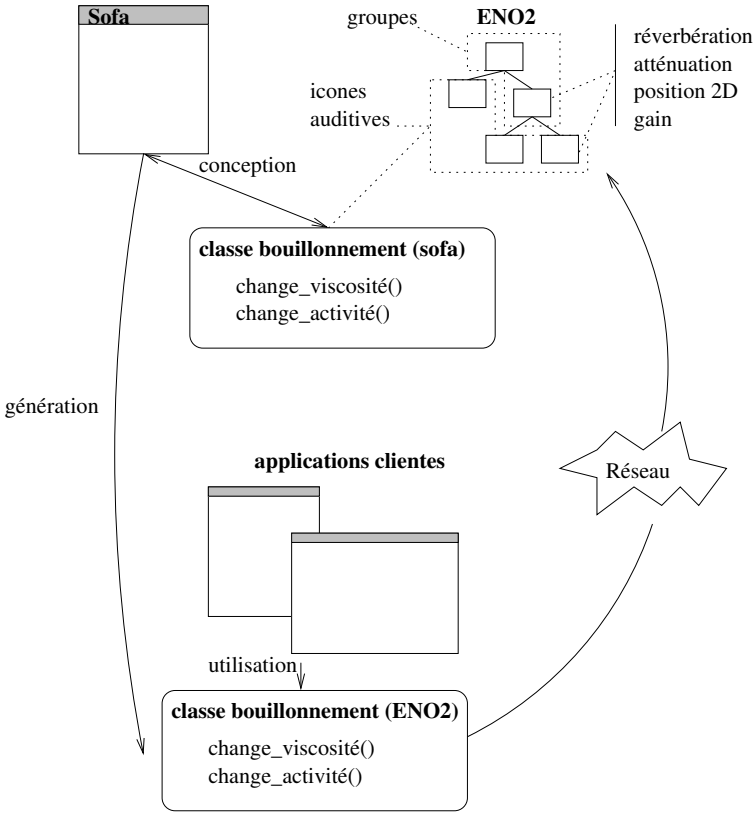


FIG. 5.4 – Relation entre Sofa, ENO2, la conception et la génération de classes de son.

définir un groupe avec une réverbération. De même, un effet d'atténutation permet de simuler un objet situé derrière un obstacle. Il est aussi possible de définir un effet de flou dans le positionnement pour qu'un son ne soit pas "situable", en décalant temporellement légèrement les échantillons gauche et droite. Cet effet peut être utilisé pour un son naturel comme le vent, ce qui donne l'impression d'en être entouré.

## 5.3 Exemples d'application

Cette section expose les types d'application que l'on peut sonoriser avec les icones auditives que nous avons conçues. Elle présente ensuite deux exemples d'applications qui utilisent Sofa et ENO2.

### 5.3.1 Exemples d'usages

Pour décider d'utiliser ou non des sons d'environnement comme le vent et les vagues dans une application, il faut prendre en compte trois caractéristiques importantes de ces sons. Premièrement, ils repésentent des phénomènes naturels sur lesquels l'homme a peu ou pas de contrôle direct. Ils sont donc plus adaptés à représenter un état du système dû à une cause extérieure qu'un processus qui se déroule sous le contrôle de l'utilisateur. Deuxièmement, ce sont des sons qui correspondent à des phénomènes de grande amplitude et qui sont permanents. Ils sont donc plus adaptés au suivi d'état du système dont les variations sont lentes, car ils ont un comportement cohérent avec les phénomènes réels. Troisièmement, les sons les moins intrusifs (faible force du vent ou faible hauteur de vague) correspondent à un environnement calme, tandis que les sons les plus dérangeants correspondent à un environnement perturbé. Il serait donc inapproprié d'utiliser par exemple un son de vent pour représenter le débit d'un réseau : en général, on souhaite que le débit soit le plus grand possible, ce qui en correspondant à un vent fort ne respecterait pas l'*affordance* apportée par celui-ci.

En prenant en compte ces caractéristiques, il apparaît que ces sons sont bien adaptés pour rendre compte d'une *charge* de travail. En général, on cherche à diminuer la charge et si celle-ci augmente ou reste importante, c'est probablement qu'il y a un problème à régler, sachant que l'effet sur la charge ne sera pas en général direct ni immédiat.

Le son d'ascenseur quant à lui est approprié pour le suivi de tâche dont la quantité de travail déjà fait n'est pas connue. Par exemple l'information sur la taille des données est parfois absente lors du téléchargement de certains fichiers. On aimerait pourtant avoir une idée de la *vitesse* de progression du processus. On peut utiliser le son d'ascenseur en faisant correspondre la vitesse de déplacement au débit de la transmission. Comme la vitesse est spécifiée par un "changement qui ne change pas" (une hauteur qui semble montée mais qui reste à un niveau moyen constant), elle peut être maintenue sans atteindre une valeur limite.

### 5.3.2 Suivi de téléchargement

Netscape permet de naviguer sur le web avec des liens hypertextes. Quand un fichier a une extension que Netscape ne connaît pas, celui-ci propose de le télécharger et de l'enregistrer sur le disque local. Lorsque le fichier est gros, le téléchargement prend du temps et évolue en fonction de la charge du réseau et de la qualité de la connexion. Pour pouvoir suivre le téléchargement, il faut soit ajouter des fonctionnalités à Netscape en utilisant son code source, soit avoir accès aux requêtes HTTP qu'il envoie aux serveurs concernés. Netscape peut être configuré pour se connecter à un proxy par lequel transitent les requêtes et les réponses. Il suffit que le proxy

analyse les requêtes pour reconnaître celles qui généreront un téléchargement à longue durée.

Pour cette application, nous nous proposons de suivre le téléchargement de fichier avec une extension ".gz", ".tgz", ".bz2", ou ".zip". Quand un téléchargement est reconnu, un son est associé. Lorsque ce n'est pas le cas, la requête est exécutée et la réponse transmise au client sans accompagenement sonore. Comme il est inutile d'essayer de suivre les téléchargement de petits fichiers, seuls les fichiers d'une taille minimum sont pris en compte.

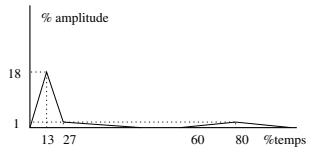


FIG. 5.5 – Enveloppe du bruit blanc pour le bruit de pas pour une durée de 1 seconde

Nous avons conçu un bruit de pas sur un manteau neigeux qui permet de donner des informations sur le déroulement du téléchargement. Le bruit d'un pas est un bruit blanc dont l'amplitude est contrôlée par une enveloppe à deux parties. La première simule l'impact du pied dans la neige et la seconde le relevé du pied lorsque la pointe sort de la neige (figure 5.5). Le son d'une marche est une séquence de son de pas. Lorsque l'on veut simuler une marche rapide, il faut non seulement accélérer le rythme de la séquence de pas, mais aussi modifier le son d'un pas pour qu'il ait un comportement réaliste. En jouant avec l'amplitude maximum des enveloppes et leurs durées, on peut simuler une entrée plus forte et plus rapide du pied dans la neige, ce qui correspond aux caractéristiques des sons de pas lorsque la marche est rapide. En revanche, une marche lente voire prudente est spécifiée par une entrée douce et une sortie un peu plus longue que le son de celle d'une marche normale.

Une marche a pour dimension la vitesse, mais on peut aussi faire passer des informations par sa régularité. Si la marche est irrégulière, c'est que la personne a du mal à marcher, sans doute parce qu'elle a un problème. De même, l'action de "trébucher" introduit une irrégularité particulière dans la marche : elle est spécifié par une succession de pas rapides et forts qui reviennent rapidement à une marche normale. Nous avons donc ajouté quatre méthodes à la classe du son de marche qui sont "change_vitesse", "change_irrégularité", "boîter" et "trébucher".

Les informations que nous transportons par le biais de l'icone auditive sont la vitesse de transfert, son degré d'acccomplissement et la qualité de la connexion. La vitesse de transfert et le degré d'accomplissement permettent d'estimer le temps nécessaire à la complétion dela tâche. Si la tâche

est presque terminée mais qu'elle nécessitait beaucoup de temps, le temps nécessaire à sa finition peut-être long par rapport à l'attente de l'utilisateur qui peut décider d'interrompre la tâche. Percevoir la qualité de la connexion peut aussi inciter l'utilisateur à interrompre la tâche pour la reprendre plus tard, ou dans d'autre conditions (avec un autre serveur par exemple).

Dans notre exemple, la vitesse de transfert est spécifiée par la vitesse de la marche. La vitesse de transfert ne correspond pas au débit du réseau, mais à une mesure de l'accomplissement instantané de l'ampleur estimée de la tâche. Le degré d'accomplissement est spécifié par la position du marcheur dans l'espace 2D en positionnant le marcheur à gauche au début, puis en le déplaçant à droite au fur et à mesure du téléchargement. Enfin, la qualité de la connexion est spécifiée par des irrégularités dans le son de marche. Ainsi, il arrive que la connexion soit tellement mauvaise que plus aucun message ne parvient au proxy. Dans ce cas, le marcheur trébuche et s'arrête.

Le but premier d'une telle interface auditive n'est pas de montrer que l'interface que nous avons conçue permet de suivre efficacement les processus de téléchargements. Pour cela, il faudrait conduire des expérimentations avec des utilisateurs, ce que nous n'avons pas fait. Les expérimentations informelles que nous avons effectuées nous ont permis de constater l'utilité d'un outil de prototypage d'icone auditive, car c'est en utilisant l'application que l'on peut régler les paramètres de l'algorithme. Par exemple, lorsque la marche est très rapide, le son entendu n'est plus reconnu comme une marche. Nous avons pensé ajouter le son du souffle du coureur lorsque la marche devient une course. Ajouter ce son n'a pas été difficile car créer le son et augmenter la classe de son correspondant au son de marche de quelques lignes de script n'a pris que quelques minutes.

### 5.3.3 Suivi de travaux d'impressions

Certaines imprimantes de notre laboratoire peuvent être interrogées sur la quantité de papier disponible ainsi que sur la quantité d'encre restant dans le toner. En combinant ces informations avec le nombre de pages cumulées des documents en attente d'impression, on peut donner des informations sur le devenir de l'état de l'imprimante plutôt que sur son état courant.

Nous avons utilisé le son de vagues et le son de vent pour présenter ces informations. En associant la quantité de papier restant et qui va être consommée par les travaux d'impression en attente avec la taille des vagues, nous pouvons être conscient que l'imprimante ne sera bientôt plus capable d'imprimer de nouveaux documents. De plus, en associant la force du son de vent avec le nombre de pages en attente d'impression, nous pouvons avoir une idée du temps que l'impression prendra. L'absence de vent spécifie que l'imprimante est disponible.

Ce type d'information sonore est destiné à être utilisé de façon opportuniste. Savoir à tout moment que l'imprimante permet d'imprimer effective-

ment des documents a pour conséquence que l'utilisateur n'a pas besoin de s'en enquérir explicitement. Si un utilisateur veut imprimer un document, il peut immédiatement se dire qu'il peut reporter l'impression au moment où l'imprimante sera à nouveau susceptible d'effectuer son office.

Evidemment, les processus d'impression ne sont pas prévisibles et peuvent avoir lieu à tout moment de la journée. Il n'est pas envisageable de jouer le son de vagues toute la journée pour avoir en permanence l'état de l'imprimante ne serait-ce que parce que d'autres processus en tâche de fond pourraient utiliser les vagues dans leur interface. La solution que nous avons testée est la suivante. En prenant le proxy du suivi de téléchargement, il est possible d'identifier les documents susceptibles d'être imprimés. Il s'agit en effet de documents dont le nom contient l'extension ".ps" ou ".pdf". Lorsqu'un tel document est téléchargé, le son de vagues est joué par le proxy avec les paramètres explicités plus haut pendant cinq minutes. Si l'utilisateur désire imprimer les documents, il peut avoir une idée de l'état de l'imprimante sans demande explicite, ni surcharge visuelle.

Le proxy ne peut pas savoir si l'utilisateur imprime effectivement le document. Pour cela, l'utilisateur doit utiliser une commande spéciale qui lance le processus d'impression et qui prévient le proxy qu'une impression a été lancée. La commande d'impression prend le relai du proxy et produit des vagues de la même façon. Les vagues sont modifiées lorsque le document en question est effectivement imprimé : leur forme change et elles semblent s'étaler plus largement. Après l'impression, les vagues reprennent leur forme du début, et sont jouées pendant 5 minutes. Si une nouvelle impression est lancée, le compteur est remis à zéro, et les vagues sont de nouveau jouées cinq minutes après la fin de la dernière impression. Cela permet à l'utilisateur d'être conscient de l'état de l'imprimante pendant une séance d'impressions.

## 5.4 Résumé

Les systèmes qui permettent la conception et l'intégration d'icones auditives doivent répondre à quatre critères. Ils doivent fonctionner en temps réel, disposer d'une interface graphique et pouvoir être contrôlés par des programmes, permettre une transition facile entre conception et intégration, et enfin être facilement déployable en étant extensible et portable.

Les systèmes existants ne proposent pas l'ensemble de ces caractéristiques. Nous avons conçu Sofa, un outil qui permet de concevoir des icones auditives paramétrées. Sofa fonctionne en temps réel et permet à des scripts de contrôler le comportement des sons. Les scripts ont notamment accès à un ordonnanceur et au graphe d'algorithmes qui définit le son.

Les sons conçus avec Sofa sont utilisable facilement par les applications interactives grâce au serveur ENO2 basé sur ENO [8], et qui permet d'utiliser les sons avec l'interface applicative définie par le concepteur. Des exemples

d'utilisations ont permis illustrer les capacités de Sofa et de ENO2.

# Chapitre 6

# Conclusion

Dans cette thèse, nous utilisons le fait que les sons pouvaient apporter des informations par le biais de dimensions écologiques, suivant en cela les préceptes de l'approche écologique de la perception. Nous proposons que les sons construits à partir de sons élémentaires possèdent eux aussi des dimensions de haut-niveau pouvant être vecteurs d'informations. Ils sont donc candidats à une utilisation dans l'interface. En stipulant que les spectres des sons sont moins importants que les invariants qu'ils contiennent, nous proposons d'utiliser une méthode de synthèse ad-hoc. Celle-ci, si elle ne génère pas des sons réalistes, prend en compte les dimensions écologiques de l'événement sonore, et permet leur intégration dans l'interface. De plus, les sons synthétisés de cette manière peuvent être inclus dans un son plus complexe possèdant ses propres invariants. Dans un son complexe, le réalisme des sons simples n'est pas un facteur critique, et leur conception par synthèse ad-hoc est adéquate.

Nous avons conçu un outil, Sofa, qui permet de créer ces sons en donnant notamment accès à un ordonnanceur et au réseau d'algorithmes composant le son. En utilisant des programmes de script, nous avons montré comment reproduire des comportements complexes de sons de base comme le vent, ou des comportements de sons simples pour générer un son complexe comme le son de bouillonnement. Cet outil inclut un serveur d'icones auditives et propose un modèle structuré permettant le positionnement en deux dimensions et l'héritage d'attributs. Le serveur se sert des icones auditives conçues avec Sofa que le programmeur peut utiliser de façon intuitive.

Bien que des examples d'application aient été donnés, il n'a pas été fait de tests en grandeur nature. En fait, il a déjà été montré que le son permet à des utilisateurs de mieux se sentir impliqués dans une métaphore [35], qu'il améliore la collaboration entre plusieurs utilisateurs [33], et qu'il permet de suivre efficacement des activités d'arrière-plan [19]. Les expériences qui ont été faites ne sont effectivement que des expériences de type laboratoire. A de très rares exceptions près comme Out-to-lunch [20], il n'existe

pas d'exemples de déploiement d'applications sonorisées, et encore moins de sonorisation d'applications d'arrière-plan. Bien que nous souhaitions dès le départ réaliser ce type de validation, nous ne l'avons pas fait pour plusieurs raisons. Premièrement, les outils dont nous disposions étaient relativement limités : pas de temps réel, peu ou pas d'extensibilité, et des fonctionnalités qui ne nous satisfaisaient pas. Deuxièmement, il est difficile de développer des icones auditives, et sans ce matériel il est difficile de faire des expériences. Troisièmement, le développement de nos propres outils a pris un temps que nous avons sous-estimé, ce qui a empêché le déploiement parmi les expérimentateurs potentiels. Enfin, nous ne souhaitions pas faire des expériences de type laboratoire, mais des études d'utilisations effectives, dans des applications que les sujets utilisent chaque jour. Ceci nécessite un effort de portage et de programmation non négligeables. De plus, nous voulions étudier les effets à long terme de l'utilisation d'une telle interface comme la captation d'informations opportunistes, ce qui prend encore plus de temps.

Néanmoins, maintenant que nous disposons de matière sous la forme d'icones auditives et d'outils même à l'état de prototype comme Sofa, il est possible d'essayer en grandeur nature d'intégrer des icones auditives dans les applications interactives. Evidemment, si les icones auditives dont nous disposons maintenant proposent un ensemble de départ utilisable dans les applications, les phases de prototypage impliqueront la conception de nouvelles icones auditives dans un futur proche. Certes, la conception d'icones auditives doit suivre les principes de l'approche écologique, et des outils comme Sofa en facilitent la création. Cependant, il faut constater que les informaticiens que nous sommes n'ont pas les compétences "artistiques" pour le faire. A tort ou à raison, une interface graphique est souvent acceptée par ses utilisateurs parce qu'elle est plaisante à regarder, le succès des thèmes des projets Open Source en atteste. On ne voit pas pourquoi il n'en serait pas de même pour les interfaces sonores, surtout que la gêne occasionée par des sons mal conçus est rédhibitoire. La conception sonore doit être effectuée par un *designer*, en suivant les conseils de concepteur d'interfaces pour que ces icones auditives soient effectivement utilisables dans une application interactive, c'est-à-dire qu'elles doivent pouvoir être contrôlées par des attributs de l'événement sonore et non des attributs "musicaux".

A terme, la puissance des machines augmentera suffisament pour que les algorithmes de synthèse ne soient plus aussi coûteux en termes de calculs. Si le champ des possibilités s'en trouvera élargi, les techniques exposées ici ne sont pas condamnées pour autant, car nous avons vu que certains sons sont difficiles à simuler quelque soient les techniques employées. De plus, nous ne sommes pas intéressés par des sons qui ont le *spectre* des sons d'événements réels, mais par des sons qui ont les *invariants* des sons d'événements réels. Cependant, l'intégration de ces techniques et le fait que les *designers* sonores sont habitués à certains outils nous laissent à penser que Sofa se transfor-

mera en serveur de contrôle d'un système de synthèse musical existant. Avec un outil de synthèse comme jMax, les concepteurs disposeront de toute la panoplie des techniques récentes de synthèse de sons, et les utilisateurs d'un système efficace et maintenu par des professionnels du domaine. Sofa ne deviendra obsolète pour autant, car son pouvoir d'expression par les scripts et les possibilités d'intégration dans les applications qu'il offre ne sont pas prises en compte dans un système comme jMax.

Nous n'avons pas étudié durant cette thèse les problèmes que peuvent poser la diffusion simultanée de plusieurs sons. Si l'"effet de cocktail" [2]à un rôle à jouer dans le processus de perception, il n'a pas été étudié dans une interface pour le suivi d'activités d'arrière-plan multiple. Les phénomènes de masquage dûs à l'empilement de signal auront sûrement des conséquences sur la conception des sons joués.

Enfin, il est nécessaire d'analyser plus finement ce qui permet à des sons d'être reconnus et ce qui permet d'évaluer leurs dimensions écologiques. Si les travaux que nous avons évoqués au premier chapitre et la taxonomie de Gaver sont les premières briques de cette analyse, beaucoup de travail reste à faire dans ce domaine. Cela permettrait à des concepteurs d'icones auditives de travailler de façon plus formelle sans passer par des phases "d'essai-erreur".

# Bibliographie


[1] The xerox star : a retrospective. *IEEE Computer 22(9)* (september 1989), 11–29.

[2] Arons, B. A review of the cocktail party effect. *Journal of the American Voice I/O Society 12* (july 1992), 35–50.

[3] Back, M., and D., D. Micro-narratives in sound design : context, character, and caricature in waveform manipulation. In *Proceedings of the International Conference on Auditory Display (ICAD'96)* (1996).

[4] Ballas, J. A. Implicit knowledge in the identification of environmental sounds : causal uncertainty and stereotypy. Tech. Rep. ONR-87-2, Department of Psychology, George Mason University, Fairfax (VA,USA), 1987.

[5] Ballas, J. A. Common factors in the identification of an assortment of brief everyday sounds. *Journal of Experimental Psychology : Human Perception and Performance 19(2)* (1993), 250–267.

[6] Ballas, J. A., and Mullins, T. Effects of context on the identification of everyday sounds. In *Human Performance* (1991), L. E. Associates, Ed., vol. 4(3), pp. 199–219.

[7] Beaudouin-Lafon, M., and Conversy, S. Audio illusions for audio feedback. In *Companion Proceedings of ACM CHI 96 Conference on Human Factors in Computing Systems* (Vancouver (Canada), 1996), pp. 299–300.

[8] Beaudouin-Lafon, M., and Gaver, W. W. ENO : Synthesizing Structured Sound Spaces. In *Seventh Annual Symposium on User Interface Software and Technology (UIST'94)* (November 2-4 1994), pp. 49–57.

[9] Bertin, J. *Sémiologie Graphique*. Gauthier-Villars, 1967.

[10] Blattner, M. M., Sumikarwa, D. A., and Greenberg, R. M. Earcons and Icons : Their Structure and Common Design Principles. In *Proceedings of CHI'90 Conference on Human Factors in Computing Systems* (April 1990), vol. 4, ACM, pp. 5.48–5.74.

[11] Bly, S. Communicating with sound. In *Proceedings of CHI'90 Conference on Human Factors in Computing Systems* (April 1990), vol. 4, ACM, pp. 5.4–5.10.

[12] Bregman, A. S. *Auditory Scene Analysis.* The MIT Press, 1994.

[13] Brewster, S. A. *Providing a Structured Method for Integrating Non-Speech Audio Into Human-Computer Interfaces.* PhD thesis, University of York, 1994.

[14] Cadoz, C., Luciani, A., and Florens, J.-L. Cordis-anima : a modeling and simulation system for sound and image synthesis - the general formalism. *Computer Music Journal 17(1)* (1993).

[15] Card, S. K., Mackinlay, J. D., and Shneiderman, B., Eds. *Readings in Information Visualization : Using Vision to Think.* Morgan Kaufmann, 1999.

[16] Carello, C., Anderson, K. L., and Kunkler-Peck, A. J. Perception of object length by sound. *Psychological Science 9(3)* (may 1998), 211–214.

[17] Chowning, J. The synthesis of complex audio spectra by means of frequency modulation. *Journal of the audio Engineering society 21* (1973), 526–534.

[18] Cohen, J. "Kirk here :" Using genre sounds to monitor background activity. In *Adjunct Proceedings, ACM Conference on Human Factors in Computing Systems* (April 24-29 1993), ACM, pp. 63–64.

[19] Cohen, J. Monitoring backround activities. In *Auditory Display*, G. Kramer, Ed. Addison-Wesley, 1994, pp. 499–531.

[20] Cohen, J. Out to lunch : further adventures monitoring background activity. In *Proceedings of the International Conference on Auditory Display (ICAD'1994)* (1994).

[21] Conn, A. Time affordances : the time factor in diagnostic usability heuristics. In *Proceedings of ACM CHI 95 Conference on Human Factors in Computing Systems* (Denver (USA), 1995), pp. 186–193.

[22] Conversy, S. Sons d'environnement naturel pour le suivi d'activités d'arrière-plan. In *Actes Neuvièmes Journées Francophones sur l'Interaction Homme-Machine, IHM'97* (Futuroscope (France), Septembre 1997), pp. 191–197.

[23] Conversy, S. Ad-hoc synthesis of auditory icons. In *Proceedings of the International Conference on Auditory Display* (Glasgow (UK), November 1998).

[24] Crease, M., and Brewster, S. Making progress with sounds - the design and evaluation of an audio progress bar. In *Proceedings of the International Conference on Auditory Display* (1998).

[25] Déchelle, F., Borghesi, R., Cecco, M. D., Maggi, E., Rovan, B., and Schnell, N. jmax : An environment for real time musical applications. *Computer Music Journal 23(3)* (1999), 50–58.

[26] de Luze, H. *L'ethnométhodologie.* Anthropos, 1997.

[27] DeFatta, D. J., Lucas, J. G., and Hodgkiss, W. S. *"Digital Signal Processing : A System Design Approach".* Wiley, 1988.

[28] Depalle, P., and Rodet, X. De la voix aux instruments. *Les cahiers de L'IRCAM 2* (1993), 121–141.

[29] Dix, A., Finlay, J., Abowd, G., and Beal, R. Predictability and observability. In *Human-Computer Interaction.* Prentice-Hall, pp. 357–361.

[30] Dix, A., Finlay, J., Abowd, G., and Beal, R. Satus/event analysis. In *Human-Computer Interaction.* Prentice-Hall, pp. 365–373.

[31] Doel, K. V. D., and Pai, D. K. Synthesis of shape dependent sounds with physical modelling. In *Proceedings of the International Conference on Auditory Display* (1996).

[32] Fouad, H., and Ballas, J. A. Perceptually based scheduling algorithms for real-time synthesis of complex sonic environments. In *Proceedings of the International Conference on Auditory Display (ICAD'97)* (1997).

[33] Gaver, W., Smith, R., and O'Shea, T. Effective sounds in complex systems : the arkola simulation. In *Proceedings ACM conference on human factors in software (chi'91)* (1991), pp. 85–90.

[34] Gaver, W. W. *Everyday listening and auditory icons.* PhD thesis, University of California, San Diego, 1988.

[35] Gaver, W. W. The SonicFinder : An Interface That Uses Auditory Icons. In *Proceedings of CHI'89 Conference on Human Factors in Computing Systems* (1989), vol. 4, ACM, pp. 67–94.

[36] Gaver, W. W. Synthetizing Auditory Icons. In *Human Factors in Computing Systems, INTERCHI'93* (April 24-29 1990), ACM, pp. 228–325.

[37] Gaver, W. W. Sound Support For Collaboration. In *Second European Conference on Computer-Supported Collaborative Work (E-CSCW'91)* (September 24-27 1991), pp. 293–308.

[38] Gaver, W. W. How Do We Hear In The World ? Explorations of Ecological Acoustics. *Journal of Ecological Psychology 5(4)* (1993), 285–313.

[39] Gaver, W. W. What In The World Do We Hear ? An Ecological Approach to Auditory Event Perception. *Journal of Ecological Psychology 5(1)* (1993), 1–29.

[40] Gaver, W. W., and Smith, R. B. Auditory Icons in Large-Scale Collaborative Environments. In *Human-Computer Interaction - Interact'90* (1993), D. Diaper and D. Gilmore and G. Cockton and B. Shackel, pp. 725–740.

[41] Gibson, J. J. The auditory system. In *The Senses Considered as Perceptual Systems*. Hougton Mifflin, Boston (USA), 1966, pp. 75–96.

[42] Gibson, J. J. *The Senses Considered as Perceptual Systems*. Hougton Mifflin, Boston (USA), 1966.

[43] Gibson, J. J. *The Ecological Approach to Visual Perception*. Lawrence Erlbaum Associates, 1986.

[44] Hopper, A., Harter, A., and Blackie, T. The active badge system. In *Human Factors in Computing Systems, INTERCHI'93* (April 24-29 1990), ACM.

[45] Hutchins E.L., Hollan J.D., N. D. Direct manipulation interfaces. In *User Centered System Design*. Lawrence Erlbaum Associates, 1986, pp. 87–124.

[46] Kevin Karplus and Alex Strong. Digital Synthesis of Plucked-String and Drum Timbres. In *The Music Machine*. MIT Press, 1989, pp. 467–479.

[47] Kunkler-Peck, A. J., and Turvey, M. Hearing shape. *Journal of experimental psychology : human perception and performance 26(1)* (2000), 279–294.

[48] Lass, N., Hinzman, A., Eastman, S., Wright, T., Mills, K., Bartlett, B., and Summers, P. Listeners' discrimination of real and human-imitated animal sounds. *Perceptual and Motor Skills 58* (1984).

[49] Lovstrand, L. Being selectively aware with the khronika system. In *Second European Conference on Computer-Supported Collaborative Work (E-CSCW'91)* (September 24-27 1991).

[50] Ludwig, F. L., Pincever, N., and Cohen, M. Extending the notion of a window system to audio. *IEEE Computer* (August 1990), 66–72.

[51] Mackay, W. E. Media spaces : environments for informal multimedia interaction. In *Computer Supported Cooperative Work*, M. Beaudouin-Lafon, Ed. John Wiley and Sons, Ltd, 1999.

[52] Maury, S., Athènes, S., and Chatty, S. Vers une interaction basée sur le rythme : les menus rythmiques. In *Actes de la 11ème conférence francophone sur l'interaction homme-machine (Tome 2)* (1999), vol. 2, pp. 29–32.

[53] Miner, N. E., and Caudell, T. P. Using wavelets to synthesize stochastic-based sounds for immersive virtual environments. In *Proceedings of the international conference on auditory display (ICAD'97)* (1997).

[54] Moore, F. R. *"Elements Of Computer Music"*. Prentice Hall, 1990.

[55] Moorer, J. A. About this reverberation business. *Computer Music Journal 7(3)* (1983), 6–15.

[56] Muller, M., and Kuhn, S. Participatory design. *Communications of the ACM 36(6)* (june 1993), 24–28.

[57] Mynatt, E. D., Back, M., and Want, R. Designing audio aura. In *Proceedings of the ACM Conference on Computer-Human Interaction, CHI'98* (1998), pp. 566–573.

[58] Mynatt, E. D., and Edwards, W. K. Mapping guis to auditory interfaces. In *ACM Symposium on User Interface and Software Technology* (1992), pp. 61–70.

[59] Norman, D. D. Cognitive engineering. In *User Centered System Design*. Lawrence Erlbaum Associates, 1986, pp. 31–61.

[60] Norman, D. D. *The psychology of everyday things*. Basic Books, Inc., 1988.

[61] Norman, D. D., and Draper, S. W., Eds. *User Centered System Design*. Lawrence Erlbaum Associates, 1986.

[62] Ousterhout, J. K. Scripting : Higher level programming for the 21st century. *IEEE Computer 31(3)* (march 1998), 23–30.

[63] Peterson, I. Some labs are alive with ... the sound of data. In *Proceedings of CHI'90 Conference on Human Factors in Computing Systems* (April 1990), vol. 4, ACM, pp. 5.12–5.14.

[64] Pluckette, M. Combining event and signal processing in the max graphical programming environment. *Computer Music Journal 15(3)* (1991), 68–77.

[65] Pluckette, M. Fts : A real-time monitor for multiprocessor music synthesis. *Computer Music Journal 15(3)* (1991), 58–67.

[66] Risset, J.-C. Paradoxes de hauteur. In *IRCAM Report no. 10*. IRCAM, 1977.

[67] Road, C., Ed. *Computer Music Tutorial*. The MIT press, 1996.

[68] Roads, C. Initiation à la synthèse par modèles physiques. *Les cahiers de L'IRCAM 2* (1993), 145–169.

[69] Roseman, M., and Greenberg, S. Building real time groupware with groupkit, a groupware toolkit. *ACM Transactions on Computer Human Interaction* (march 1996).

[70] Scaletti, C. Sound synthesis algorithms for auditory data representations. In *Auditory Display*, G. Kramer, Ed. 1994, pp. 185–251.

[71] Scaletti, C., and Johnson, R. An interactive environment for object-oriented music composition and sound synthesis. In *Proceedings of the conference on object-oriented programming, systems, languages and applications (OOPSLA'88)* (1988), pp. 222–232.

[72] Scheifler, R. W., and Gettys, J. The x window system. *ACM Transaction on Computer Graphics 5(2)* (april 1986), 79–109.

[73] Schneiderman, B. Direct manipulation : A step beyond programming languages. *IEEE Computer* (august 1983), 57–69.

[74] Serra, X. Musical sound modeling with sinusoids plus noise. In *Musical Signal Processing*. Swets and Zeitlinger, 1997, pp. 91–122.

[75] Serra, X., and Smith, J. Spectral modeling synthesis : a sound analysis/transformation/synthesis system based on a deterministic plus stochastic decomposition. *Computer Music Journal 14(4)* (1990), 12–24.

[76] Smith, J. O. Physical modelling using digital waveguides. *Computer Music Journal 16* (1992), 74–91. Special issue on physical modelling.

[77] Smith, J. O., and Cook, P. R. The Second-Order Digital Waveguide Oscillator. In *Proceedings of the International Computer Music Conference* (1992).

[78] Streri, A. Comment l'homme perçoit-il le monde ? In *L'homme cognitif*, A. Weil-Barais, Ed. Presses universitaires de France, 1993, pp. 89–193.

[79] Turvey, M., Burton, G., Amazeen, E., Butwill, and Carello, C. Perceiving the width and height of a hand-held object by dynamic touch. *Journal of experimental psychology : human perception and performance 24* (1998), 34–35.

[80] Van Derveer, N. Ecological acoustics : Human perception of environmental sounds. In *Dissertation Abstracts International* (1979), U. M. N. 8004002, Ed., vol. 40/09B, 4543.

[81] Walker, B. N., Kramer, G., and Lane, D. M. Psychological scaling of sonification mappings. In *Proceedings of the International Conference on Auditory Display (ICAD'2000)* (2000).

[82] Warren, W., and Verbrugge, R. Auditory perception of breaking and bouncing events : A case study in eclogical acoustics. In *Journal of Experimental Psychology : Human Perception and Performance* (1984), vol. 10, pp. 704–712.