Modèle sonore pour applications interactives

Stéphane Conversy

1995

Table des matières

Intoduction **5**
Intoduction 5
Le son dans l'informatique 5
Plan 6

1 Acoustique et Psycho-Acoustique **8**
1.1 Acoustique 8
1.1.1 Représentation du son 8
1.1.2 Enveloppe d'amplitude 10
1.2 Psycho-Acoustique 11
1.2.1 La hauteur 11
1.2.2 Le volume 11
1.2.3 Le timbre 12
1.2.4 La localisation 12

2 Le son et l'informatique **14**
2.1 Les Synthèses Sonores 14
2.1.1 Synthèse additive 14
2.1.2 Synthèse soustractive 15
2.1.3 Synthèse non linéaire 15
2.1.4 Synthèse par modèle physique 16
2.2 Exemples d'utilisation du son non-parlé 16
2.2.1 Les messages codés 16
2.2.2 Les indicateurs d'état et de contrôle 17

3 ENO **24**
3.1 Principe 24
3.2 Description détaillée 25
3.2.1 Les sources et l'arbre des sources 25
3.2.2 L'héritage d'attribut 25
3.2.3 Les sources primitives 26
3.2.4 Les sources de haut-niveau 26
3.2.5 L'espace sonore 27

3.2.6 Développements futurs 27

4 Utilisation et extension du modèle sonore 28
4.1 Les ascenceurs 28
4.1.1 Description du problème 28
4.1.2 Une solution sonore 29
4.1.3 Les Sheppard-Risset Tones 30
4.1.4 Le son d'ascenseur 30
4.1.5 Utilisation dans les barres de défilement 31
4.1.6 Comparaison avec la solution de Brewster 32
4.1.7 Conclusion 32
4.2 Les Environnements 33
4.2.1 Description du problème 33
4.2.2 Une solution sonore 33
4.2.3 Exemples d'environnement 34
4.2.4 Construction des sons 36
4.2.5 Comparaison avec la solution de Brewster 40
4.2.6 Conclusion 40
4.3 Les Pièces 40
4.3.1 Description 41
4.3.2 L'algorithme 41
4.3.3 Complexité 44
4.3.4 Utilisation 44

Conclusion 45

Introduction

Introduction

Dans la vie de tous les jours, nous utilisons tous nos sens pour acquérir des informations sur le monde qui nous entoure. En particulier, le système visuel nous donne des données détaillées sur une partie de ce monde, et le système auditif nous fournit des données générales qui ne peuvent être perçues par la vue. La combinaison de ces deux sens nous donne la plupart des informations importantes sur notre environnement.

Il est donc intéressant d'adapter ces deux vecteurs d'informations que sont la vue et l'ouïe à l'interaction homme-machine. Ainsi, comme dans un environnement naturel, l'utilisation de ces deux sens permettrait d'avoir un supplément d'information qu'un seul médium ne peut nous fournir : pendant que nous concentrons notre attention visuelle, pour éditer un document par exemple, nous pourrions surveiller l'état d'autres processus en tâche de fond (arrivée de courrier électronique, état d'une impression, etc). Cependant, le développement des interfaces homme-machine a privilégié l'aspect graphique au détriment de l'aspect sonore. Couramment, presque toutes les informations sont affichées à l'écran, conduisant à une surcharge de l'écran. La présence des deux média pourrait nous aider à présenter les informations de la façon la plus naturelle et la plus efficace possible.

Le son dans l'informatique

L'utilisation de sons sur les ordinateurs n'est pas un fait nouveau, bien qu'il soit parfois fortuit. Un des exemples favoris des partisans du son est celui du bruit du disque dur ou de l'imprimante. Les utilisateurs peuvent dire quand une opération de sauvegarde ou de copie a pris fin, grâce au bruit que le disque dur produit. Cela leur permet de faire autre chose pendant que l'opération se déroule. Certes, le son doit être conçu de manière plus « formelle » que cet exemple, mais il faut retenir qu'il est un excellent médium d'information.

L'évolution de l'informatique permet aujourd'hui l'utilisation de sons dans l'interface. En effet, la plupart des postes de travail récents comportent des circuits sonores spécialisés, comme les stations NeXT, les Apple Macintosh AV, les Silicon Graphics Indigo (SGI) ou même les PC. De plus, l'avènement du standard MIDI

dans le monde de la musique électronique et des synthétiseurs permet un contrôle toujours plus facile de ces machines. Malheureusement, ces capacités ne sont utilisées essentiellement que par des musiciens professionnels ou dans les jeux vidéos.

Il est important de noter que nous abordons uniquement la communication sonore dans le sens ordinateur → utilisateur (on exclut ainsi la reconnaissance de la parole ou de motifs sonores), et que ces sons sont **non parlés** (pas de synthèse vocale). Pourquoi exclure l'utilisation de textes parlés ? Tout d'abord, présenter des informations grâce à la parole est un processus plutôt lent, à cause de sa nature sérielle. De plus, une importante limitation du texte est son manque d'expressivité : il faut souvent plusieurs mots pour décrire quelque chose de simple. C'est pourquoi les interfaces graphiques sont apparues : il est plus facile de repérer une icône parmi d'autres qu'un nom dans une liste. Évidemment, la synthèse vocale est parfois utile, comme dans le cas ou il faut communiquer avec des non-voyants. Néanmoins, cette discussion ne prend pas en compte cette catégorie.

On peut diviser le monde des sons non-parlés en trois groupes :

les alarmes et les avertissements sont des signaux qui ont priorité sur toute autre information. Ils ont pour but d'interrompre les tâches courantes et d'alerter un utilisateur que quelque chose requiert son attention immédiate.

les messages d'état et de contrôle donnent des informations sur une tâche qui se déroule.

les messages codés servent à présenter des données numériques sous forme de motifs sonores. Ces sons complexes et variables tranchent par rapport aux sons simples des alarmes ou ceux plus répétitifs des sons de contrôles.

Ce stage ne concerne que la deuxième catégorie, à savoir celle des messages d'état et de contrôle. Nous étudierons plus loin des cas se rattachant à ces trois catégories, car chacun peut apporter de précieux renseignement sur la manière d'utiliser les sons dans une interface.

Plan

Le premier chapitre est une introduction aux concepts et aux termes se rapportant à l'étude du son. Nous allons voir ce qu'est un son, les différentes manières de le représenter, ses caractéristiques et la manière dont nous le percevons. Cette partie peut être sautée par les lecteurs familiers du domaine.

Le deuxième chapitre aborde l'univers de l'audio en informatique. Les différents types de synthèses sonores y sont présentées, puis divers systèmes de sonifications sont étudiés.

Le chapitre suivant est consacré à une application particulière, le serveur Unix ENO. Son principe et une description détaillée y sont exposés.

Enfin, le dernier chapitre présente les travaux effectués pendant ce stage sont présentés. L'objet du stage était d'étudier des méthodes permettant d'enrichir l'interaction homme-machine grâce aux sons. Pour cela, nous avons étudié trois problèmes particuliers des interfaces : la navigation par le biais des barres de défilement, la gestion des modes d'une application et la gestion d'un environnement multi-fenêtres. Pour chacun d'entre eux, nous avons apporté une solution par des algorithmes de synthèse sonore. Cette approche se démarque de la plupart des autres travaux dans ce domaine par l'emploi de techniques de synthèse temps réel, aujourd'hui rendues possibles par la puissance sonore des postes de travail. L'objectif à plus long terme est de définir des primitives sonores comparables à celles qui existent pour la construction d'applications graphiques.

Chapitre 1

Acoustique et Psycho-Acoustique

Ce chapitre est une introduction aux termes et aux concepts utilisés dans les discussions sur le son et l'écoute des sons. Dans la première section, nous présentons les différentes manières de représenter un son, ainsi que ses caractéristiques principales. Dans la deuxième section, nous étudions comment un son est perçu par un être humain et les retombées qu'ont ces effets sur la conception d'interfaces sonores.

1.1 Acoustique

1.1.1 Représentation du son

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

Le son montré en Figure 1.1 est une sinusoïde. Ce type de son est rarement rencontré dans la nature. Néanmoins, cette onde a des propriétés mathématiques extrêmement pratiques. En effet, grâce aux travaux de Fourrier [DLH88], on montre que les sons complexes, sous certaines conditions (il faut que leur forme d'onde

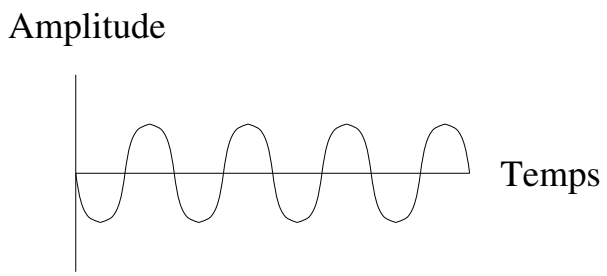


FIG. 1.1 - *Une forme d'onde sinusoïdale. Ce type de graphe représente le son dans le domaine temporel*

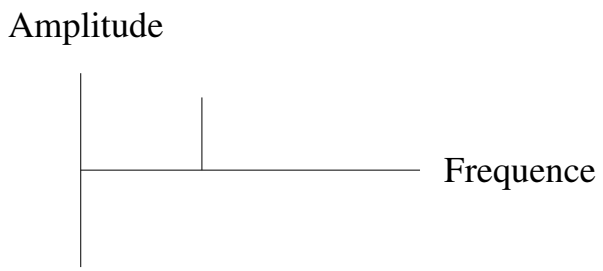


FIG. 1.2 - *Le spectre d'une sinusoïde. Ce type de graphe représente le son dans le domaine fréquentiel*

soit périodique), peuvent être exprimés comme la somme d'un certain nombre de sinusoïdes d'amplitudes, de fréquences et de phases différentes. Grâce à la transformée de Fourrier, on peut retrouver ces sinusoïdes avec leurs caractéristiques respectives à partir d'une forme d'onde. On peut donc aussi représenter un son grâce à son spectre (amplitude en fonction de la fréquence). Ainsi, la forme d'onde de la Figure 1.1 peut être représentée par la Figure 1.2.

Il est important de noter que la transformée de Fourrier permet de passer du domaine temporel au domaine fréquentiel et vice-versa (grâce à la transformée de Fourrier inverse). De plus, il n'y a pas de perte d'information lors de ces deux opérations. On peut remarquer que la fréquence est en quelque sorte l' « inverse » du temps : donc 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 Fourrier permet de passer d'une représentation en fonction d'une variable à une représentation en fonction de l'inverse de cette variable.

Avec le théorème de Fourrier et la transformée de Fourrier, on peut représenter une forme d'onde complexe de façon beaucoup plus explicite (cf. Figure 1.3), car on peut voir effectivement la décomposition du son en sinusoïdes.

Les composantes fréquentielles d'un son sont appelées des *partiels*. Très souvent, il n'existe aucune relation entre ces partiels, ce qui produit des sons *inharmoniques*. Cependant, certains sons ont des partiels qui sont des multiples entiers du partiel de plus basse fréquence (la fréquence *fondamentale*) : ces sons sont dits *harmoniques*. Les instruments traditionnels sont généralement conçus de manière à ce qu'il produisent ce type de sons, et c'est ce qui les rend agréables à écouter.

Si l'on regarde à nouveau la Figure 1.3, 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. Le spectre (amplitude = f(fréquence)) est donc plus important que la forme d'onde.

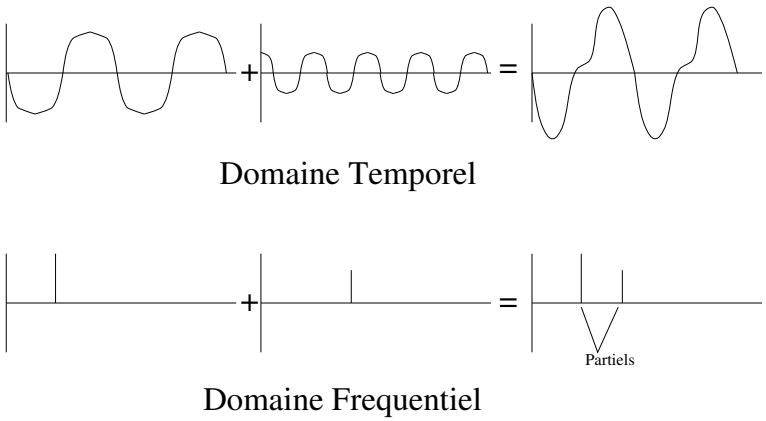


FIG. 1.3 - *Un forme d'onde complexe. Le domaine des fréquences est plus facile à manipuler*

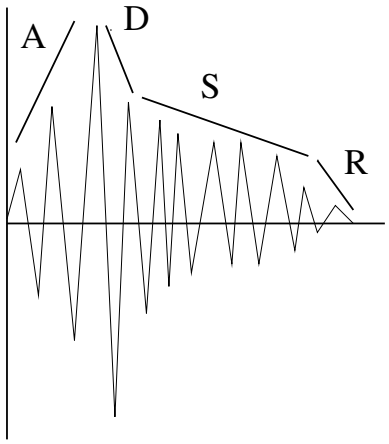


FIG. 1.4 - *Une enveloppe ADSR*

1.1.2 Enveloppe d'amplitude

Cependant, cette représentation ne tient pas compte de l'évolution du son au cours du temps. En effet, les sons naturels ne « stagnent » pas, ne serait-ce que parce qu'ils ont 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 1.4). La représentation la plus répandue est celle dite ADSR (Attack, Decay, Sustain, Release) ; cependant, on n'est pas obligé 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 1.5). Les évolutions relatives de l'amplitude de chaque partiel permettent de caractériser la « couleur » d'un son, autrement dit son *timbre*.

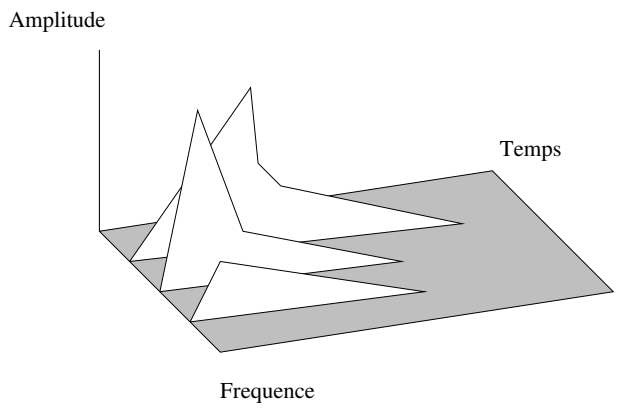


FIG. 1.5 - *Un spectre variable au cours du temps*

1.2 Psycho-Acoustique

La psycho-acoustique est l'étude des phénomènes régissant la perception des caractéristiques d'un son par l'homme. La prise en compte des effets psycho-acoustiques est nécessaire lors de la conception d'une interface sonore. En effet, l'ignorance de ces effets peut conduire au fait qu'un utilisateur ne peut différencier deux sons, ou ne peut les entendre. On considère généralement quatre caractéristiques psycho-acoustiques : la hauteur, le volume, le timbre et la localisation [Gav90a].

1.2.1 La hauteur

La hauteur correspond à la fréquence, mais sa perception n'est pas linéaire par rapport à la fréquence. Ainsi, tripler une fréquence entre 1 kHz et 3 kHz ne fait que doubler l'impression subjective de hauteur. De plus, la hauteur d'un son est influencée par son intensité : à moins de 2 kHz, une augmentation d'intensité diminue la hauteur perçue. Enfin, les timbres « brillants », c'est-à-dire ceux des sons dont les hautes fréquences ont une grande énergie, paraissent plus haut que les sons sourds.

En général, notre oreille n'est pas « absolue », c'est-à-dire qu'ils ne peuvent trouver la fréquence du son qu'ils viennent d'entendre. En revanche, beaucoup de gens ont une oreille « relative » : ils peuvent différencier deux sons de hauteurs différentes (bien que certaines soient incapables de dire lequel des deux est le plus haut). Ces deux derniers faits indiquent qu'il ne faut pas essayer de communiquer des valeurs absolues avec des hauteurs ; l'affichage de l'évolution d'une fonction selon une dimension (que l'on associera au temps) est plus appropriée à ce type de médium [Bly90].

1.2.2 Le volume

Le volume correspond à l'amplitude d'un son selon la relation

$$L = kI^{0.3}$$

avec L, volume en Sones (une unité subjective), I l'intensité en dB et $k \approx 0.3306$. Cette relation dit qu'une augmentation d'intensité de 10 dB double le volume.

Les autres facteurs qui influent sur le volume sont la fréquence et la largeur de bande (phénomène de masquage). Le volume dépend de la fréquence : ainsi, pour une intensité donnée, un son à 1000 Hz n'aura pas le même volume qu'un autre à 100 Hz. La relation entre le volume et la fréquence est très compliquée, il suffit pour s'en convaincre de se référer à [Gav90a]. Quant au phénomène de masquage, il se traduit par l'atténuation d'un son par un autre. Il dépend des fréquences mises en jeu par ces sons : pour les sons dont la fréquence tombe dans une bande critique, l'énergie est additionnée, alors qu'à l'extérieur de cette bande, c'est le volume qui est additionné. De plus, les sons avec une large bande de fréquence sonnent plus fort que ceux avec une bande étroite. Donc si une information doit être présentée avec le volume, le son correspondant doit être assez fort pour éviter le masquage par des bruits ambiants.

En général, les êtres humains ne sont pas capables de juger absolument le volume d'un son. De plus, l'échelle des jugements relatifs est limitée à trois niveaux. Ces pauvres capacités et les phénomènes vus plus haut obligent les concepteurs à être très prudents quant à l'utilisation du volume. De plus, il ne faut pas oublier que les utilisateurs ont souvent un contrôle du volume général d'une machine, et qu'ils peuvent de ce fait perdre des informatisons si le niveau est trop bas.

1.2.3 Le timbre

Le timbre occupe une place particulière au sein des termes descriptifs d'un son, car il n'existe ni définition précise, ni relation mathématique qui puisse le caractériser formellement. La définition qu'en donne l'ANSI [ano73] est d'ailleurs plutôt vague : c'est « l'attribut auditif qui permet de distinguer deux sons de même hauteur et de même fréquence ». Pourtant, le timbre est une des caractéristiques d'un son les plus immédiates et les plus facilement reconnaissables.

Le timbre reste donc en grande partie inconnu.On peut dire qu'il dépend au moins du spectre et des enveloppes de chaque partiels. Actuellement, les chercheurs dirigent leur travaux sur la source du son plutôt que le son lui-même, afin de mieux contrôler ce dernier : c'est une approche par modèle physique, que nous verrons plus loin.

1.2.4 La localisation

La localisation est la capacité à déterminer la position d'une source sonore dans l'espace. La localisation droite-gauche se fait grâce à deux phénomènes : la Différence Interaurale d'Intensité (DII), et la Différence Interaurale Temporelle (DIT).

En fait, comme l'homme dispose de deux oreilles espacées de quelques centimètres, les sons ne sont pas perçus de la même façon par chaque oreille.

La DIT provient du fait que le son ne met pas le même temps pour parvenir à chaque oreille : il existe une différence dont notre cerveau se sert pour construire la perception spatiale. Cette différence est d'autant plus importante que la source est située sur les côtés de la tête (c'est-à-dire que la source est proche d'une plan passant par les oreilles et les pieds) et que la source est proche de nous. En effet, plus la source est éloignée, moins la différence temporelle est importante. La DIT est donc négligeable pour les sources éloignées mais utile pour celles qui sont proches.

La DII quant à elle, est la différence perçue entre les intensités. En fait, elle est due à l'ombre sonore que crée la tête d'une personne. Si l'onde une longueur d'onde plus grande que la largeur de la tête, la différence ne sera pas perçue. En revanche, si elle est plus petite, une ombre sonore se forme, différenciant ainsi les sons reçus par chaque oreille.

L'impression de profondeur est due à l'affaiblissement de l'intensité du signal et à l'appauvrissement du spectre. En effet, les fréquences basses voyagent mieux que les fréquences plus hautes, dont les pertes sont facilement constatables. Il résulte de ce fait que les sons sourds sont perçus plus éloignés que les sont brillants.

D'autres apects interviennent de manière importante dans la localisation, en particulier la façon dont les sons sont filtrés par l'oreille externe, la tête et le haut du corps. Des systèmes comme le *convoltron* [S.H91] modèlisent ces effets à l'aide de filtres, mais ils demandent beaucoup de puissance de calcul.

Chapitre 2

Le son et l'informatique

Ce chapitre à trait à l'utilisation de l'informatique pour la création sonore, et à l'utilisation du son dans un environnement informatique. Bien que leur formulation laisse à penser que ces deux parties sont antinomiques, l'étude de l'une influence l'étude de l'autre. En effet, l'informatique est un outil de modélisation sonore, et les sons crées sont utilisés par les concepteurs d'interface. Les champs nouveaux découverts grâce à l'informatique nourrissent les besoins sonores des interfaces utilisées par cette même informatique.

Ce chapitre se compose donc de deux parties : les différentes techniques de synthèse sonore, puis les différentes utilisations du son dans les interfaces.

2.1 Les Synthèses Sonores

Comme on l'a vu plus haut, les sons sont des variations de pression dans un milieu élastique. Le signal que représente l'amplitude en fonction du temps est donc un signal analogique. La numérisation de signaux analogiques permet leur traitement par l'informatique. On peut donc représenter le processus de synthèse sonore comme un système ingérant une suite de nombres représentant un signal, et produisant une sortie correspondant à une transformation mathématique de l'entrée.

Cette section décrit les diverses classes de synthèses utilisées par les informaticiens et les musiciens. Pour des informations plus détaillées, voir [Moo90].

2.1.1 Synthèse additive

Le principe de la synthèse additive est dérivé du théorème de Fourrier. Celui-ci dit que toute fonction périodique peut s'écrire comme la somme de fonctions sinusoïdales de fréquence, d'amplitude, et de phase différentes. Chaque composant est très simple à synthétiser, mais leur addition permet de produire des timbres très complexes, très riches et, en théorie du moins, identiques aux instruments simulés.

En effet, il suffit d'analyser un son pour déterminer chaque composante avec ses caractéristiques respectives, puis de les reproduire à l'identique pour obtenir un

son très proche de l'original. C'est le principe qu'adopte le *phase vocoder*, ou *vocoder de phase* : on découpe la bande passante en plusieurs bandes étroites, qui sont analysées indépendamment (ou presque). On peut reconstruire le signal original en utilisant une sinusoïde par bandes, chose plus facile que d'imiter le son en entier. Le vocoder de phase est aussi utilisé pour créer des effets spéciaux. En effet, plutôt que de rejouer le son original, on peut appliquer un traitement sur certaines de ses composantes seulement, ce qui peut donner par exemple des voix robotisées.

Cette technique est donc très puissante mais elle a un défaut de taille : elle est trop gourmande en calculs. Ainsi, pour un son de piano de qualité CD (44,1 kHz), il faut 400 sinusoïdes pour synthétiser le La le plus grave. Si l'on veut une polyphonie de 32 voies, le nombre de partiels à traiter simultanément reste hors de portée des techniques actuelles. Cependant, de nouvelles techniques basées sur des composants VLSI voient le jour, et semblent assez prometteuses [SC92].

2.1.2 Synthèse soustractive

La synthèse additive produit un son à partir de sinusoïdes primitives. La synthèse soustractive est basée sur l'idée complémentaire de passer un signal complexe dans un filtre afin de produire la forme d'onde voulue. C'est une technique de sculpture, le matériau étant le son en entrée, le filtre jouant le rôle du marteau et du ciseau. Ce filtre est généralement variable dans le temps.

On associe souvent cette technique au principe de *modèle physique*, car plusieurs instruments fonctionnent de cette façon. Une corde de violon est une excitation qui est filtrée par le corps du violon, la voix humaine est un bruit modulé par les cordes vocales...

Tout l'art du programmeur est de concevoir des filtres qui imitent le comportement de tel ou tel instrument. Un des exemples les plus réussis est celui de l'algorithme de Karplus et Strong [Kev89], qui synthétise des cordes pincées ou frappées. Cet algorithme est très simple et peu coûteux, et pourtant le son produit parait « réel », « acoustique » ; en effet, son principe est tel que deux notes identiques jouées à la suite avec un tel algorithme ne sont pas exactement identiques.

2.1.3 Synthèse non linéaire

Les synthèses vues précédemment sont dites linéaires. Elles peuvent être décrites comme un système recevant une entrée et produisant une sortie. Si pour un tel système, une entrée x_1 produit une sortie y_1 et une entrée x_2 produit une sortie y_2, alors une entrée $ax_1 + bx_2$ produira une sortie $ay_1 + by_2$. Les systèmes qui ne respectent pas cette règle sont dits *non-linéaires*.

Les synthèses non-linéaires sont généralement des générations de formes d'ondes complexes à partir de formes simples grâce à des opérations non-linéaires. La synthèse non-linéaire la plus connue est sans doute la synthèse *FM* (*Frequency Modulation* ou *Modulation de Fréquence*) de Chowning [Cho73] , popularisée par Yamaha

en 1983 avec son DX7. Elle consiste en une fréquence de base f_c (*la porteuse*) modulée par une autre fréquence f_m (*la modulation*) :

$$f(n) = f_c + \delta f cos \frac{2\pi n f_m}{R}$$

avec $f(n)$ la fréquence instantanée à l'échantillon n, δf la déviation maximum par rapport à la porteuse, et R le taux d'échantillonnage.

Ce type de synthèse produit des spectres complexes, mais les formes d'onde obtenues ne correspondent pas exactement à un type de son réel précis. Néanmoins, les méthodes non-linéaires donnent un moyen efficace de contrôle de paramètres acoustiques, comme la largeur de bande d'un spectre harmonique.

2.1.4 Synthèse par modèle physique

Les moyens informatiques toujours plus puissants et moins coûteux ont permis à un nouvelle forme de synthèse d'émerger ces dernières années. Certains chercheurs ont en effet décrit la source sonore plutôt que le son lui-même. La création de sons acoustiques y a beaucoup gagné car la technique utilisée imite le comportement d'un instrument. En effet, une description complète de l'instrument est introduite dans l'ordinateur (un *modèle physique*), et une excitation « virtuelle » lui fait produire un son [Roa94].

Les modèles mathématiques des instruments utilisent généralement des systèmes d'équations différentielles. Cependant, leurs résolutions demandent des techniques d'approximations successives très gourmandes en temps de calcul. Par exemple, les cordes sont modélisées par le modèle des masses et des ressorts. De ce fait, la moindre simulation de pincement ou de frottement demande un travail énorme des calculs complexes. Une technique plus rapide semble cependant émerger et pourrait avoir des retombées industrielles d'ici peu. Il s'agit de la technologie *Waveguide* ou *Guide d'Onde*, dont le principe résoudrait en grande partie les problèmes de résolutions d'équation différentielles [Smi92].

Malgré cette complexité, cette nouvelle approche est très utile, et son concept très intéressant pour la conception d'interfaces sonores, comme nous le verrons plus loin.

2.2 Exemples d'utilisation du son non-parlé

Il existe plusieurs exemples d'utilisation des sons dans l'interface homme-machine. On peut rassembler ces cas en deux ensembles : les messages codés et les messages d'état et de contrôle.

2.2.1 Les messages codés

Les messages sonores codés sont utilisés comme une alternative au graphique pour transmettre des données.

Ainsi, Sara Bly estime que certaines fonctions sont difficiles à représenter graphiquement : les fonctions multidimensionnelles, celles variant en fonction du temps, ou les fonctions logarithmiques [Bly90].

Pour les fonctions multidimensionnelles, Bly utilise sept caractéristiques sonores : hauteur, volume, durée, forme d'onde, enveloppe d'attaque, et addition d'une 5e et d'une 9e harmonique. Lors d'un test, elle des données à six dimensions sont présentées graphiquement, avec du son ou avec les deux à la fois. L'expérience montre que l'utilisation du son combinée au graphisme augmente sensiblement la capacité d'une personne à dire si une variable appartient à un ensemble ou à un autre.

Bly utilise une technique similaire pour les données temporelles. Cependant, elle ne fait varier que la hauteur et le volume pour une fonction donnée tout en attribuant à chacune de ces fonctions un timbre qui permet de les différencier. Ainsi, il est plus facile de comparer l'évolution relative de deux fonctions au cours du temps.

Enfin, pour les données logarithmiques, elle ne fait varier que la hauteur. En effet, la hauteur est elle-même une échelle logarithmique si on la représente en fonction de la fréquence, elle s'applique donc bien à ce type de fonction.

Mezrich, quant à lui, utilise la technique de Bly pour les données dépendantes du temps : ainsi, les données sont présentées par des mélodies, dont les notes correspondent à leurs valeurs [Bly90]. Comme pour Bly, les tests se sont révélés concluants : l'utilisation du son permet aux utilisateurs une meilleure distinction des fonctions que si elles sont affichées avec le graphisme seulement.

Ces techniques de présentation permettent à des étudiants en chimie non-voyants de reconnaître des spectres infra-rouges de diverses substances [Pet90] [Bly90] 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... ».

2.2.2 Les indicateurs d'état et de contrôle

Les exemples vus précédemment concernaient la présentation de données par un moyen autre que le graphisme. Cette section aborde l'utilisation du canal auditif pour étendre et enrichir l'interaction homme-machine.

SoundTrack

L'une des applications des interfaces sonores est l'accès aux ordinateurs par les aveugles et mal-voyants. L'étude de tels systèmes est non seulement utile pour les non-voyants, mais aussi pour les voyants, car elle montre les chemins à suivre quand

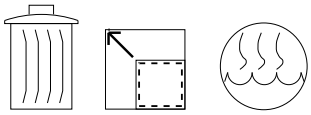


FIG. 2.1 - *Les différents types d'icônes (représentatif, abstrait, semi-abstrait)*

l'écran est surchargé et que nous devenons nous aussi d'une certaine manière partiellement mal voyants.

Ainsi, Edwards propose SoundTrack [Edw90], une interface sonore pour un traitement de texte. SoundTrack utilise à la fois des sons parlés et des sons musicaux. L'écran est divisé en zones correspondant à des menus. Quand le curseur de la souris passe au-dessus d'une de ces zones, un son caractéristique de cette zone est émis : c'est une information brève de l'endroit où se trouve le curseur. Les sons utilisés sont des *siganux carrés* (sons simples à synthétiser) dont la hauteur varie de gauche à droite et de haut en bas. Si l'utilisateur veut plus d'information, il clique sur la zone survolée et son titre est dicté par le synthétiseur vocal. Pour aller plus loin, l'utilisateur double clique sur le menu et se retrouve avec un autre menu.

Lors des tests, il est apparu que la hauteur des signaux n'était pas utilisée. Les utilisateurs comptent le nombre de changements de tons (c'est-à-dire le nombre de zones traversées), pour savoir où se trouve le curseur.

Les Earcons

Dans [BSG90], Blattner, Sumikawa et Greenberg définissent l'équivalent sonore des icônes : les *earcons*. Ils servent à donner des informations sur des objets, des opérations ou des interactions informatiques (fichier, éditer, éditer un fichier).

Les auteurs considèrent trois classes d'icônes (cf. Figure 2.1) :

représentatives : ce sont de simples images familières d'objets ou d'opérations. Elles sont facilement reconnaissables, mais malheureusement, tous les objets ou toutes les opérations n'ont pas de représentation picturale évidente. De plus, ces images peuvent apporter des informations parasites, car elles peuvent être trop détaillées.

abstraites : ce sont des combinaisons de formes géométriques décrivant un objet ou une opération qui ne peut être représentée par une image.

semi-abstraites : ce sont soit un mélange d'icônes représentatives et d'icônes abstraites, soit une icône représentative tellement simplifiée que l'on peut la considérer comme abstraite.

Les auteurs admettent que l'échelle est continue (il est parfois difficile de classer une icône).

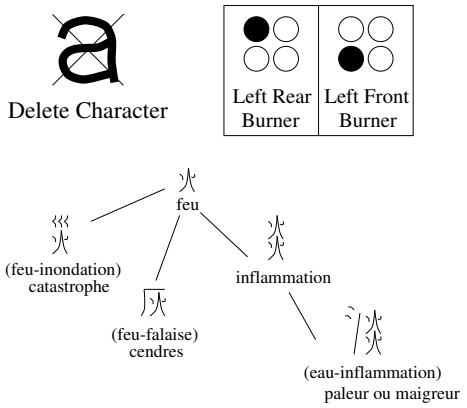


FIG. 2.2 - *Les compositions d'icônes (combinaison, transformation, héritage)*

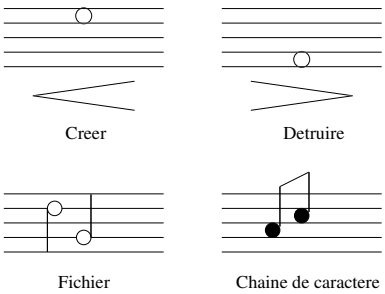


FIG. 2.3 - *Des motifs*

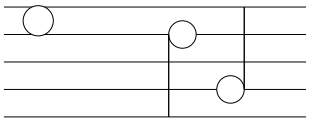
Selon eux, les icônes sont soit des éléments, soit des compositions d'éléments. Ils présentent trois façons de créer des icônes composées (cf. figure 2.2) :

la combinaison : comme son nom l'indique, c'est un mélange d'icônes ;

la transformation : il s'agit d'une modification légère d'une icône connue. Les icônes composées grâce à une transformation représentent souvent un objet avec un état associé (cf. l'exemple 2 Figure 2.2) ;

l'héritage : c'est une transformation telle que l'ensemble des transformations se rapportant à une icône peut s'organiser en arbre hiérarchique. Plus on s'enfonce dans l'arbre, plus la signification d'une icône se spécialise. Le système d'écriture japonais, dont un exemple est donné Figure 2.2 fonctionne en partie grâce à l'héritage.

Blattner reprend ces principes pour ses *earcons* : les éléments sont des *motifs* (cf. Figure 2.3) (petites mélodies de deux ou trois notes aisément reconnaissables par leur rythme, leur hauteur, et leur volume), qui peuvent être combinées, transformées ou héritées.



Creer un Fichier

FIG. 2.4 - *Une combinaison de motifs*

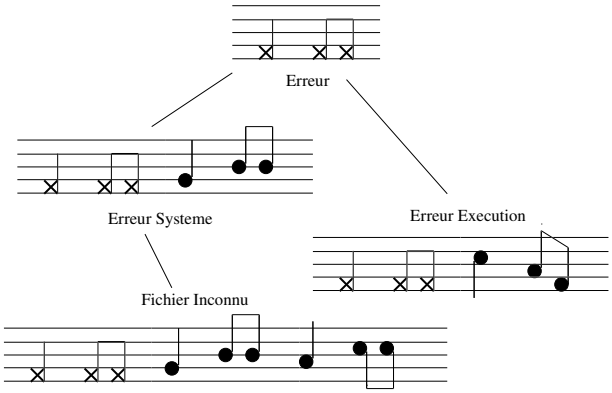


FIG. 2.5 - *Un exemple d'héritage*

Ainsi, si un mi *crescendo* (augmentation du volume) correspond à *créer* et si deux blanches, ré et sol correspond à *fichier*, ces deux notes jouées après un mi crescendo correspondent à *créer un fichier*. C'est un exemple de combinaison, comme pour les icônes (cf. Figure 2.4)

L'héritage est une peu plus complexe : chaque niveau correspond à un attribut (dans l'ordre hiérarchique : rythme, hauteur, timbre, dynamique et registre ou octave). Les familles de messages sont distinguées avec 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 positive. 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 un dynamique ou jouée dans un autre registre (Figure 2.5)

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 correspondant. 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 bien s'entraîner au moins une fois à reconnaître les types d'earcons, la famille à laquelle ils 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.

Auditory Icons

Pour présenter des informations, Bly utilisait les caractéristiques du son en lui-même comme la hauteur, le volume, le timbre etc. Ceci résultait en des sons qui n'avaient aucune correspondance avec notre environnement naturel. De la même façon, les earcons n'ont aucun analogue dans notre vie quotidienne. Ils nécessitent donc un apprentissage.

Pourtant, quand une porte se ferme, nous n'entendons pas le son en terme de hauteur, de volume, ou de timbre... Nous entendons une porte d'une certaine taille, d'une certaine matière qui se ferme avec une certaine force dans une pièce d'un certain volume. En fait, nous utilisons le son afin d'identifier les événements qui l'ont crée. Les caractéristiques de bas niveau d'un son nous intéressent peu : ce qui nous importe, ce sont les caractéristiques de la source sonore et de l'interaction sur cette source qui a donné naissance à ce son. Donc, si nous raisonnons en termes de sources et d'interactions, plutôt qu'en termes du son en lui-même, nous pouvons esprérer communiquer des informations beaucoup plus naturellement.

En fait, un son fournit des informations sur des objets interagissant à un endroit dans un environnement. Cette approche s'inspire de l'*approche écologique* [Gav93] et est utilisée par Gaver pour ses *Auditory Icons* (icones auditives) [Gav86]. Les icônes auditives sont des imitations de sons naturels comme des impacts ou des frottements. Le terme « imitation » est préferable au terme « reproduction » : il n'y a nullement besoin d'une reproduction parfaite. En effet, ces sons essaient de capturer les principales caractéristiques des sons naturels de telle façon qu'elles soient aisément identifiables. Ils correspondent en fait aux caricatures graphiques, ou aux dessins animés, où certains traits sont accentués.

Dans le système de classification de Blattner, il existe un ordre de facilité d'apprentissage. Ainsi, les icônes représentatives sont plus facilement reconnaissable que les icônes semi-abstraites, elles-même plus facilement reconnaissables que les icônes abstraites. Comme nous l'avons vu, la même hiérarchie peut être utilisée pour les sons, les earcons étant classées dans la troisième catégorie. Le fait que Gaver utilise des sons naturels, que nous avons appris à connaître et à reconnaître, permet une représentation beaucoup plus directe que les earcons. On peut donc classer les Auditory Icons dans la première et deuxième catégorie (représentative et semi-abstraite).

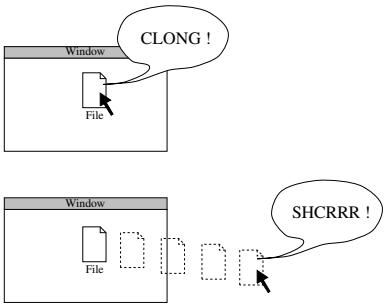


FIG. 2.6 - *Sonic Finder : Sélection et déplacement d'un fichier*

En réalisant le SonicFinder [Gav89], Gaver a implémenté les Auditory Icons sur l'interface du Macintosh. Un exemple d'utilisation est celui de la destruction d'un fichier (cf. Figure 2.6). Pour détruire un fichier sur un Macintosh, on le sélectionne d'abord en cliquant dessus. Cette action est accompagnée d'un son d'impact, ce qui parait naturel. Le type du fichier et sa taille sont véhiculés par le bruit du choc. Par exemple, un fichier texte a un son mat, comme un bout de bois, alors qu'une application a un son brillant. De plus, plus le fichier est gros, plus on a l'impression que l'on a tapé sur un objet gros (le son est plus grave). Ainsi, le son est défini grâce à une source (fichier, objet) et une interaction (pointer, taper).

Le fait qu'à un type d'objet informatique (fichier texte, application) correspond un objet d'une certaine matière (bois, métal), classe cette représentation dans la catégorie semi-abstraite : le son est reconnu, mais la correspondance doit être apprise. En revanche, la correspondance entre la taille de l'objet informatique et la taille de la source sonore est totalement naturelle, donc à classer dans la catégorie représentative.

Il faut maintenant déplacer le fichier dans la poubelle, ce qui se réalise en faisant glisser le fichier sur le bureau. Ici, la source n'a pas changé (c'est toujours le même fichier). En revanche, l'interaction est passé du type « pointer » au type « glisser », il faut donc faire évoluer le son en conséquence. Le son correspondant à l'action « glisser » est un frottement. Mais le changement d'interaction ne change pas la perception des caractéristiques de l'objet que l'on avait acquises lors de la première interaction. Si l'objet est gros, on entend un frottement sur un objet gros. S'il est en bois, on entend un frottement sur du bois. Il est important de différencier les caractéristiques de l'objet de celles de l'interaction. Ainsi, il doit être possible de reconnaître le même objet dans deux interactions différentes.

Gaver utilise aussi ses Auditory Icons dans des tâches de collaboration : ARKola et EAR. [Gav91]. Ainsi, avec la simulation ARKola, il modélise une usine de soda, dont le but est de fabriquer du cola et de le mettre en bouteille. Mais l'usine est conçue de telle manière qu'il faut au moins deux écrans pour la surveiller en entier. On fait donc collaborer deux personnes, qui doivent s'entendre afin de gérer au mieux l'usine. Généralement, une tâche de collaboration est rendue difficile par

le fait que l'on veut permettre un travail individuel, et le fait que l'on doit toujours penser au but commun. L'apport d'un retour d'information sonore permet d'atténuer cette frontière. Ainsi, dans ARKola, le fait de travailler sur une partie de l'usine n'empêche pas de se tenir au courant de la partie invisible, grâce aux sons qu'elle produit. On se tient au courant de l'état global de l'usine. Par exemple, on peut très bien entendre des sons de bouteilles qui se brisent, et le signaler à son collègue pour qu'il corrige le défaut.

Avec EAR, les sons sont utilisés pour signaler des événements à des personnes travaillant dans un même bâtiment. Ainsi, si un groupe de personnes veut prendre le thé et aimerait qu'une personne particulière se joigne à eux, elles lui font savoir en lui envoyant un son de thé versé dans une tasse. Le début d'une réunion est signalée par le son de murmures augmentant en nombre et se terminant par un bruit de maillet. Ces sons n'interrompent pas le travail, et peuvent être facilement ignorés. De même, des événements relatifs à l'utilisation d'un Mediaspace peuvent être signalés par des signaux sonores. Ainsi, quand quelqu'un se connecte sur une caméra, on entend un grincement de porte qui s'ouvre, et quand il quitte la connexion, un son de claquement de porte se produit. Cet environnement sonore permet à chacun de se tenir au courant des activités de ses collègues

Enfin, le concept d' Auditory Icons a été adapté au monde Unix avec le serveur ENO [BLG94]. Ce serveur gère un espace sonore que se partagent plusieurs applications. Ces dernières créent des sources à l'intérieur de cet espace et interagissent avec elles. J'ai utilisé ENO pour mettre en œvre mes algorithmes : le prochain chapitre lui est consacré.

Chapitre 3

ENO

Ce chapitre est consacré à ENO[BLG94], un serveur audio sur Unix. Il reprend les principes d'approche écologique [Gav93] et d'Auditory Icons [Gav86] vus dans le chapitre précédent. Dans un premier temps, nous verrons le principe de la structure du serveur, une description plus détaillée étant donnée en deuxième partie.

3.1 Principe

Comme nous l'avons vu, ENO est basé sur la compréhension de l'origine des sons : les sons sont issus de *source sonore* sur lesquelles une *interaction* se produit. Les sources et les interactions sont caractérisées par un type et des attributs, comme la taille d'un objet, ou la force d'un impact. Un impact peut être considéré comme une interaction élémentaire. Des structures plus complexes peuvent être construites, en associant plusieurs sons de sources différentes (parallèlement ou séquentiellement), ou plusieurs sons d'une même source disposée temporellement d'un façon particulière (comme pour produire un effet de rebond).

Les sources sont placées dans un espace sonore (cf. Figure 3.1), ce qui permet non seulement de les distinguer quand plusieurs d'entre elles jouent en même temps, mais aussi de leur attribuer de nouvelles caractéristiques aisément identifiable. Par exemple, on peut émettre des messages urgents en premier plan alors que des sons de surveillance se situeront plutôt en arrière plan. Ou encore, les sons générés par une application viennent tous d'un même endroit. Beaucoup de systèmes d'écoute 3D ont été réalisés, mais ils demandent beaucoup de ressources matérielles et surtout nécessitent l'utilisation d'un casque. La solution retenue dans ENO est un espace sonore en 2D (un plan horizontal), solution suffisante en regard de l'usage de la spatialisation.

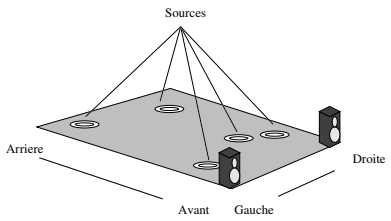


FIG. 3.1 - *Espace sonore d'ENO.*

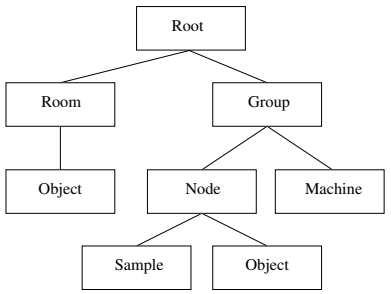


FIG. 3.2 - *Structure d'arbre d'ENO. Les attributs sont hérités le long des branches.*

3.2 Description détaillée

3.2.1 Les sources et l'arbre des sources

L'élément principal de l'architecture de ENO est la *source*. Ces sources produisent un son grâce une excitation, ou *interaction*. Les sources et les interactions ont un type et une ensemble d'attributs.

Les sources sont organisées en arbre, ce qui permet de partager des attributs par héritage, et de contrôler les *sources primitives* (les sous-sources) par les sources de *haut niveau* (cf. Figure 3.2)

3.2.2 L'héritage d'attribut

Les attributs sont hérités par les sources inférieures : si un attribut n'est pas défini par une source, celle-ci reprend la valeur de celui de sa source mère. Certains attributs sont relatifs, ce qui signifie que la valeur d'un tel attribut pour une source est une combinaison des attributs de ses parents. Ce principe reprend celui du système de fenêtrage X Window : les positions des fenêtres à l'écran sont déterminées par rapport aux positions des fenêtres parentes. De même, dans ENO, le gain final d'une source est la somme des gains de ses parents et du gain propre à cette source. Ainsi, on peut contrôler le gain d'un ensemble de source en modifiant le gain d'une source parente commune.

3.2.3 Les sources primitives

Les sources primitives sont les feuilles de l'arbre. Elles sont pour l'instant au nombre de trois (leurs attributs sont indiqués entre parenthèses) : des objets (taille, matériau), des machines (taille, vitesse, travail maxi, travail mini, rapport) et des échantillons (nom du fichier). Il y a deux interactions sur les objets, l'impact (force) et le frottement (force et vitesse), et une sur les machines et les échantillons, la mise en marche (travail maxi, vitesse) et la lecture (nombres de boucles). Les attributs de gain, de position en x et en z sont communs à toutes les sources.

L'une des difficultés lors de la conception d'un tel serveur, est de faire en sorte qu'il produise des échantillons à une vitesse supérieure à celle de la fréquence d'échantillonage. Sinon, des «clics» seront audibles lors de la reproduction. C'est pour cela que les algorithmes de synthèse se doivent d'être performant.

Gaver a conçu deux algorithmes [Gav90b] :

les objets : Gaver constate que le son produit par des objets ont le même spectre pour les deux interactions (impact et frottement). Il décrit donc les objets à l'aide d'une banque de filtres qui accentue ces fréquences dont les caractéristiques dépendent de celles de l'objet (taille et matériau). Le signal traité par cette banque est soit une sinusoïde de courte durée (impact), soit un bruit blanc (frottement).

les machines : Pour ce son, la synthèse FM est utilisée. En effet, une machine produit un bruit qui est assez répétitif et qui semble avoir un mouvement cyclique. En utilisant la modulation autour de la porteuse, on peut reproduire ces caractéristiques.

Les sons des objets sont d'un grand réalisme, alors que ceux des machines le sont moins. Ceci est dû aux approches différentes qu'a eu Gaver pour les concevoir. Pour le premier cas, il a étudié le son de plusieurs objets de taille, de forme et de matériau différents, ce qui lui a permis d'en définir les principales propriétés. Pour le deuxième cas, il n'a pas étudié de vraie machine (pour la bonne raison qu'il y en a une multitude et qu'aucune n'est vraiment familière à tout le monde). Il s'est donc interessé aux caractéristiques qu'il voulait inclure et a créé un son qui les reprenait, sans qu'il existe vraiment. En fait, l'important n'est pas que les sons soient d'un grand réalisme : il suffit qu'on les reconnaisse et que leurs caractéristiques influent réellement sur le son produit.

Par ailleurs, les algorithmes de synthèse des sons d'objets et de machines permettent un contrôle des paramètres des sources et des interactions, même pendant que le son est joué. Cette particularité donne aux applications un moyen de modifier en temps réel un son, et de répercuter ainsi un changement d'état du système.

3.2.4 Les sources de haut-niveau

Deux types de sources de haut-niveau ont été implémentés : les *nœuds* et les *groupes*. Les nœuds sont de simples regroupement de sources sous un ancêtre com-

mun, ce qui permet de manipuler le gain et la position de l'ensemble facilement. Les groupes sont des nœuds qui permettent de jouer des sources de deux façons : séquentiellement et parallèlement. Il est prévu un troisième type de séquencement, le rebond.

3.2.5 L'espace sonore

Toutes les sources sont positionnées dans l'espace sonore, quelque soit leur type. Comme nous l'avons vu, cet espace est en deux dimensions : la notion de hauteur n'est pas incluse dans ENO, car elle nécessiterait et des calculs trop importants et le port d'un casque (ou l'utilisation de quatre hauts-parleurs), contrainte que l'on souhaite éviter.

Le positionnement droite-gauche est obtenu en faisant varier les amplitudes des canaux stéréo : plus un son se situe sur un des côtés, plus l'écart de volume entre le haut-parleur droit et gauche est important.

L'impression d'éloignement fait appel à deux techniques. La première est l'atténuation du volume en fonction de la distance. Plus le son est faible, plus il nous paraît lointain. La deuxième part de la constatation que les fréquences graves sont moins sensibles à la distance que les fréquences aigues : un son lointain est plus étouffé. Il faut donc filtrer le son en lui enlevant un certain nombre de ses plus hautes fréquences en fonction de la distance.

Ces techniques ont l'avantage d'être peu gourmandes en calcul, pour des résultats très satisfaisants.

3.2.6 Développements futurs

Les auteurs de ENO pensent rendre celui-ci un peu plus qu'un simple serveur d'icônes auditives : l'intégration de sons parlés, de connexion MIDI avec des synthétiseurs, ou même d'*earcons* doit être réalisée. Ceci transformerait ENO en un support de toute information présentée auditivement.

Chapitre 4

Utilisation et extension du modèle sonore

Ce chapitre est consacré au travail que j'ai réalisé durant mon stage. Il se divise en trois parties, chacune concernant un problème particulier d'interface : les barres de défilement, la gestion des modes et la gestion des fenêtres. Pour chaque problème exposé, une solution sonore est proposée. La conception des sons et leur complexité sont analysés. Enfin, pour deux de ces problèmes, Brewster [Bre94] a des solutions auxquelles sont comparées celles présentées ici. Les implémentations de ces solutions ont été réalisées comme des extensions à ENO.

4.1 Les ascenceurs

Dans cette section, nous allons montrer comment un problème d'interface peut être réglé grâce à l'apport du son, et en particulier grâce à une icône auditive.

4.1.1 Description du problème

Nous connaissons tous la *barre de défilement*, l'*objet interactif* qui permet de faire défiler une vue d'un document trop grand pour tenir entièrement à l'écran (cf. Figure 4.1). On peut l'utiliser de différentes façons. L'utilisateur peut cliquer sur les flèches du haut et du bas, cliquer sur la zone de défilement ou faire glisser l'ascenseur à l'endroit désiré. Ces manipulations ne sont pas sans problème :

Se trouver hors de la zone de défilement : Quand on manipule l'ascenseur, on peut être amené a sortir de la zone qui prend en compte le déplacement. Le fait de sortir de cette zone annule l'effet escompté, c'est-à-dire attribuer une nouvelle position à l'ascenseur. Lors du relâchement du bouton de la souris, le texte ne défilera pas : l'utilisateur, qui peut très bien ne pas s'être aperçu de sa mauvaise manipulation, ne comprendra pas ce qui s'est passé.

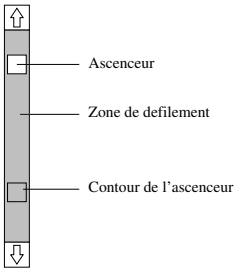


FIG. 4.1 - *Une barre de défilement*

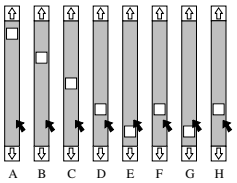


FIG. 4.2 - *Effet de bond sur place : l'ascenseur saute de part et d'autre du curseur en F, G et H*

État de la position : Lors du défilement, il est difficile de repérer sa position dans le document.

Les sauts : On a vu que l'on pouvait manipuler l'ascenseur en cliquant dans la zone de défilement. Ceci déplace l'ascenseur selon une valeur arbitraire, en général l'équivalent d'une page. Si le déplacement est grand par rapport à la taille de l'ascenseur, celui-ci fait des « bonds ». Si le pointeur se trouve entre deux bonds, l'ascenseur saute au-dessus du curseur indéfiniment, car pour une position, il est au-dessus du curseur, et pour l'autre il est au-dessous (figure 4.2). L'utilisateur, qui regarde défiler son document, ne s'aperçoit pas qu'il regarde toujours les deux même pages : à partir d'un certaine vitesse de défilement il n'y a guère de différences entre les pages, et il est souvent difficile d'en reconnaître une.

Ces trois problèmes sont dus au fait que la manipulation et le contrôle sollicitent tous les deux la vision. En effet, il faut surveiller ce que l'on fait sur la barre de défilement, et regarder le document pour savoir où l'on en est dans le défilement. Malheureusement, la vue n'est pas partageable : on ne peut concentrer son attention visuelle que sur une seule chose. Comme de toute façon la surveillance du document est obligatoire dans cette opération, il faut un autre moyen de surveiller la barre.

4.1.2 Une solution sonore

On a vu que le média visuel est occupé par la surveillance du document. Pourquoi ne pas utiliser le son pour surveiller l'état de la barre de défilement ? C'est la

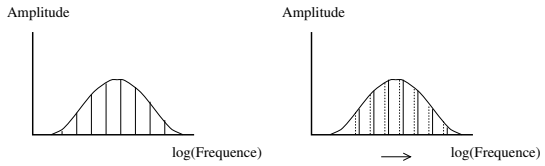


FIG. 4.3 - *Composition d'un Sheppard-Risset Tone : pour un son qui « monte », les partials glissent en suivant la courbe en cloche.*

solution que je propose : une imitation du son d'un ascenseur. Ce son utilise une illusion acoustique découverte par Sheppard et Risset : les *Sheppard-Risset Tones* [Ris77].

4.1.3 Les Sheppard-Risset Tones

Tout le monde connaît des illusions d'optique, comme la roue du carrosse qui tourne à l'envers au cinéma. Pourtant, bien peu de gens connaissent leurs analogues sonores. Les Sheppard-Risset Tones en sont un : ce sont des sons qui nous donnent l'impression de « monter » ou de « descendre » en hauteur indéfiniment.

Comment sont-ils fabriqués ? Ils se composent de plusieurs partiels (cf. Chapitre 1) dont le spectre semi-logarithmique suit une courbe en cloche. Il suffit de faire « glisser » ces partiels tout en conservant la forme de la courbe pour obtenir cet effet. Prenons l'exemple d'un son qui monte (Figure 4.3). Les partiels graves ont une amplitude faible, ce qui fait que leur « arrivée » se fait progressivement, sans que l'on s'en aperçoive. De même, les partiels les plus aigus disparaissent subrepticement. Pourtant, inconsciemment, le mouvement de ces partials nous font croire que le son monte, alors que globalement il reste à la même fréquence. On peut donc utiliser ce son indéfiniment tout en ayant l'impression qu'il évolue.

4.1.4 Le son d'ascenseur

Pour synthétiser un tel son, il suffit donc d'émettre des partiels à des fréquences précises et de les faire évoluer. Il y a plusieurs paramètres :

La fréquence de base C'est la fréquence moyenne du son. Plus cette fréquence est basse, et plus le son évolue dans les fréquences graves.

La direction C'est le sens de déplacement des partiels, à droite ou à gauche.

La vitesse C'est la vitesse de déplacement des partiels.

Le nombre de partiels

La répartition des partiels C'est le rapport entre un partiel et son voisin (slon une échelle logarithmique).

Pour cet algorithme, j'ai approximé la forme de cloche par une pyramide : les calculs sont plus simples, pour des résultats similaires. Le son produit est une sorte de « buzz », imitant plutôt imparfaitement le son d'un ascenseur réel (on dira que c'est un ascenseur de Star Trek), mais l'important n'est pas là : ses caractéristiques sont intéressantes pour l'utilisation souhaitée.

Il faut maintenant faire correspondre aux caractéristiques du son des attributs qui puissent s'adapter à sa source, à savoir le déplacement de l'ascenceur. La fréquence de base sert à déterminer l'*étage* auquel se trouve l'ascenceur : plus la fréquence est haute, plus l'ascenceur sera haut. La direction de déplacement des partiels correspond à la *direction* de marche de l'ascenceur. Enfin, on détermine la *hauteur* du bâtiment dans lequel évolue l'ascenceur en nombre d'étages.

La vitesse, le nombre des partiels et leur répartition ne font pas partie des attributs de haut-niveau de l'ascenceur. Les deux derniers parce qu'ils ne correspondent à rien intuitivement et que leur variation n'est pas très significative. Il faut donc les fixer : pour un son simulant bien les Sheppard Tones, six partiels au moins sont nécessaires. Utiliser plus de partiels n'apporte rien et rend le son pénible. Leur répartition ne doit pas être trop étendue, sous peine de couvrir trop de spectre, et donc de ne plus pouvoir utiliser la fréquence de base efficacement. Un rapport de 1.2 entre les fréquences semble la bonne solution. Quant à la vitesse, elle n'est pas un attribut sémantique : seule une valeur fixée et bien choisie permet l'efficacité des Sheppard-Risset Tones (si elle est trop rapide ou trop lente, l'effet de glissement est annulé).

Comme les partiels sont au nombre de six, il faut calculer six sinusoïdes par échantillons. Mais l'utilisation de tables préalablement calculées ne pénalise pa trop la performance de l'algorithme. De plus, l'utilisation d'une pyramide à la place d'une cloche permet des calculs plus rapides.

4.1.5 Utilisation dans les barres de défilement

Le premier problème était de savoir si l'on se trouvait toujours dans la zone de défilement. Avec l'usage d'un son, quelqu'il soit, il est facile de le résoudre : il suffit en effet de produire un son lorsque tout marche bien, et de laisser place au silence lorsque l'utilisateur n'est plus dans la zone adéquate. On utilise ici le fait que le moyen de contrôle du document est différent de celui de la surveillance de la barre. C'est d'ailleurs la justification principale de l'utilisation du son dans les interfaces.

Le troisième problème est celui des « sauts » de l'ascenseur de part et d'autre du curseur. En utilisant les Sheppard-Risset Tones, on connait à tout instant la direction de déplacement. Donc si l'on fait correspondre le déplacement de l'ascenseur graphique avec celui de l'ascenseur sonore, l'effet de bond se traduira par un va-et-vient sonore entre descente et montée.

Le second problème (déterminer la position de la vue dans le document) est résolu par la fréquence globale du son. En associant une fréquence grave au début du document et une fréquence haute à sa fin, il suffit de définir l'écart de fréquence entre deux pages et de jouer la fréquence intermédiaire correspondant à la position

de l'ascenseur. Pour ce qui est du choix des fréquences, il y a deux alternatives :

1. soit les fréquences de base sont fixées une fois pour toutes, quelles que soit la taille du document, et l'on adapte l'écart à celle-ci. Ceci convient bien aux documents de petites taille. Malheureusement, pour les gros documents, on ne pourra pas suivre efficacement le déplacement de l'ascenseur, car la différence de hauteur entre deux pages ne sera pas assez importante.

2. soit l'écart est fixé et l'on définit les fréquences de base en fonction du nombre de pages. Ici, il faut définir un écart suffisamment important pour arriver à discerner deux pages voisines. Mais encore une fois, les gros documents donneront des fréquences de base soit trop basse (difficilement audible), soit trop haute (difficilement supportable).

Le fait que le son utilise les Sheppard-Risset Tones nous fait pencher vers la première solution. En effet, le problème était l'impossiblité de déterminer la direction de déplacement dans un gros document. Mais les Sheppard-Risset Tones indique justement ces déplacements : si l'on se déplace, la hauteur globale évolue peu, mais l'impression de déplacement est présente. Le fait de fixer les hauteurs de début et de fin permet d'apprendre les positions relatives pour tous les ascenceurs : pour un son de telle fréquence, l'utilisateur saura où il se trouve, quelle que soit la taille du document.

4.1.6 Comparaison avec la solution de Brewster

Brewster [Bre94] utilise une earcon pour sonoriser la barre. Le timbre des sons est celui d'un orgue électronique. Le défilement vers le haut se traduit par un son aigu (assez fort pendant un court instant pour prévenir l'utilisateur, puis plus faible), un son grave traduisant un déplacement vers le bas. Cette technique permet d'éviter les sauts et la sortie de zone. En ce qui concerne la position, comme pour l'ascenceur, on lui fait correspondre une hauteur de son.

Les deux méthodes sont à peu près équivalente. L'apport de celle présentée ici est le fait qu'elle est plus intuitive au niveau du déplacement : les Sheppard-Risset Tones donnent vraiment l'impression que l'on descend ou que l'on monte, alors que le rapport entre les deux hauteurs utilisées par Brewster est artificiel.

4.1.7 Conclusion

Le son présenté répond donc aux problèmes posés. Cependant, même s'il a un côté plus naturel que celui de Brewster, il reste assez artificiel : malgré ses attributs de haut niveau, il se classe à la limite des icônes auditives. Enfin, il faudrait une expérimentation contrôlée auprès des utilisateurs pour réellement évaluer cette technique.

4.2 Les Environnements

4.2.1 Description du problème

Un des problèmes bien connus des interfaces graphiques est la gestion des modes. Les modes sont les états d'un système qui correspondent à des interprétations différentes d'une même action d'un utilisateur. Par exemple, dans VI, l'éditeur de texte UNIX, le fait de taper un « d » dans le mode « éditer » affichera un « d », alors que taper un « d » en mode « commande » actionnera la commande « delete » (effacement). Le problème avec les modes est que l'utilisateur peut, soit avoir manqué l'événement relatif au changement de mode, soit l'avoir oublié. Par exemple, dans le premier cas, une personne qui ne sait pas très bien taper au clavier ne regarde pas l'écran, et peut ne pas s'apercevoir du changement. Dans le deuxième cas, si l'on abandonne son ordinateur un moment, pour une pause par exemple, on peut revenir en oubliant le mode dans lequel on l'a laissé.

Plusieurs défauts d'interface sont des problèmes de mode :

Les boîtes de dialogue modales : Ce sont des objets graphiques dont on doit s'occuper avant toute autre chose : toute action qui ne leur est pas destinée n'est pas prise en compte. Ainsi, si dans un traitement de texte, une boîte s'ouvre demandant une confirmation d'une action (bouton « ok » et « cancel »), toute touche du clavier autre que *Enter* ou tout clic de souris autre que sur les boutons de la boîte sera perdu.

Les fenêtres Dans certains gestionnaires de fenêtres, il suffit de positionner le curseur au-dessus d'une fenêtre pour la rendre active. Ceci implique que si pour une raison ou une autre, le curseur glisse sur le fond d'écran ou sur une autre fenêtre sans que l'utilisateur ne s'en aperçoive, les actions de ce dernier ne seront pas redirigées vers l'application propriétaire de la fenêtre voulue.

Plusieurs solutions graphiques ont tenté de résoudre ces problèmes. Malheureusement, c'est justement le fait que ces solutions soient graphiques, donc surchargeant ce média, qui les rende inefficaces.

4.2.2 Une solution sonore

Ici encore, l'utilisation du son peut résoudre en partie le problème des modes. On a vu que les erreurs de l'utilisateur étaient dues soit à une inattention, soit à un oubli. Le fait que l'oubli soit une des causes implique que la notification de l'état du système doit être permanente. L'un des avantages du graphisme est que ce média se prête bien à la notification permanente : une icône ou une couleur différente n'est pas perturbante. Malheureusement, ces informations sont souvent insuffisantes, car elles ne sont pas affichées au point d'attention.

Le problème est d'émettre un son en continu sans pour autant qu'il devienne insupportable. En s'inspirant de l'approche écologique, il m'est apparu que l'on

pouvait utiliser des sons de notre environnement naturel. Lorsque l'on se trouve immergé dans une environnement sonore, si celui-ci n'est pas perturbant ou ennuyeux, on a tendance à ne plus faire attention aux sons émis : leur perception tombe en arrière-plan de notre conscience. En revanche, un changement notable de cet environnement fait remonter sa perception en avant-plan de notre conscience.

Le problème des fenêtres peut donc être résolu grâce à des environnements sonores. En effet, si à chaque fenêtre on associe un environnement, le changement de fenêtre active s'accompagne du changement de l'environnement actif correspondant. Comme les sons sont émis en permanence, on ne peut pas se tromper de fenêtre.

Quant au problème des modes, sa résolution découle du précédent. On a vu que l'on pouvait différencier les applications à l'aide d'environnement. Les sons émis tombent dans l'arrière-plan de notre conscience. Pour avertir d'un changement de mode, il suffit de modifier légèrement l'environnement, de manière à ne pas faire croire que l'on a changé d'application. La modification fait revenir la perception des sons en avant-plan de notre conscience. Le but recherché, c'est-à-dire prévenir l'utilisateur, est bien atteint. Mais cette modification doit être permanente, pour ne pas que l'utilisateur oublie le mode dans lequel il est.

Ainsi, on définit deux états pour un environnement : *calme* et *perturbé*. Par exemple, le mode normal d'un traitement de texte est l'édition : lorsque l'on écrit, on entend un environnement calme. L'appartition d'une boîte de dialogue se traduit par la perturbation de l'environnement. Tant que la boite reste à l'écran, empêchant l'édition, l'environnement reste perturbé. Dès que le problème est traitée, l'environnement revient en mode calme (figure 4.4).

4.2.3 Exemples d'environnement

Actuellement, j'ai implémenté deux environnements :

La Mer Cet environnement simule un bors de mer. En mode calme, on entend des sons de vagues se brisant sur la grève. Quand l'environnement est perturbé, le souffle du vent s'y ajoute.

StarWar Cet environnement simule le bruit d'un vaisseau spatial. En mode calme, on entend la combustion d'un réacteur et des bips évoquant les sons d'un ordinateur de Star Trek. En mode perturbé, les sons précédant sont gardés, mais des sons de lasers, d'explosions et d'autres vaisseaux sont ajoutés, donnant l'impression que l'on assiste à une bataille spatiale.

Il est très important de soigner la conception des sons, car ils perçus continuement. S'ils sont ennuyeux à écouter, les utilisateurs voudront les supprimer. En ce qui concerne l'environnement StarWar, pour l'instant, la technique utilisée est l'échantillonnage, c'est-à-dire que ces sons ont été précédemment enregistrés. Malheureusement, à cause de cette technique, ces sons ne sont pas variés ce qui fait que ra-

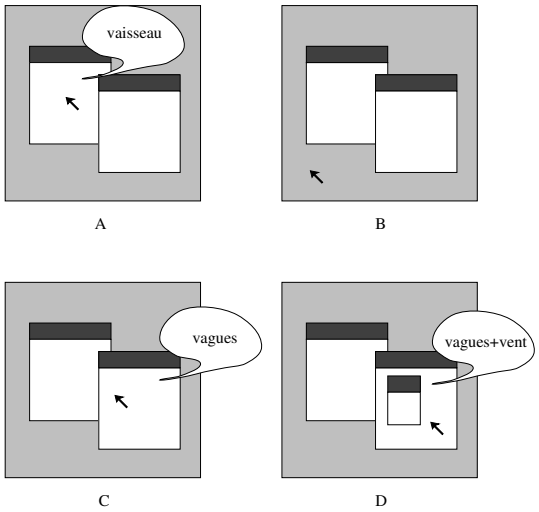


FIG. 4.4 - *Utilisation d'Environnements Sonores. En A, le pointeur est sur une application produisant un son de vaisseau spatial. En B, le pointeur est sur le fond : aucun son n'est produit. En C, il s'agit d'une autre application, dont l'environnement est la mer. En D, l'environnement est « perturbé » à cause de l'apparition d'une boîte de dialogue modale.*

pidement la monotonie s'installe. Je compte synthétiser prochainement les sons de cet environnement pour pouvoir les contrôler et éviter cette monotonie.

Pour l'environnement Mer, les sons de vagues et de vent sont synthétisés, ce qui les rend très variés. On peut contrôler la taille et la forme des vagues, ainsi que la nature de la plage sur laquelle elles s'échouent. En tirant au hasard pour chaque vague ses caractéristiques, on produit des sons tous différents. Pour le vent, on définit sa force, et les écarts de souffle sont calculés au hasard en fonction de cette force. Ce son est continu et ne nécessite aucun contrôle pour le rendre varié, à moins que l'on ne veuille qu'il forcisse ou qu'il faiblisse.

Il reste beaucoup de sons naturels à imiter, qui n'ont pas été implémenté faute de temps. Néanmoins, on peut déjà penser à certains environnements comme :

La Nature : Chants d'oiseaux en mode calme, et vent en mode perturbé ;

Le Monde Sous Marin : Bruit de fond, de bulles et de sonar en mode calme, sirène de sous-marin en mode perturbé ;

etc.

Il est évident que l'environnement StarWar n'est pas « naturel » au sens propre (de plus, il n'y a pas de bruits dans l'espace). Mais c'est un son qui n'a pas besoin d'être appris : tout le monde dans sa vie a vu un film ou une série avec des vaisseaux spatiaux. Ce genre de sons est à rapprocher des *genre sound* de Cohen [Coh93],

qui construit des correspondances entre les événements informatiques et les sons associés provenant de la télévision ou du cinéma. De plus, comme on l'a vu, ces sons ne doivent pas être des imitations parfaites, le fait que l'on puisse les reconnaître facilement suffit.

4.2.4 Construction des sons

J'ai crée deux algorithmes de synthèse : les vagues et le vent. Ce sont essentiellement des bruits blancs (c'est-à-dire contenant tout le spectre possible) filtrés par un filtre IIR («Infinite Impulse Response» ou filtre récursif) variable au cours du temps. On peut définir un filtre passe-bande, c'est-à-dire laissant passer certaines fréquences du spectre par l'équation :

$$y = G \times (x_n - R \times x_{n-2}) + b_1 \times y_{n-1} + b_2 \times y_{n-2}$$

avec :

$$\begin{aligned} R &= e^{\frac{-\pi \times B}{S}} \\ G &= 1 - R \\ b1 &= 2 \times R \times cos(\frac{2 \times \pi \times f}{S}) \\ b2 &= -(R \times R) \end{aligned}$$

S est la fréquence d'échantillonnage, y_i le ième échantillon de sortie, x_i le ième échantillon d'entrée, f la fréquence qui est accentuée par le filtre, et B la largeur du pic de fréquence accentuée. Il suffit de faire varier les deux derniers paramètres (f et B) pour avoir un bon contrôle sur le filtre :

```
tant qu'il reste des échantillons à produire
   calculer B(n) et f(n)
   si B(n) ou f(n) a changé par rapport à l'iteration précédente
      calculer R, G, b1 et b2
      calculer y
      jouer l'échantillon y
   finsi
   n = n+1
fin tantque
```

Les vagues

Pour le bruit de vague, j'ai retenu trois enveloppes :

Volume : il est contrôlé par cinq points répartis le long du son (figure 4.5. La vague grossit d'abord (volume haut), puis s'écrase sur la plage (volume haut) : il faut

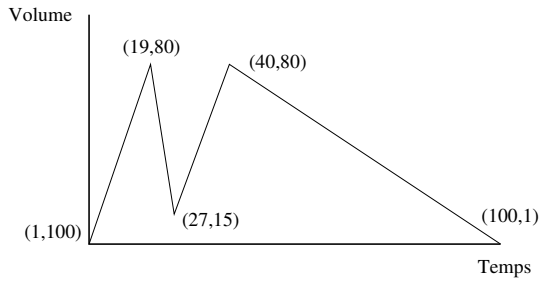


FIG. 4.5 - *Enveloppe de volume. L'unité de l'abscisse est le pourcentage de la longueur du son et l'unité de l'ordonnée le pourcentage du volume maximum.*

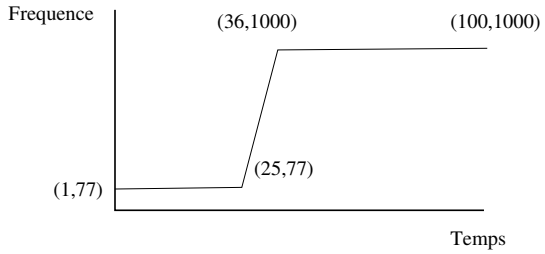


FIG. 4.6 - *Enveloppe de fréquence. L'unité de l'abscisse est le pourcentage de la longueur du son et l'unité de l'ordonnée est le Hz.*

bien séparer ces deux pics, donc on y ajoute un «trou». Enfin, la vague meurt. Les coordonnées des cinq points sont indiquées sur la figure 4.5.

La fréquence : à son arrivée, la vague gronde, d'où un son grave. Ensuite, la vague s'écrase, produisant un son plus aigu. On peut donc réaliser cette enveloppe avec quatre points (figure 4.6).

La largeur de bande : plus la larguer de bande est grande, plus on a une impression de «bruit». Quand la vague s'écrase, il faut simuler le frottement sur le sable : une bonne façon de le faire est d'introduire plus de bruit à la fin du son (figure 4.7).

Le bruit blanc est synthétisé en tirant des valeurs au hasard. Le son final est produit par le filtrage du filtre IIR dont les paramètres S et B suivent les enveloppes de fréquence et de largeur de bande, et dont le volume suit l'enveloppe de volume.

Pour rester dans la ligne des Auditory Icons, il faut décrire une vague avec des paramètres «naturels». J'en ai retenu trois :

La taille : elle contrôle la durée du son. Comme les points définissant les enveloppes se répartissent le long du son selon des pourcentages, la forme globale n'est pas changée.

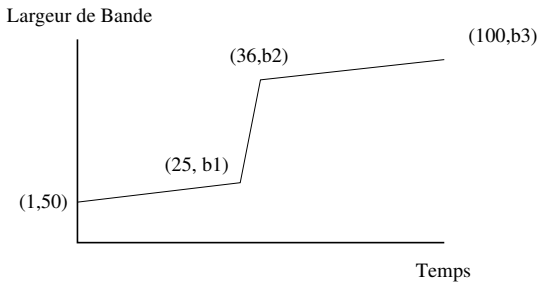


FIG. 4.7 - *Enveloppe de bande de fréquence. L'unité de l'abscisse est le pourcentage de la longueur du son et l'unité de l'ordonnée est le Hz. Les b_i varient entre 400 Hz et 4000 Hz. Leur valeurs dépendent du paramètre forme de la vague*

La forme : c'est un paramètre qui contrôle les valeurs des ordonnées des trois derniers points de l'enveloppe de largeur de bande. Il ne correspond pas à une entité descriptible d'une vague, mais il sert à introduire un peu de variété : deux vagues de formes différentes ne finiront pas de la même façon. Le problème est de créer 3 valeurs ($b_1,\ b_2,\ b_3$) à partir d'une seule (la *forme*). Il suffit de dire que chaque b_i peut prendre quatre valeurs: 64 permutations sont donc possible entre les b_i. Soit

$$\beta_i = \frac{forme}{4^{(i-1)}} \pmod 4$$

avec $i \in \{1, 2, 3\}$. Chaque β_i est donc compris entre zéro et trois. La valeur finale de b_i est donnée par :

$$b_i = 400 \times (1 + \beta_i^2)$$

ce qui donne quatre valeurs possibles :400Hz, 800Hz, 2400Hz et 4000Hz.

La plage : ce dernier paramètre ne fait pas partie de la source, mais de l'interaction, c'est-à-dire de l'écrasement de la vague sur la plage. Il peut prendre quatre valeurs correspondant à la nature de la plage (sable, graviers, galets, rochers...). Il agit en fait sur le bruit : la valeur tirée au hasard est maintenue plus ou moins longtemps (elle est maintenue pendant *plage* échantillons, *plage* étant compris entre 1 et 4), modifiant ainsi le bruit obtenu.

En créant des sons de vagues successives avec des caractéristiques *taille* et *forme* tirées au hasard, on peut simuler la mer.

Le son obtenu est à comparer avec celui de *machines* de ENO. On a vu que ces machines n'existaient pas réellement, et les sons produits n'étaitent pas d'un grand réalisme. Cependant, en les entendant on les associe immédiatement à une machine, dont les caractéristiques sont transmises par celles du son. On obtient un

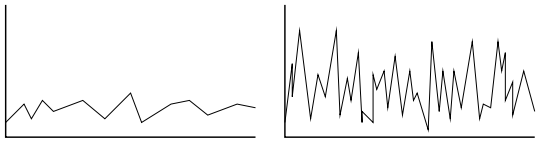


FIG. 4.8 - *Le Vent :courbe des fréquences. Le vent représenté à gauche est plus faible que celui de droite.*

résultat similaire avec le bruit des vagues : on reconnait tout de suite la nature du son, et on sait si une vague est plus grosse qu'une autre, ou qu'elles n'ont pas les même formes.

Le vent

La simulation du bruit du vent fait appel aux même techniques que pour les vagues. Le même filtre est utilisé, mais on fixe la largeur de bande (à 60Hz) pour ne faire varier que la fréquence stimulée, selon une enveloppe de fréquence similaire à celle des vagues (c'est-à-dire des points reliés par des droites) dont les points ont des coordonnées choisies au hasard. Ainsi, au temps t_n correspond une fréquence f_n et au temps t_{n+1} une fréquence f_{n+1}. La fréquence instantanée f au temps t est interpolée linéairement entre (t_n, f_n) et(t_{n+1}, f_{n+1}) :

$$f = (t - t_n) \times \frac{f_{n+1} - f_n}{t_{n+1} - t_n} + f_n$$

Quand $t = t_{n+1}$, on recommence en initialisant t_n à la valeur de t_{n+1} et f_n à celle de f_{n+1}, et en choisissant de nouvelles valeurs au hasard pour t_{n+1} et f_{n+1}.

Les valeurs f_{n+1} et t_{n+1} choisies au hasard dépendent du paramètre *force* du vent, dont la valeur maximum est 100 :

$$f_{n+1} = 100 + rand \times force \times 10$$

et

$$t_{n+1} = t_n + rand \times (110 - Force) \times 500$$

$rand$ est un nombre choisi au hasard compris entre 0 et 1. Plus le vent est fort, plus les variations sont rapides et importantes, donc plus les différences entre t_n et t_{n+1} sont petites et plus les différences entre f_n et f_{n+1} sont grandes. Des exemples de tracés d'enveloppe sont donnés en figure 4.8. Le fait que l'on choisisse des valeurs au hasard permet d'obtenir des effets de souffle très varié.

Complexité des algorithmes

Ces algorithmes nécessitent une opération *cosinus* et une *exponentielle* à chaque modification du filtre, ce qui est relativement coûteux. Néanmoins, il est possible de diminuer le nombre de ces opérations, en diminuant le nombre de changement de

paramètres du filtre, surtout quand ces variations ne sont pas significatives. De plus, l'emploi d'une table précalculée du *cosinus* peut réduire encore le temps de calcul.

4.2.5 Comparaison avec la solution de Brewster

Brewster [Bre94] propose une solution pour le problème des fenêtres : à chaque fenêtre est associé un timbre distinct. Un son continu est joué avec peu de volume pour ne pas être ennuyeux quand une fenêtre est active. Lors d'un changement de fenêtre, un son plus fort est joué de façon à être sûr que l'utilisateur a été averti.

La solution présentée ici est plus naturelle, dans la ligne des icônes auditives. De plus, elle résoud le problème des modes.

4.2.6 Conclusion

Les environnements sonores permettent de différencier les applications et deux états de chaque application, ce qui semble suffisant : il suffit que l'utilisateur sache que l'application n'est pas dans son état normal. On pourrait représenter des degrés dans l'état de perturbation d'un système et associer des sons aux différents degrés. Cependant, cela nécessiterait un apprentissage précis de chaque environnement, donc un effort de la part de l'utilisateur. Il faudrait aussi trouver plusieurs sons pour chaque environnement, tâche relativement difficile voire impossible.

Par ailleurs, on peut trouver d'autres utilisations pour ces environnements. Ainsi, Gaver suggére l'utilisation de sons naturels pour aider à la navigation dans un système dont la représentation graphique occupe une surface supérieure à celle d'un écran [GS93]. Si l'on dispose des environnements sur cette surface et que le volume de chacun est inversement proportionnel à la distance qui le sépare du curseur, l'utilisateur peut se repérer facilement et agréablement.

Enfin, le son du vent peut servir à faire passer des informations telles que le débit d'une connexion. Par exemple, si l'on sonorisait Netscape, la violence du vent serait fonction de la vitesse de chargement d'un document.

4.3 Les Pièces

On a vu dans le chapitre 2 que l'approche écologique consistait à définir un son comme une *interaction* de *matériaux* à un *endroit* dans un *environnement*. Cette section concerne l'implémentation d'un environnement dans ENO, les *pièces*. Dans ENO, les sons sont produits comme si la source etait à l'air libre. Dans une *pièce*, les sons sont transformés, comme si les sources étaient enfermées. Ainsi, on distinguera les sources non seulement par le son qu'elles produisent, mais aussi par l'environnement dans lequel il se produit.

La correspondance entre une *pièce* et une fenêtre graphique permet de mieux gérer un environnement multi-fenêtres. Si un son est joué, la façon dont il est transformé peut nous aider à repérer la fenêtre dans laquelle se trouve la source l'origine de ce son. Imaginons que l'on entende le son d'un choc d'un maillet sur un bout de

bois. Selon que celui-ci est plus ou moins réfléchi, on peut dire qu'il provient d'une fenêtre plus ou moins grande. Si la réflexion est sourde, c'est que le son provient par exemple d'une fenêtre en arrière plan, ou de telle application.

4.3.1 Description

Un son produit dans une pièce a des caractéristiques différentes d'un son produit dans un environnement sans obstacle. Dans un endroit clos, le son se propage et vient frapper les murs. Ceux-ci réfléchissent tout ou partie du son, comme le ferait un miroir avec de la lumière. Il y a donc un écho plus ou moins marqué selon les dimensions de la pièce. Pour simuler le chemin des ondes sonores dans une pièce, il existe plusieurs techniques dont :

Le lancer de rayons : À l'instar de son homologue visuel, le lancer de rayons sonore consiste à calculer les chemins des ondes de la source à nos oreilles.

Les filtres : Cette technique consiste à simuler le fond sonore restant après le son initial à l'aide d'une banque de filtres [Moo83].

Les calculs mis en jeu lors du lancer de rayons sont énormes et ne peuvent faire l'objet d'une implémentation permettant le temps réel.

La deuxième solution, si elle est moins ambitieuse, est donc plus raisonnable et largement suffisante pour nos besoins. Les concepteurs de l'algorithme sont partis de la constatation qu'à partir d'un moment assez court, l'enchevêtrement des ondes réfléchies par les murs était tel qu'il était complètement inutile de calculer avec précision ces réflexions, comme pouvait le faire le ray-tracing. Il est plus facile de simuler le bruit ambiant résultant avec une batterie de filtres aux caractéristiques judicieusement choisies.

4.3.2 L'algorithme

Les types de filtres utilisés sont au nombre de deux :

Le filtre en peigne passe-bas : Un filtre en peigne (ou *comb filter*) est un filtre IIR [DLH88] comportant une ligne à délai de D échantillons et un retour de coefficient $g < 1$ (figure 4.9). Intuitivement, il permet de simuler un écho qui baisse progressivement en intensité. Un filtre en peigne passe-bas (ou *low-pass comb filter*) est un filtre dans lequel est inséré un filtre passe-bas : selon la valeur du coefficient de retour $g1$, on peut simuler une décoloration du signal réflechi (figure 4.9).

Le Allpass Filter : ce filtre a la particularité de laisser passer toutes les fréquences, c'est-à-dire que le spectre n'est pas modifié (figure 4.10). En revanche, les phases des fréquences sont changées en fonction de ces fréquences, ce qui se traduit par un «étalement temporel» du son, rendant celui-ci plus réaliste.

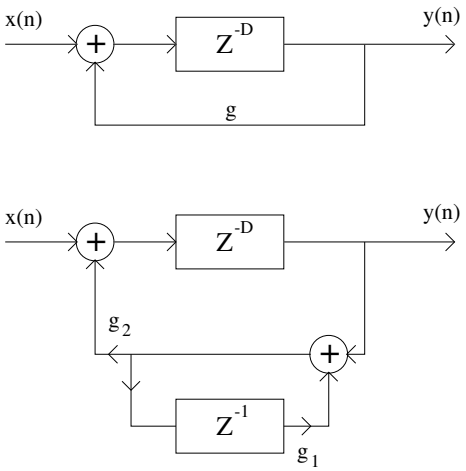


FIG. 4.9 - *Un Filtre en Peigne (en haut) et un Filtre en Peigne Passe-Bas (en bas)*

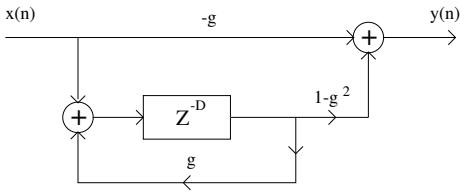


FIG. 4.10 - *Un Allpass Filter*

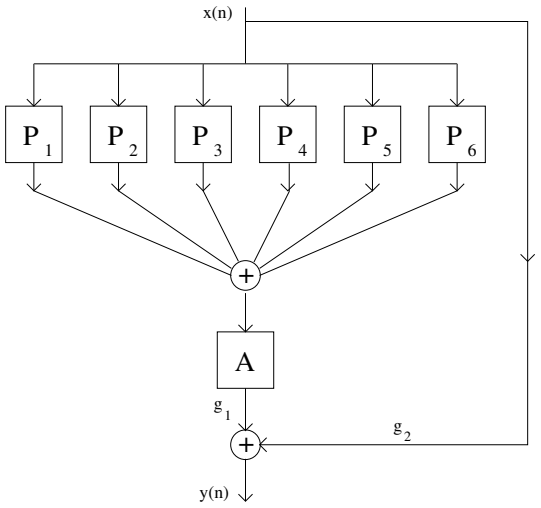


FIG. 4.11 - *Banque de Filtres pour une pièce.*

On peut simuler un écho avec une batterie de *filtres en peigne* et le faire passer dans le *allpass filter* pour l'étaler et simuler l'enchevêtrement des ondes évoqué plus haut. Les auteurs de cet algorithme [Moo83] ont constaté que l'emploi de six *filtres en peigne* suffisait à rendre réaliste le son obtenu. Les filtres sont disposés comme le montre la figure 4.11. On fait passer le signal dans les six *filtres en peigne* dont les valeurs de D et g sont différentes, puis la somme des signaux de sortie est traitée par le *allpass filter*. Le signal obtenu est pondéré par $g1$, et le signal initial par $g2$: le son final est la somme des deux sons.

Pour rester cohérent avec le principe d'ENO et des Auditory Icons, il faut contrôler les pièces par des caractéristiques « naturelles ». J'en ai retenu trois :

La largeur et la profondeur : La largeur et la profondeur servent à définir la taille de la pièce. De cette taille dépend la longueur de la réverbération. De plus, elles définissent les limites des positions des objets contenus dans la pièce. Ces positions sont donc relatives, et décrivent des pourcentages : un objet de position $(90, 10)$ dans une pièce de position (x_p, y_p) est en fait à $(0.9 \times l_p + x_p\ ,\ 0.1 \times p_p + y_p)$.

La texture des murs : Elle permet de définir la « coloration » des réflexions des sons (perte de certaines fréquences).

Moorer [Moo83] avoue que son algorithme présente quelques défauts, en particulier l'extrême dépendance du résultat par rapport aux variables des filtres. Ainsi, l'idéal serait que les délais D des filtres en peigne soient tous premiers entre eux, pour être sûr qu'aucun phénomène de résonnance ne se produise. Cependant, le fait que ces paramètres se doivent d'être variable rend très difficile la vérification de la primalité. Néanmoins, les phénomènes observés ne sont pas trop gênants. Une répartition de 1:1.5 des différences entre les délais semblent donner des résultats satisfaisants. Intuitivement, la largeur et la profondeur influent sur la durée de la réverberation, et sur sa rapidité à arriver. Ces paramètres subjectifs sont contrôlés par les délais D. Leurs valeurs sont calculées en fonction du périmètre de la pièce. Sachant que la largeur et la hauteur maximum est 100, la formule donnant les D_i en ms est :

```
d[1] = 1+((largeur+profondeur)/3);
diff = 2+d[1]/8;
for( i = 2; i¡=6;i++)
  d[i] = d[i-1]+diff;
  diff = diff*1.5;
```

D_1 est compris entre 1 et 70 ms, et la différence de départ entre 2 et 10 ms : ces valeurs permettent d'obtenir des sons réalistes.

La coloration dépend des coefficients g_1 des filtres en peigne passe-bas : plus ils sont grands, plus les fréquences hautes sont atténuées. Sachant que la texture a une valeur comprise entre 1 et 10, la formule donnant la valeur des coefficients est :

```
g1[1] = (wall*90)/1000.0;
```

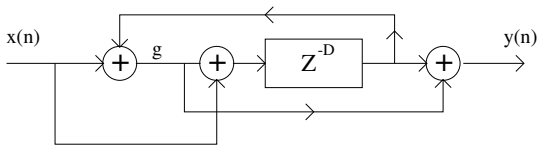


FIG. 4.12 - *Un Allpass Filter à Une Multiplication*

```
diff = 0.016;
for( i = 2; i¡=6;i++)
  g1[i] = g1[i-1]+diff;
```

Le filtre en peigne passe-bas est un filtre IIR dont le coefficient global est $g_2/(1-g_1)$. Pour que ce filtre soit stable, il faut que ce coefficient soit inférieur à 1. En se donnant un coefficient $g < 1$ global à tous les filtres en peigne (afin de leur donner un comportement cohérent), on peut calculer g_2 : $g_2 = g \times g_1$. g est fixé à 0,83.

Les valeurs des paramètres du filtre allpass sont fixes : $g = 0,7$ et le delai est de 6 ms.

Une convention a été décidée pour la restitution du son dans ENO: le son original est localisé selon la méthode vue dans le chapitre consacré à ENO, tandis que le son dû à la réverbération n'est pas localisé (il semble provenir du « fond » de l'espace sonore).

4.3.3 Complexité

Le filtre en peigne passe-bas nécessite deux multiplications et deux additions. Comme il y a six filtres de cette sorte, il faut douze multiplications et douze additions pour la banque entière. Le filtre allpass requiert trois multiplications et deux additions sous la forme montrée figure 4.10. Cependant, il existe une autre forme nécessitant moins de calcul (figure 4.12) : une seule multiplication et trois additions. Le coût total de l'algorithme s'élève donc à treize multiplications et quinze additions.

4.3.4 Utilisation

Le concept de pièce peut être utilisé pour établir une analogie avec celui des fenêtres graphiques. Cependant, d'autres applications peuvent faire l'objet d'emploi de *pièces*. Ainsi, Gaver donne l'exemple d'un message électronique arrivant dans une boîte aux lettres [Gav86] : le son joué nous renseigne sur la nature du message (texte, application...) et sur ses caractéristiques (taille...). L'écho accompagnant ce son peut témoigner du taux de remplissage de la boîte au lettres (écho large, semblant provenir d'une pièce vide = boîte vide). Le déplacement d'un fichier sur un disque peut aussi nous renseigner sur le taux d'occupation du dique, suivant l'écho accompagnant le choc du fichier sur le disque.

Conclusion

Dans ce stage, des solutions sonores ont été apportées a plusieurs probème d'interaction homme-machine. Ces solutions s'inspirent du concept des icônes auditives [Gav86]:

1. Les ascenceurs : grâce aux Sheppard-Risset Tones, ils permettent de savoir la direction de déplacement et de se repérer dans le document.

2. Les environnements : ces sons continus pallient le problème des modes. Ils sont conçus de manière à *prévenir* l'utilisateur d'un changement de mode et à *fournir un contrôle continu* sur le mode courant d'un système. Mais ils permettent aussi d'aider à la gestion d'un environnement multi-fenêtres.

3. Les pièces : conçues pour résoudre le problème des fenêtres, elles peuvent être employées pour d'autre usages, en particulier pour informer d'un taux d'occupation.

Bien qu'elles aient été créées dans un but précis, certaines des solutions ont d'autres usages. Cependant, des tests auprès d'utilisateurs doivent être effectués, afin de vérifier leur efficacité.

De plus, des recherches doivent être menées pour étendre le domaine des sons présentés ici. Ainsi, la création d'autres environnements permettrait d'en avoir un nombre suffisant à proposer aux applications, et de pouvoir gérer au mieux les systèmes multi-fenêtres. Ou encore, des études poussées de sons complexes à synthétiser (eau qui coule, par exemple) doivent être faites afin de couvrir un plus large panorama de sons naturels [Gav93].

Enfin, des effets (comme la *distorsion*, ou *l'effet Doppler* pour les sources qui se déplacent) et leur utilisation pourraient être étudiés et faire l'objet d'une implémentation dans ENO.

Bibliographie

[ano73] anonymous. Ameican national psychoacoustical terminology. Technical Report S3.20, American National Standards Institute, New York, 1973.

[BLG94] Michel Beaudouin-Lafon and William W. Gaver. ENO: Synthesizing Structered Sound Spaces. In *Seventh Annual Symposium on User Interface Software and Technology (UIST'94)*, pages 49–57, November 2-4 1994.

[Bly90] Sara Bly. Communicating with sound. In *Proceedings of CHI'90 Conference on Human Factors in Computing Systems*, volume 4, pages 5.4–5.10. ACM, April 1990.

[Bre94] Stephen Anthony Brewster. *Providing a Structured Method for Integrating Non-Speech Audio Into Human-Computer Interfaces*. PhD thesis, University of York, 1994.

[BSG90] Meera M. Blattner, Denise A. Sumikarwa, and Robert M. Greenberg. Earcons and Icons: Their Structure and Common Design Principles. In *Proceedings of CHI'90 Conference on Human Factors in Computing Systems*, volume 4, pages 5.48–5.74. ACM, April 1990.

[Cho73] John Chowning. The synthesis of complex audio spectra by means of frequency modulation. *Journal of the audio Engineering society*, 21:526–534, 1973.

[Coh93] J. Cohen. "Kirk here:" Using genre sounds to monitor background activity. In *Adjunct Proceedings, ACM Conference on Human Factors in Computing Systems*, pages 63–64. ACM, April 24-29 1993.

[DLH88] David J. DeFatta, Joseph G. Lucas, and William S. Hodgkiss. *"Digital Signal Processing: A System Design Approach"*. Wiley, 1988.

[Edw90] A. Edwards. Soundtrack. In *Proceedings of CHI'90 Conference on Human Factors in Computing Systems*, volume 4, pages 5.30–5.46. ACM, April 1990.

[Gav86] William W. Gaver. Auditory Icons : Using Sound in Computer Interfaces. In *Proceedings of CHI'86 Conference on Human Factors in Computing Systems*, volume 2, pages 167–177. ACM, 1986.

[Gav89] William W. Gaver. The SonicFinder: An Interface That Uses Auditory Icons. In *Proceedings of CHI'89 Conference on Human Factors in Computing Systems*, volume 4, pages 67–94. ACM, 1989.

[Gav90a] William W. Gaver. Acoustics and Psychoacoustics. In *Proceedings of CHI'90 Conference on Human Factors in Computing Systems*, volume 4, pages 2.1–2.15. ACM, 1990.

[Gav90b] William W. Gaver. Synthetizing Auditory Icons. In *Human Factors in Computing Systems, INTERCHI'93*, pages 228–325. ACM, April 24-29 1990.

[Gav91] William W. Gaver. Sound Support For Collaboration. In *Second European Conference on Computer-Supported Collaborative Work (E-CSCW'91)*, pages 293–308, September 24-27 1991.

[Gav93] William W. Gaver. What In The World Do We Hear ? An Ecological Approach to Auditory Event Perception. *Journal of Ecological Psychology*, 5(1), 1993.

[GS93] William W. Gaver and R. B. Smith. Auditory Icons in Large-Scale Collaborative Environments. In *Human-Computer Interaction - Interact'90*, pages 725–740. D. Diaper and D. Gilmore and G. Cockton and B. Shackel, 1993.

[Kev89] Kevin Karplus and Alex Strong. Digital Synthesis of Plucked-String and Drum Timbres. In *The Music Machine*, pages 467–479. Curtis Roads, 1989.

[Moo83] James A. Moorer. About this reverberation business. *Computer Music Journal*, 7(3):6–15, 1983.

[Moo90] F. Richard Moore. *"Elements Of Computer Music"*. Prentice Hall, 1990.

[Pet90] Ivars Peterson. Some labs are alive with ... the sound of data. In *Proceedings of CHI'90 Conference on Human Factors in Computing Systems*, volume 4, pages 5.12–5.14. ACM, April 1990.

[Ris77] Jean-Claude Risset. Paradoxes de hauteur. In *IRCAM Report no. 10*. IRCAM, 1977.

[Roa94] Curtis Roads. Initiation à la synthèse par modèles physiques. *Les cahiers de L'IRCAM*, pages 145–169, 1994.

[SC92] Julius O. Smith and Perry R. Cook. The Second-Order Digital Waveguide Oscillator. In *Proceedings of the International Computer Music Conference*, 1992.

[S.H91] S.H. Foster and E.M. Wenzel and R.M. Taylor. Real Time Synthesis of Complex Acoustic Environment. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, October 20-23 1991.

[Smi92] Julius O. Smith. Physical modelling using digital waveguides. *Computer Music Journal*, 16:74–91, 1992. Special issue on physical modelling.