Sons d'environnement naturel pour le suivi d'activités d'arrière-plan

Stéphane Conversy

Laboratoire de Recherche en Informatique - URA 410 du CNRS
LRI - Bâtiment 490 - Université de Paris-Sud
91405 Orsay Cedex, France
conversy@lri.fr

RÉSUMÉ

Cet article concerne l'utilisation d'icônes auditives pour le suivi d'activités d'arrière-plan. Pour être utilisables, ces icônes auditives doivent présenter des caractéristiques essentielles, telles que la variété et la non-intrusion. Cet article présente deux sons naturels, ceux du vent et des vagues, se prêtant bien à cette tâche. Ces sons sont synthétisés en temps réel, et ont été implémentés dans le serveur ENO. La technique de synthèse utilisée est décrite pour chaque son, ainsi que la complexité de leur calcul à chaque échantillon. Enfin, des exemples d'applications sont proposés.

MOTS CLÉS : Suivi d'activités, icônes auditives, sons naturels.

INTRODUCTION

La présentation sous forme graphique d'informations concernant l'état d'un système peut conduire à une surcharge de nos moyens visuels. D'une part, la taille de l'écran est faible par rapport au champ visuel humain : des informations comme la taille des fichiers dans une interface icônique ne sont pas représentées car elles alourdissent l'affichage. D'autre part, le centre d'interêt de la vision est unique : ainsi, l'utilisation des barres de défilement pour contrôler la vue sur un document impose de partager son attention visuelle entre la barre et le document. Cette surcharge du canal visuel peut être réduite par l'utilisation du canal sonore. Ansi, le SonicFinder [8] montre que la taille des fichiers dans une interace icônique peut être présentée de façon sonore lorsque l'on sélectionne un fichier, et l'usage du son lors de l'interaction avec une barre de défilement permet de limiter les changements de points d'attention lors du défilement d'un document [2].

On peut classer l'usage des sons dans les interfaces homme-machine en trois grandes catégories [1] :

- feed-back des actions de l'utilisateur ;
- information sur l'état du système ;
- collaboration avec d'autres utilisateurs du système.

Le SonicFinder et les barres de défilement sonorisées illustrent les utilisations dans la première catégorie. D'autres travaux comme Ear et Arkola [9] se sont interessés à la troisième catégorie. La seconde catégorie inclut la notification d'évènements comme l'arrivée de courrier électronique, mais également le suivi d'activités d'arrière-plan comme la réception, l'impression ou la transmission de documents, l'évolution de la charge de la machine ou du réseau, etc. Dans cet article, nous nous intéressons aux sons continus (par opposition aux sons ponctuels) permettant d'assurer le suivi d'activités d'arrière-plan.

Le son possède deux caractéristiques essentielles qui le rendent particulièrement adapté à cette tâche. D'une part, plusieurs sons indépendants peuvent être perçus simultanément. En vision, un objet se situant derrière un autre est caché, donc les informations qu'il donne ne sont plus accessibles. En revanche, un son peut être entendu simultanément avec un autre, à condition qu'il n'y ait pas d'effets de masquage dus aux spectres et à la localisation de ces deux sons [3]. En se concentrant sur ce son, l'associant ainsi à une *stream* [3], on peut en déduire les informations qu'il contient. D'autre part, un son continu n'évoluant pas est entendu inconsciemment. Si le son n'évolue pas, il est toujours présent, en arrière-plan de notre conscience, mais il revient en avant-plan quand une des ses caractéristiques change. Donc, s'il n'y a pas d'évènement, on oublie le son (mais on peut lui porter attention si l'on veut), et l'évènement « le son a changé » est signalé de lui-même en provoquant une prise de conscience de la part de l'auditeur.

Le fait que ces sons soient continus implique qu'ils ne doivent pas être trop distrayants ni trop répétitifs pour l'utilisateur. C'est pourquoi il faut choisir soigneusement les sons à utiliser et les produire de manière non-intrusive et non-monotone.

Nous avons choisi l'approche des icônes auditives [12], selon laquelle les sons quotidiens sont interprétés en terme des évènements physiques qui les produisent. Nous avons donc exploré l'usage de sons naturels comme les vagues, la pluie et le vent qui nous renseignent sur

l'état de notre environnement. Ces sons satisfont le premier impératif cité plus haut : lorsqu'ils ne correspondent pas à des éléments déchaînés, ils sont familiers et peu perturbants. Nous avons choisi de les synthétiser en temps réel et de définir des variables de contrôle correspondant aux caractéristiques de l'environnement qui les produit : taille des vagues, force du vent et de la pluie, etc. La suite de l'article est organisée comme suit. Dans la première partie, nous étudions les différentes solutions proposées pour sonoriser des activités d'arrière-plan. Ensuite, nous décrivons les algorithmes de synthèse des sons de vague et de vent. Enfin nous présentons quelques applications, et les développements futurs de ce travail.

SURVEILLER LES ACTIVITÉS D'ARRIÈRE-PLAN

Parmi les travaux de recherche concernant l'usage du son dans les interfaces, un certain nombre d'auteurs se sont interessés à la surveillance des activités continues.

Earcons

Les Earcons sont des mélodies dont les caractéristiques musicales (rythme, hauteur, timbre...) sont organisées hiérarchiquement, afin de pouvoir les classer en famille [4].

Un des problèmes des systèmes multi-fenêtres et multi-applications est que l'utilisateur peut avoir oublié la fenêtre active, soit parce qu'il n'a pas fait attention au pointeur de souris qui a glissé, soit parce qu'il n'a pas utilisé son ordinateur depuis un certain temps. Brewster [5] propose d'associer un son de timbre différent à chaque application, et de jouer ce son à un niveau situé juste au-dessus de la limite de perception.

Les problèmes de suivi d'activité ont surtout été résolus avec des icônes auditives, c'est-à-dire des sons plus ou moins réalistes évoquant ceux entendus dans la vie quotidienne. Ils font référence aux *causes* du son, c'est-à-dire aux sources sonores et aux interactions ayant produit le son, plutôt qu'aux caractéristiques physiques du signal sonore comme l'amplitude, la hauteur ou le timbre. C'est par le biais des caractéristiques de ces sources et de ces interactions inférées par l'auditeur que l'information est transmise [12].

SonicFinder

Dans le SonicFinder [8], Gaver accompagne la copie de fichier d'un son de bouteille qui se remplit. On peut connaître l'état du processus grâce au changement de hauteur du son au cours de la copie. Ainsi, une jauge de progression représentant le niveau atteint par la copie n'est pas nécessaire.

ARKola

Dans ARKola [9], des utilisateurs doivent collaborer pour surveiller une usine de fabrication de boissons. Gaver et Smith montrent que l'utilisation de sons indiquant l'état des machines de l'usine rend la tâche plus facile. Ainsi, chaque machine a un son propre correspondant à son fonctionnement normal, auquel s'ajoutent des sons spécifiquent quand un incident se produit. Lors de la simulation, jusqu'à quatorze sons sont produits simultanément, et permettent aux utilisateurs de surveiller efficacement l'état de la production. Cette expérimentation montre que le canal sonore est approprié à la surveillance d'activité, même si les utilisateurs sont parfois induits en erreur lorsque le trop grand nombre de sons ne permet pas de se rendre compte d'un évènement. Les sons utilisés sont échantillonnés, mais le nombre minimal de sons simultanés est suffisamment grand pour que la monotonie ne s'installe pas comme elle pourrait le faire avec un seul son. Les utilisateurs perçoivent le son général de l'usine, à partir duquel il est possible de dire si l'usine « tourne rond » ou pas.

ShareMon

Avec ShareMon [6], Cohen teste un moyen de surveiller le partage de fichiers sur une machine locale. Des sons sont associés aux différents états de la tâche, dont certains représentent le suivi d'activités d'arrière-plan : pour rappeler la présence d'un client, Cohen a successivement utilisé le bruit d'un raclement de gorge, un bruit rose (un bruit blanc filtré), le son d'un homme qui marche, et des sons de vagues. Les utilisateurs ont trouvé le son de vagues très agréable, certains ne l'ayant même pas remarqué. Les autres sons proposés sont loin d'avoir provoqué une telle réaction positive : ils ont soit été confondus avec les sons réels, soit été trouvés ennuyeux par les utilisateurs.

Sons de machine

Gaver a défini un algorithme de synthèse d'un son évoquant une machine [11]. Un tel son peut être associé à un processus dont les différentes phases contrôlent les caractéristiques de la machine, comme une compilation ou une impression [1]. Le fait qu'il soit synthétisé permet de le contrôler efficacement et de manière naturelle, grâce à des caractéristiques de haut-niveau : taille et vitesse de la machine, quantité de travail fournie.

Les illusions auditives

Enfin, Beaudouin-Lafon et Conversy utilisent une illusion auditive (les sons de Shepard-Risset [16]) pour sonoriser une barre de progression [2]. Ce son donne l'impression que sa hauteur monte ou descend indéfiniment, alors qu'elle est globalement constante. Il ressemble à un ascenceur un peu particulier. Le son est synthétisé en temps réel et est contrôlable avec des paramètres de haut-niveau : vitesse et direction de la progression.

De nouvelles icônes auditives

Les icônes auditives a surtout été utilisée pour notifier des évènements, plutôt que des états. Ainsi, dans Ear [9], les utilisateurs d'un mediaspace sont avertis d'évènements se produisant dans un même batiment et susceptibles d'intéresser tout le monde. Les sons utilisés, pour ne pas être trop intrusifs, sont courts, et comportent peu de hautes fréquences. Le problème est de trouver des icônes auditives peu perturbantes. Ainsi, les râclements de gorge de ShareMon ont déplu à un nombre important d'utilisateurs.

Nous proposons d'utiliser des sons d'environnement naturel pour les tâches nécessitant un contrôle continu. En effet, dans la nature ces sons ne sont pas dérangeants (dans la limite où ils sont rendus avec un niveau raisonnable, et qu'ils ne dénotent pas un environnement très perturbé) et sont variés car leur évolution résulte essentiellement du hasard. Ainsi, on n'entend pas deux fois de suite le même son de vague, ni même un motif répétitif de vagues déferlantes.

Nous avons étudié deux sons d'environnement naturel : le son du vent et le son des vagues. Ces sons sont synthétisés en temps réel, et peuvent être contrôlés avec des paramètres de haut-niveau. La section suivante décrit la technique de synthèse utilisée.

SYNTHÈSE DE SONS NATURELS

Il existe deux moyens de produire des sons sur un ordinateur : l'échantillonnage et la synthèse.

L'échantillonnage consiste à enregistrer un son pour le rejouer ensuite. Cette technique produit des sons de haute qualité mais trop statiques et trop gourmands en mémoire. Il est pratiquement impossible de décrire un son avec des paramètres de haut-niveau lorsque l'on utilise un son échantillonné à moins d'enregistrer toutes les variations intéressantes pour les rejouer ensuite à la demande. Mais le nombre d'échantillons nécessaires devient vite rédhibitoire, ce qui conduit à réduire le nombre d'échantillons, donc à terme à une monotonie lors de la reproduction.

La synthèse consiste à calculer chaque échantillon du son à produire, à partir d'un algorithme de synthèse et de paramètres de contrôle. S'il est souvent difficile de synthétiser un son de façon suffisamment réaliste, lorsqu'on y arrive les avantages sont grands par rapport à l'échantillonnage, car les problèmes de variation et de mémoire sont éliminés. En revanche la qualité du son obtenu est souvent moins bonne car les algorithmes de synthèse sont coûteux en temps calcul, spécialement pour des fréquences d'échantillonnage élevées. Dans le cas qui nous concerne, la qualité intrinsèque du son n'est pas le facteur le plus important : il ne s'agit pas d'une écoute musicale pour laquelle la qualité sonore, voire de l'instrument et du musicien est prépondérante. Il suffit que l'on puisse reconnaître la nature du son et ses caractéristiques sans se tromper. De plus, comme le soulignent Gaver et Cohen, il arrive que des sons de bonne qualité soient considérés comme venant du monde réel et non de l'interface: il faut alors dégrader la qualité des sons présentés [6] [13] !

Les deux sons que nous avons dévelopé utilisent la technique de synthèse dite soustractive [15] : un bruit blanc est filtré par un filtre IIR (« infinite impulse response », ou filtre récursif) dont les caractéristiques évoluent suivant une enveloppe définie. C'est un filtre passe-bande (*Fig 1*), c'est-à-dire laissant passer certaines fréquences du (*Fig 2*). Son équation est :

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

avec

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

$$G = 1 - R$$

$$b_1 = 2 \times R \times \cos(\frac{2 \times \pi \times f}{S})$$

$$b_2 = -(R \times R)$$

où S est la fréquence d'échantillonnage, y_i le ième échantillon de sortie, x_i le ième échantillon d'entrée (ici le bruit blanc), f la fréquence accentuée par le filtre et B la largeur de bande du pic de fréquence accentuée [15]. En faisant varier f et B au cours du temps, on peut synthétiser un bruit de vent et de vague.

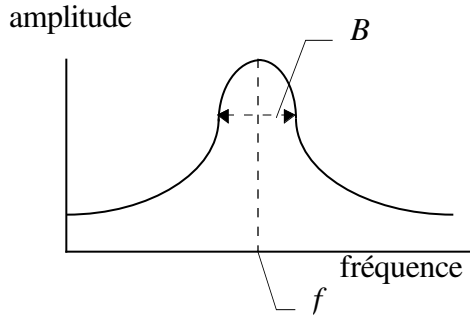

Figure 1: Spectre du filtre passe-bande. *B* dénote la largeur de bande et *f* la fréquence accentuée.

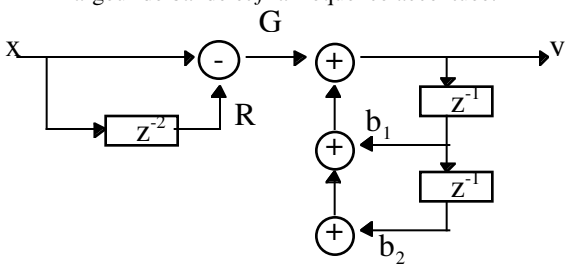

Figure 2: Schéma du filtre passe-bande.

Le vent

La largeur de bande permet de spécifier la quantité de bruit perceptible dans le son. Pour simuler le bruit du vent, il n'est pas nécessaire de faire varier cette valeur. En la fixant à une valeur de 60 hz, on obtient un bon compromis entre un son trop bruité et un son qui accentue trop la fréquence *f* (résultant en un sifflement peu réaliste). On ne fait donc varier que la fréquence de filtrage *f*.

L'enveloppe de fréquence est constituée de points successifs (temps, fréquence) reliés entre eux par interpolation linéaire. On tire les valeurs de ces points au hasard, tout en contraignant les valeurs obtenues selon la force du vent :

$$f_{n+1} = 100 + rand \times force \times 10$$
$$t_{n+1} = t_n + rand \times (110 - force) \times 500$$

avec f_{n+1} la prochaine fréquence à atteindre, t_{n+1} le temps auquel cette fréquence sera atteinte, *rand* la fonction qui tire au hasard une valeur comprise entre 0 et 1, et *force* la force du vent comprise entre 1 et 100. Plus la valeur de la force est grande, plus les écarts entre les temps sont petits et plus ceux entre les fréquences sont grands. Ceci renforce l'aspect erratique du vent lorsqu'il est fort. De plus, plus la force est grande, plus les fréquences sont hautes, correspondant à une plus grande vitesse de l'air (*Fig 3*). Le fait que l'on choisisse des valeurs au hasard permet d'obtenir des effets de souffle variés, empêchant ainsi le son d'être monotone.

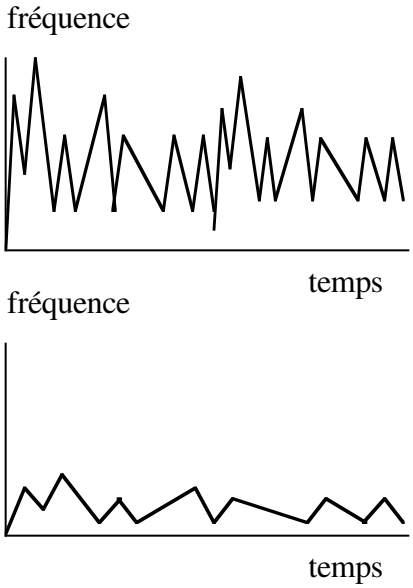

***Figure 3**:* Le vent, courbe d'évolution des fréquences en fonction du temps. La courbe du bas représente un vent faible, celle du haut un vent fort.

Les vagues

Le son de vague utilise trois enveloppes temporelles (*Fig 4*):

Enveloppe de volume: elle consiste en cinq points répartis le long du son. La vague s'écrase (volume haut) puis elle roule (le volume baisse) avant de mourir sur le sable (le volume augmente à nouveau).

Enveloppe de fréquence: à son arrivée la vague s'écrase en grondant (son sourd) puis elle roule sur le sable. On définit cette enveloppe avec quatre points.

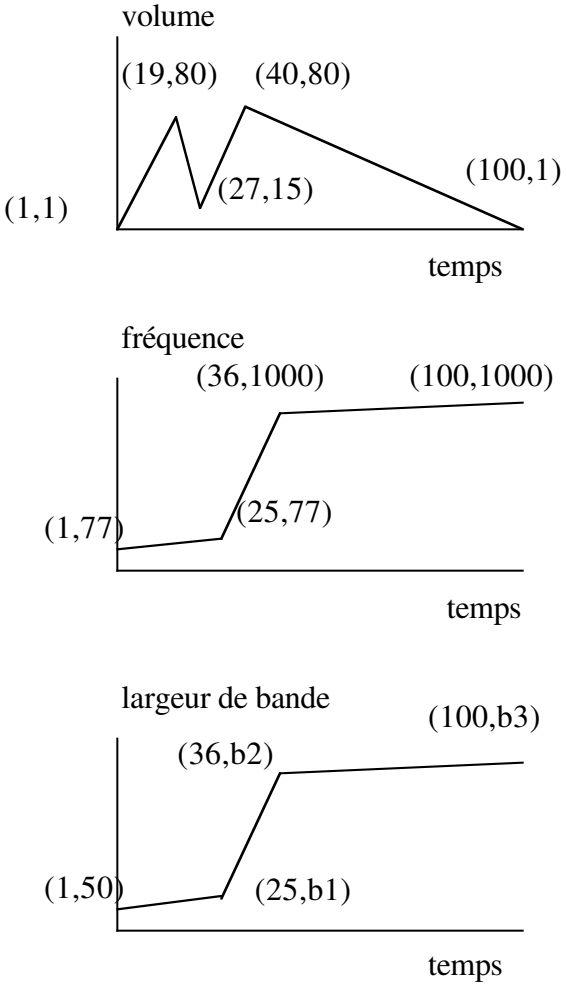

Figure 4: Les enveloppes de paramètres de filtre pour le son de vague. Les points de contrôle ont des abscisses en pourcentage de la durée totale du son.

Enveloppe de largeur de bande: plus la largeur de bande est grande, plus on a une impression de bruit. Quand la vague arrive sur la plage, il faut simuler le roulement du sable ou des galets : une bonne façon d'obtenir cet effet est d'introduire plus du bruit à la fin du son. Les valeurs des temps des points de contrôle peuvent varier d'une vague à l'autre pour les différencier plus facilement.

Ces trois enveloppes ont des points dont les abscisses sont des temps, en pourcentage de la durée totale du son. Trois paramètres de haut-niveau permettent de décrire un son de vague :

La taille: elle contrôle la durée totale du son. Plus la taille est grande, plus le son est long.

La forme: elle contrôle les abscisses des points de l'enveloppe de largeur de bande. La forme ne correspond pas à une caractéristique « réelle » d'une vague, mais sert à introduire un peu de variété. Les écarts entre b1, b2 et b3 sont construits en tirant une valeur au hasard et en la multipliant par une valeur plus ou moins grande selon la forme, qui peut prendre quatre valeurs (cf *Table 1*): ainsi, si la forme est égale à 1, l'écart est petit à la fois pour b_2-b_1 et b_3-b_2. b_1 est fixe à 400 Hz.

La plage: elle indique le type de plage sur laquelle s'écrase la vague (4 valeurs possibles de 1 à 4). Ce paramètre agit sur le bruit filtré : au lieu de tirer une valeur au hasard à chaque échantillon, on n'en tire une nouvelle que tous les 1, 2, 3 ou 4 échantillons.

forme	*b2-b1*	*b4-b3*
1	1000 x *rand*	1000 x *rand*
2	2000 x *rand*	1000 x *rand*
3	1000 x *rand*	2000 x *rand*
4	2000 x *rand*	2000 x *rand*

Table 1: valeur des différences entre b_i selon le paramètre *forme* de la vague.

En créant des sons de vagues successifs avec des tailles et des formes tirées au hasard, on simule le bruit de la mer. Le son obtenu permet de reconnaître facilement le son caractéristique d'une vague, et l'on peut dire si deux vagues ont la même durée ou la même forme.

Implémentation

Les sons décrits plus hauts ont été intégrés à ENO, un serveur implémentant un modèle de son structuré [1]. ENO s'inspire du concept des icônes auditives, bien que le modèle puisse représenter également des sons musicaux ou parlés. Les sons sont produits en créant des *interactions* avec des *sources.* Chaque source et chaque interaction est décrite par un type et des paramètres de haut-niveau. Les paramètres sont contrôlable en temps réel par l'application. Par exemple, une source de type *objet* est décrite par un *matériau* (par exemple du bois) et une *taille*. Une interaction possible avec une telle source est le *frottement*, caractérisé par une *vitesse* de frottement et la *rugosité* du contact. Les sources sont placées dans un plan horizontal faisant face à l'utilisateur. Les sons joués sont spatialisés en fonction de la position de la source qui les produit.

L'ajout des sons environementaux dans ENO consiste à définir le vent comme une source sans attribut ; une seule interaction est possible, le faire *souffler* avec une certaine *force*. Une vague est une source à deux attributs (taille et forme); elle peut *déferler* (interaction) sur une *plage* (attribut de l'interaction).

La complexité des calculs est un facteur important, car tous les sons de ENO sont synthétisés en temps réel et il faut laisser du temps à la machine pour d'autres tâches. Il faut donc minimiser les opérations simples comme l'addition et la multiplication, et éviter les appels de fonctions trigonométriques ou transcendentales.

Le calcul d'un échantillon à partir des coefficients du filtre (G, R, b_1 et b_2) requiert quatre multiplications, trois additions, et un appel à la fonction *rand* pour la génération du bruit blanc en entrée du filtre. Ces chiffres sont incompressibles, que ce soit pour le vent ou pour les vagues.

Le calcul des coefficients du filtre nécessite une opération *exponentielle* et *cosinus*, cinq multiplications (en ne comptant pas $2\pi/S$ et $-\pi/S$ qui sont des constantes) et une addition. Pour le vent, B est fixée donc on peut calculer une fois pour toutes R et G. Finalement, à chaque changement des paramètres du filtre, pour le vent, le calcul des coefficients peut être fait en deux multiplications et une opération *cosinus*. L'emploi d'une table précalculée permet d'éviter le calcul du *cosinus*. En ce qui concerne l'opération *exponentielle*, on peut restreindre B à des valeurs entières comprise entre 1 et 5000, et faire une table de 500 valeurs indexées par $B/10$ pour le calcul de R, sans nuire à la qualité du son produit. On arrive donc à un calcul comprenant quatre multiplications et une addition par échantillon pour les vagues.

	échantillon	contrôle
vent	4 mult, 3 add, 1 rand	3 mult, 1 add, 1 ind
vague	4 mult, 3 add, 1 rand	7 mult, 4 add, 2 ind

Table 2: Récapitulatif des coûts des algorithmes. Mult=multiplication, add=addition, rand=random, ind=indexation dans une table.

Le calcul des enveloppes requiert une multiplication et une addition par segment ($ax+b$). Pour le vent, il n'y a qu'un segment; pour les vagues, il y a trois enveloppes

donc on a besoin de trois multiplications et trois additions pour calculer les paramètres du filtre. Il est possible de diminuer la fréquence de calcul des paramètres pour les enveloppes à un dixième de la fréquence d'échantillonnage sans réduire la qualité du son de façon perceptible.

En résumé, pour les calculs des coefficients du filtre, il faut trois multiplications, une addition et une recherche dans une table pour le vent et sept multiplications, quatre additions et deux recherches dans une table pour les vagues (cf *Table 2*).

APPLICATIONS

Il existe divers domaines d'application dans lesquels les utilisateurs ont besoin ou bénéficieraient d'un suivi d'activités d'arrière-plan. C'est le cas notamment des applications de contrôle de processus. Dans ce type d'application, l'utilisateur doit s'assurer que le processus contrôlé fonctionne normalement, et le système doit le prévenir en cas de dysfonctionnement ou de fonctionnement anormal. L'usage d'alarmes, qui est le plus fréquent, n'est pas toujours approprié car une alarme est par nature événementielle et se déclenche lorsqu'une condition exceptionnelle est satisfaite, c'est-à-dire trop tard ou presque trop tard.

Bien souvent, des signes avant-coureur permettraient d'éviter le déclenchement de l'alarme si l'utilisateur pouvait avoir une représentation (visuelle, auditive ou autre) de l'ensemble du processus. L'expérience Arkola [9] a ainsi montré que les utilisateurs avaient une image auditive correspondant au fonctionnement normal de l'usine. Un autre exemple d'application est le contrôle du trafic aérien [14]. Des alarmes sont prévues lorsque des avions sont trop proches ou sont sur des trajectoires conflictuelles. Mais les contrôleurs cependant anticipent ces conflits et organisent en fait une partie de leur activité autour de leur gestion. Leur but est de diminuer la charge du trafic en réduisant le nombre de conflits potentiels.

Pour décider d'utiliser ou non des sons d'environnement dans une application, il faut prendre en compte trois caractéristiques importantes de ces sons. Premièrement, ils représentent des phénomènes naturels sur lesquels l'homme a peu ou pas de contrôle direct. Ils sont donc plus adaptés à représenter un état du système dû à une cause extérieure qu'un processus qui se déroule sous le contrôle de l'utilisateur.

Deuxièmement, ce sont des sons qui correspondent à des phénomènes naturels qui varient relativement lentement, qui ont une grande amplitude, et qui sont permanents. Ils sont donc plus adaptés au suivi d'un état du système dont les variations sont lentes.

Troisièmement, les sons les moins intrusifs (faible force du vent ou faible hauteur des vague) correspondent à un environnement calme, tandis que les sons les plus dérangeants correspondent à un environnement perturbé. Il serait donc inapproprié d'utiliser par exemple un son de vent pour représenter le débit d'un réseau : en général, on souhaite que le débit soit le plus grand possible, ce qui correspondrait à un vent fort donc dérangeant pour l'utilisateur.

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

Nous avons donc expérimenté l'usage du son de vagues pour représenter la charge d'un ordinateur multi-tâches, et le son de vent pour représenter la charge du trafic du réseau local. Plus l'ordinateur a une charge élevée, plus la mer est agitée ; plus le réseau est chargé, plus le vent est fort. En situation normale, non chargée, les sons ne sont pas intrusifs et passent pratiquement inaperçus. Lorsque l'un ou l'autre des sons change, l'utilisateur le remarque et peut en tirer des conséquences. Ainsi, lorsque le vent est fort, il vaut mieux sans doute éviter de générer du trafic réseau supplémentaire et ne pas « sortir » sur le Web.

Vers des environnements sonores

L'évaluation informelle que nous avons pu faire a mis en évidence le besoin de fournir des sons plus variés pour permettre une meilleure utilisation du canal audio. Par exemple, ShareMon [6] utilise différents sons pour contrôler le partage de fichiers. Mais ces sons ne sont pas liés entre eux par un facteur commun qui permettrait de les associer à une seule et même tâche. Si l'on souhaite surveiller plusieurs activités en même temps au moyen du canal audio, il faut pouvoir grouper ces sons par activité. Il faut pour cela concevoir des sons de façon telle que l'on puisse inférer qu'ils appartiennent au même environnement. Les sons naturels présentés ici sont ainsi une première pierre à la définition d'*environnements sonores*.

Un environnement sonore est un ensemble d'icônes auditives qui sont liées par la nature des sons (les événements physiques qu'ils représentent) et l'état de leur évolution (les caractéristiques des icônes évoluent ensemble). Par exemple, un environnement *maritime* comporterait des sons de vagues, de vent, des mouettes, des cornes de brumes, etc. L'augmentation de la force du vent ou la présence des cornes de brume indiquerait une perturbation de l'environnement, donc un évènement à prendre en compte. L'utilisation de plusieurs environnements permet d'associer ces évènements à des

tâches particulières. Ainsi, on pourrait définir des environnements naturels comme la nuit (grillon, hibou en mode calme, coq en mode perturbé), le feu (bruit sourd et craquements de bois puis sirène de pompiers), ou même des environnements de fiction comme l'espace (postcombustion et bips d'ordinateur puis communication de type talkie-walkie et rayons lasers), etc. Ces environnements sonores permettraient donc de combiner les sons continus et événementiels dans un cadre cohérent.

CONCLUSION ET TRAVAUX FUTURS

Les sons naturels que nous avons présenté dans cet article ont été conçus pour des tâches de contrôle continu d'activité. La technique utilisée pour les construire est relativement peu coûteuse en temps calcul et permet un bon contrôle des paramètres de haut-niveau de ces sons. L'usage de ces sons dans des applications réelles a montré qu'ils doivent être intégrés au sein d'environnement sonores plus complets et plus variés.

Il reste donc à trouver des algorithmes de synthèse efficaces pour d'autres sons d'environnement naturels (le feu, la pluie, une rivière qui coule etc.). Pour cela, on peut envisager des techniques de synthèse soustractive comme pour le vent et les vagues. Une forme de synthèse additive faisant appel à une transformée de Fourrier inverse semble cependant plus prometteuse pour les icônes auditives en général, en réunissant le réalisme des sons échantillonnés et le pouvoir de contrôle des sons synthétisés [7].

Une fois définis de tels environnements, des tests utilisateurs devront être conduits afin de voir si ces sons sont effectivement non-intrusifs et assez variés pour être utilisés dans une interface.

BIBLIOGRAPHIE

1. Michel Beaudouin-Lafon and William W. Gaver. ENO: Synthesizing Structered Sound Spaces. In *Proc. ACM Symposium on User Interface Software Technology, UIST'94*, pages 49-57, November 2-4, 1994.
2. Michel Beaudouin-Lafon and Stéphane Conversy. Auditory llusion for Audio Feedback. In *Companion Proceedings, ACM Human Factors in Computing Systems, CHI' 96*, Vancouver(Canada), pages 299-300, April 1996.
3. Bregman, A.S. (1990). *Auditory Scene Analysis. Cambridge*, Mass., MIT Press.
4. Blattner M., Sumikawa D., Greenberg R. Earcons and Icons: Their Structure and Common Design Principles. In *Human Computer Interaction*. 4(1) (1989). Pages 11-44.
5. Stephen Antony Brewster. Providing a Structured Method for Integrating Non-speech Audio Into Human-Computer Interfaces. *PhD thesis*, University of York, 1994.
6. Jonathan Cohen. Monitoring Background Activities. In *Proc. International Conference on Auditory Display, ICAD'92*, pages 499-531. Held in Santa-Fe, October, 1992.
7. Freed A., Rodet X., Depalle P. Synthesis and Control of Hundreds of Sinusoidal Partials on a Desktop Computer without Custom Hardware. In *Proc. International Conference on Signal Processing Applications & Technology, ICSPAT,* 1993. DSP Associates, Boston, MA.
8. William W. Gaver. The Sonic Finder: An Interface that Uses Auditory Icons. In *Human Computer Interaction*. 4(1) 1989. Pages 67-94.
9. William W. Gaver. Effective Sounds in Complex Systems: The ARKola Simulation. In *Proc. ACM Human Factors in Computing Systems*, CHI 1991. Held in New Orleans, April 28-May 2, 1991. New-York: ACM, 1991.
10. William W. Gaver. Sound Support for Collaboration. In *Proc. the Second European Conference on Computer-Supported Collaborative Work*. Held in Amsterdam, September 24-27, 1991. Dordrecht: Kluwer, 1991.
11. William W. Gaver. Synthesizing Auditory Icons. *Proc. Human Factors in Computing Systems, INTERCHI'93*. Held in Amsterdam, April 24-29, 1993. New-York: ACM, 1993.
12. William W. Gaver. What In The World Do We Hear ? An Ecological Approach to Auditory Source Perception. *Ecological Psychology* (5)1 (1993).
13. William W. Gaver and R. B. Smith. Auditory Icons in Large-Scale Collaborative Environments. In *Proc. Human-Computer Interaction, Interact'90*, pages 735-740. Amsterdam: North Holland, 1990.
14. Hopkins V.D. *Human Factors in Air Trafic Control*. Taylor & Francis, London, 1995.
15. F. Richard Moore. *Elements of Computer Music*. Prenctice Hall, 1990.
16. Risset J-C., Paradoxes de Hauteur, *IRCAM Report no. 10*, IRCAM, 1977.