

Rapport de TER

Sonification de signaux financiers

Par **Baptiste BOHELAY**, sous la responsabilité de **Laurent DAUDET**, du 11 octobre 2007 au 7 janvier 2008.

1-Sujet

Ce projet a pour but de « sonifier » à l'aide du logiciel Pure Data, des signaux financiers pour l'entreprise Arbitragis. Il s'est déroulé sous la responsabilité de Laurent DAUDET, chercheur en traitement du signal au LAM.

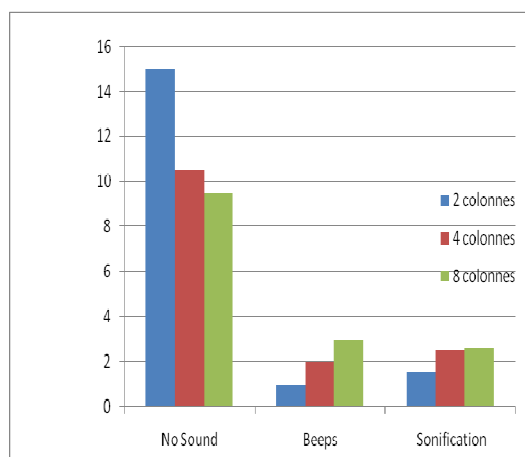
a-Qu'est ce que la sonification ?

Dans certain métiers ou dans le quotidien, nous recevons une multitude d'informations que nous traitons ; et nous agissons en fonction. Un géologue, par exemple, doit surveiller une multitude de données constamment pour pouvoir prévoir un séisme ou une irruption volcanique. La vue est notre sens le plus stimulé pour recevoir ces données. Le feu devient vert, je peux passer. La voie de mon train s'affiche, je peux me diriger vers ma rame. Et c'est tout à fait naturel puisque c'est le sens pouvant capter le plus de subtilités, avec le plus de précision. Mais nous sommes rapidement saturés d'informations visuelles, et notre oreille est là pour compléter notre vision. Il peut être alors utile de sonifier ces informations, en jouant sur les quatre dimensions du son, qui sont le temps (ou durée), la hauteur, le timbre et l'intensité.

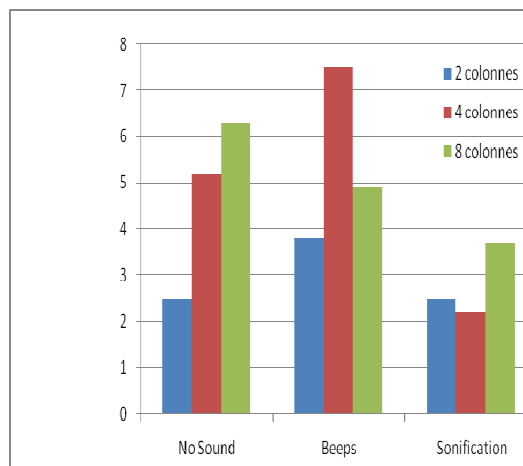
Une étude du Department of Psychological and Brain Sciences and Center for Cognitive Neuroscience, faite en 2004 au Dartmouth College [01] a porté sur l'impact d'une sonification sur notre vitesse de réaction face à une information et l'erreur de lecture de celle-ci. L'expérience consistait à mettre un sujet devant un écran contenant 2,4 ou 8 colonnes (selon l'expérience) d'une ligne de nombres. Le sujet devait déterminer, lorsqu'un chiffre changeait, de quelle colonne s'agissait-il et si le nombre était positif ou négatif. Pour simuler la distraction que peut subir un trader (par exemple par un coup de fil), le sujet avait, pendant l'expérience, devant les yeux 4 nombres et devait déterminer si l'un de ces quatre nombres était dans l'une des colonnes.

Trois conditions ont été observées. Dans la première, appelée « No Sound », la colonne dont le nombre avait changé, s'illuminait. Dans la deuxième, appelée « Beeps », lorsqu'un changement avait lieu, le sujet était averti par deux « bip » de même hauteur, plus l'illumination de la colonne. Dans le troisième cas, appelé « Sonification », en plus de l'illumination de la colonne changeante, deux sons prévenaient d'un changement. Le timbre d'un instrument était spécifique à une colonne précise. Le deuxième des deux sons était plus haut ou plus bas selon le signe du nombre.

Pourcentage de réponses perdues



Pourcentage d'erreur de signe



Les résultats obtenus montrent qu'un signal sonore, qu'il soit un simple bip ou qu'il soit plus complexe aide à être plus alerte puisque nous voyons que le pourcentage de réponses perdues passe de environ 12% sans son à environ 2% avec son. Nous voyons aussi que le changement de timbre aide à déterminer le signe du nombre puisque, quelque soit le nombre de colonnes à surveiller, les erreurs de signe sont moins élevées pour la sonification.

Les avantages psychomoteurs étant maintenant mis en évidence, nous pouvons insister sur le gain de temps et de confort qu'offre la compréhension auditive de ce qu'il se passe à l'écran. Avec une sonification intuitive et assez complexe, l'utilisateur pourra se détacher de son écran et suivre ce qu'il se passe tout en s'occupant d'autre chose.

La sonification est un champ de recherche qui a beaucoup progressé ces dernières années, à la fois d'un point de vue théorique et des applications, avec une conférence internationale annuelle sur le sujet (International Conference on Auditory Display).

b- Qu'est ce que le LAM ?

Le Laboratoire d'Acoustique Musicale est un organisme public de recherche sous la tutelle de l'Université Pierre et Marie Curie, du CNRS et du Ministère de La Culture et de la Communication. Les activités de recherche du LAM à caractère fortement pluridisciplinaire se regroupent selon trois thèmes : Instruments de musique et voix, techniques audio, cognition et perception[03]. Laurent a eu la gentillesse de me faire visiter les locaux qui regorgeaient d'instruments divers et variés. Je pense notamment au bureau concernant les recherches sur les supports d'enregistrements qui contenait, entre autre, un ancêtre du vinyle, de forme cylindrique qui devait dater de 1910 ! Une bibliothèque à disposition, c'est dans ces locaux que j'ai effectué une partie de mon travail.

c-Qu'est ce que Arbitragis ?

Arbitragis est une entreprise composée de 5 informaticiens et mathématiciens et dirigée par Tuan Nguyen, spécialiste de finance quantitative. Tuan était très intéressé par l'idée de sonifier les signaux de la bourse. Ils cherchent et exploitent des algorithmes qui, en fonction de plusieurs données vont automatiquement acheter ou vendre en bourse, en fonction d'informations qui leur proviennent toutes les 3 secondes. Comme dans tous les algorithmes, il y a une part d'erreur dans ces transactions. C'est pourquoi il est nécessaire d'être constamment en alerte sur ce qu'il se passe en bourse et sur les transactions faites par l'ordinateur. Tuan espère de la sonification plus qu'un avertissement de ce qu'il se passe. Il aimerait un logiciel qui permettrait, de faire le lien entre ce qu'on entend des signaux financiers et ce que fait l'algorithme.

d-Que sont ces signaux financiers ?

Voici les informations, ou signaux, que Tuan souhaite que je traite. Il s'agit du nom de l'action, de l'heure et de divers quantités et prix correspondants aux carnets d'ordres (vendeurs et acheteurs). Ces données sont actualisées toutes les 3 secondes :

CAC40,09:21:54,5819.5,3,2,5819,26,5820,24,5818.5,29,5820.5,14,5818,55,5821,26,5817.5,4
2,5821.5,22,5817,48,5822,11

(Les couleurs sont pour la lisibilité)

Ils sont ensuite affichés de façon plus lisible dans un tableau comme suit :

Situation à 09:21:54

	Bid Size	Bid Price	Last Price	Last Size	Ask Price	Ask Size
Level 5					5822	11
Level 4					5821.5	22
Level 3					5821	26
Level 2					5820.5	14
Level 1					5820	24
Level 0			5819.5	3		
Level 1	26	5819				
Level 2	29	5818.5				
Level 3	55	5818				
Level 4	42	5817.5				
Level 5	46	5817				

Les Bid sont les acheteurs. Les Ask sont les vendeurs. Les deux colonnes Size correspondent au nombre d'action demandées et proposées respectivement par les acheteurs (Bid) et vendeurs (Ask). Les colonnes Price sont les prix demandés et proposés. Par exemple, ici, 26 actions sont demandées à 5819\$, 29 à 5818.5\$, etc. Une transaction de 3 actions est en cours à 5819.5 \$.

Comment les traiter ?

En bourse, il existe deux termes pour désigner si le marché est actif ou en hausse (c'est bon pour nous), ou en baisse (idem) : respectivement Bullish et Bearish. Il existe plusieurs facteurs déterminant si le marché est bullish ou bearish. Et Tuan souhaite être averti des plus importants :

- Si la somme du nombre d'action des acheteurs est supérieure à la somme des actions des vendeurs, le marché est actif

Donc si :

$$\frac{\sum bid_size}{\sum ask_size} > 1 \rightarrow \text{Bullish}$$

$$\frac{\sum bid_size}{\sum ask_size} < 1 \rightarrow \text{Bearish}$$

- Si le prix transacté est celui **proposé par les bid**, c'est **Bullish**. Si c'est celui **proposé par les ask**, c'est **bearish**. Si ce n'est aucun des deux (comme dans l'exemple) c'est Bullish quand même car il y a des ventes : le marché est actif.
- Plus le nombre de transactions instantanées est élevé, plus le marché est bullish.

e-Qu'est ce que Pure Data ?

Créé par Miller Puckette, Pure Data est un environnement de programmation graphique en temps réel pour la création musicale et multimédia[04]. C'est un programme qui fait des programmes. Les possibilités de créations paraissent infinies. Assez semblable à Max MSP, les seules limites, dit-on, sont celles de l'imagination. Pure Data est utilisé, par exemple pour :

- modéliser des instruments électroniques (synthétiseurs, sampler, effets, séquenceur midi, ...)
- applications multimédias, interfaces, interactions (jeux, instruments, robotique, design d'interaction)
- Concerts, performance, compositions, installations (vidéo /sonore), conception sonore (sound design)
- Outil technique de mesures acoustiques
- Outil pédagogique (acoustique, synthèse, ...)

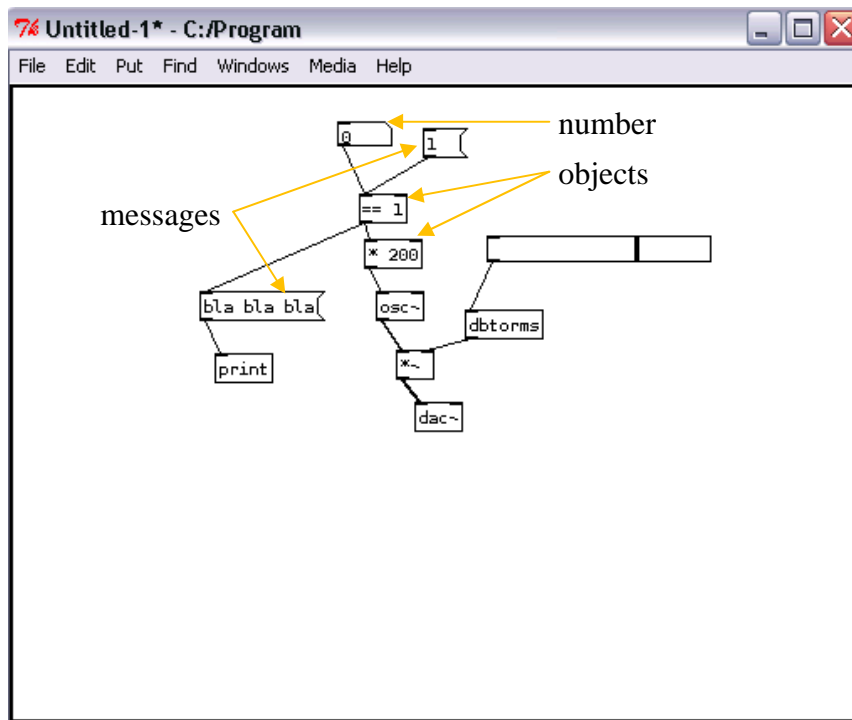
Ce logiciel est libre et marche sur la plupart des systèmes d'exploitation ce qui est un avantage conséquent, compte tenu qu'il sera utilisé par l'entreprise. Du fait que les signaux sonores doivent être traités en temps réel, c'est tout naturellement que nous nous sommes tournés vers ce programme.

Assez grossièrement, programmer sur Pure Data consiste à mettre des cases, que l'on remplit selon le rôle qu'elles vont jouer et qu'on relie à d'autres cases. La (ou les) sortie(s) de chaque cellule est(sont) en bas et les entrées en haut. La sortie peut être un signal sonore, un graphique ou bien des chiffres ou du texte. Il existe 5 types de cases que l'on peut reconnaître à leur forme:

- les « object », que nous utilisons le plus fréquemment car elles contiennent toutes les instructions
- les « number », qui contiennent des réels, et les envoie en sortie. Elles peuvent être modifiées par l'utilisateur, en mode d'utilisation ou par le programme.

- Les « message », qui ne sont pas modifiables en mode utilisation, et qui envoient leur contenu dès la réception d'une impulsion (ou « bang ») ou lorsqu'on clique dessus.
- Les « commentaires », les commentaires.
- Les « symbols », que je ne connais et n'utilise pas encore.

Exemple :



Une fois le programme fait, nous nous mettons en mode édition (utilisation). Nous pouvons alors faire varier avec la souris la valeur de la case « number » en haut à gauche ou cliquer sur le « message » 1 en haut à droite. La dernière valeur modifiée est envoyée à l'opérateur ==1, qui correspond à un if. Si l'une des deux valeurs est 1, la sortie de l'opérateur renvoie une **impulsion** (ou « bang ») et l'entier 1. L'impulsion fait que le message « bla bla bla », à gauche sera écrit sur la console. La valeur 1 est multipliée par 200, grâce à l'impulsion (sans « bang », pas de calcul), et est envoyée comme

fréquence à un oscillateur sonore (osc~). Tous les opérateurs audio contiennent le symbole « ~ ». Cette oscillation a comme amplitude 1, ce qui correspond au seuil maximum. Elle sera donc envoyée à un multiplicateur audio, qui va multiplier l'amplitude de sortie de osc~ par une valeur, que l'on peut régler avec le curseur de droite. Cette barre envoie une valeur allant de 0 à 128 (que l'on détermine dans les propriétés), que l'on convertit en dB avec l'opérateur dbtorms. L'amplitude finale est envoyée en sortie audio via l'opérateur dac~. Donc, si nous cliquons sur le 1 ou que nous mettons la case number à 1, nous entendrons un son pur de 200Hz et nous pourrions lire « bla bla bla » sur le moniteur.

Il est possible (et même vivement conseillé), de séparer son programme en plusieurs parties, qui communiquent les unes avec les autres, par souci de lisibilité.

Ce programme peut lire les formats audio midi, wave, lire des fichiers texte (des réels ou des caractères) ou une socket (des données provenant d'un autre logiciel, d'internet, etc.). Pour la lecture des fichiers audio, il est important de savoir que le logiciel ne modifiera pas leur structure, sinon les importants calculs excluraient le temps réel. C'est en fait la façon de lire le morceau que l'on va programmer, en agissant sur la vitesse de lecture, la répétition, etc.

L'annexe 1 contient un lexique de la syntaxe permettant de mieux comprendre le programme.

2- Journal

Premières idées :

Après un apprentissage de la syntaxe et une fois que j'ai compris l'essentiel des capacités du programme (grâce à de nombreux exemples, bien expliqués, inclus dedans), j'ai pu avoir une idée de ce que je voulais. Il se trouve que mon programme final n'a pas beaucoup à voir avec l'idée du début. Il est vrai que, compte tenu de mon manque d'expérience, le programme s'est fait un peu à tâtons. Mais j'avais pour idée directrice d'utiliser des sons que j'enregistrerais et mixerais moi-même. Premièrement, ça me permettrait d'allier à ce sujet, un de mes centres d'intérêt : l'électroacoustique, et deuxièmement, je pense que ces sons, plus riches, sont aussi plus chaleureux. Il y avait trois données différentes à sonifier (le rapport $\frac{\sum bid_size}{\sum ask_size}$, le signal avertissant d'une transaction ainsi que le nombre de transactions, et finalement, le domaine de transaction).

Pour le rapport, je souhaitais que l'écoute de la sonification ne soit pas trop fatigante. C'est pourquoi, j'ai spontanément été réticent à sonifier le rapport $\frac{\sum bid_size}{\sum ask_size}$ en continu comme le souhaitait Tuan, avec qui j'étais en contact régulier

Je suis donc parti sur l'idée d'un « cric », enregistré et mixé des mois auparavant, émis par le programme uniquement toutes les 3 secondes (rythme de réception des données), de hauteur croissante et de tempo décroissant pour un rapport >1 , comme si nous tendions un mécanisme ; et inversement pour un rapport <1 , comme un relâchement. La variation de hauteur et de tempo aurait été fonction du rapport.

Exploration de la première « structure » :

Une structure trouvée dans les exemples de programme m'a beaucoup plu, de par sa sonorité et j'ai décidé de l'utiliser. Elle permet de faire varier indépendamment le tempo, la hauteur, et de donner un grain particulier au son (voir explication détaillée plus loin). Il n'a pas été immédiat de trouver un « cric » adéquat. Une fois lu par la structure, un bruit de mécanisme remonté, pris sur un jouet, peut être perçu comme un menaçant armement de revolver.

Après diverses manipulations et résolutions de problèmes, mon résultat me plaisait assez mais je pensais pouvoir faire mieux. J'ai décidé d'essayer avec des sons de bulles. Leur son d'éclosion me semblait assez intéressant pour symboliser un bullish. De plus, la dynamique du son de bulle a un sens assez prononcé. Selon qu'on l'écoute à l'envers ou à l'endroit, nous le percevons très différemment. Il pouvait donc éventuellement servir aussi pour le bearish, en le lisant à l'envers.

Il se trouve qu'enregistrer une bulle n'était pas si simple car la plupart contenaient dans leur son un « plic » assez désagréable de par sa dynamique d'impulsion alors que la globalité d'un son de bulle est plutôt doux. La méthode consistant à mettre un verre plein d'air sous l'eau et de laisser échapper les bulles petit à petit s'est avérée infructueuse. Après plusieurs essais de techniques différentes j'ai pu obtenir un résultat satisfaisant en séparant deux assiettes collées par une couche d'eau qui faisait ventouse. Puis le son a été traité sur

Adobe Audition pour enlever un léger bruit dû au micro. Le son m'a moins plu que celui du « cric » et j'ai donc décidé de continuer mes recherches dans le cric.

Avec la seule structure de base choisie, il était en effet possible d'en créer une infinité et de trouver des sonorités sans cesse différentes. Une fois le programme bien changé, la curiosité m'a poussé à réessayer avec la bulle. Il s'est trouvé que lue très lentement avec une certaine hauteur (ou pitch) et un grain particulier, on percevait un son assez « psychédélique » (pour reprendre l'expression de Laurent Daudet), mais que j'appréciais beaucoup. Le grain varie aléatoirement sur un domaine restreint pour, tout en gardant le sens du son, éviter à l'utilisateur de se lasser. Je décidai de le garder et même de le mettre en continu car sa dynamique très lente excluait une utilisation ponctuelle. Le son étant un peu fatigant, j'ai mis une option pour avoir un son plus doux et un seuil de rapport, défini par l'utilisateur, en dessous duquel l'intensité est diminuée d'une valeur choisie par l'utilisateur.

Pour avoir une idée plus exacte du résultat et pour calibrer les fonctions de mon programme, il me fallait entrer des valeurs proches des données réelles. Tuan m'a envoyé un fichier Excel contenant toutes les données groupées. Je les ai donc séparées et fixées une à une à la main dans mon programme. Cela prenait du temps (10, 15 secondes par groupe) et ne me permettait pas d'avoir une audition continue du résultat. J'ai alors créé un mode aléatoire qui générait des valeurs toutes les 3 s avec une statistique proche des signaux réels. Quelques semaines plus tard, j'ai appris à lire les fichiers texte et ai pu écouter les vrais signaux financiers. Cela m'a permis de calibrer plus finement pour avoir un résultat le plus agréable possible.

Deuxième structure : le choix de sons synthétiques

Pour le signal des transactions, j'ai souhaité un son léger, moins fatigant. Pour qu'il contienne l'information du nombre de transaction, je me suis dirigé vers un rapport de hauteur. Il y aurait donc un premier son d'une hauteur définie constante, de référence, puis un deuxième quelques millisecondes après, d'une hauteur et d'une durée proportionnelle au nombre de transactions instantanées. Pour une seule transaction, nous aurons donc deux sons de même hauteur, pour deux transactions, un premier assez bas et un sensiblement plus haut, etc. Là aussi, une structure trouvée dans les exemples a été utilisée comme base. Elle peut sommer des sinusoïdes dont on détermine la fréquence relative, l'intensité et la durée relative. On peut ensuite faire varier indépendamment pendant l'utilisation programme, le coefficient multiplicateur de la fréquence et de la durée. Les fréquences relatives ont été volontairement choisies légèrement enharmoniques pour donner plus de naturel au son.

J'ai décidé d'inclure dans le signal du nombre de transaction l'information concernant qui, des bid ou des ask achète dans le domaine de l'autre. Quelques semaines auparavant, Laurent Daudet m'avait soumis l'idée d'utiliser la dimension spatiale du son. J'ai choisi d'utiliser son idée à cet effet. Le son de cloche était entendu à gauche, à droite ou au milieu selon le domaine de transaction.

Mon programme était terminé mais, frustré de n'avoir pas pu utiliser mon son de cric, je l'ai inclus dans le réglage du volume des deux signaux. Le maniement de la barre de volume produit donc un bruit décalé mais assez semblable à celui d'un mécanisme que l'on remonte.

Présentation du résultat :

Une fois présenté à Tuan, le programme a dû subir quelques changements :

- Il semblait préférer le son de bulle plus doux, j'ai donc mis le plus riche en option.
- Il préférerait ne pas avoir le premier son de cloche de référence. Je l'ai mis en option.

Après quelques difficultés pour recevoir le flux de la bourse, lire les données et les séparer le programme marchait et il est maintenant compris dans le logiciel développé par Tuan.

Quelques problèmes, assez faciles à résoudre demeurent cependant :

- Il n'est pour l'instant pas possible de choisir de suivre les données d'une bourse ou d'une autre. Seul le CAC40 est « audible ».
- Nous ne recevons que les 5 premières quantités d'acheteurs et vendeurs. Il est arrivé qu'une grosse quantité de vendeurs du 5ème rang oscille entre le 5ème et le 6ème, apparaissant et disparaissant régulièrement. Comme cette quantité faisait tout basculer, même si elle n'avait pas beaucoup d'importance, il s'est avéré qu'il faudrait pondérer les quantités selon leur rang. Nous n'avons pas encore eu le temps de faire cela.

3- Le programme

Le programme est composé de 12 parties séparées pour plus de lisibilité. Dans cette partie, je vais présenter les patches les plus importants et les plus intéressants, en laissant de côté ceux qui ne concernent que des calculs (par exemple le calcul de la somme des quantités d'acheteurs). Nous commencerons par le patch du rapport $\frac{\sum bid_size}{\sum ask_size}$, qui a été le premier à être

inclus dans le programme, mais aussi le plus modifié. Puis nous verrons la sonification du nombre de transaction, suivi par la sonification du domaine de transaction qui, comme nous le verrons, sont très liés. Nous passerons rapidement sur le bruitage des curseurs de volume et finirons par une explication de l'interface qui, malgré sa simplicité, a son importance puisque c'est normalement la seule partie que verra et maniera l'utilisateur.

Sonification du rapport $\frac{\sum bid_size}{\sum ask_size}$ (annexes 2 et 3)

Avant toutes choses, il faut savoir que lire un fichier audio wave en pure data se fait via un signal sawtooth (dents de scie) d'amplitude le nombre d'échantillons du fichier (44100*(la durée en secondes) pour un wave). Le fichier est donc lu point par point, linéairement, lorsque la valeur du sawtooth augmente. Tout le fichier est ainsi parcouru et lu en boucle.

Le patch que l'on peut voir en annexe 1 contient la structure principale concernant la lecture du fichier wave. La structure, visible en **annexe 2**, est composée de 5 parties interagissant les unes sur les autres :

- Partie 1 : gère la vitesse de lecture globale
- Partie 2 : gère la hauteur
- Partie 3 : gère le grain
- Partie 4 : gère le volume
- Partie 5 : reçoit vitesse de lecture, hauteur, volume et grain et lit le fichier

Cette structure va en fait lire le sample à une vitesse voulue, en lisant des « grains » du sample à une autre vitesse, qui va agir sur la perception hauteur.

Partie 1 :

Cette partie reçoit (voir lexique) la variable *precession*. Cette variable est proportionnelle au rapport bid/ask. L'objet *phasor~* produit le signal sawtooth d'amplitude 1 et de fréquence l'entrée (ici, $precession/(2460*0.001)$), en Hz. 2460 correspond à la durée du sample en millisecondes. Elle est donc convertie en seconde ($*0.001$) et inversée pour correspondre à une fréquence. Cette fréquence de lecture est donc deux fois plus élevée si la variable *precession* est égale à 2. L'amplitude du sawtooth est multipliée par la durée du signal et sera, dans la partie 5, multipliée à nouveau par 44100 pour parcourir tout le sample.

Partie 2 :

L'entrée *transposition* est elle aussi proportionnelle au rapport. Elle est ensuite injectée dans le calcul $(2^{(transposition/120)} - precession) / chunksize$. Nous savons que tout ça va jouer sur la hauteur. La *transposition* changera la hauteur d'un 120ème d'octave par unité. Les termes *precession* et *chunksize* servent à équilibrer les perturbations que vont créer ces deux entrées. La valeur de $(2^{(transposition/120)} - precession) / chunksize$ est ensuite envoyée comme fréquence à un *phasor* (signal sawtooth). Le signal sawtooth est ensuite dupliqué en deux signaux déphasés : Le premier, envoyé à phase est le même que le signal d'origine. Le deuxième est incrémenté de 0.5 (on rappelle que le sawtooth est d'amplitude 1) puis traduit en modulo 1 (commande *wrap~*). Le signal est donc déphasé d'une demi période.

Partie 3 :

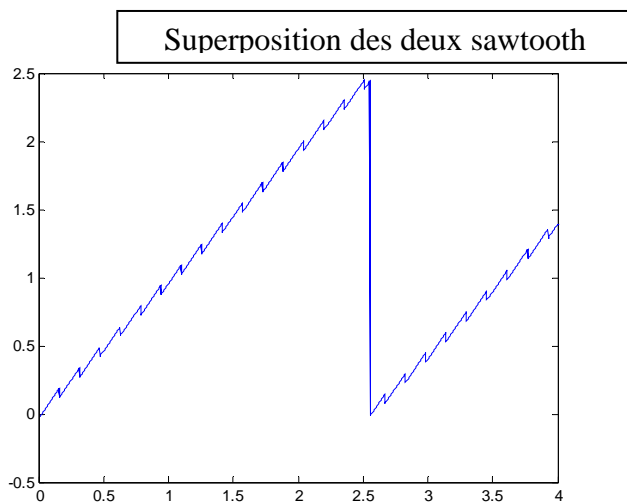
Ici, on définit la résolution des « grains » Le patch « sonorite », accessible via l'objet « *pd sonorite* » (voir **annexe 3**) contient en fait un petit programme qui ajoute ou enlève aléatoirement jusqu'à une unité à la variable « sonorite », qui sera comprise entre 17 et 24 (choisis à l'oreille) qui seront ensuite divisées par 1000 (car un grain de 1 correspondrait à tout le sample). La valeur d'un grain variera donc de 1.7 à 2.4% du sample. Le grain est envoyé à la partie 5 via *chunk-size*

Partie 4 :

Ici, le volume général est modulé par une valeur variant selon les propriétés choisies par l'utilisateur et servant à diminuer la fatigue auditive. Tout cela est expliqué plus loin. L'opérateur « *line* » associé au « *pack f 2000* » fait que la variation de volume est progressive, linéaire, et s'établit en 2000 ms. Ceci servant à éviter les trop brusques variations de volume.

Partie 5 :

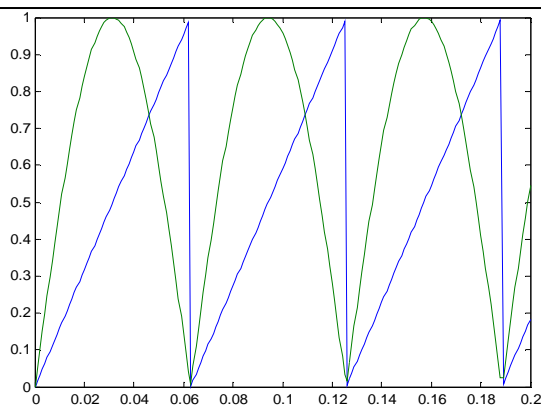
La structure reçoit chunk-size. La valeur de chunk-size est envoyée et multipliée dès que « phase » décroît (opérateur « samphold »). Chunk-size est multipliée par « phase ». Nous avons donc en sortie du « *~ » un sawtooth d'amplitude chunk-size. Nous voyons donc l'utilité du samphold : Il permet d'avoir en sortie, un sawtooth régulier sur sa période même si chunk-size varie entre temps. A ce signal est ajouté le sawtooth de la partie 1. Nous avons donc une somme de deux sawtooth : le premier d'amplitude la durée du grain et de fréquence $(2^{(transposition/120)} - precession) / chunksize$, le second d'amplitude la durée totale du fichier et de période la durée modulée par precession. Nous aurons donc un signal à l'allure suivante :



La structure va lire (objet « tabread ») un grain de sample à une vitesse définie par la variable transposition, puis revenir en arrière et lire un autre grain, décalé du premier par la lente variation du grand sawtooth. Le résultat va être une vitesse de lecture globale indépendante de la hauteur perçue mais avec de nombreuses discontinuités inhérentes au sawtooth. Pour palier à cette discontinuité, à chaque fois que le signal sawtooth « rapide » revient à 0, un cosinus annule l'amplitude du signal de sortie de façon plus douce.

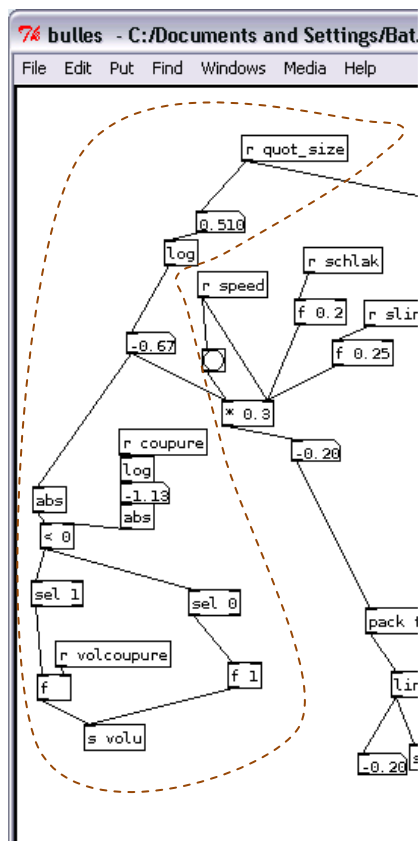
Ce cosinus a pour variable le sawtooth, auquel on soustrait 0.5 et divisé par 2. C'est-à-dire qu'elle va de -0.25 à 0.25. Cette variable est multipliée par l'opérateur par 2π . Voir ci-dessous l'allure du cosinus et du sawtooth.

Corrélation entre le cosinus et le sawtooth



Pour palier au problème de la rapide variation de volume, le même signal, avec le sawtooth rapide décalé d'une demi-période (vu en partie 2) est additionné au premier. Le tout est filtré et envoyé en sortie audio. Le volume semble alors constant et il n'y a aucune impulsion, ou craquement dans ce qu'on entend.

Partie gérant le volume de seuil :



Une brève explication de la partie gauche du patch « bulles » (les deux parties à droites sont en fait le traitement du rapport, qui est ensuite traduit en transposition et precession) :

Il s'agit de la partie servant à diminuer la fatigue en imposant un seuil de rapport en dessous duquel, le volume est diminué. Ici, j'ai opté pour une traduction en logarithme de rapport pour ne pas différencier deux rapports inverses (0.5 et 2 par exemple), qui ont la même importance. Donc le rapport est envoyé via quot_size, il passe par un logarithme puis l'on prend sa valeur absolue avec l'opérateur « abs ». L'opérateur « < 0 » va comparer l'inlet gauche et l'inlet droit. Si le gauche est inférieur au droit, il enverra 1 en sortie. L'objet « sel x » envoie une impulsion si son entrée est x. L'objet « f x » envoie l'inlet gauche ou, par défaut, x, lorsqu'il reçoit une impulsion.

Au final, la variable « volu » recevra :

- 1 si le rapport est supérieur au rapport de coupure
- La variable volcoupure (défini dans l'interface et compris entre 0 et 1) si il est inférieur

Nous avons vu plus haut que volu était multiplié par le volume global et défini par volume de sortie.

Sonification du nombre de transactions (Annexe 4)

Pour cette sonification, c'est un son synthétique de cloche qui va être émis. Il s'agit de sinusoïdes dont auxquelles on donne la dynamique d'une cloche. J'ai, pour cette partie, utilisé comme base un autre programme contenu dans les exemples. Il permettait de définir la durée et de modifier la fréquence des sinus.

La structure se décompose en 6 parties :

- Partie 1 : Définitions des amplitudes, durées relatives, fréquence relative et décalage de fréquence
- Partie 2 : fonction concernant la hauteur du son de cloche
- Partie 3 : fonction déterminant la durée du son de cloche
- Partie 4 : volume de la cloche

- Parties 5 et 5bis : actionnement de la cloche
- Partie 6 (dans le patch medium) : Sortie du son (commune à la sonification du domaine de transaction).

Partie 1 :

Les définitions des sinus font appel à une fonction, appelée « partial » intégrée dans le programme. Les objets « partial x1 x2 x3 x4 » vont envoyer les variables x1, x2, x et x4 à une fonction assez simple. En bref : x1 va multiplier un sinus, de fréquence $x3 * frequency + x4$. frequency est une variable envoyée par la partie 2 et sera modulée par le nombre de vente. L'amplitude du sinus décroît et atteint 0 en un temps $x2 * duration$. duration étant définie dans la partie 3 et est aussi fonction du nombre de ventes. x1 est donc l'amplitude, x2, la durée relative, x3 la fréquence relative et x4 le décalage de fréquence.

Partie 2 :

Soit nbtrans le nombre de transaction. La variable ring est égale à $60 + nbtrans * 10$. « zero » et « on » reçoivent des impulsions. Quand « on » reçoit une impulsion (déclenchement du son de cloche), le réel 60 est converti d'unité midi en hertz via « mtof » puis envoyé à frequency (dans la fonction partial). C'est la hauteur de référence. Plus tard, la variable « zero » reçoit à son tour une impulsion et alors c'est la valeur de ring, stockée avec l'opérateur « f » qui est convertie puis envoyée à frequency. Elle s'appliquera au second son de cloche. Notons pour la suite que la valeur de la fréquence est envoyée à la partie 4 (calculant le volume) via la variable « voleq ».

Partie 3 :

Cette partie fonctionne comme la partie 2 mais gère la durée du son de cloche. Comme précédemment, quand « on » reçoit une impulsion, le coefficient multiplicateur de la durée est de 10 (multiplicateur car envoyé à partial et multiplié par la variable x2 des sinus). Plus tard, c'est « zero » qui reçoit l'impulsion et la durée relative devient $nbtrans * 5$.

Partie 4 :

Cette structure, gérant le volume, a une fonction assez complexe. Premièrement, elle atténue les variations trop rapides de volumes responsables de certains craquements. Deuxièmement, elle coupe le son s'il n'y a pas de transaction. Et finalement, elle compense la baisse de sensibilité de l'oreille aux basses fréquences en pondérant l'intensité afin d'en obtenir une perception indépendante de la hauteur de la cloche.

Quand « on » est envoyé, la variable « line » croît jusqu'à atteindre la valeur 1 en 30 millisecondes. Ceci est déterminé par le message « 1 30 ». 2500 millisecondes plus tard (objet « delay 2500 »), line décroît et atteint 0 en 100 millisecondes. Cette variation résout le problème des trop brusques hausses d'intensité. La valeur de sortie de line est ensuite multipliée par le volume général de la cloche « volcloche », déterminé par l'utilisateur dans l'interface. Ce volume est modulé par une fonction empirique de « voleq » (la fréquence).

Après 30 millisecondes, en sortie de l'opérateur « / », on a donc :

$Volcloche / (0.25 * \log(voleq / 88) + 1)$.

Cette valeur est ensuite multipliée : - par 1 si nbtrans \neq 0
- par 0 si nbtrans = 0

Il n'y aura pas de son si il n'y a pas de transaction.

La partie en bas à gauche ne sert qu'en cas de test du son de cloche et ne sert finalement pas à grand chose.

Parties 5 et 5bis:

Il est possible dans l'interface de décider d'enlever le son de cloche de référence. Pour cela, il suffit de cocher une case, qui enverra à la variable « refer » la valeur 1 si elle est cochée et 0 sinon. Dans 5bis, l'objet « sel 1 0 » va envoyer un bang (impulsion) à la variable « avec » si refer = 1 et un bang à « sans » si refer = 0.

Partie 5 : Nous allons étudier les deux cas selon que ce soit « avec » ou « sans » qui envoie un bang.

Si c'est « sans », « on » en haut à gauche va envoyer à zero un bang avec un délais de 5 ms car le « sans » au dessus de delay envoie le réel 5 via l'objet « f 5 ». Il envoie, 15 ms après que zero ait reçu l'impulsion une autre impulsion à trigger qui va déclencher le son de cloche.

L'objet « moses 1 » va empêcher l'autre « on » juste au dessus d'envoyer le bang à trigger.

L'objet « moses x » envoie son entrée :

- A la sortie gauche si l'entrée est $< x$
- A la sortie droite si l'entrée est $\geq x$

Donc si refer = 0 (cas sans), rien n'est envoyé à droite et le bang est bloqué.

Nous n'aurons donc qu'un son de cloche dont la hauteur et la durée sont déterminées par le nombre de transactions.

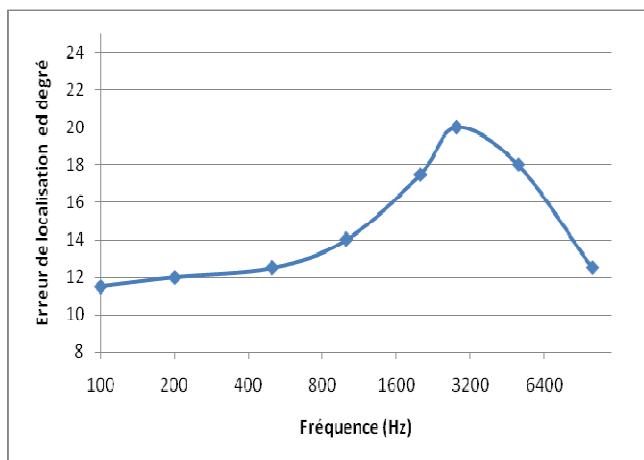
Si c'est « avec » qui reçoit l'impulsion, le « on » au dessus du « moses 1 » n'est pas bloqué et il déclenche le premier son de cloche de référence avec un délai de 20ms. Ce délai sert à ce que l'initialisation de la hauteur et de la durée n'arrive pas en même temps que le déclenchement de la cloche, auquel cas le comportement du programme risque d'être imprévisible. Le « on » en haut à gauche envoie au zero l'impulsion avec 250ms de délai (le temps que le premier son soit fini), puis, 150ms plus tard (ou 400ms après le bang du « on »), il envoie le second son de cloche différent du premier grâce au « zero ».

Partie 6 :

Voir sonification du domaine.

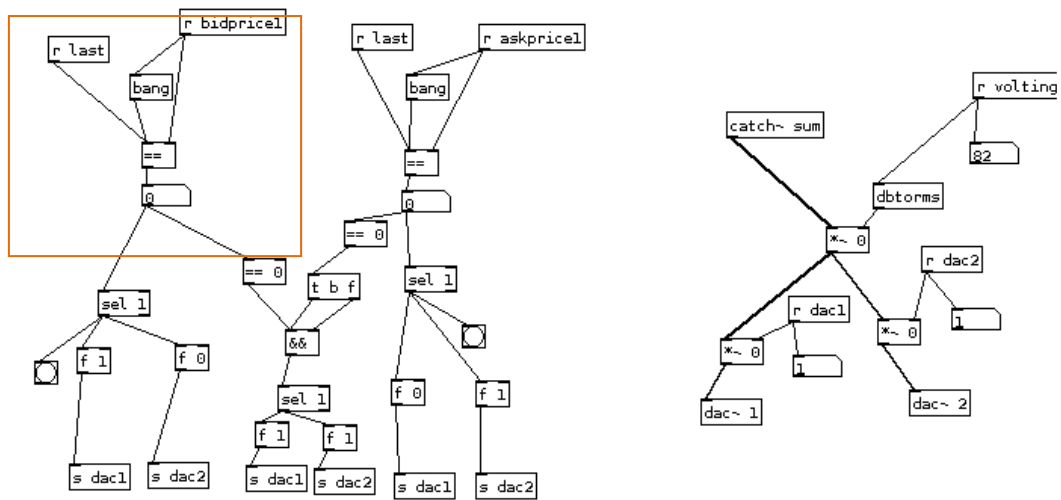
Sonification du domaine de transaction

L'information concernant le domaine de transaction (prix du bid, du ask ou entre les deux) était l'occasion idéale d'utiliser la spatialisation du son.



Ci-coté sont représentés les résultats des expériences de localisation d'une source sonore en fonction de la fréquence, faites par Stevens et Newman en 1936 [05]. Contrairement au programme, la spatialisation des sons de ces expériences ne se limite pas à une différence d'intensité. Pour les basses fréquences, l'auditeur a pu s'aider de la différence de phase perçue entre les deux oreilles. Nous pouvons supposer que en s'aidant uniquement de l'intensité, l'erreur se situera plutôt aux alentours de 20°, ce qui offre une précision de l'information assez restreinte.

Avec une information à trois degrés et des hauts parleurs probablement situés avec un angle de 90°, nous obtenons un domaine de 30° par degré d'information, ce qui est bien au-dessus des 20° nécessaires. L'utilisateur ne fera pas d'erreur. Comme la spatialisation peut être intégrée au son de cloche nous évitons de surcharger l'espace sonore et de fatiguer l'auditeur. Pour que la compréhension du signal soit instinctive, si le domaine d'achat est le ask, le son de cloche sera à gauche, à droite pour le bid et au milieu si c'est au milieu, comme dans le tableau vu plus haut.



Partie de gauche :

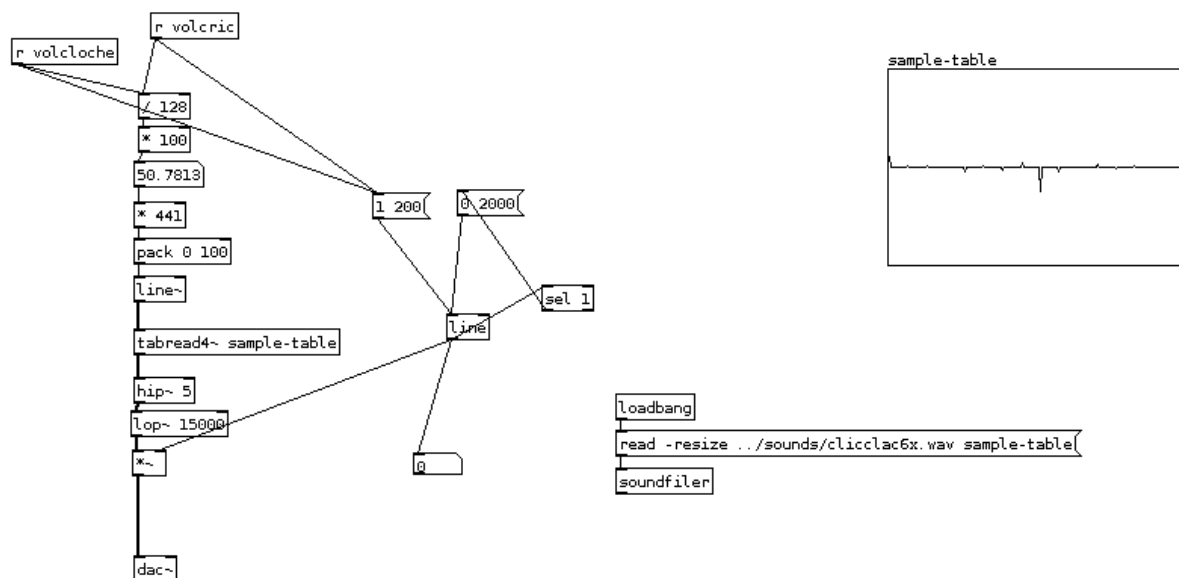
Les variables last, bidprice1 et askprice1 sont respectivement le prix transacté, le prix des bid le plus haut et le prix des ask le plus bas. Les offres des bid sont comparées avec le prix transacté avec l'opérateur « == ». Le number en dessous donne 1 si ils sont égaux et 0 sinon (voir encadré). La même opération est faite avec les offres des ask et le prix transacté à droite.

L'objet « sel 1 » va envoyer un bang si l'égalité est vraie. Alors les variables dac1 et dac2 vont prendre respectivement les valeurs 1 et 0 si c'est le bid qui est égal au ask, ou 0 et 1 si c'est le ask qui est égal au bid. Si les deux égalités sont fausses, alors la sortie du && est 1 et dac1 et dac2 sont égaux à 1.

Partie de droite :

Dans la partie de droite nous retrouvons le volume de la cloche calculé précédemment (partie 4) et le signal audio des cloches calculé avec la fonction « partial ». Chaque sinusoïde est envoyés sous le nom « sum » et est sommée avec les autres avec l'objet « catch~ ». Le signal est multiplié par le volume converti puis envoyé dans le haut parleur gauche (dac~1) et droit (dac~2) après avoir respectivement été multipliés par dac1 et dac2. Le son viendra donc de la gauche et/ou de la droite selon le domaine de transaction.

Sonification du bouton de volume :



Pour rendre le programme plus agréable à utiliser et pour m’amuser, j’ai aussi sonifié les boutons de volume, qui agissent maintenant comme des têtes de lecture d’un petit sample constitué d’une dizaine d’impulsions de mécanisme tirés d’un jouet pour le bain. Les variables volcric (appelé comme ça car devait être le son d’un cric au départ) et volcloche sont, je le rappelle, comprises entre 0 et 128 et varient proportionnellement avec la position de la barre de volume (voir interface dans annexe). Leur valeur est donc divisée par 128 puis multipliée par le nombre de points à lire (ici 44100 car le morceau dure approximativement une seconde). Si la barre est au milieu, l’entrée de « pack 0 100 » sera donc le numéro du point du milieu du morceau. Ce nombre va en fait être la valeur à atteindre d’un line~ en 100ms. Cette line~ va lire un tableau dans lequel il y a mon cric samplé.

Donc le programme va lire en 100ms (choisi à l’oreille) l’intervalle séparant les deux valeurs consécutives définies par la barre. Comme nous la faisons bouger en continu, le son paraît être lu à la vitesse de translation de la barre de volume.

Comme la même structure gère les deux barres de volume, si je pousse l’une au maximum et que je décide de pousser l’autre, qui est à 0, l’entrée va passer de 128 à 0 instantanément. Le programme va lire tout le sample en 100ms, ce qui n’est pas agréable à entendre. Pour cela, à chaque variation de volume, l’intensité va passer de 0 à 1 en 200ms puis une fois qu’elle aura atteint 1, va décroître jusqu’à 0 en 2000ms. Comme la décroissance est plus longue que la croissance, si je fais varier mon volume en continu, l’intensité va rester à 1. Mais si je passe de 128 à 0 instantanément, les 100ms de lecture non voulues seront non audibles.

L'interface (Annexe 5)

L'interface que l'on peut voir en annexe est celle vue par l'utilisateur mais certains éléments n'ont pas de grande utilité.

Les simulations, par exemple, qui peuvent être lancées via la partie 1 servent uniquement à se donner une idée du résultat du programme. Deux simulations ont été créées. La première, dite aléatoire (**voir annexe 6**) est lancée avec le bouton vert. Elle est en fait créée par un programme tenant compte de résultats que m'a donné Tuan et les simulant aléatoirement. Bien sûr, la bourse n'est pas aléatoire et le son obtenu n'est pas tout à fait probable mais il m'a été utile pour calibrer les sons. On peut le voir en annexe. Plus tard, j'ai pu lire les fichiers .txt et j'ai donc créé un deuxième programme de simulation qui lisait toutes les 3 secondes les nombres de la ligne suivante. Il peut être lancé avec l'interrupteur bleu et réinitialisé avec l'interrupteur à gauche. L'interrupteur rouge interrompt les simulations.

Les messages « pd dsp 1 » et « pd dsp 0 » respectivement ouvrent et ferment les sorties audio.

Toutes les fenêtres en haut à gauche sont les différents patches qui contiennent les structures.

La partie 2 concerne uniquement le son de cloche et les données correspondantes. Sur la colonne de gauche sont visibles les prix de transactions ainsi que les offres, puis le nombre de transactions instantanées. Sur la colonne de droite, nous pouvons, de haut en bas, régler le volume global de la cloche, la tester, déterminer un nombre de transaction de seuil (inachevé) et mettre ou non le son de référence.

La partie 3 concerne la sonification du rapport. Sur la colonne gauche, nous pouvons surveiller le rapport, la somme des acheteurs, des vendeurs et opter ou non, en cochant la case pour un son différent (plus lourd à écouter car plus riche mais plus amusant). Sur la colonne de droite, nous faisons varier le volume globale, le rapport de coupure, l'atténuation du volume en dessous de ce rapport de coupure, la vitesse relative, c'est-à-dire la sensibilité, puis choisir entre un son de bulle ou un autre, moins intéressant à mon goût.

4- Conclusion

J'ai créé un programme de sonification qui fonctionne. Il permet de déterminer, à un dixième près le rapport $\frac{\sum bid_size}{\sum ask_size}$, de savoir combien il y a eu de transaction à une prés, et de savoir exactement dans quel domaine la transaction a été faite. Cependant, pour l'instant, il

n'a pas de grande utilité. Il n'est pas assez complexe, ne tient pas compte d'assez de variables et surtout, je n'ai pas encore sonifié les achats faits par le logiciel de Tuan. On ne peut donc pas encore observer de corrélation entre ce qu'on entend et ce qui se produit. En ce qui me concerne, je pense qu'à la longue, écouter mon programme doit être assez fatigant, même avec les options que j'ai rajoutées..

Tuan semble néanmoins être satisfait du résultat et nous discutons d'un stage éventuel ayant pour but de compléter le programme. Il est très intéressé par la complexité de notre perception et souhaite étudier la corrélation des résultats des calculs compliqués de son algorithme avec ce que nous percevons du signal sonore. Avec le temps que m'offrirait ce stage, nous pourrions sans doute obtenir des résultats.

Ce projet m'a énormément intéressé. La découverte de Pure Data a été une révélation. Je continue à créer des petits programmes et suis constamment surpris des résultats obtenus. Je pense continuer mon apprentissage de ce programme et je suis certain qu'il me sera utile. Mise à part la sonification, l'interactivité qu'il propose peut probablement être exploitable dans les jeux vidéo. J'ai d'ailleurs contacté Quantic Dream, un studio réalisant des jeux vidéos, situé à Paris, reconnu pour ses innovations et sa recherche dans le langage. Le langage et la psychoacoustique sont des domaines dans lesquels je vais me plonger et qui m'aideront à créer des programmes peut être plus intuitifs et plus efficaces.

Quoiqu'il en soit ce projet aura été une expérience enrichissante. Il m'a donné une idée de ce qu'était d'avoir à rendre un résultat pour une entreprise. J'ai eu l'impression que ce que je faisais était utile et je pense que c'est à mettre dans mon cv.

Je remercie Laurent Daudet de m'avoir trouvé un sujet aussi intéressant et pour son aide, je pense notamment aux contacts qu'il m'a donnés. Je remercie Tuan pour son ouverture et son enthousiasme. Il m'a laissé la liberté dont j'avais besoin dans la réalisation du programme et m'a beaucoup aidé dans tout ce qui concernait les problèmes de type informatiques (je suis novice en ce domaine).

References :

- [01] Jonata and Edward CHILDS, "Marketbuzz: Sonification of real-time financial Data" in *Proceedings of ICAD 04-Tenth Meeting of the international Conference on Auditory Display, Sydney, Australia, July 6-9, 2004*
- [02] Agnieszka Roginska, Edward Childs, Micah K. Johnson, "Monitoring Real-Time Data: A Sonification Approach", in *Proceedings of the 12th ICAD, London, UK, July 20-23, 2006*
- [03] www.lam.jussieu.fr
- [04] Jérôme Abel, « Pure Data Initiation », septembre 2006, http://impala.utopia.free.fr/pd/patches/PureData_Initiation_fr.pdf
- [05] *George Canévet* «Audition binaurale et location auditive, Aspects physiques et psychoacoustiques » dans *Psychologie et perception auditive*

Bibliographie :

M.C Botte, G Canévet, L Demany, C Sorin, *Psychoacoustique et perception auditive*

M. Chion, *Guide des objets sonores, Pierre Schaeffer et la recherche musicale*

C. Hugonnet, P. Walder, *La prise de son stéréophonique.*