le cnam
The graduate school of games and interactive media
enjmin

# RAPPORT DE STAGE

**MASTER JEU ET MÉDIAS INTERACTIFS NUMÉRIQUES**
**CONCEPTION SONORE - 2EME ANNEE**

Florent Dumas
@ AudioGaming

# REMERCIEMENTS

Je tiens à remercier particulièrement **Amaury Laburthe** de m'avoir offert l'opportunité de travailler sur des projets innovants voire passionnants dont les tâches dépassaient les compétences d'un simple sound designer du jeu vidéo et s'accommodaient parfaitement avec mon profil un peu plus technique qui tend à se dessiner au fil de mes expériences. Pour tout ce qu'il m'a appris et son professionnalisme.

Merci à **l'équipe d'AudioGaming** : Amaury Laburthe, Damien Henry, Chunghsin Yeh et Robin Picou, mais aussi aux stagiaires qui ont travaillés avec moi cet été : Victorien Aubret, Adrien Tisseraud, Kévin Loustau, Paul-Arthur Sauvageot et Julien Tallon pour leur accueil, leur bonne humeur et leur professionnalisme.

Merci au corps enseignant de l'ENJMIN pour leur soutien et leur suivi durant nos projets de Master 2. Et enfin, merci à tous les étudiants de l'ENJMIN.

# SOMMAIRE

# INTRODUCTION

La recherche d'un stage est souvent source de frustration et de stress. Celle de mon stage de Master 2 n'a pas fait exception.

D'abord gagnant d'un stage de six mois chez **Dont-Nod Entertainment** grâce à un concours étudiant, je me sentais libre et soulagé d'être à l'abri, bien avant les autres, presqu'égoïstement. Seulement voilà, les plus gros studios ne sont pas intouchables et ne contrôlent pas leurs ventes. Suite à l'échec commercial de leur premier jeu, le studio a été placé en redressement judiciaire en Janvier 2014.

Début Mars, après un entretien Skype et la lecture de quelques écrits sous NDA, je me préparais psychologiquement à mon départ pour Shanghai où je devais effectuer un stage extrêmement formateur avec une douzaine de mes camarades dans une start-up chinoise nommée **Arts United** sous l'aile d'un grand du jeu vidéo, Tore Blystad, papa de Hitman entre autres. Malheureusement, suite à une mésentente entre Tore et ses associés, il fut contraint de partir, et d'annuler les stages promis.

Je me suis alors tourné vers les personnes les plus proches de l'école susceptibles de sauver mon année. Par chance, **AudioGaming** avait un créneau estival pour un profil tel que le mien !

Ainsi, mon objectif était de m'intégrer au sein de l'équipe d'AudioGaming afin d'assurer une partie de la production d'un certain nombre de projets en cours ou à venir jusqu'à leur livraison finale. J'ai travaillé sur trois projets détaillés dans ce rapport dont deux plug-ins et une expérience interactive sur tablette.

Cette activité correspondait à mes attentes puisque je souhaitais mettre à profit mon goût pour la conception sonore dans son coté plus technique et m'éloigner de l'idée d'"usine à assets" à laquelle j'ai été assigné l'été dernier pour mon premier stage, mais aussi parce que le projet sur tablette se situe dans la continuité directe de mes travaux sur mon projet de Master 2 "Ascent".

# AUDIOGAMING

**L'ENTREPRISE**

AudioGaming
Sound design, Music & Technology

AudioGaming est une jeune société Toulousaine lauréate du concours de création d'entreprises innovantes et soutenue par l'incubateur Midi-Pyrénées. L'entreprise relève les défis d'une nouvelle génération d'outils audio pour sound designers et développeurs de jeux vidéo. L'entreprise a été fondée à partir du constat que le processus de production du son était en fort retard par rapport à l'image. Les images étant synthétisées en temps réel depuis leur début, le son est construit à partir de fichiers préenregistrés, alourdissant le processus et limitant considérablement les possibilités créatives.

AudioGaming crée des outils audio innovants, articulés autour d'un moteur audio qui place le contrôle du son au cœur de la conception, apportant une aide à des tâches fastidieuses et répétitives en offrant des outils intelligents qui sont capables de comprendre et de traiter le son en fonction de ses propriétés intrinsèques.

AudioGaming a été fondée en mars 2009 par **Amaury La Burthe** et **Damien Henry**.
Amaury La Burthe CEO (Chief Executive Officer) a travaillé comme chercheur assistant à l'Ircam et Sony Computer Science Laboratory et comme responsable design audio pour Ubisoft.
Damien Henry CTO (Chief Technology Officer) a travaillé en Recherche et Développement dans le domaine de l'acoustique pour Brüel Kjær et aussi en tant que directeur technique chez Voxler.

**2009** a été une année de gestation où le but de la société a été défini : « développer des outils innovants interactifs pour la création audio». AudioGaming a commencé la recherche de financement et le projet a été élu en tant que vainqueur du concours annuel de création d'entreprise innovante du ministère de la Recherche dans la catégorie de projets émergents. Grâce à ce prix la société à pu investir pour concrétiser ses idées dans la réalisation de prototypes.

**2010** a été une année de consolidation. Cette année là, la société a été élue de nouveau gagnante du concours annuel du ministère de la Recherche pour l'innovation, cette fois-ci dans la catégorie «Création & Développement». La subvention du concours d'innovation a été amplifiée avec l'arrivée de deux investisseurs privés qui ont permis à AudioGaming de se développer et de commencer à embaucher.

**2011** a été une année de développement, les efforts des années précédentes ont été fructueux et la société a attiré l'attention de partenaires tels que Firelight Technologies et Audiokinetic.

# AUDIOGAMING

**2012** marque le lancement du premier produit, **AudioWeather**, synthétiseur temps réel des sons de vent et pluie. Les premiers plug-ins dirigés vers la post-production, **AudioWind** et **AudioRain** ont également été lancés cette année avec succès y compris postérieurement avec l'utilisation de AudioWind dans "Django Unchained" de Quentin Tarantino. AudioGaming a également participé à deux jeux mobiles : "Chicken Doom" et "Journey To Hell" où la société à fait tout l'audio (contenu et programmation).

**2013** La société s'est plus engagée dans la production de jeux vidéo. AudioGaming a attiré l'attention avec l'ambiance sonore du jeu **Type_Rider**, fait en association avec la chaîne de télévision Arte France, les studios BulkyPix et Ex Nihilo et élu meilleur jeu mobile de 2013 par Le Parisien. Les plug-ins sont aussi en développement continu : **AudioSteps** et une deuxième version d'**AudioMotors** sont lancés.

**2014** Le nom : "Audiogaming" ayant une consonance trop réductrice par rapport à la diversité des activités de l'entreprise, cette dernière lance "**Novelab**", une marque qui regroupe toutes les activités de services, recherche et développement facilitant ainsi la communication avec les clients.

## SERVICES ET PRODUITS

### Plug-ins audio

AudioGaming développe des plug-ins Virtual Studio Technology (VST) et Real TimeAudioSuite (RTAS) innovants et originaux pour les jeux vidéo et les créateurs de médias interactifs. Les solutions se concentrent sur la prestation d'audio interactif et dynamique dans des configurations en temps réel. Dans cette catégorie ont été lancés les produits **AudioSteps** pour les bruits de pas, **AudioMotors** pour les bruits de moteur, **AudioWind** et **AudioRain** pour la conception des bruits de vent et de la pluie.


### Technologies audio

AudioGaming développe également des librairies d'audio procédural pour les jeux vidéo et les créateurs de médias interactifs. Dans cette catégorie, on trouve les librairies AudioWeather, et bientôt **AudioFire** et **AudioElec**, ainsi qu'**AudioGesture**, un intergiciel pour la sonorisation temps réel des gestes. AudioWeather est notamment disponible dans Fmod, pouvant être utilisé au travers de nombreux langages de programmation et prenant en charge de nombreuses plates-formes.

# AUDIOGAMING

**Services**
Le studio partage aussi son expertise audio sous forme de services avec la prise en charge globale de l'audio : création d'assets sonores, composition musicale et intégration dans des environnements **Unity3D** par exemple. Depuis le lancement de **Novelab**, AudioGaming propose des applications mobile : jeux vidéo, serious game, réalité augmentée, modeling 3D, webdesign avancé, des applications de réalité virtuelle avec par exemple l'**Oculus Rift**, reconnaissance vidéo, mapping vidéo... En bref tout ce qui gravite autour des nouveaux médias et supports interactifs modernes.

# AUDIOGAMING

### WORKFLOW

L'organisation de l'entreprise est très horizontale. Directeurs, employés et stagiaires travaillent au même endroit et tous sont investis dans le travail. Cette proximité permet un très bon suivi du travail et de la gestion d'équipe du côté des directeurs de la société. Du côté des stagiaires, cela aide à l'organisation du travail à réaliser.

Comme les membres de l'équipe travaillent souvent sur des projets différents, une réunion hebdomadaire est réalisée tous les lundis matin avec comme support le **WeekScrumboard** récapitulant les objectifs de la semaine et le **YearScrumboard** situant les objectifs globaux dans le temps à l'échelle annuelle. Le but de cette réunion est que toute l'équipe prenne compte du contexte général des projets en cours et par conséquent puisse avancer ensemble efficacement.

A chaque réunion, on discute des tâches accomplies lors de la semaine précédente et on attribue des objectifs à atteindre pour le sprint suivant. Cela permet aussi de garder un historique de l'évolution au cours des semaines, puisqu'un objectif prévu qui n'a pas été atteint est maintenu pour la semaine suivante.

### COMMUNICATION

Les locaux d'AudioGaming sont basés dans une pépinière d'entreprises et contiennent deux salles en openspace. Nous sommes très peu nombreux et pouvons sans soucis communiquer directement avec la personne voulue, faire des réunions ou des points projets sur place. Néanmoins pour éviter le dérangement excessif de nos collègues, nous communiquons beaucoup par Skype pour les questions ou petites discussions et ainsi respecter le travail de chacun.

| Week goals | Project Name | Who | To do | Priority | Livrable | Done? |
|---|---|---|---|---|---|---|
| | PureData | fd/alb | Plan de match finition elec/blender | 0 | | Ok |
| | PureData | fd | Finition Elec/Blender selon retours alb | 0 | | Ok |
| | DoseVST | fd | Intégrations des buzz pour elec | 1 | | No |
| | Novelab/AG | fd/alb | Proto binaural pour tester différentes tech | 0 | | En Cours |
| | PureData | fd/va | Debug fire, elec, blender | 3 | | En Cours |
| | PureData | fd | Sound design pour blender, elec | 2 | | En attente |
| | AG | fd | Rapport de stage | 1 | | Ok, presque fini |
| | | | | | | |
| | GrooveExtractor | jt | Reprise/debug code python | 0 | | Ok YES |
| | GrooveExtractor | jt | reprise test drum_analyzer pour comparaison python/C++ | 0 | | En cours |
| | GrooveExtractor | jt | Tester les modèles C++ avec différents réglages (normalisation, etc...) | 1 | | No |
| | GrooveExtractor | jt | Amélioration algos générale | 2 | | No |
| | | | | | | |
| | Projet Doublage | cc | Polish/debug appli doublage | 0 | | ok |
| | Projet Doublage | cc | Modification dernière minutes | 0 | | ok |
| | | | | | | |

*Aperçu du WeekScrumboard*

# AUDIOELEC

## LE PROJET

AudioElec est un plug-in audio **procédural**. Son but est de générer des sons de types électriques en temps réel. Dans la même veine que ses aînés AudioWind, Audio-Rain et bientôt AudioFire, il est destiné aux concepteurs sonores travaillant sur la postproduction de films, la production de jeux vidéo bien sûr, mais aussi tout type d'installations sonores interactives ou pas.

Six membres de l'équipe ont travaillé sur ce projet : Victorien et moi avons travaillé conjointement sur la partie programmation sous **PureData**, tandis que je me suis occupé seul de la prise de son et du sound design pour alimenter la librairie de samples du lecteur granulaire. D'autre part, Paul-Arthur était en charge de la partie développement en C/C++, et Kévin et Adrien de la partie interface graphique et de son intégration. Amaury quant à lui, supervisait de près le projet tout au long de son développement.

J'ai donc travaillé notamment en étroite collaboration avec Victorien durant mon premier mois de stage où j'ai dû réapprendre et renforcer mes compétences en Pure-Data avant de prendre efficacement la relève après son départ fin Juin.

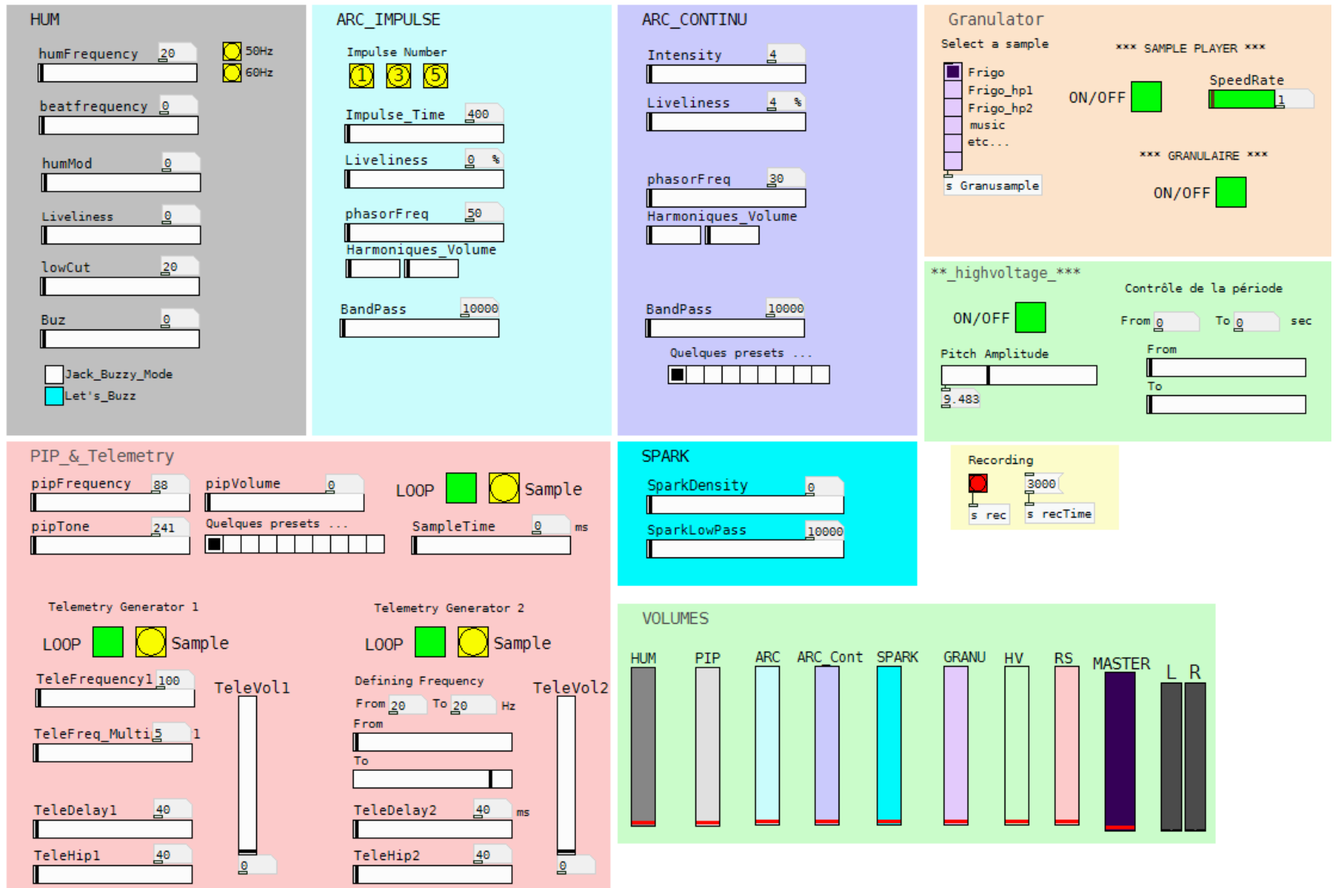
# AUDIOELEC

### STRUCTURE DU PATCH

Le moteur derrière AudioElec est **PureData**. Son interface graphique permet à un non-initié à la programmation de créer des patchs simples facilement. Avec l'expérience, il est possible de réaliser des patchs gigantesques et très complexes.

Ci-contre, l'interface principale du patch d'AudioElec. Cette interface regroupe tous les contrôles que l'on donnera à l'utilisateur dans le plug-in final. Elle représente un prototype de ce que verra l'utilisateur finalement.

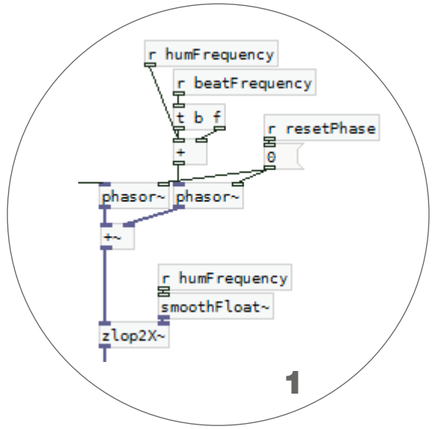
Les encadrés de couleurs représentent chacun des **modules** du plug-in, que je détaillerai dans la partie suivante, ainsi qu'une partie "Mixer".



# AUDIOELEC

### LES MODULES

Sept modules composent le plug-in, chacun ayant un rôle différent. Chaque module est une **abstraction** du patch, c'est-à-dire un patch dans le patch, ou un sous-patch. Je tenterai, dans cette partie , de vous détailler le fonctionnement de chacun d'eux. Certains des patchs entiers se trouvent en annexes.




**Hum**
Ce module est chargé de reproduire un bourdonnement basse fréquence, notamment celui du secteur. Il est composé de six parties.

Le hum est généré simplement par deux phasors dont on contrôle les fréquences. Le signal est ensuite filtré par un lowpass. (1)

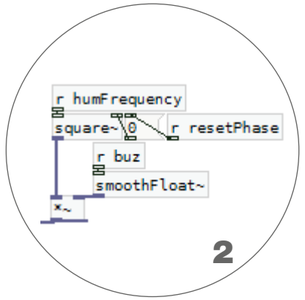
En y ajoutant un signal carré, on génère un buzz. (2)

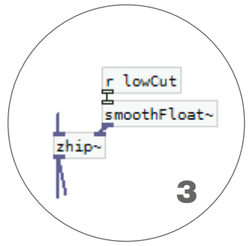
L'utilisateur peut contrôler un highpass s'il le souhaite. (3)

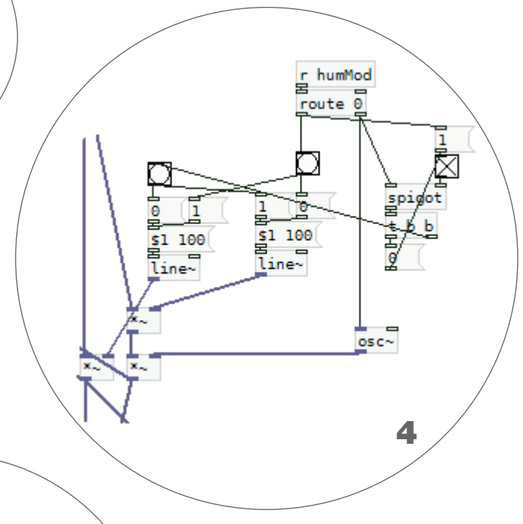
La modulation d'amplitude permet de faire varier le volume du hum via un sinus. On utilise pour cela deux fois le même signal, l'un sans modulation et l'autre avec. Les enveloppes permettent de switcher entre les deux signaux proprement. (4)

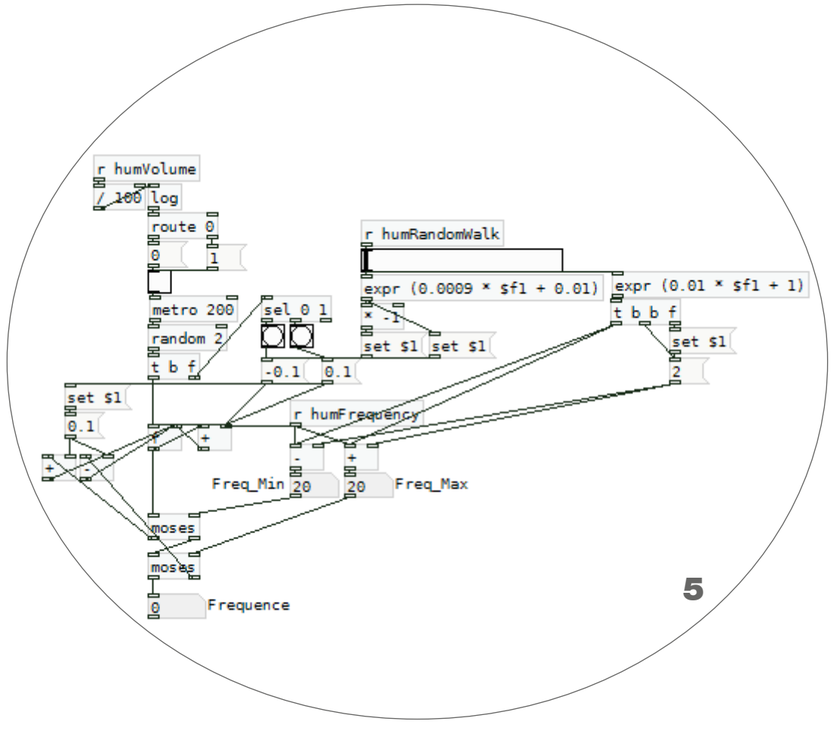
Le liveliness contrôle ce que l'on appelle un "Random Walk". La fréquence du hum n'est pas fixe, elle oscille autour d'une fréquence donnée (entre un fMin et fMax). Le Random Walk incrémente ou décrémente cette fréquence d'un pas. Plus le liveliness est élevé, plus ce pas est élevé et donne donc plus de vie au hum. (5)

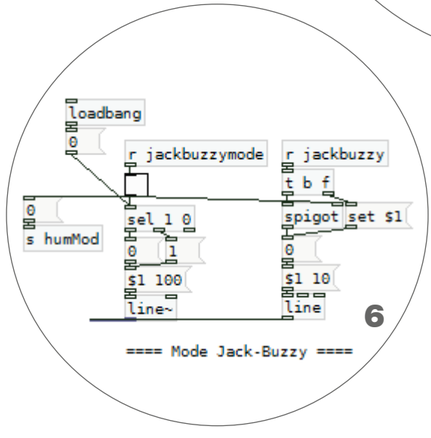
Enfin, le mode savamment appelé "Jack Buzzy Mode" permet de simuler le contact du doigt avec un câble jack branché. (6)



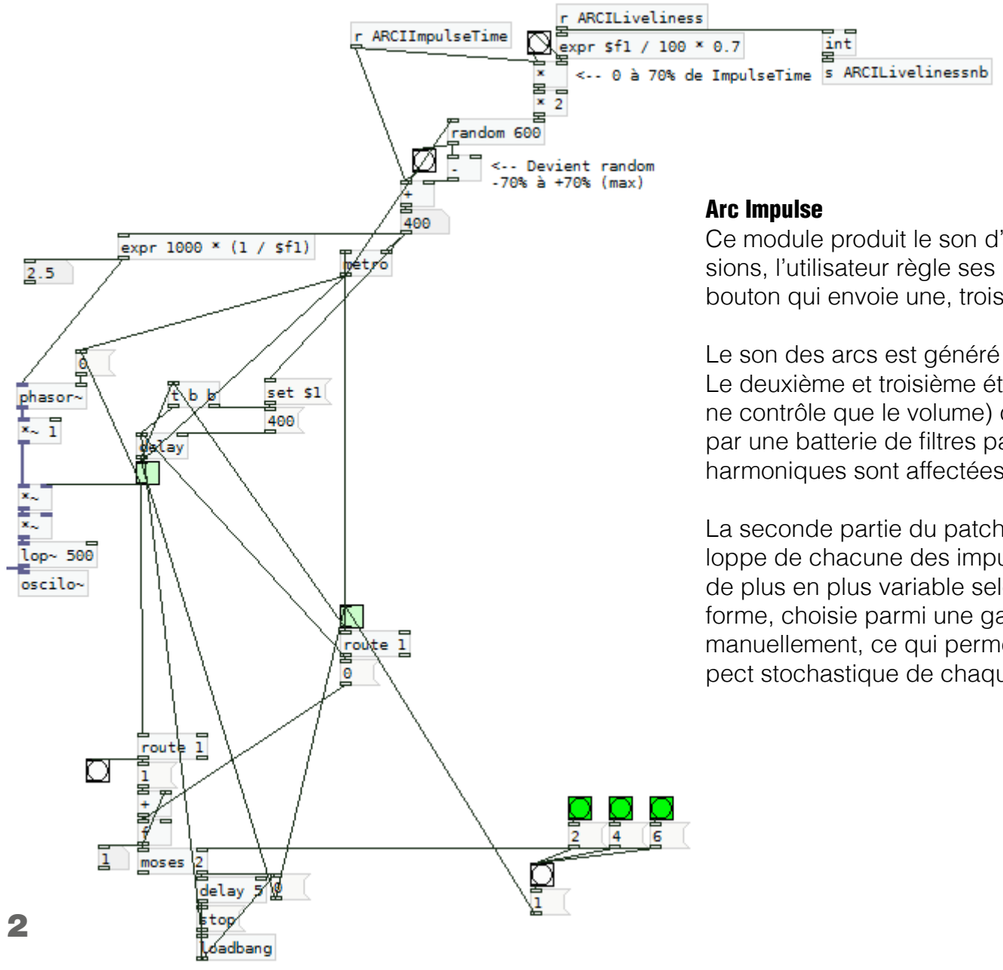



# AUDIOELEC

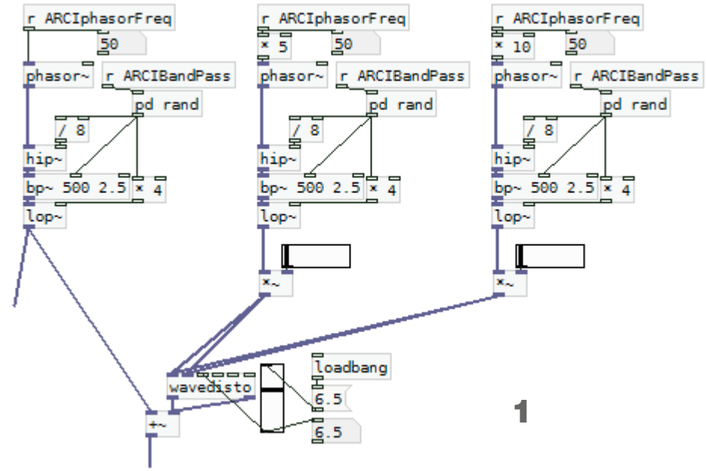



2

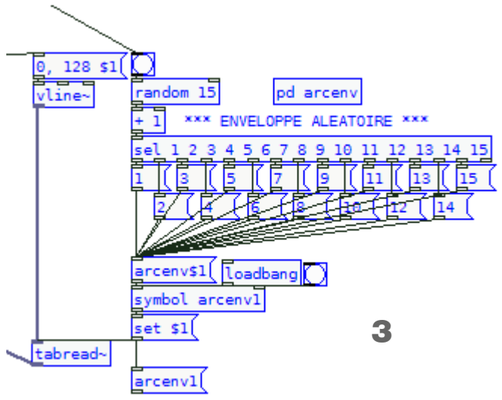
**Arc Impulse**
Ce module produit le son d'arcs électriques par impulsions, l'utilisateur règle ses paramètres et appuie sur un bouton qui envoie une, trois ou cinq impulsions à la suite.

Le son des arcs est généré par un, deux ou trois phasors. Le deuxième et troisième étant les harmoniques (dont on ne contrôle que le volume) du premier. Chacun passe par une batterie de filtres partiellement contrôlables. Les harmoniques sont affectées d'une légère distorsion. (1)

La seconde partie du patch permet de contrôler l'enveloppe de chacune des impulsions. A savoir leur temps, de plus en plus variable selon le liveliness (2) et leur forme, choisie parmi une galerie d'enveloppes dessinées manuellement, ce qui permet d'augmenter encore l'aspect stochastique de chaque impulsion. (3)

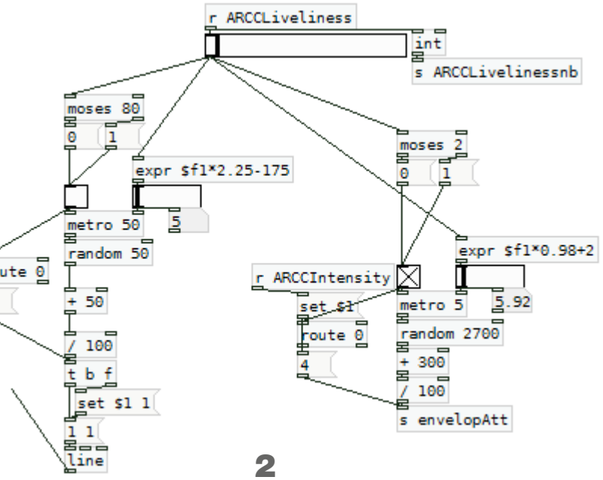



1




3

# AUDIOELEC

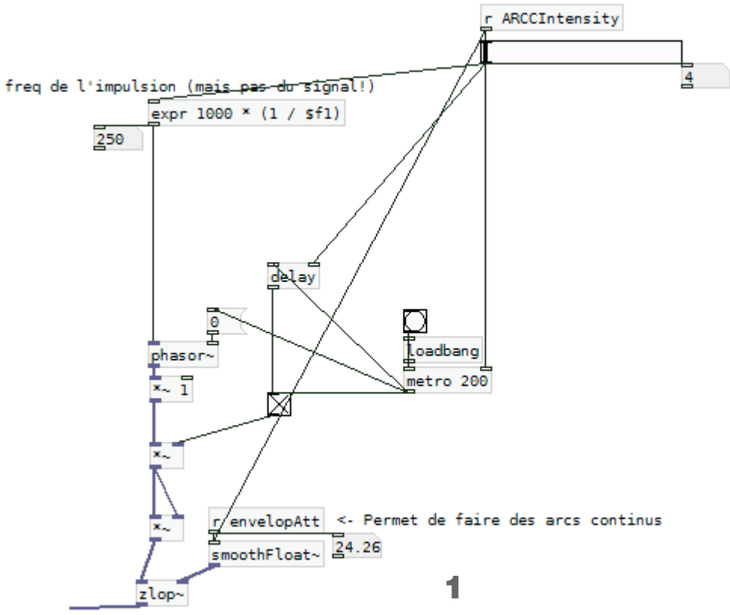



2

**Arc Continu**
Sur la même base que les arcs par impulsions, ce module possède le même générateur sonore. Mais ici, il ne lance pas de longues impulsions, mais cherche plutôt à générer un grésillement continu en envoyant beaucoup d'impulsions très courtes de l'ordre de la dizaine de millisecondes.

Un lowpass permet d'adoucir l'enveloppe des impulsions pour donner un meilleur rendu de grésillement. (1)

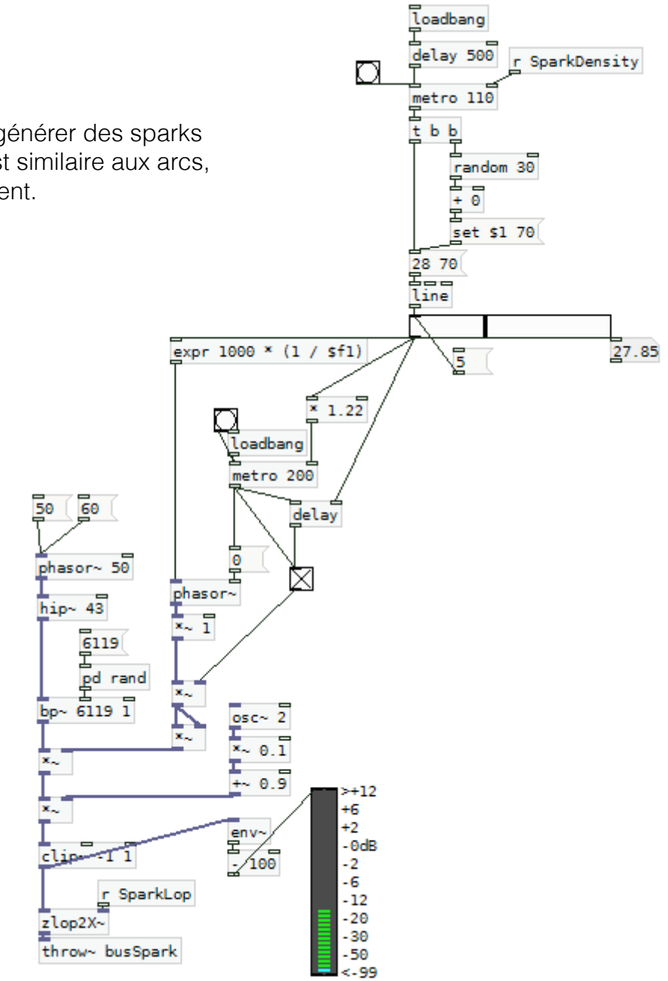
Le liveliness de ce module est en deux temps. De 0 à 80% (0 exclu), il envoie des valeurs aléatoires (que nous avons restreintes) au lowpass toutes les 5 vers 100 millisecondes ce qui a pour effet de donner plus d'intensité et de vie au grésillement. De 80 à 100%, il envoie des valeurs aléatoires (retreintes aussi) au volume du signal sortant du module. (2)



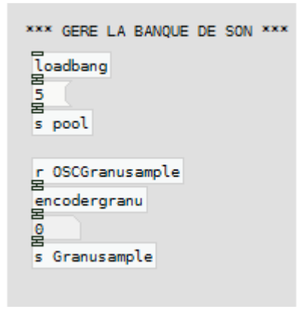

1

# AUDIOELEC

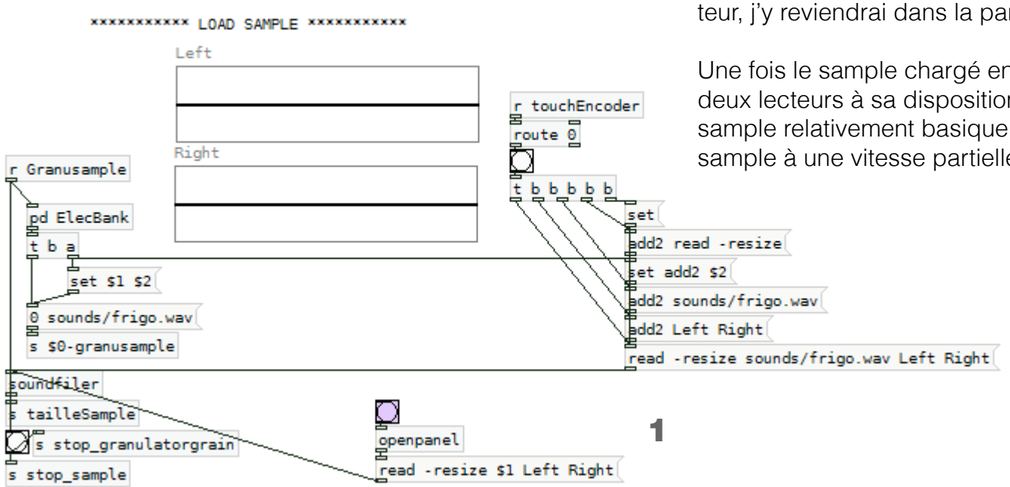
**Sparks**
Ce module a pour simple but de générer des sparks assez vifs. Son fonctionnement est similaire aux arcs, mais le son de synthèse est différent.



# AUDIOELEC




2



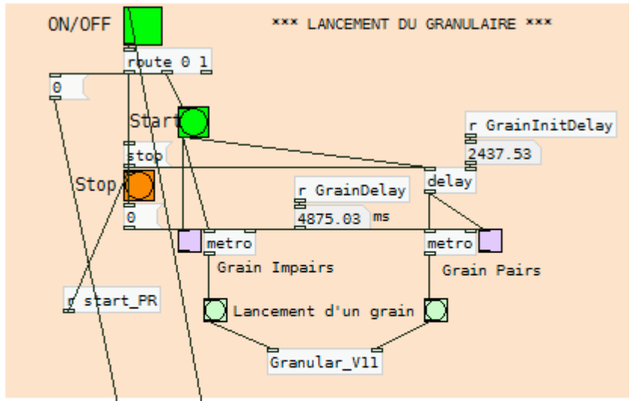

1

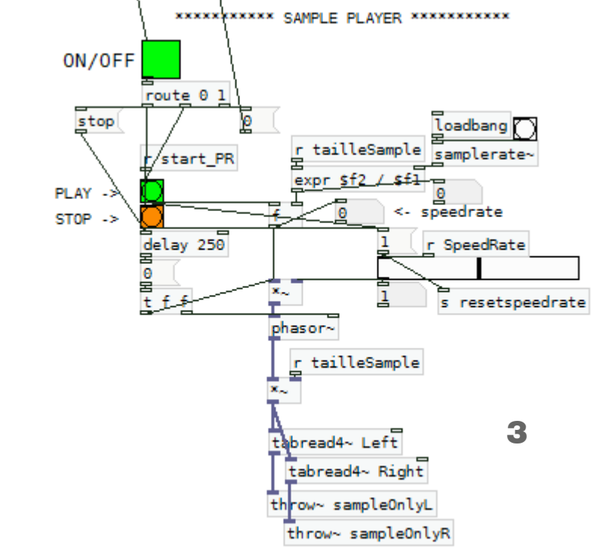
**Lecteur Granulaire**
Ce module comporte plusieurs parties. Un lecteur de sample et un lecteur granulaire indépendants l'un de l'autre et un chargeur de sample pour les deux lecteurs.

Le chargeur de sample charge en mémoire dans deux array (Left/Right) le son choisi par l'utilisateur parmi une banque prédéfinie (Object ElecBank), du moins dans ce qu'il a été possible de programmer via PureData. En effet, l'utilisateur pourra normalement charger ses propres sons, mais cette partie se programme plus tard, en C/C++. (1)

Le bloc gris sert à récupérer la variable donnée par l'encoder, un contrôle un peu spécifique offert à l'utilisateur, j'y reviendrai dans la partie "Les contrôles". (2)

Une fois le sample chargé en mémoire, l'utilisateur a deux lecteurs à sa disposition, tout d'abord un lecteur de sample relativement basique qui se contente de lire le sample à une vitesse partiellement contrôlable. (3)



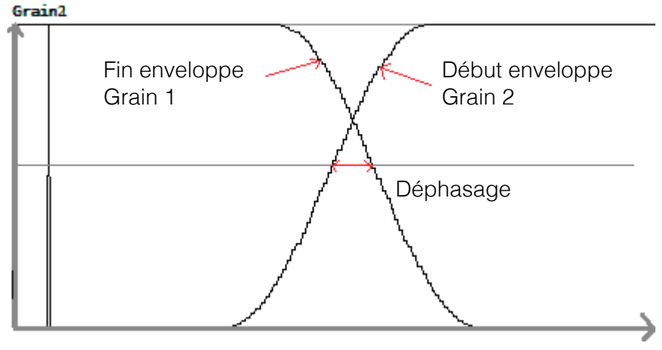




3

# AUDIOELEC

Le lecteur granulaire quant à lui, fabrique des échantillons à partir du sample afin de les enchaîner les uns après les autres par des crossfades. Nous n'avons pas laissé le choix à l'utilisateur de contrôler les valeurs du lecteur granulaire, par soucis de simplicité pour celui-ci. Ce lecteur, particulièrement, est destiné à créer des nappes comme par exemple un buzz de néons, le bruit d'un frigidaire etc. Aussi, la taille des échantillons est fixé à 3000 ms, le temps des fades in et out à 450 ms et les fades sont de types Hanning.

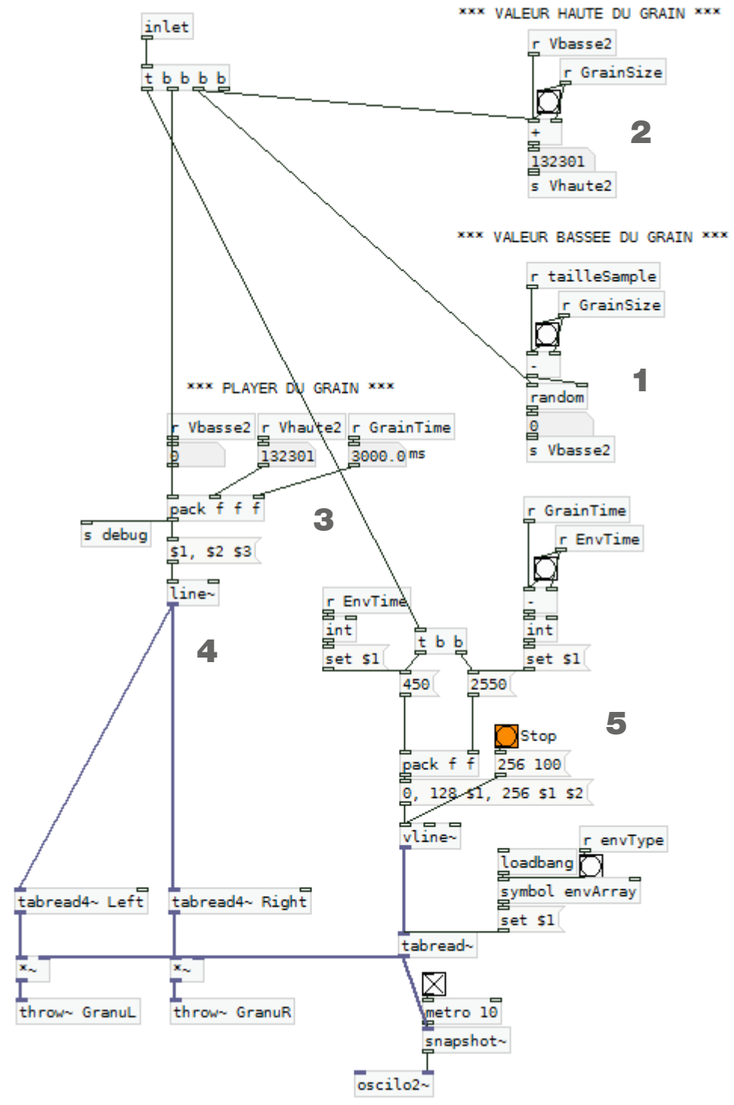
Au lancement, les grains sont lancés à intervalle d'un delay calculé selon la taille des grains, des fades et selon un déphasage supplémentaire afin qu'ils s'enchaînent parfaitement. Le déphasage permet simplement de décaler le crossfade pour qu'à l'oreille on n'entende pas de différence de niveaux entre les deux grains.




A chaque bang d'un nouveau grain, la machine choisit une valeur "basse" (1) du grain et en déduit sa valeur "haute"(2). Ces deux valeurs ainsi que le temps du grain sont packés (3) et envoyés dans un objet line qui va lire l'endroit du sample où se trouve le grain. (4)

L'enveloppe de chaque grain est commandé par un objet vline qui va lire la forme de l'enveloppe (fixée de type Hanning) sur un array. (5)

Pour terminer, il n'est pas possible de lancer à la fois le lecteur de sample et le lecteur granulaire. Cela n'aurait pas vraiment de sens et provoquerait plus de la confusion qu'autre chose…



# AUDIOELEC

**Pip & Telemetry**
Un module assez lourd en termes de contrôles, mais qui est en fait assez simple.
Tout d'abord la partie "pip", génère un pip tone avec un signal carré. (1)

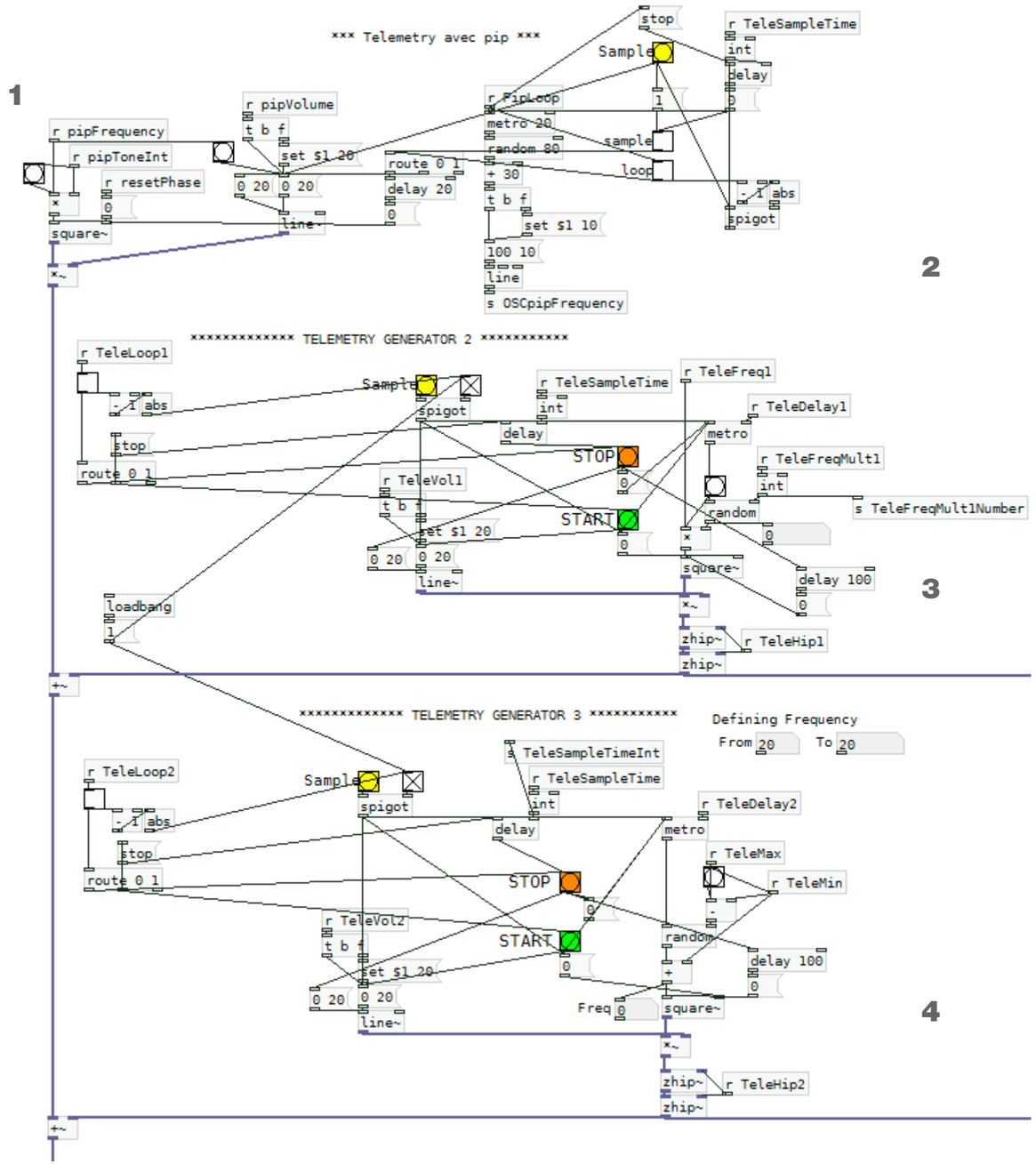
Au-delà de ça, le module comprend trois générateurs de télémétrie, idéal pour fabriquer très rapidement de petits sons d'interfaces pour les jeux vidéo par exemple. Chacun des trois générateurs peut être activé sur un temps défini par l'utilisateur mais il existe un mode loop permettant de générer du son continuellement.

Le premier est basé sur le signal carré vu précédemment. Il envoie des fréquences aléatoires au signal carré toutes les 20 ms. (2)

Le second multiplie une fréquence choisie d'un signal carré par un multiplicateur aléatoire. (3)

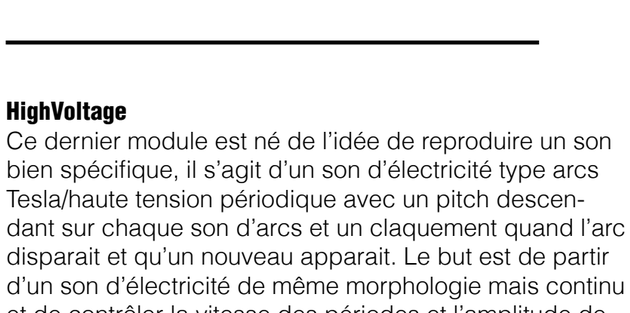
Enfin le troisième choisit, de façon aléatoire, une fréquence d'un signal carré entre deux valeurs définies par l'utilisateur. (4)

Dans les deuxième et troisième générateurs, l'utilisateur peut choisir un delay (très court) qui sera l'intervalle entre chaque nouvelle fréquence envoyée au signal carré. L'utilisateur peut aussi utiliser un highpass sur chacun d'eux s'il le souhaite.



# AUDIOELEC

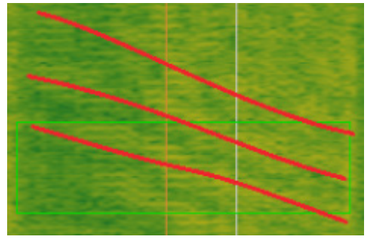
**HighVoltage**
Ce dernier module est né de l'idée de reproduire un son bien spécifique, il s'agit d'un son d'électricité type arcs Tesla/haute tension périodique avec un pitch descendant sur chaque son d'arcs et un claquement quand l'arc disparait et qu'un nouveau apparait. Le but est de partir d'un son d'électricité de même morphologie mais continu et de contrôler la vitesse des périodes et l'amplitude de la rampe de pitch. Ce n'est pas vraiment évident à l'écrit, le mieux reste d'écouter le résultat !


3

Pour ce faire, on lit le sample continu avec un lecteur de sample similaire à celui que l'on a vu précédemment. Et on lui applique une rampe de pitch périodiquement. Celle-ci est générée à partir d'un phasor inversé. (1)

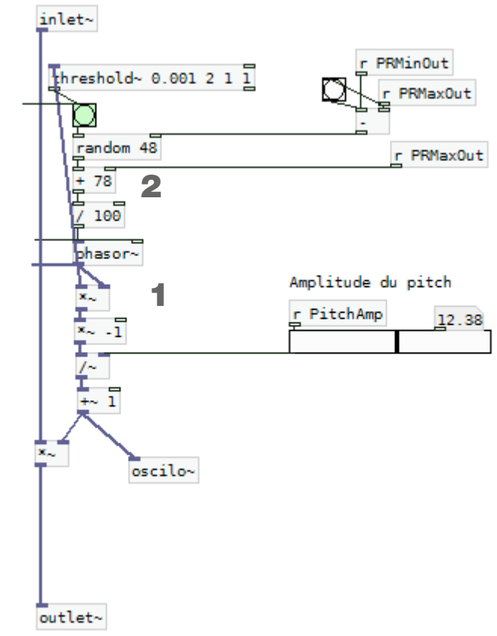
Pour obtenir un ordre d'idée de la taille d'une période, on analyse le spectre du signal que l'on veut reproduire, et on transforme les temps observés en fréquence. Ensuite, on donne à l'utilisateur le contrôle de cette période par rapport à notre résultat. (2)

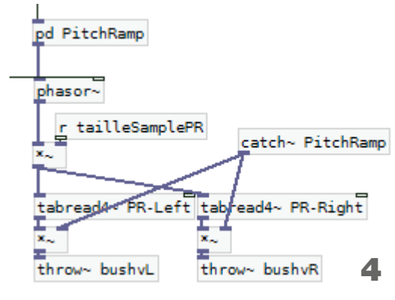


*La fréquence harmonique diminue, d'où le down-pitching.*

L'amplitude du pitch est obtenue en divisant le signal sortant du phasor, ce qui a pour effet d'atténuer plus ou moins la rampe. (3)

Pour créer le claquement final de chaque période, on utilise le signal en sortie du phasor (avant de le passer au carré et de l'inverser) comme enveloppe finale, qui aura donc la même période. (4)



# AUDIOELEC

## LES CONTRÔLES

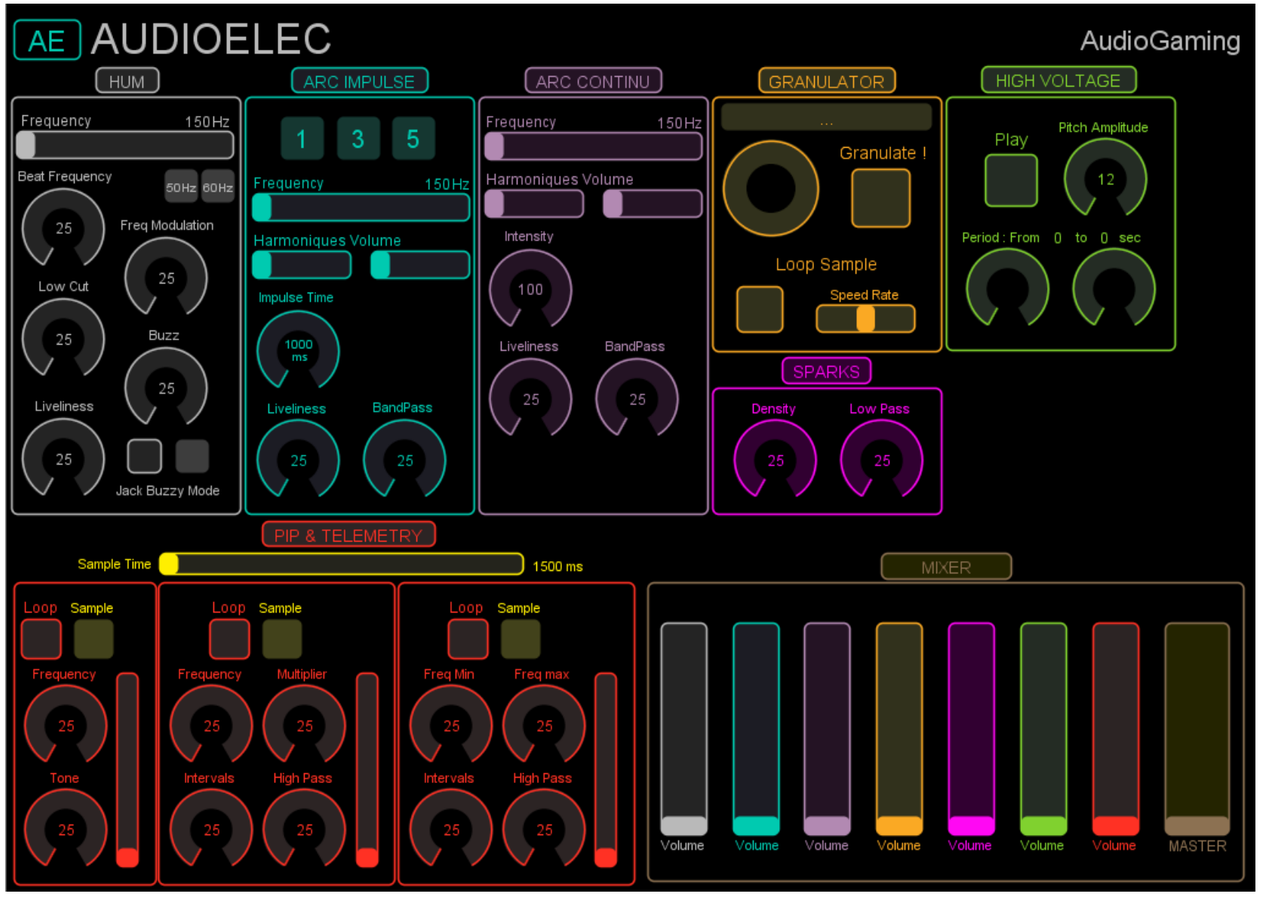
Le but du plug-in est d'être à la fois simple à prendre en main et assez large en choix de contrôles. Il faut trouver le bon compromis entre **palette de contrôles** et **ergonomie**. Une surcharge de contrôles ajouterait des tonnes de possibilités, mais rendrait la recherche de sonorités laborieuse et confuse. Nous avons donc restreint un certains nombre de contrôles pour cibler les sonorités recherchées par module.

Pour contrôler les paramètres de chacun des modules, l'utilisateur a accès à plusieurs types de contrôles :
Les **faders** ou **sliders**, sont à l'origine des potentiomètres linéaires (à bande), c'est donc un héritage de l'analogique.
Les **push buttons**, ou bouttons poussoir, envoient une impulsion, 1 ou 0 logique, et servent de déclencheur.
Les **toggle** et **multi-toggle**, sont des interrupteurs, qui à la différence des push buttons conservent leur état entre deux appuis. On les utilise notamment pour les solo/mute et on/off.
Les **knobs** sont à l'origine des potentiomètres axiaux à axe vertical ou des résistances variables crantées. Ils sont utilisés pour toutes sortes de contrôles y compris ceux couverts par les autres catégories.
Les **encodeurs** ressemblent aux knobs mais n'ont pas de fin de course, ils sont issus de l'électronique numérique et sont souvent couplés avec un afficheur. Ils permettent de naviguer dans des menus et/ou de contrôler plusieurs paramètres.

Deux interfaces sont proposées à l'utilisateur : d'une part l'interface du plug-in directement sur l'écran d'ordinateur, donc contrôles à la souris. Pour cela, l'abstraction "parameter" permet au programmeur C/C++ de récupérer les variables contrôlables et de communiquer avec le patch.

# AUDIOELEC

D’autre part une interface tactile sur iPad via le protocole OSC, disponible gratuitement sur le site d’AudioGaming. Ici, nous devons gérer la communication par OSC grâce à deux abstractions “OSCsend” et “OSCreceive”.



# BLENDER

## LE PROJET

Blender est un plug-in audio **d'expérimentation de sound design**. Le principe du plug-in est de sélectionner de un à trois samples et de contrôler une **batterie d'effets** et **la spatialisation** appliqués à ces sons en temps réel. Cette application est destinée aux sound designers de tous bords (cinéma, télévision, documentaire, web, jeux vidéo…) et tend à innover en termes de recherche et d'expérimentation en offrant un résultat immédiat.

Blender est conçu comme une application tactile sur iPad où chacun des effets est contrôlé par des "**pads XY**", un nouveau type de contrôle très intuitif. Ces pads sont présents sur la version PC mais ne peuvent pas être contrôlés simultanément en temps réel sans interface tactile. Mais il est bien sûr possible de passer par des automations.

L'équipe porteuse du projet est la même que celle d'AudioElec.

J'ai repris le projet après le départ de Victorien qui l'avait bien entamé, j'ai donc dû apprendre à lire son patch afin de me l'approprier et de comprendre son fonctionnement avant de travailler dessus.
Le patch étant bien plus conséquent qu'AudioElec, je ne le détaillerai pas entièrement mais m'attarderai sur son fonctionnement, son contenu et ses contrôles d'un point de vue plus large que le détail d'AudioElec.

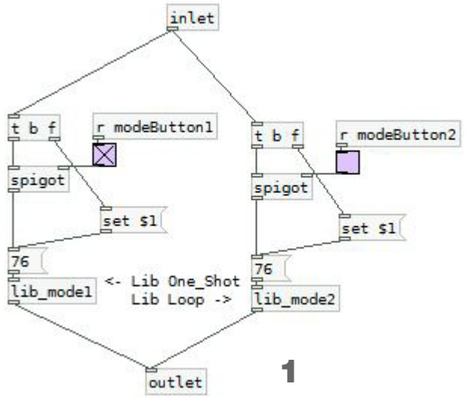

*Une vidéo du prototype est disponible ici :* http://youtu.be/u--eeOdgrwI

# BLENDER

## FONCTIONNEMENT

### Banque de sons

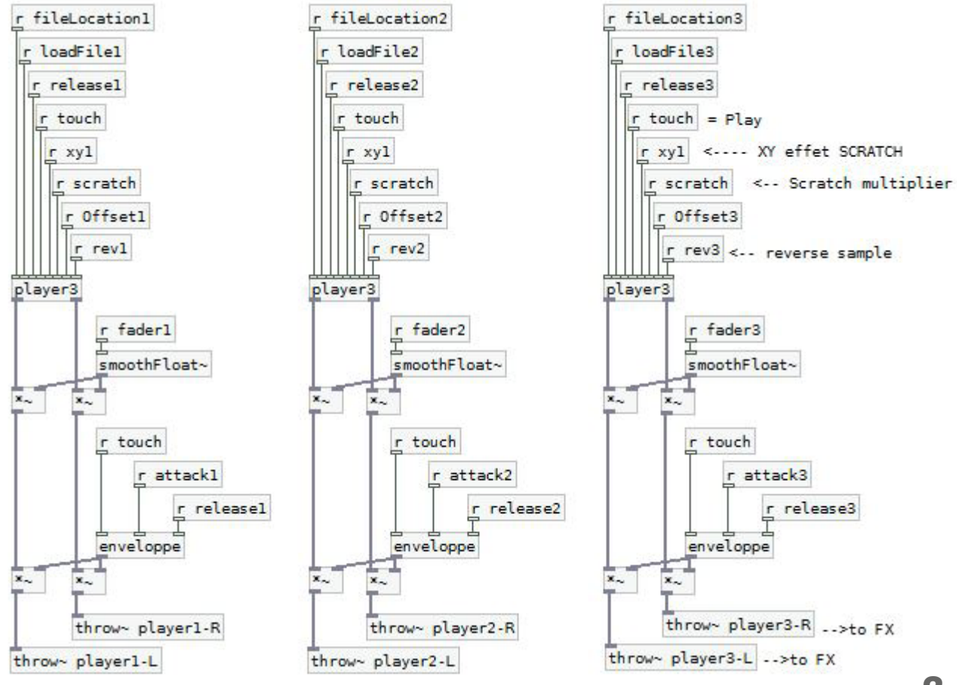
On sélectionne des sons dans la librairie offerte par l'application. La librairie est divisée en deux catégories : "**One Shot**" et "**Loop**". La première comprend des samples non bouclables comme un cri d'oiseau, une voix, un impact… A l'inverse, la seconde catégorie contient des nappes, des sons de moteurs, d'eau et autres sons bouclables.
Tous les samples offerts par Blender sont répertoriés dans des messages et envoyés dans les arrays des players une fois choisis. (1)




1

### Players

Blender propose **trois samplers** identiques à l'utilisateur. Ils permettent de charger et jouer les samples choisis et de définir plusieurs **paramètres** avant l'application des effets, à savoir l'attaque, la release, l'offset, s'il est joué en reverse ou pas, le volume.
Les trois samplers sont activés en même temps lorsque l'utilisateur lance la lecture.
Au niveau du patch, les players sont similaires à ceux d'AudioElec. (2)

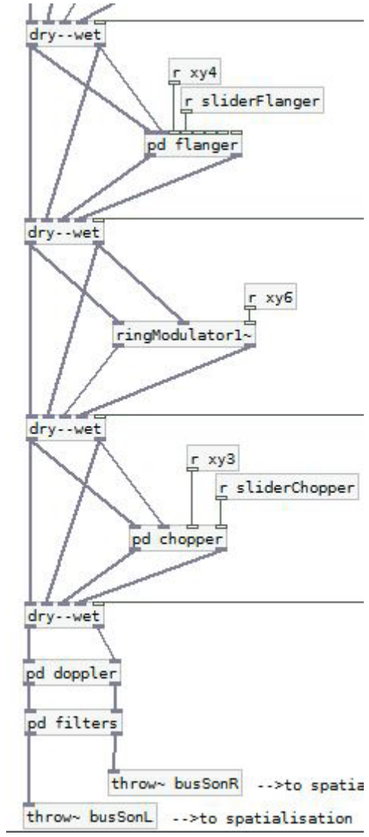



2

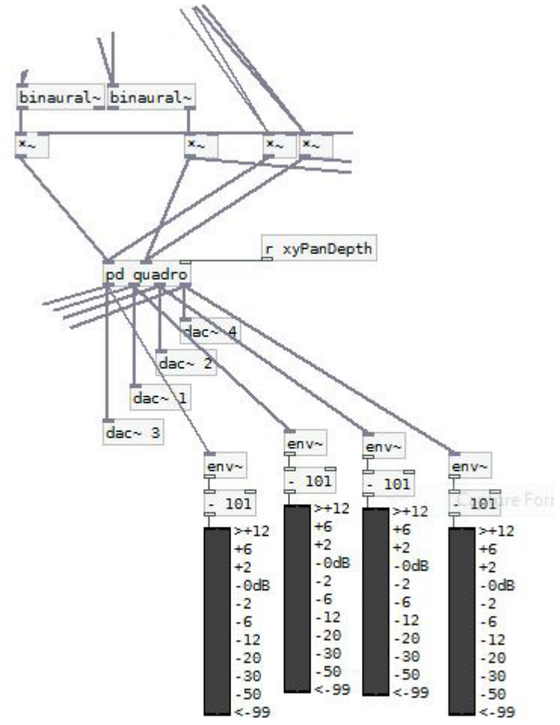
# BLENDER

**Effets**
Les effets sont ensuite appliqués, en série, aux sons joués. Il est donc possible d'appliquer une dizaine d'effets à la fois sur les sons joués ! La banque d'effets comprend une distorsion et un chorus par son, un scratch, un pitch granulaire, un chopper, un flanger, un multidelay, un ring modulator, un doppler et des filtres (un bandpass, un low pass et un high pass) tous contrôlables en temps réel. Je n'ai créé que les patchs du doppler et des filtres personnellement, les autres effets viennent de sources du web ou ont été fabriqués par mon collègue Victorien. Chaque effet possède un Dry/Wet et peut être simplement bypassé. (3)

**Spatialisation**
Enfin, le signal est réparti selon la spatialisation choisie dans notre système de diffusion préféré. Blender propose deux modes de spatialisation : **multicanal 4.0** et **binaural**. Il est possible de jouer avec la spatialisation en temps réel de la même manière que les effets ! (4)




3

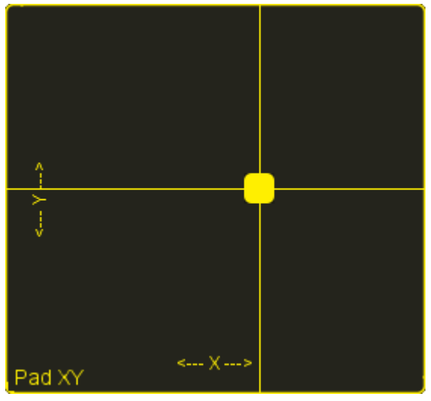



4

# BLENDER

### LES PADS XY

L'interface de Blender est caractérisé par sa palette d'effets dont la plupart sont contrôlables avec un pad XY. On pourrait le définir comme un **double slider**, horizontal et vertical. En effet, le contrôle récupère à lui seul deux valeurs, une en X et l'autre en Y. Deux paramètres sont donc contrôlables dans un carré avec un seul point, et donc, avec un seul doigt.




### LES EFFETS

La dizaine d'effets proposés par Blender et leurs contrôles intuitifs à résultat immédiat offrent des possibilités quasi-infinies en termes d'expérimentation sound design. Voici une petite description de chacun d'eux.

#### Distortion
Plus précisément ici, il s'agit d'une distorsion d'amplitude, et s'obtient par un écrêtage du signal.

#### Chorus
Dans Blender, le chorus additionne deux signaux identiques à celui d'origine avec un léger déphasage et une légère variation de fréquence.

#### Scratch
Le scratch diminue ou augmente la vitesse de lecture du sample.

#### Pitch granulaire
L'intérêt du pitch granulaire est de pitcher le sample sans avoir les inconvénients du changement de vitesse de lecture. Pour cela, le patch découpe le sample en grains qui sont alors transposés par resynthèse et enchaînés avec des enveloppes adaptées afin de conserver la vitesse de lecture d'origine.

#### Chopper
Le rôle du chopper est de découper le volume du son joué par battements plus ou moins fréquents avec plus ou moins d'intensité. Et comme son nom l'indique, le rendu ressemble au son d'un hélicoptère.

#### Flanger
Le flanger additionne à un signal ce même signal avec un léger delay variant périodiquement dans le temps. Dans Blender, on y contrôle le taux de signal traité injecté au signal d'origine, la variation périodique du delay (qui donne le son caractéristique du flanger) et son intensité (ou feedback) qui ajoute une partie du signal de sortie au signal d'entrée.

#### Multidelay
Delay possédant plusieurs lignes de retard. Le rendu agit comme un multi-écho voire même une reverb.

#### Ring modulator
Le ring modulator multiplie le signal d'entrée par une sinusoïde. Dans Blender, on y contrôle la fréquence de la sinusoïde et un simple filtre bandpass qui lui est associé.

#### Filtres
Blender propose trois types de filtres. Un bandpass, un lowpass et un highpass. On contrôle la fréquence de coupure de chacun des filtres, et le facteur de qualité du bandpass. Chaque filtre possède trois "forces". Chacune ajoute un filtre supplémentaire en série, dans le but d'augmenter la force du filtre en question.

# BLENDER

**Doppler**

L'effet doppler correspond au changement de la fréquence d'un signal  reçu par un observateur mobile (respectivement fixe) par rapport à une source émettrice fixe (respectivement mobile). Dans le cas de Blender, l'observateur (listener) est fixe et la source mobile est contrôlée par l'utilisateur. Il est à noter que les calculs suivants n'ont pas pour but d'être physiquement exacts en vue de donner l'effet doppler le plus réaliste possible. Le but de l'effet est d'offrir un rendu qui semble réaliste, bien sûr, mais sans forcément passer par des calculs trop scientifiques !

Le doppler est un effet dont le contrôle est très intéressant dans Blender. Il est contrôlé par un slider horizontal dont le centre représente la position du listener. La longueur du slider représente le segment parcouru par la source. Le rendu de l'effet doppler est donné par l'augmentation du pitch du son joué lorsque la source approche du listener et inversement, et l'apparition d'un filtre lowpass fonction de la distance au listener.

# BLENDER

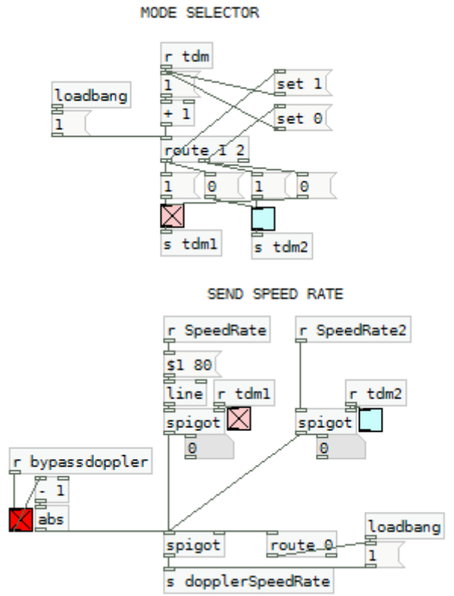
Nous avons mis en place deux modes de calculs différents. En effet, notre doigt contrôle la position de la source sur le slider en temps réel. (1) Ainsi, le premier mode calcule la vitesse de la source en temps réel pour en déduire le pitch à appliquer au son.

On calcule d'abord la distance entre deux points x1 et x2 à deux instants t1 et t2 séparés d'un intervalle très court (60 ms), ce calcul est renouvelé à chaque instant. (2)

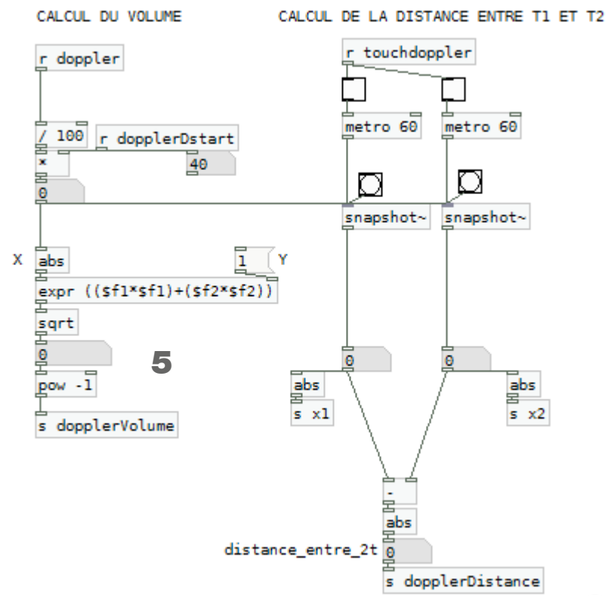
On en déduit alors une direction, si la source s'éloigne ou au contraire se rapproche du listener. (3)

De la même distance, on calcule un taux fonction de la longueur du segment (paramètre modifiable par l'utilisateur) qui s'ajoutera à la vitesse de lecture du son; c'est ce qui créera le pitch en fonction de la vitesse du glissé du doigt de l'utilisateur sur le slider. (4)
On notera que la vitesse de lecture du son n'est pas censée être changée. Mais au moment où j'écris ces lignes, nous n'avons pas implanté de pitch granulaire dans l'effet doppler à la place de la vitesse de lecture.

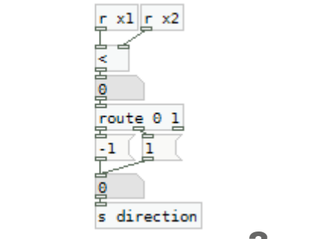
Le volume est une fonction 1/x calculée en fonction de la position de la source. (5)



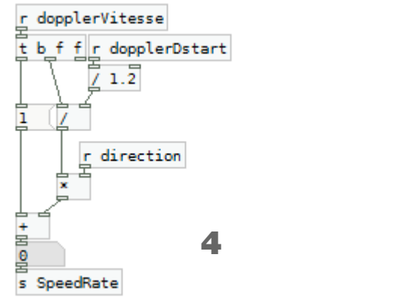

1




2




3



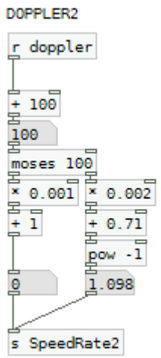

4

# BLENDER

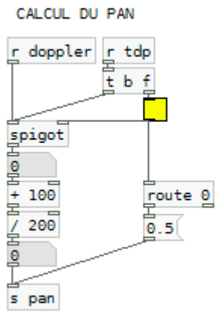
Le deuxième mode est de type fixe, c'est-à-dire que quelle que soit la vitesse du glissé de l'utilisateur sur le slider, le pitch reste inchangé. Il est précalculé et ne dépend que de la position de la source par rapport au listener. (6)

Il est possible d'associer le panoramique avec la position de la source sur le slider. On aura alors l'impression que la source passe devant nous. (7)

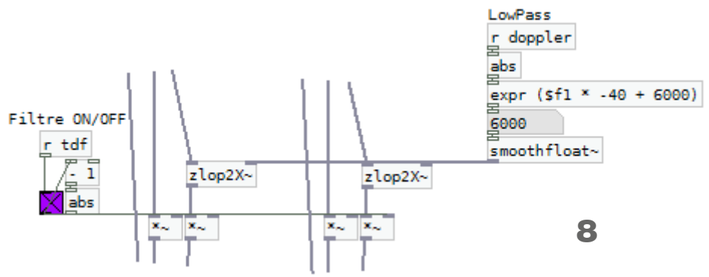
Enfin, un lowpass est placé en sortie du doppler. Plus la source s'éloigne, plus la fréquence de coupure est élevée. (8)




6




7




8

# BLENDER

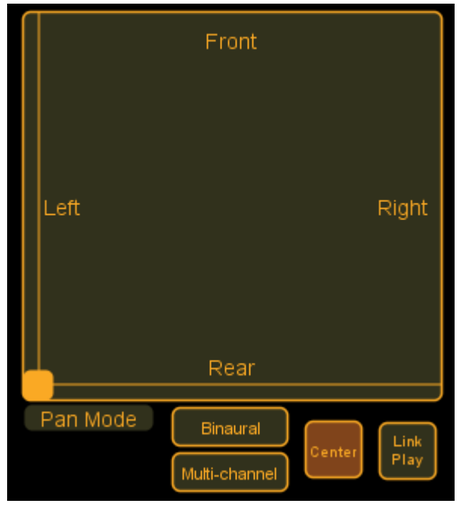
### LA SPATIALISATION

Contrôler la spatialisation de la même manière que les effets offre de nombreuses possibilités pour les sound designers, typiquement pour la création de whooshes spatialisés.

La spatialisation se contrôle comme la plupart des effets avec un “pad XY”. La posititon du listener est imaginée au centre du pad et le carré représente donc l’espace autour de nous avec l’avant, l’arrière, la gauche et la droite.

Deux modes de spatialisations sont proposés par Blender. Le mode **multicanal 4.0** permet de répartir le signal par modification du volume selon la position de la source sur le pad. On envoie simplement plus ou moins de son dans les enceintes.

Le mode **binaural** quant à lui utilise une banque de données HRTF (head-relative transfer function) et ITD (interaural time difference) pour spatialiser le son (sur les azmiths uniquement). Ce mode prend son sens, bien sûr, pour des sons à écouter au casque. Je reviendrai sur la spatialisation binaural dans la partie dédiée au projet suivant. La petite difficulté mathématique a été, pour ce mode, de faire correspondre le pad XY ([0-100];[0-100]) avec un cercle trigonométrique ...



# BLENDER

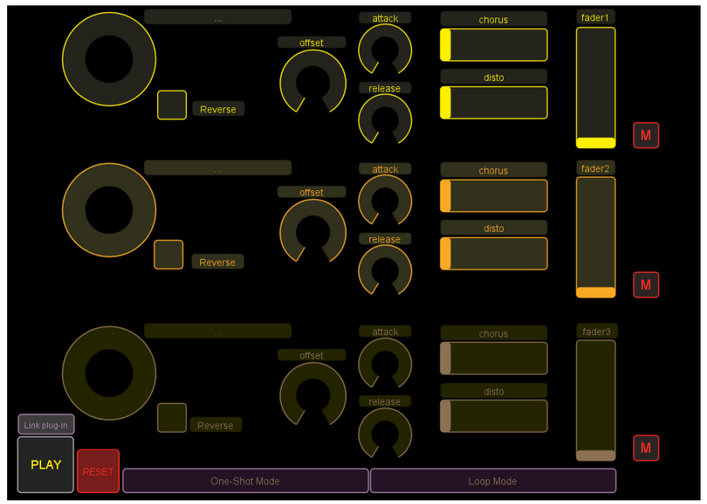
## LES CONTRÔLES

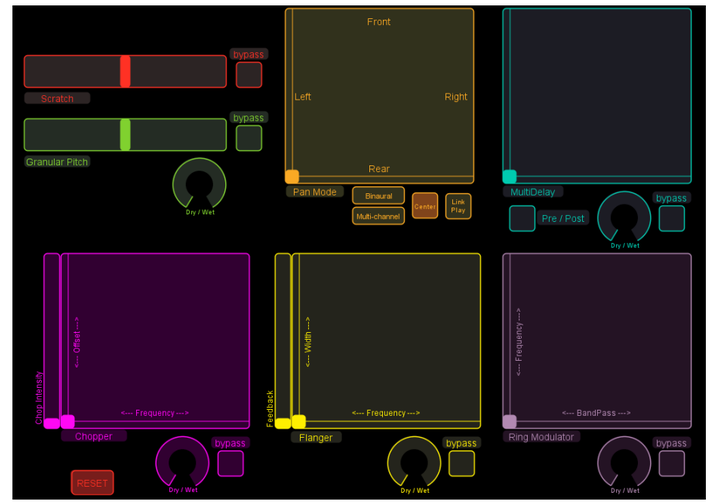
A l'instar d'AudioElec, les contrôles se doivent d'être variés pour offrir une palette de possibilités assez grande, sans trop laisser de leste à l'utilisateur, qui, si surchargé de paramètres modifiables mettraient trop de temps à obtenir un résultat rapidement et efficacement.

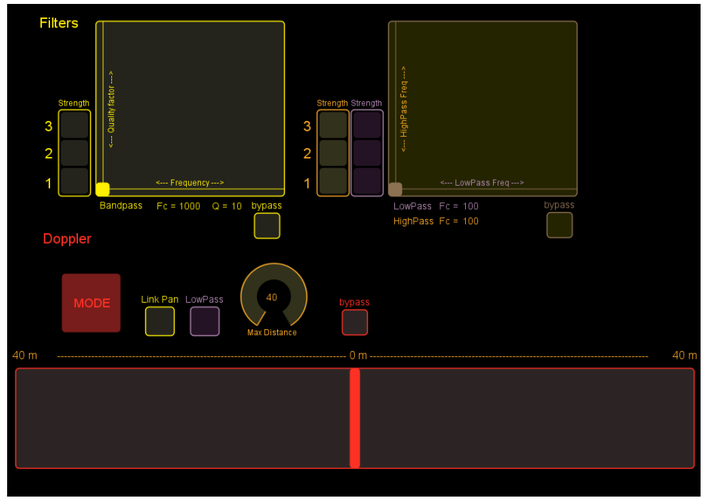
Blender donne donc à l'utilisateur le choix de jongler entre un à trois paramètres par effets maximum, ce qui se résume en termes de contrôles à un pad XY et un slider maximum.

Les types de contrôles sont les mêmes que ceux décrits dans AudioElec. Blender possède aussi deux interfaces, une pour l'écran d'ordinateur et une pour iPAD.

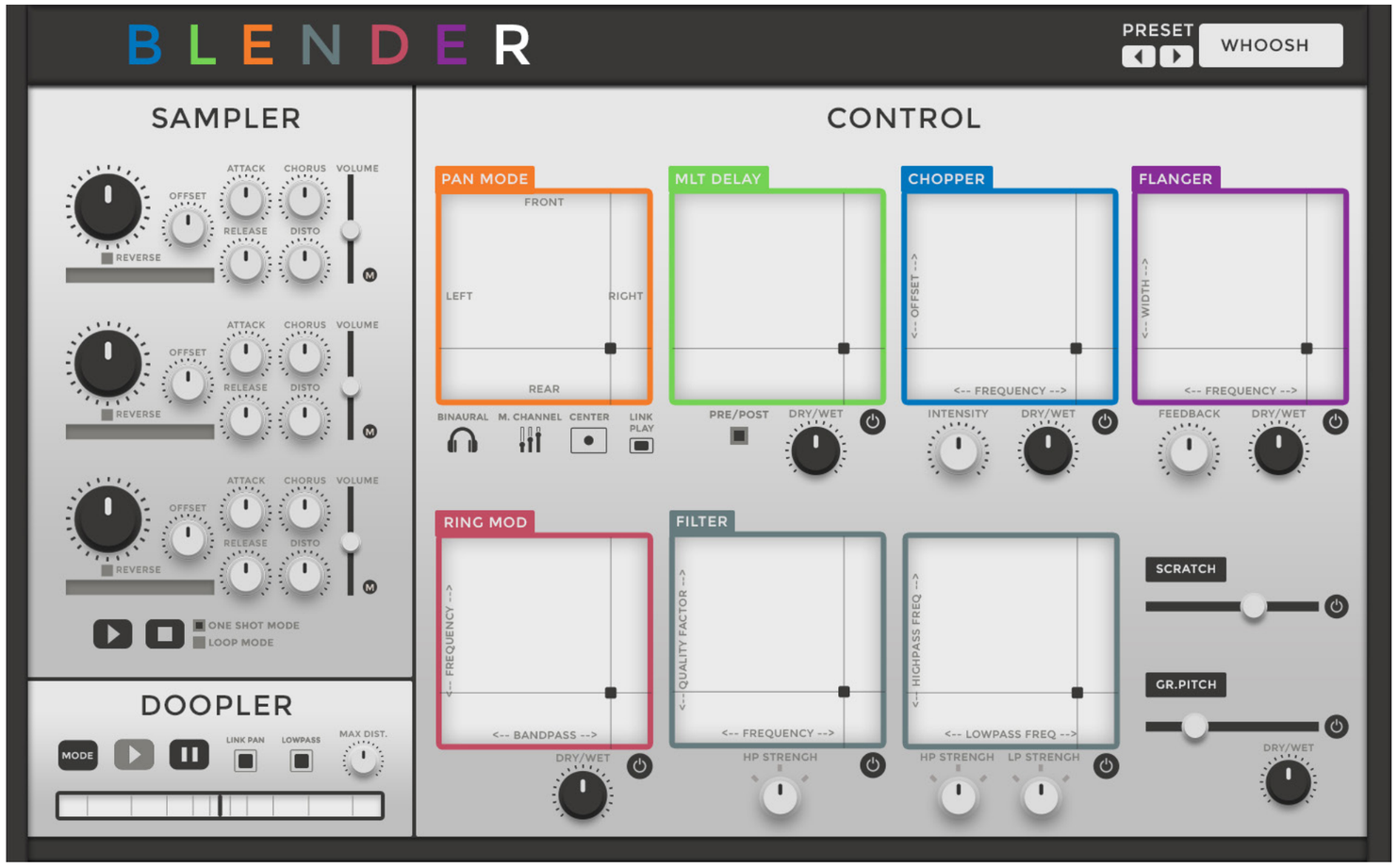
Les contrôles tactiles sont très importants dans Blender, c'est ce qui rend son utilisation si accessible et c'est aussi ce qui donne des résultats si intéressants. En effet, la notion de mouvement des doigts sur la tablette est quasi-impossible à simuler avec des automations à la souris.



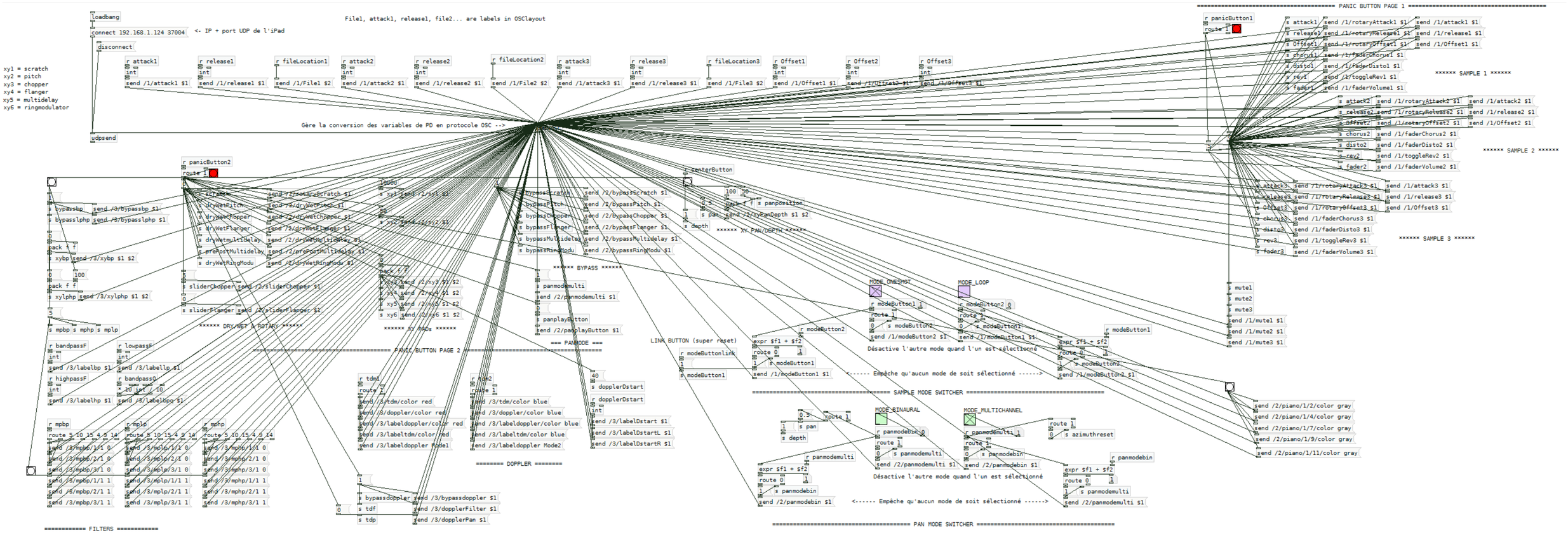



# BLENDER




*Interface All-In-One proposé par Kévin*

# BLENDER




*Le patch permettant d'envoyer des informations à l'iPAD. Le fameux phénomène "Spider web" de PureData.*

# NOTES ON BLINDNESS

**LE PROJET**

Notes On Blindness est une **expérience vidéo-ludique** sur iOS/Android proposant d'incarner un personnage aveugle. Le projet est inspiré d'un court métrage réalisé à partir de journaux sonores enregistrés par John Hull, écrivain et théologien devenu complètement aveugle en 1983. Vous pouvez voir la vidéo du court en suivant ce lien : http://vimeo.com/84336261

Ce projet interactift fait parti d'un documentaire Transmedia produit par **ARTE**, qui avait fait de même avec **Type_Rider**. Le documentaire comprend, entre autres, un film, une partie web et un projet interactif.

Le gameplay de l'expérience n'est pas encore tout à fait défini, il s'agira de traverser des espaces intérieurs et extérieurs en ne se repérant qu'au son. C'est pour cela que le projet s'intéresse grandement à la **spatialisation binaurale**.

Sur ce projet je travaille avec Robin, programmeur de la promotion 2010-2012 de l'ENJMIN. Et bien sûr avec Amaury.

# NOTES ON BLINDNESS

**PRÉCÉDEMMENT EN BINAURAL**

Il est important pour moi de revenir sur mon travail concernant la spatialisation binaurale effectué sur mon projet de Master 2 “Ascent”. En effet, celui-ci aura une influence certaine sur le projet.

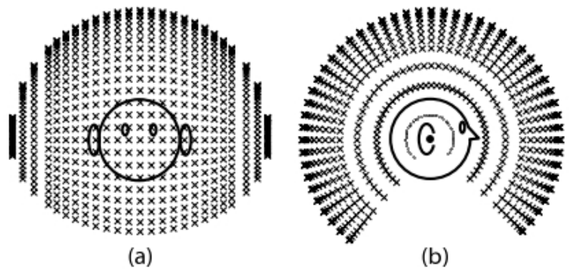
L’idée d’utiliser un **système de spatialisation binaurale** vient du fait qu’Ascent est un jeu entièrement conçu pour **Oculus Rift**. Il était donc plus interessant d’utiliser un casque pour jouer et afin de renforcer l’immersion procurée par l’Oculus Rift, nous avions choisi de créer notre propre système binaural. Pourquoi notre système ? Car à l’époque, il n’existait pas vraiment de système binaural fonctionnel simple à utiliser pour les jeux vidéo, et nous voulions que le système soit compatible avec notre moteur son **Wwise** d’Audiokinetic. Depuis, Audiokinetic a bien simplifié le problème et l’a plus ou moins solutionné par l’apparition de deux plug-ins, j’y reviendrai.

Avant de rentrer dans le vif du sujet, quelques définitions:
**La synthèse binaurale** est la technique de spatialisation sonore la plus proche de l’écoute naturelle. Le principe de cette technique repose sur la reproduction au niveau des oreilles d’un auditeur de toutes les informations nécessaires pour la construction d’une image sonore extra crânienne. La synthèse binaurale permet une localisation précise des sources sonores en trois dimensions ainsi qu’un rendu fidèle de l’environnement sonore (effet de salle). Contrairement aux autres systèmes de spatialisation sonore qui nécessitent parfois un nombre important de haut-parleurs, la diffusion des informations binaurales est réalisée grâce à un casque stéréophonique classique. L’utilisation d’un casque, et d’un système de suivi de mouvement, permet une immersion totale de l’auditeur dans une scène sonore sans interaction avec le monde extérieur et la synthèse n’est pas réduite à une zone de reconstruction optimale.

**Une fonction de transfert relative à la tête** (HRTF - head-related transfer function) caractérise les transformations apportées aux ondes sonores par le corps d’un auditeur, principalement la tête, le pavillon de l’oreille et le conduit auditif, qui permettent à l’être humain de repérer l’origine d’un son horizontalement (on parle d’azimuth) et verticalement (on parle d’élévation).

**La différence interaurale de temps** (ITD - Interaural Time Difference) est la différence de temps que met le son à atteindre une oreille par rapport à l’autre.

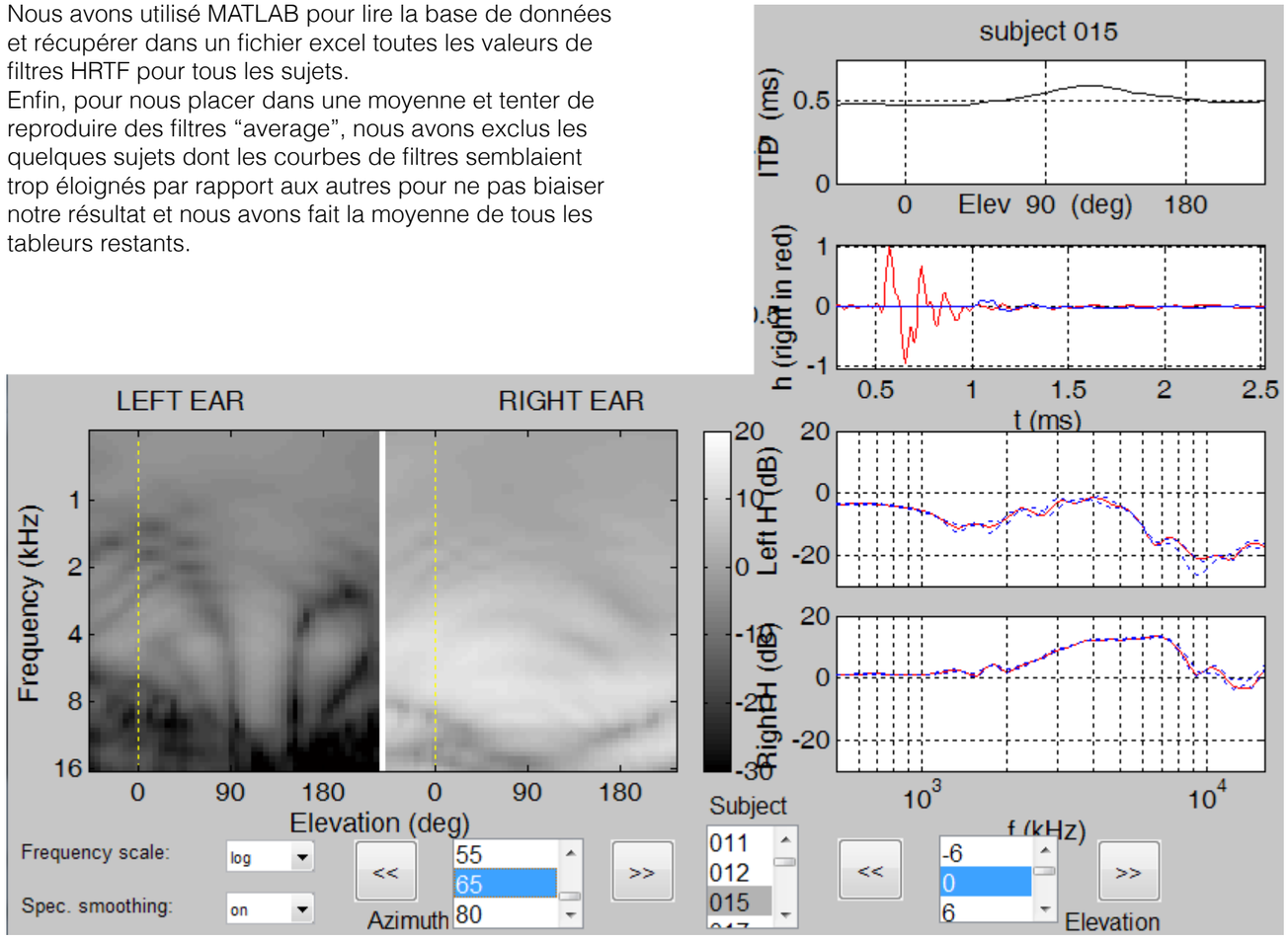
Afin de reproduire un système binaural, il faut donc d’abord recréer les filtres HRTF. Pour cela, nous avions utilisé une base de données HRTF. Celle-ci contient plusieurs centaines de mesures de réponses impulsionnelles relatives à la tête pour plusieurs sujets. Pour notre système, nous avions utilisé la base de données HRTF du CIPIC Interface Laboratory basé en Californie.



(a) (b)

# NOTES ON BLINDNESS

Nous avons utilisé MATLAB pour lire la base de données et récupérer dans un fichier excel toutes les valeurs de filtres HRTF pour tous les sujets.
Enfin, pour nous placer dans une moyenne et tenter de reproduire des filtres “average”, nous avons exclus les quelques sujets dont les courbes de filtres semblaient trop éloignés par rapport aux autres pour ne pas biaiser notre résultat et nous avons fait la moyenne de tous les tableurs restants.

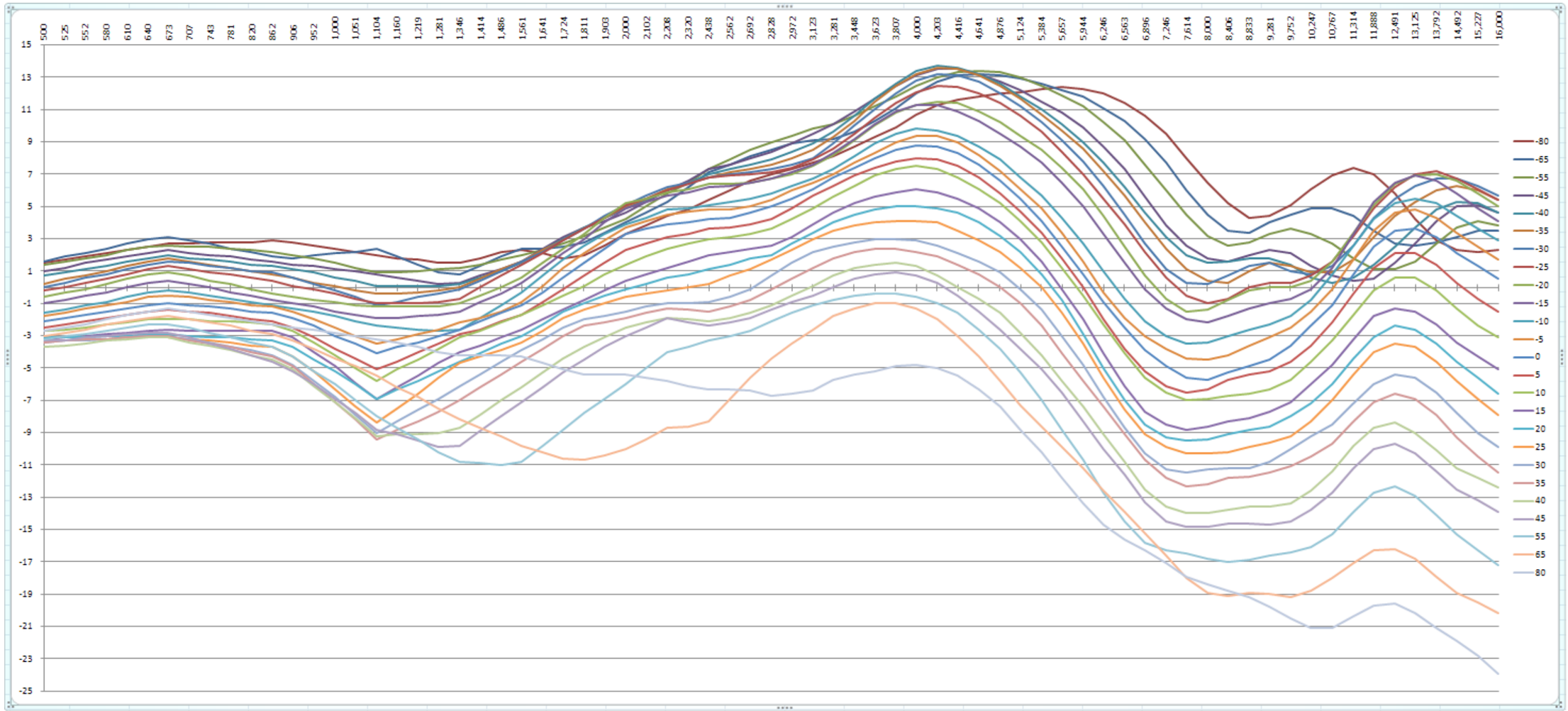



Une fois les tableurs transformés en graphiques, nous obtenons de très belles courbes de filtres ! En abscisse, nous avons un balayage du spectre fréquentiel audible en Hertz (Hz). En ordonnée, les niveaux en décibel (dB). Chaque courbe de couleur représente un angle à partir duquel a été envoyée l’impulsion.

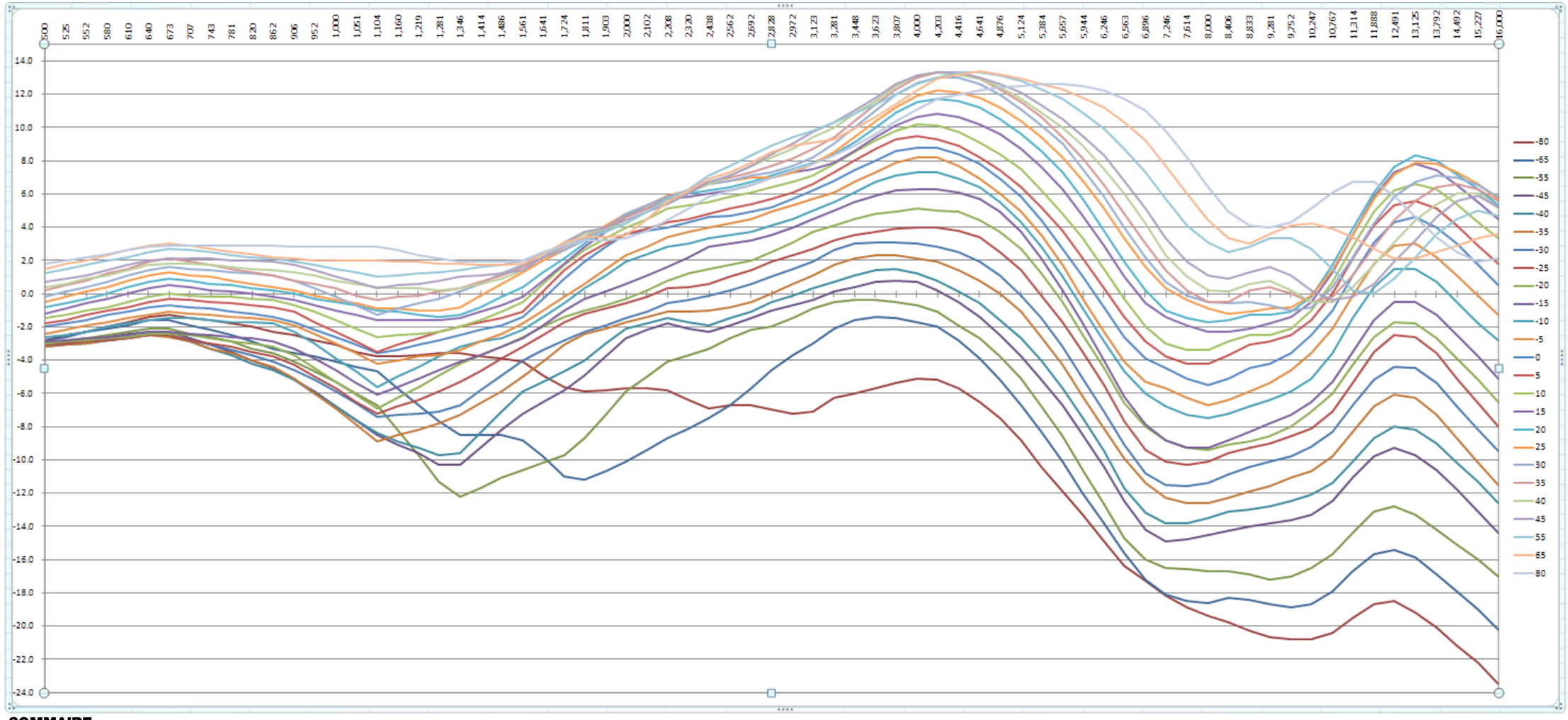
Nous avons besoin de quatre filtres. Trois concernent les azimuths : les HRTF affectant l’oreille gauche, celles affectant l’oreille droite et le filtre arrière. Le dernier concerne l’élévation. Il faut bien penser que l’on considère trois axes qui sont gauche/droite, avant/arrière et haut/bas. Donc dans notre moteur son, il s’agira d’additionner les filtres de chacun des axes.

Pour les azimuths -90° à 90° (de gauche à droite), nous avons gardé les filtres tels quels, mais pour chaque angle d’élévation, il faut soustraire le filtre zéro, c’est à dire le filtre à azimuth 0° et élévation 0°, car il existe bel et bien. Le fait d’avoir la source en face de notre tête n’empêche pas notre tête de filtrer. Sans cela, nous aurions une addition de deux fois le même filtre. Et pour ne pas non plus interférer avec le filtre arrière, on ne prend en compte pour l’élévation que la partie avant de notre tête, de -45° (devant vers le bas) à 90° (au-dessus de la tête). Pour l’arrière, nous n’avons dans la base de données que les valeurs à élévation -180° donc dans notre moteur son on reportera, pour des azimuths de -90° à -180° et de 90° à 180° (autrement dit l’arrière gauche et l’arrière droit) les valeurs allant de 0 à ces valeurs de façon linéaire. Cela est évidemment perfectible, mais le résultat est plutôt probant.

# NOTES ON BLINDNESS

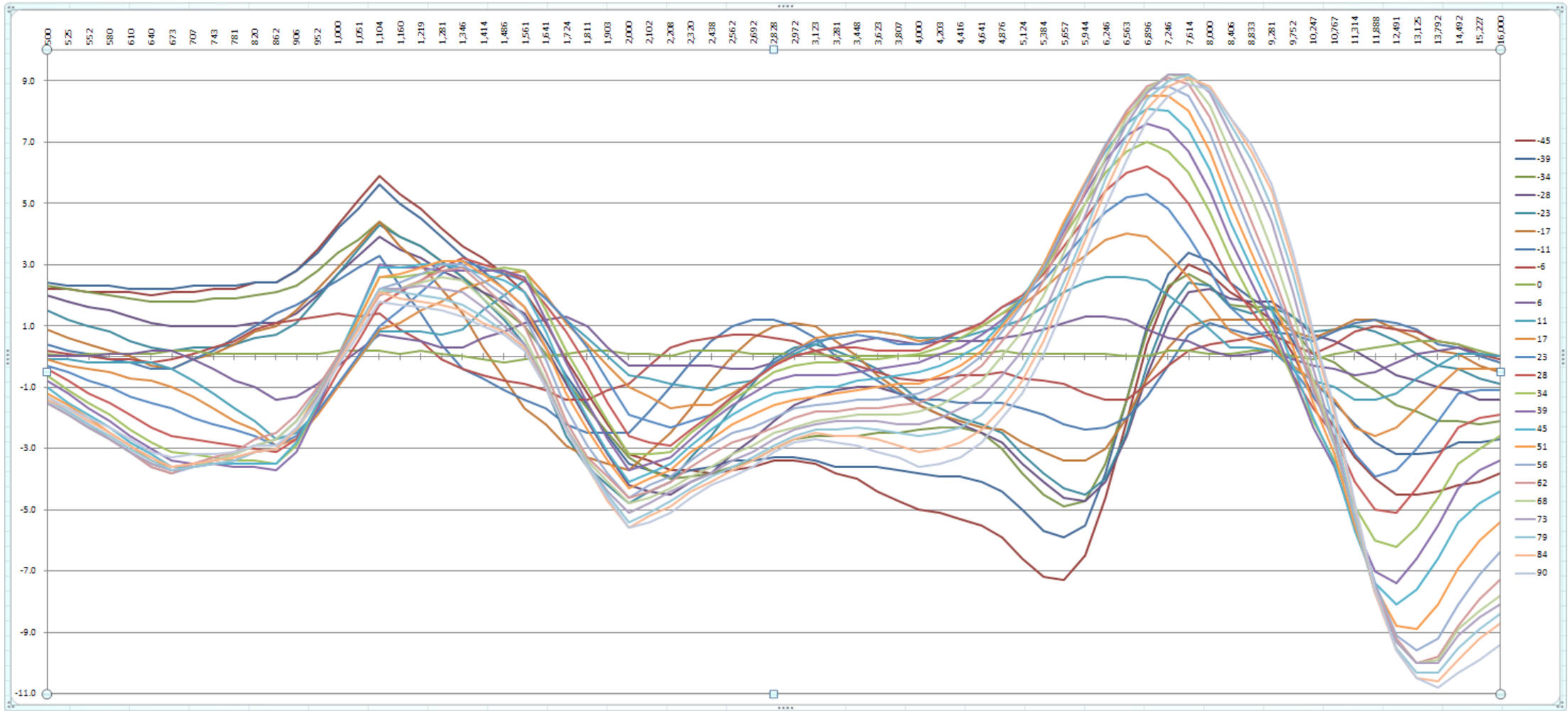


# NOTES ON BLINDNESS




*Evolution des filtres HRTF de l'oreille droite selon les Azimuths*

# NOTES ON BLINDNESS

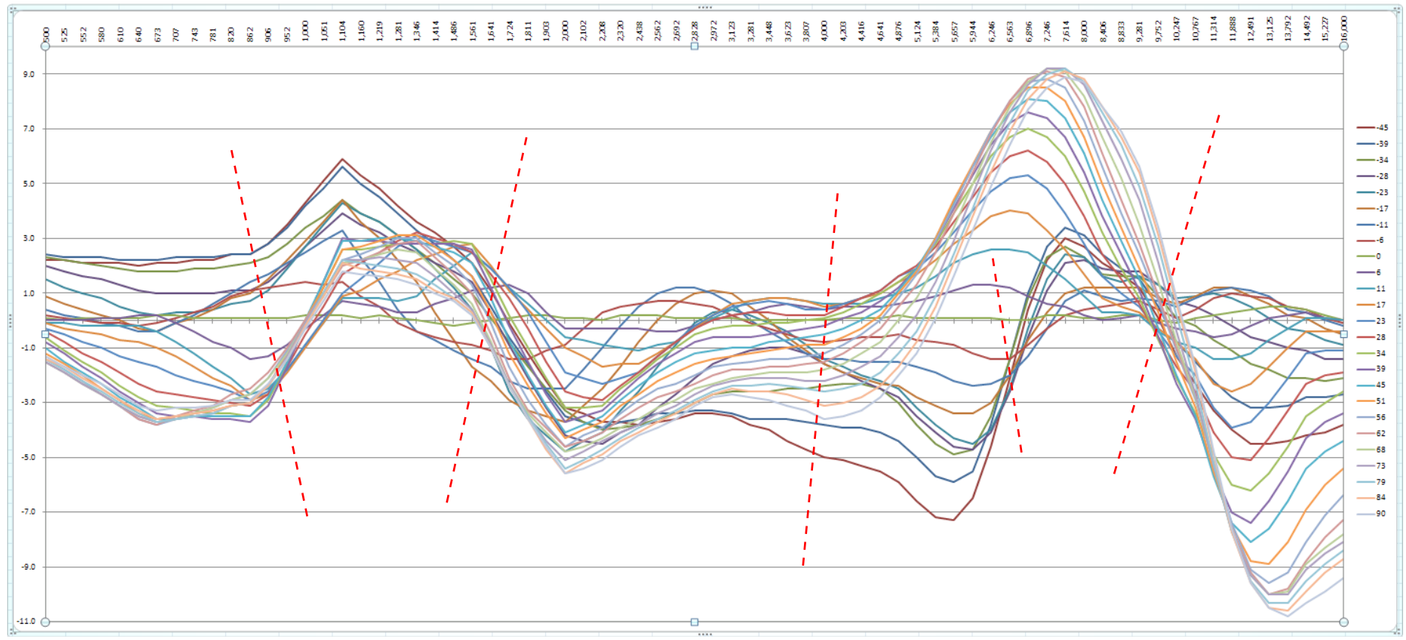



*Evolution des filtres HRTF des deux oreilles selon l'Élévation*

# NOTES ON BLINDNESS

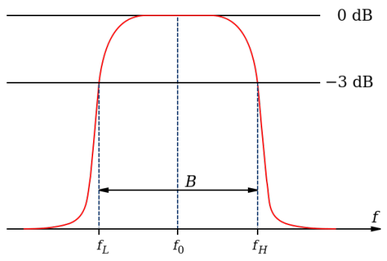
On veut alors reporter ces filtres dans Wwise.

Pour cela il faut en premier lieu effectuer un relevé de valeurs minutieux. On regarde combien de filtres on peut distinguer sur les graphes. Personnellement, j'ai associé ces filtres à des 6-Band, que ce soit pour les azimuths ou pour l'élévation. On différencie les types de filtres si besoin (bandpass, low-shelf, high-shelf, low-cut, high-cut). Dans l'exemple de l'élévation ci-contre, j'ai considéré les filtres le plus à gauche et le plus à droite respectivement comme un low-shelf et un high-shelf.



On repère, pour chaque filtre, la fréquence de coupure et on relève son évolution selon l'angle dans un tableau.

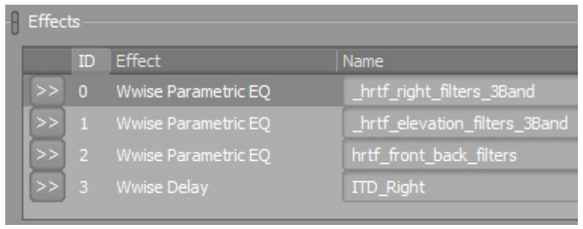
Pour calculer le facteur de qualité, on utilise la formule : Q = f0/(fH-fL) selon le schéma ci-dessous.




Le Q étant relativement constant, une moyenne de quelques valeurs suffiront à nous en donner une valeur unique.

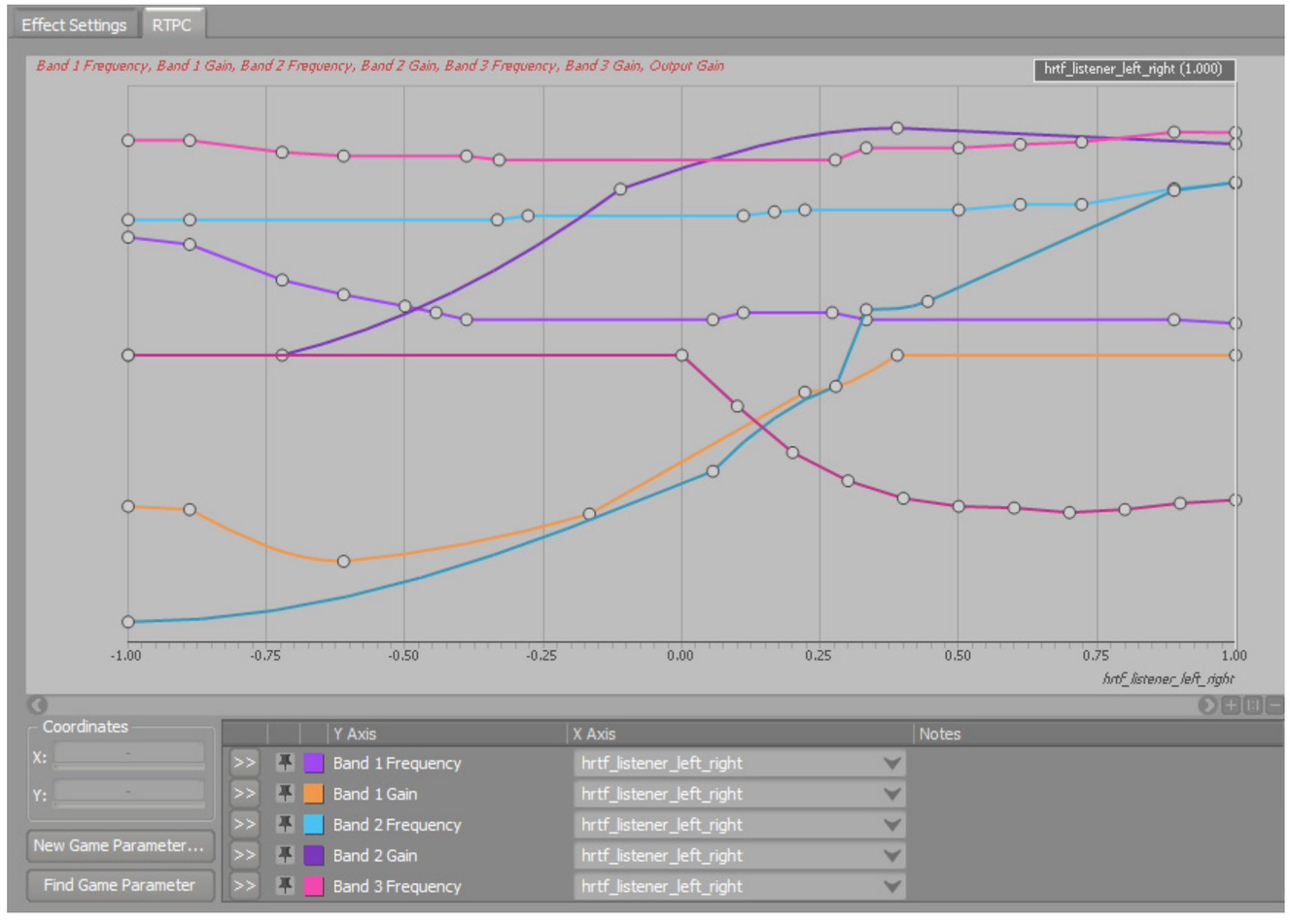
# NOTES ON BLINDNESS

Wwise ne propose que des filtres 3-Band. Il faudrait donc 5 x 3-Band pour les azimuths, l'élévation et le filtre arrière donc cinq slots d'effets par sources plus un slot supplémentaire pour le delay. Or Wwise ne propose que quatre slots par source. Ceci est le problème majeur que nous avons rencontré.




La solution ici a été de réduire les 6-Band à des 3-Band en éliminant les filtres les moins impactants. Le rendu en est légèrement affecté mais le delay (ITD) compense cette perte.

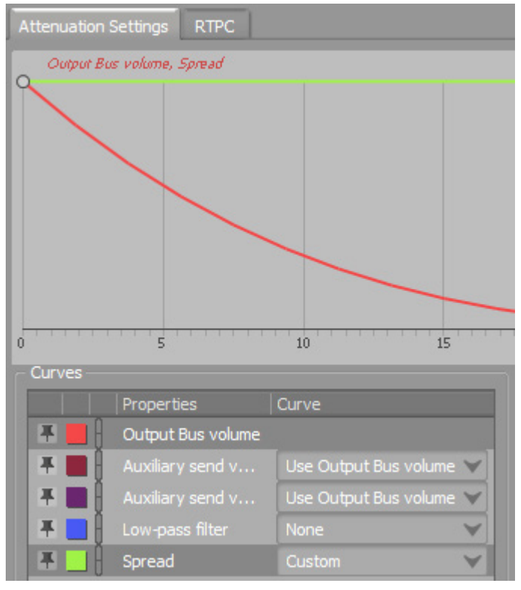
On reporte donc les valeurs de notre tableau dans des RTPCs (real time parameter control) qui seront contrôlés, dans le jeu, par la position de la caméra par rapport aux sources.

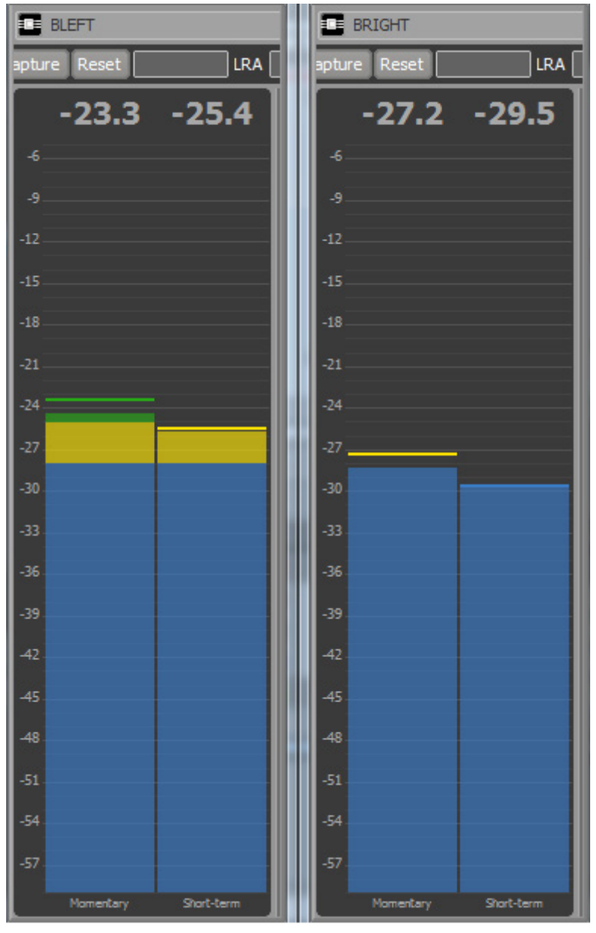


# NOTES ON BLINDNESS

De plus, nous ne voulons pas que le **volume** entre les deux signaux gauche et droite soit trop important pour ne pas biaiser le rendu. On admet une différence allant jusqu'à **3dB** mesuré au Vu Meter (RMS) et Loudness Meter (momentary et Short-term), et on modifie le **spread** à 100% dans le paramètre d'atténuation afin que l'effet panoramique n'ait aucun effet.

L'atténuation du volume entre les deux oreilles est un paramètre qu'il serait intéressant d'étudier plus en profondeur. On parle ici de l'ILD (Interaurale Level Difference).




Enfin, c'est le **delay** qui offre la majeure partie de ce rendu si réaliste. Le plug-in de delay de Wwise descend jusque 1ms de delay et permet d'être contrôlé en temps réel via le paramètre Wet/Dry avec un RTPC.

Pour terminer, je dois créer deux fois la même source 3D. Une pour l'oreille gauche et une pour l'oreille droite. A la première j'applique les filtres "HRTF_Elevation" "HRTF_Arrière", "HRTF_Left" et "ITD_Left" et à la deuxième "HRTF_Elevation", "HRTF_Arrière", "HRTF_Right" et "ITD_Right".

J'envoie la source «Left» dans un bus pané complètement à gauche, et inversement pour la source «Right».

Cela est assez contraignant surtout si on doit jongler entre plusieurs sons dans une boîte random !

Pour conclure sur ce système de spatialisation binaural fait maison, nous avons donc quatre filtres 3-Band pour les azimuths (un pour l'oreille gauche et un pour l'oreille droite, l'un étant approximativement le miroir de l'autre), l'élévation et l'arrière, et un delay. Ces quatre effets sont contrôlés par trois RTPCs qui représentent les trois axes d'écoute : Gauche/Droite - Avant/Arrière - Haut/Bas.

# NOTES ON BLINDNESS

### LE PROTOTYPE

Le premier objectif de la production de Notes On Blindness est de créer un prototype simple où le joueur doit trouver son chemin. Nous créons donc trois lieux familiers (une cuisine, un salon et une rue). Le joueur devra se retrouver, partant de la cuisine, vers le salon puis vers l'extérieur de la maison. Ce prototype permettra de discerner quels pourront être les possibilités et les limites du gameplay de l'expérience que l'on veut apporter.

#### Technologies testées

Avant de se lancer dans le prototype, nous devons d'abord tester différentes technologies permettant de reproduire la spatialisation binaurale. Nous en avons sélectionné quelques unes dont mon système maison.


**3Dception** est un système de synthèse binaurale développé par Two Big Ears. C'est un plug-in autonome compatible avec Unity pour le moment. Le rendu est très propre sur les trois axes, mais il souffre de quelques soucis de clics lors du calcul de la position de la source quand on tourne la tête sur certains sons, notamment ceux avec une amplitude assez forte et constante (comme une sirène de police). L'effet Doppler intégré fonctionne plutôt bien mais souffre des mêmes problèmes. Même si la sirène de police est un cas assez extrême, il faudra faire attention aux sources que l'on veut utiliser dans l'expérience.


**Astound 3D RTI** d'AstoundSound est un plug-in integré au moteur Wwise. Il a l'avantage d'être, justement, integré à un excellent moteur son. Le rendu binaural est plutôt bon sur les axes des azimuths mais complètement inexistant au niveau de l'élévation. De plus, il m'a été possible de le tester plus en détail sur Ascent et j'ai eu beaucoup de petits soucis inexplicables et très énervant au niveau de son intégration qui ne me donne pas envie de réés-sayer.


**Auro3D** développé par Auro Technologies est très prometteur sur papier. Mais il n'est disponible qu'avec Wwise 2014 qui ne sort que dans une semaine au moment ou j'écris ces lignes.
L'avantage d'Auro3D et 3D RTI sur Wwise 2014 par rapport à mon système est qu'Audiokinetic a facilité leur utilisation avec un nouveau système : le «Mixer Plug-in Framework». Avec ce système, on applique le plug-in binaural directement sur le bus de sortie, et on n'envoie qu'une seule source dedans. C'est Wwise qui se charge de l'instanciation des effets gauche/droite.

**Mon système maison** est fonctionnel, le rendu n'est pas parfait notamment à cause de la limitation à quatre slots d'effets par sources, mais le rendu n'en reste pas moins intéressant, sur les trois axes. Et il fonctionne sur Wwise aussi.

# NOTES ON BLINDNESS

**Assets sonores**
Nous avons déjà créé une scene Unity avec 3Dception, et j'ai créé une quinzaine d'assets, environ cinq par salles.

**LA SUITE**

Malheureusement, je ne peux pas en dire plus sur ce que sera le projet. Au moment où j'écris ces lignes, il est à peine commencé. Il s'agira à priori d'une expérience où le joueur sera confronté à des espaces assez simples et devra avancer à travers celui-ci avec pour seuls repères des sources 3D. Mon rôle sera de créer les assets sonores du jeu, les ambiances, les sources 3D, et de m'assurer de la bonne intégration de la bande son dans le projet.

# CONCLUSION

Au travers des deux projets PureData j'ai pu me rendre compte à quel point j'aime la conception de systèmes sonores, et au delà, la programmation, qui est au cœur de toute application. Pour moi, la programmation est totalement complémentaire avec la création de concepts audio, et je suis content qu'elle ne me bloque pas. En apprenant les bases du codage en autodidacte avant d'entrer à l'ENJMIN, j'imaginais déjà plusieurs concepts basés sur le son. Cela m'attire particulièrement et je pense vraiment pouvoir m'épanouir pleinement dans une équipe en tant qu'intégrateur sonore ou designer de concepts audios.

Concernant Notes on Blindness, je marche en terrain presque conquis. Je suis bien conscient des problématiques liées au système de synthèse binaural et à la production d'assets sur tablette. Le projet m'interesse d'autant plus qu'il réunit mon dossier d'admission à l'Enjmin, mon stage de Master 1 et mon projet de Master 2 !

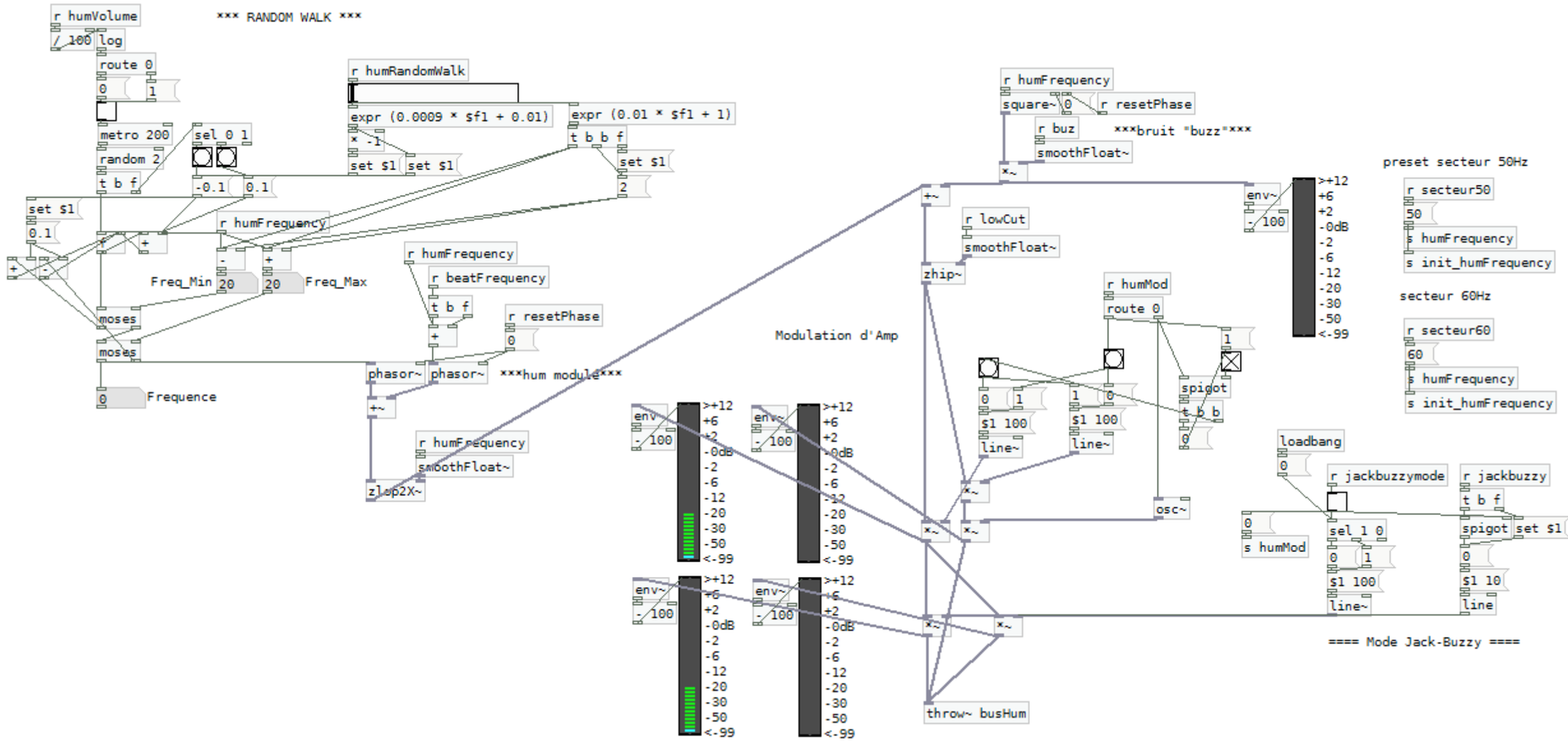
J'ai eu quelques difficultés à rentrer dans mon premier projet, en effet, j'étais un peu fâché avec PureData. Bien que je l'ai utilisé quelques temps pendant ma licence, j'ai dû presque tout reprendre de zéro. Or il a été délicat de travailler presque directement sur le projet en cours, déjà très fourni. Heureusement, mon collègue Victorien était toujours à l'écoute pour m'aider à avancer, à retrouver mes marques et finalement à progresser. De plus, les points projets effectués avec Amaury nous apportaient toujours de nouvelles idées à exploiter pour les améliorer. Ils me donnaient constamment de nouveaux défis à relever dans PureData, me poussant toujours plus loin dans mon apprentissage du logiciel.

Lors de mon stage de Master 1 chez Mando Productions, j'ai travaillé sur des jeux tablettes, malheuresement très classiques (mais propres au studio), où je ne faisais que des assets en ayant très peu de pouvoir sur leur intégration et sur l'itération qu'elle pouvait engendrer. Chez AudioGaming, j'ai utilisé la tablette pour tout autre chose. J'ai conçu des systèmes sonores dont j'ai proposé les contrôles via le device. J'ai alors été confronté à plus de responsabilités que la simple qualité des assets à fournir comme la gestion du bon fonctionnement de l'application ou la programmation de solutions pour rendre les contrôles les plus ergonomiques possibles.

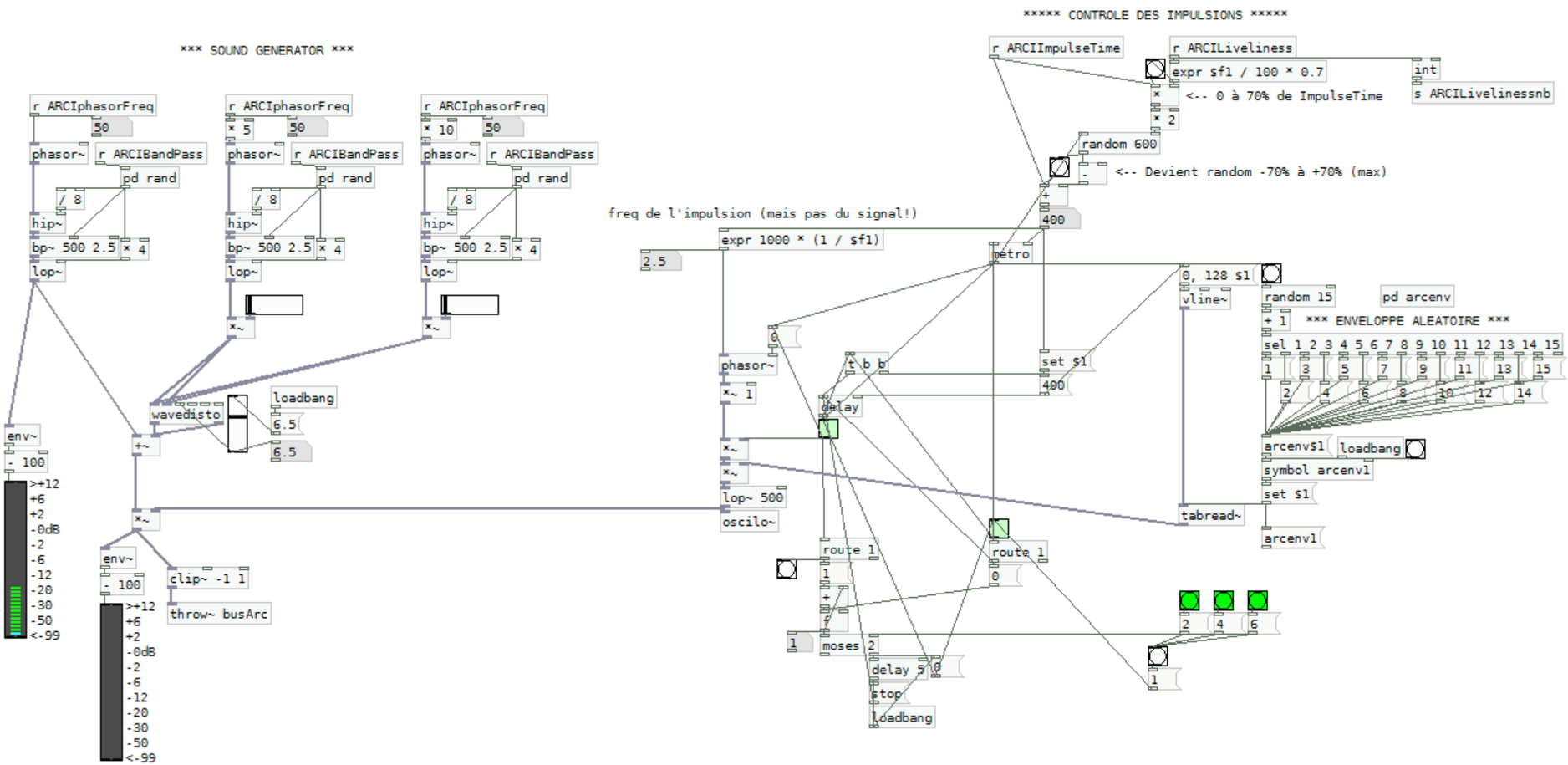
La suite de mon stage sur le projet Notes on Blindness marque une des plus belles fins d'études que je pouvais imaginer. Le projet rassemble tout ce que j'aime dans le sound design, à savoir le design d'espaces et d'environnements sonores 3D avec un système de spatialisation qui m'interesse particulièrement, de l'intégration sonore, et peut-être même quelques défis à relever concernant la mise en place de systèmes audio plus poussés, et qui plus est peut-être avec Wwise. Enfin, je vais peut-être même faire un peu de game design !

En conclusion, je suis très satisfait du stage que m'a offert Amaury. Outre ses nombreux apports et enrichissements, il m'a conforté dans la direction que prend mon profil et dans ce que j'aimerais faire dans la conception sonore.

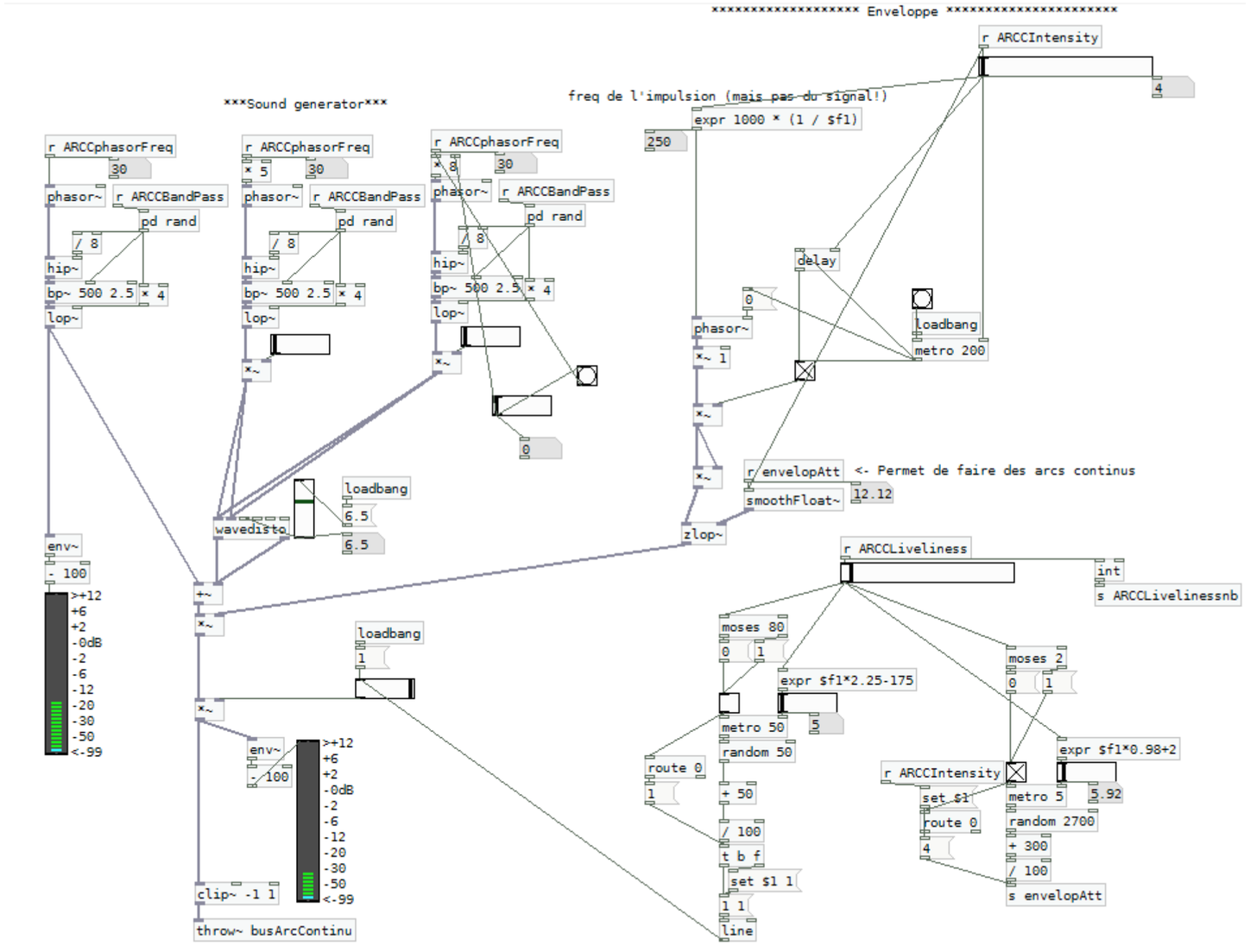
# ANNEXES



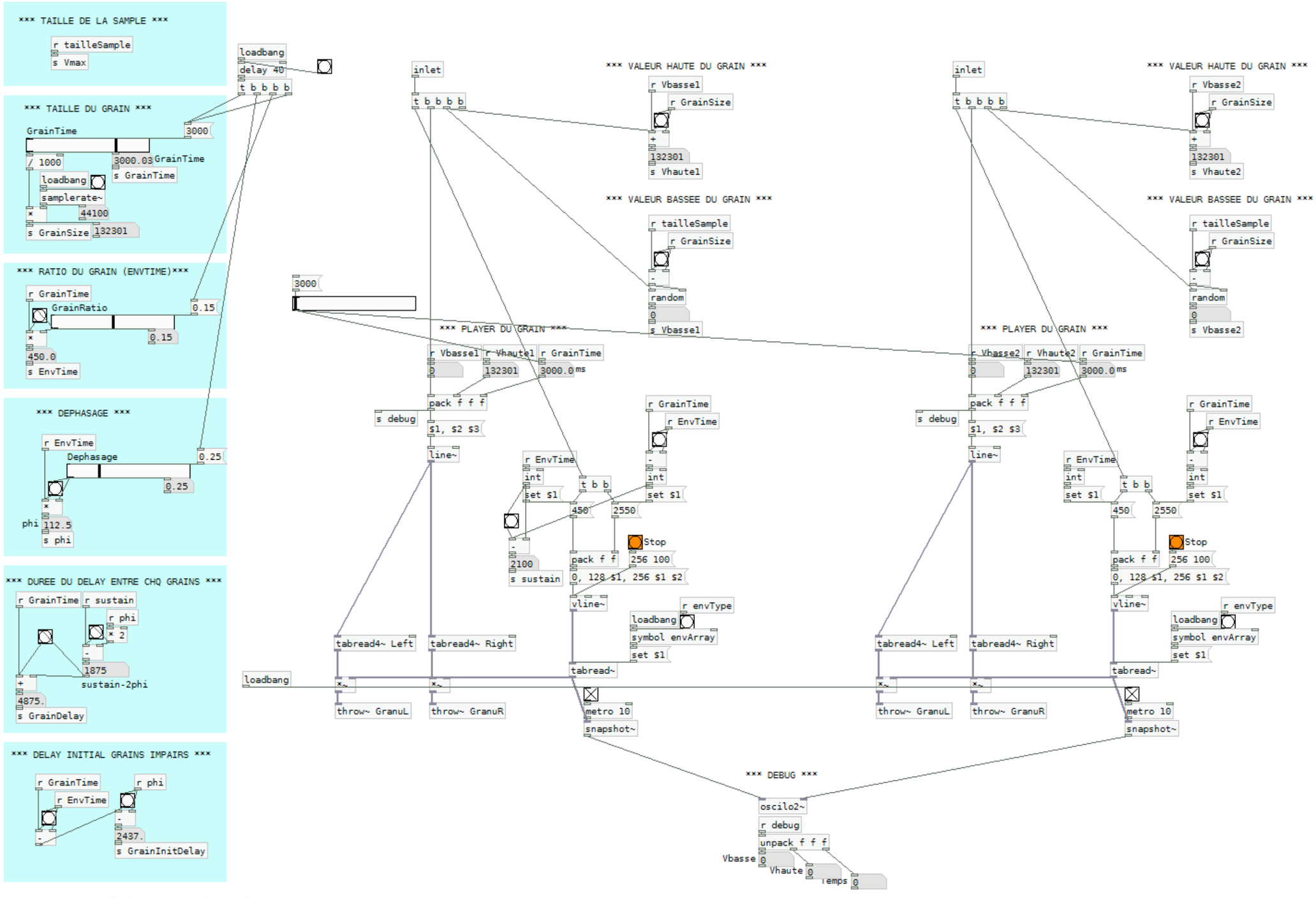

*ANNEXE 1 - AudioElec : Patch du module Hum*

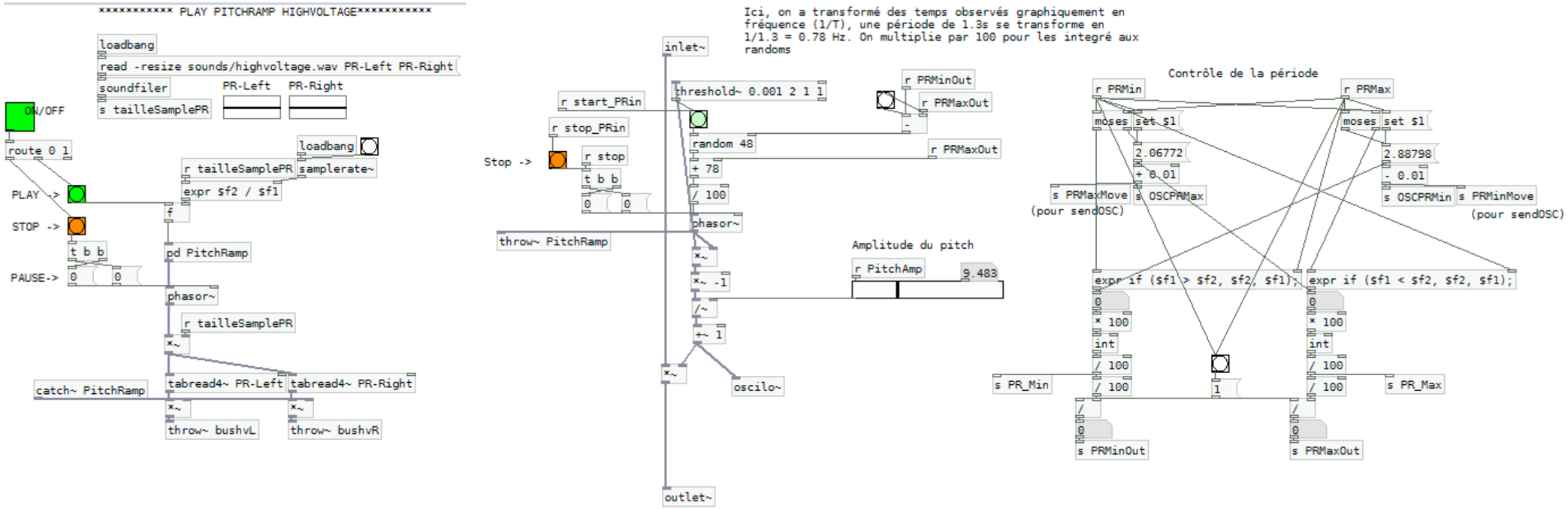



*ANNEXE 2 - AudioElec : Patch du module Arc Impulse*




*ANNEXE 3 - AudioElec : Patch du module Arc Continu*


*** TAILLE DE LA SAMPLE ***
r tailleSample
s Vmax
*** TAILLE DU GRAIN ***
GrainTime
3000
/ 1000
3000.03 GrainTime
s GrainTime
loadbang
samplerate~
x
44100
s GrainSize
132301
*** RATIO DU GRAIN (ENVTIME)***
r GrainTime
GrainRatio
0.15
x
0.15
450.0
s EnvTime
*** DEPHASAGE ***
r EnvTime
Dephasage
0.25
0.25
x
phi
112.5
s phi
*** DUREE DU DELAY ENTRE CHQ GRAINS ***
r GrainTime
r sustain
r phi
x 2
-
1875
sustain-2phi
+
4875.
s GrainDelay
*** DELAY INITIAL GRAINS IMPAIRS ***
r GrainTime
r EnvTime
r phi
-
-
2437.
s GrainInitDelay
loadbang
delay 40
t b b b b
3000
inlet
t b b b b
*** VALEUR HAUTE DU GRAIN ***
r Vbassel
r GrainSize
+
132301
s Vhautel
*** VALEUR BASSEE DU GRAIN ***
r tailleSample
r GrainSize
-
random
0
s Vbassel
*** PLAYER DU GRAIN ***
r Vbassel
r Vhautel
r GrainTime
0
132301
3000.0 ms
pack f f f
s debug
$1, $2 $3
line~
r EnvTime
int
set $1
t b b
450
2550
-
2100
s sustain
Stop
pack f f
256 100
0, 128 $1, 256 $1 $2
vline~
r envType
loadbang
symbol envArray
set $1
tabread~
tabread4~ Left
tabread4~ Right
x~
x~
throw~ GranuL
throw~ GranuR
metro 10
snapshot~
inlet
t b b b b
*** VALEUR HAUTE DU GRAIN ***
r Vbasse2
r GrainSize
+
132301
s Vhaute2
*** VALEUR BASSEE DU GRAIN ***
r tailleSample
r GrainSize
-
random
0
s Vbasse2
*** PLAYER DU GRAIN ***
r Vbasse2
r Vhaute2
r GrainTime
0
132301
3000.0 ms
pack f f f
s debug
$1, $2 $3
line~
r EnvTime
int
set $1
t b b
450
2550
-
int
set $1
Stop
pack f f
256 100
0, 128 $1, 256 $1 $2
vline~
r envType
loadbang
symbol envArray
set $1
tabread~
tabread4~ Left
tabread4~ Right
x~
x~
throw~ GranuL
throw~ GranuR
metro 10
snapshot~
*** DEBUG ***
oscilo2~
r debug
unpack f f f
Vbasse
0
Vhaute
0
Temps
0




*ANNEXE 5 - AudioElec : Patch du module HighVoltage*