PURE DATA & CREATION D'APPLICATIONS POUR SMARTPHONES

(IOS ET ANDROID)

STEREOLUX 2011

Avant Propos :

Ce projet vise à introduire des outils émergents pour la création d'applications mobiles. Je tiens à remercier l'ensemble de l'équipe de Stereolux qui s'est impliqué dans le projet, qui a permis la rédaction de ce document et l'organisation des ateliers.

Ce projet n'aurait pas été possible sans l'ensemble de la communauté Pure Data, de nombreux exemples sont tirés de travaux des membres de l'équipe de développement, et des membres de la communauté. Les erreurs qui peuvent subsister sont les miennes.

N'hésitez surtout pas à me contacter pour des erratums en tout genre (orthographe, coquilles, ajouts, clarifications …), de même si vous reconnaissez votre code et que celui-ci a été mal attribué. Je compte en effet compléter cette version de la documentation.

Vous pourrez retrouver la version web de ce document (accompagné de vidéos démonstratives) à l'adresse suivante :

http://www.stereolux.org/net-art/berenger-recoules/applications-mobiles-de-creation-numerique-car

1-ARTS NUMERIQUES, POURQUOI FAVORISER LE DEVELOPPEMENT ET L'UTILISATION D'OUTILS LOGICIEL ET MATERIEL LIBRES ("OPEN SOURCE" ET "OPEN HARDWARE") ?

Les logiciels libres ont fait leur apparition au début des années 80 avec la création par l'informaticien Richard Stallman, de la première license libre Gnu. Stallman s'opposait à l'appropriation, par les compagnies, du code source des programmes informatiques. Cacher le code source et en rendre illégal la modification lui paraît immoral et inconsistant avec la nature même du code source informatique qui peut théoriquement être partagé sans frais (contrairement à des objets matériels, tels que du papier journal ou un vélo). Dans le manifeste GNU, il écrit:

« Extraire de l'argent des utilisateurs d'un programme en restreignant leur utilisation du programme est destructif parce que, au bout du compte, cela réduit la quantité de richesse que l'humanité tire du programme. {...} Puisque je n'aime pas la situation qui résulte d'une rétention générale de l'information, il me revient de considérer comme immoral d'agir ainsi. »

Contrairement aux logiciels propriétaires, les logiciels libres donnent la liberté aux utilisateurs d'exécuter le programme, d'en étudier le fonctionnement, et d'en distribuer des copies.

Dans le milieu artistique, les logiciels libres ont pris une ampleur considérable depuis les années 90, à commencer par le logiciel Pure Data. Outre ce dernier, plusieurs logiciels ont fait leur apparition dans les dernières années et sont utilisés par des artistes professionnels: Processing, Arduino, Toonloop, Gephex, Freej, etc. Ces logiciels sont utilisés par les artistes car ils sont à la fois gratuits, performants et faciles d'utilisation. Parce qu'ils sont ouverts, ils facilitent leur appropriation par les artistes et favorisent la recherche et la création de nouvelles formes d'art. De plus en plus d'artistes intègrent dans leur pratique artistique le développement de logiciels et de matériaux libres: ils créent leurs propres outils plutôt que de travailler avec des outils déjà existants.

C'est dans cet esprit que va se construire un projet de développement d'applications mobiles de création numériques. L'idée et donc de s'appuyer sur des outils libres pour promouvoir leur utilisation, ainsi que de développer leur documentation.

2-PURE DATA ET SES DIFFERENTES FORMES

Pure Data est un logiciel libre et gratuit et multiplateforme (Linux, Windows, OS, Android, iOS...) de programmation. Ses champs d'application vont de la création multimédia interactive, à toutes formes de musiques électroniques. Pd est développé et distribué dans plusieurs versions :

- la version **vanilla**, encore développée par Miller Puckette.

- la version **extended**, qui regroupe des contributions de la communauté.

- une version sous forme de librairie, qui permet d'utiliser Pure-Data comme librairie dans plusieurs langages. **Libpd** permet donc d'exploiter les possibilités de pd sur les plateformes mobiles en tant que librairie DSP (Digital Signal Processing) en temps réel, multiplateforme, et aussi extrêmement compacte.

Hormis un intérêt certain dans le domaine artistique PD a de plus déjà été utilisés dans certaines applications commerciales. L'application Inception sur l'app store (qui utilise libpd), mais surtout Spore d'EA Games. EA games a developpé sa propre librairie propriétaire à partir du code source de pd, ils ont ensuite travaillé avec Brian Eno pour la construction d'un moteur audio adaptatif. (la sortie de libpd est largement postérieure à la sortie de Spore).

Le projet présente donc de nombreux avantages pour quiconque souhaitant se lancer dans la création numérique :

- un projet open source gratuit et cross-plateform avec une communauté installée très active et sympathique.

- traitement du son en temps réel, et notamment synthèse sonore en temps réel, ce qui permet une meilleure adéquation entre les actions et le contenu audio délivré. Ce qui devrait amener à une meilleure expérience utilisateur globale.

- construction de moteur audio sur mesures, et donc exactement adaptés aux besoins, sans options superflues.

- pas de dépendances : à partir du moment ou on utilise un compileur en C, on obtient une librairie légère qui ne dépendra de rien d'autre.

3-LE PROJET POUR STEREOLUX

Sur une période de six mois nous allons essayer de découvrir pas à pas l'outil de création pure-data à travers des exemples didactiques de mini-applications pour smartphones (android et iOS), dans l'esprit DIY (DoItYourself).

Nous allons aborder différents thèmes allant de la synthèse sonore au développement d'interfaces utilisateurs ; le coeur du projet se situera autour de technologies libres, du partage de ressources à travers la communauté Pure-Data et les différents projets qu'elle a pu porter ces dernières années.

Dans un premier temps, des articles seront publiés tous les mois par support internet, afin de présenter des petites applications pour smartphones. Le code source, ainsi que des commentaires et des ressources pour aller plus loin accompagneront ces publications. L'objectif n'est pas ici d'apprendre à utiliser Pure Data, beaucoup de ressources et tutoriaux sont déjà disponibles sur internet, nous n'allons pas les ré-écrire, mais nous essaierons de rendre notre travail le plus compréhensible possible à travers des commentaires et des ressources qui seront proposées tout au long de ces travaux. Vous n'avez besoin d'aucune connaissance pour utiliser les programmes developpés, si ce n'est savoir utiliser un ordinateur et un smartphone. Cependant si vous souhaitez aller plus loin et que vous ne connaissez pas Pure Data, je vous invite à commencer la lecture du floss manual français consacré à Pure Data, si vous ne connaissez pas Pure Data, les parties introduction et prise en main vous permettront de saisir les concepts de bases qui seront utiles à l'étude des exemples ci-après, sinon vous risquez d'être quelque peu désabusés http://fr.flossmanuals.net/Puredata/Introduction

Dans un second temps, nous vous proposerons un rendez-vous mensuel dans les locaux de Stereolux, sous forme d'un workshop gratuit et ouvert à tous. Il vous suffira de venir avec votre smartphone pour pouvoir essayer les applications, et commencer à en développer. Vos contributions pourront être si vous le souhaitez proposées au téléchargement sur le site de Stereolux.

Enfin nous vous proposerons une installation artistique, utilisant ces différentes technologies en contexte.

PRESENTATION DES OUTILS

Voici le site officiel de Pure Data, vous y trouverez de nombreuses informations, ainsi que des liens pour le téléchargement des différentes versions : http://puredata.info/

Ci-après le lien vers la page de téléchargement :

http://puredata.info/community/projects/software/by-category/distribution

Les projets qui vont nous intéresser sont les suivants :

- **Pure Data** « **vanilla** », sera la version de Pure Data à installer sur son ordinateur pour pouvoir construire des programmes avant des les importer sur son téléphone.

- **ScenePlayer** (android) ou **RJDJ** (iOS à travers l'app store) : il va nous permettre de charger un programme Pure Data (ou « patch » comme on l'appelle) et d'utiliser les informations provenant du Touchscreen, ou des capteurs de mouvement du téléphone, puis d'enregistrer la sortie audio. Il ne nous permettra par contre pas de construire des interfaces utilisateur.

http://autobuild.puredata.info/pdlab/libpd/

- **PdDroidPArty** (android seulement) : va permettre de faire fonctionner un patch tel qu'il a été developpé sous Pure Data avec une émulation des éléments d'interface disponibles dans pd. Ce projet en est encore qu'à ses débuts et ne supporte pas encore l'intégralité des fonctionnalités de Pure Data.

http://droidparty.net/

- **pd-webkit** (android) permet d'associer un patch Pure Data à un fichier HTML, on developpe alors le son dans Pure Data et l'interface utilisateur en HTML.

http://code.google.com/p/pd-webkit-droid/downloads/list

Sous Android, quelque soit le projet le principe est toujours le même :

1- on télécharge une application sous forme d'un .apk et on l'installe.

2- on crée des patchs avec pd.

3- on copie ces patchs sur la carte SD du smartphone. (soit par cable usb soit avec un système de partage de la carte SD en wifi type File Expert : https://market.android.com/details?id=xcxin.filexpert)

4- l'application va ajouter automatiquement les nouveaux patchs.

Sous Android, vous pouvez également télécharger les fichiers depuis le site sur votre téléphone et utiliser Linda manager pour extraire les zip, déplacer les fichiers ou insaller les .apk. https://market.android.com/details?id=com.lindaandny.lindamanager

Sous iOS, nous vous conseillons de télécharger l'application RJDJ depuis l'appstore, puis d'utiliser le système de serveur local pour copier vos patchs sur le téléphone : http://blog.rjdj.me/pages/pd-utilities

Si votre iphone est « jailbreaké » vous pouvez alors utiliser un logiciels tel que iphone explorer(http://www.macroplant.com/iphoneexplorer/) pour copier directement le dossier mascene.rj dans le dossier de l'application RJDJ qui se trouve dans /Root Directory/var/mobile/Applications/ le dernier dossier est chiffré et change à chaque fois que l'on y accède, il faut donc tester les dossiers un par un... Le dossier recherché contient un dossier appelé « RJDJ.app ». Une fois ce dossier identifié il faut aller dans /Documents/rjdj_scenes/ et vous pouvez finalement copier le dossier ici.

Note : nous allons essayer de travailler sur les « plateformes mobiles », cela peut se comprendre principalement par Android et un peu iOS. Ces deux plateformes présentent des philosophies radicalement différentes : iOS est propriétaire (au niveau logiciel et matériel) stable et performant il manque cependant un peu d'ouverture pour faire tout ce que l'on souhaiterait sans entrainer des difficultés ; android quand à lui est plus ouvert mais du à la multiplication des modèles matériels il et parfois difficile de pouvoir s'assurer que ces applications pourront fonctionner de manière optimale sur tous les modèles. De plus les performances techniques de ce type d'appareil évoluent très rapidement, certains modèles plus anciens risquent de manquer de puissance de calcul; cependant n'hésitez pas à venir nous voire lors des workshops pour éclaircir tout cela.

4-EPISODE 1 : L'UTILISATION DE SAMPLES

Dans ce premier épisode nous allons nous familiariser avec Pure Data, son interface et son fonctionnement. Nous allons apprendre à lire un fichier audio en synchronisation avec un autre à l'aide d'objets déjà crées, ainsi qu'à contrôler son volume.Nous allons aussi découvrir les possibilités d'interaction qui s'offrent à nous.

4.1- FONCTIONNEMENT (VIDEOS + SNAPSHOTS + FICHIERS A TELECHARGER)

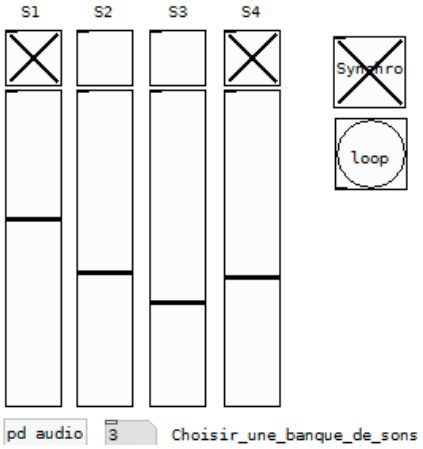
L'application permet de mixer 3/4 boucles audio ensemble, 3 pour la version rjdj et 4 pour la version « patch » Pure Data pour DroidParty. Pour la version rjdj (iOS et Android) on a trois boucles dont le volume suit l'orientation du téléphone, pour la version droidparty on a une interface avec un interrupteur et un slider permettant de régler le volume pour chaque boucle. Il y a aussi la possibilité d'avoir plusieurs packs de sons et de passer de l'un à l'autre (en touchant l'écran tactile pour la version rjdj et à l'aide d'une boîte de sélection pour DroidParty). Les boucles ont été crées avec des patchs pure data que vous pourrez retrouver dans la section ressources. L'application a été developpé dans un soucis de fonctionner sur des téléphonnes d'entrée de gamme ou relativement modestes en termes de performances (cela ne sera peut-être pas toujours le cas), si vous avez un téléphone plus puissant, un écran plus grand etc. vous pourrez facilement modifier le code pour gérer beaucoup plus de boucles.

Vous pouvez donc choisir de télécharger les patchs et de les jouer sur votre ordinateur en téléchargeant Pure Data (lien), en utilisant l'application RJDJ (pour les possesseurs d'iphone et de smartphone android), en utilisant l'application DroidParty (android seulement).

http://rjdj.me/sharescene/sample-mixer/

4.2- GRANDES LIGNES DU CODE (SNAPSHOTS)

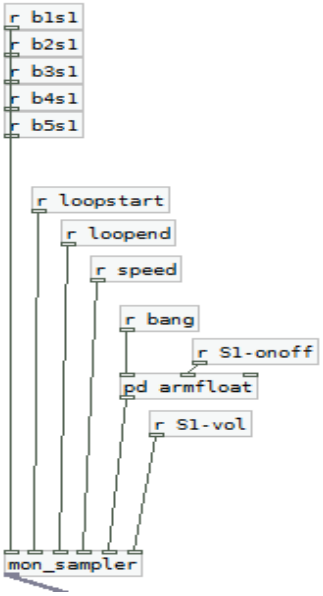
Pour plus de détails, référez vous aux commentaires directement inscrits dans le patch. Ici l'objectif n'est pas de comprendre le détail mais d'avoir une idée préalable de ce qui a été implémenté.



Fenêtre principale de l'application

Le principe général du patch DroidParty est assez simple. L'idée et de pouvoir lancer une série de boucles que l'on aura préenregistré et qu'elles se jouent de façon synchronisée. C'est-à-dire que si l'on veut lancer la boucle suivante, il faudra attendre que celle qui soit jouée revienne au début (c'est le principe qu'utilise Abbelton Live pour la lecture de ses clips).

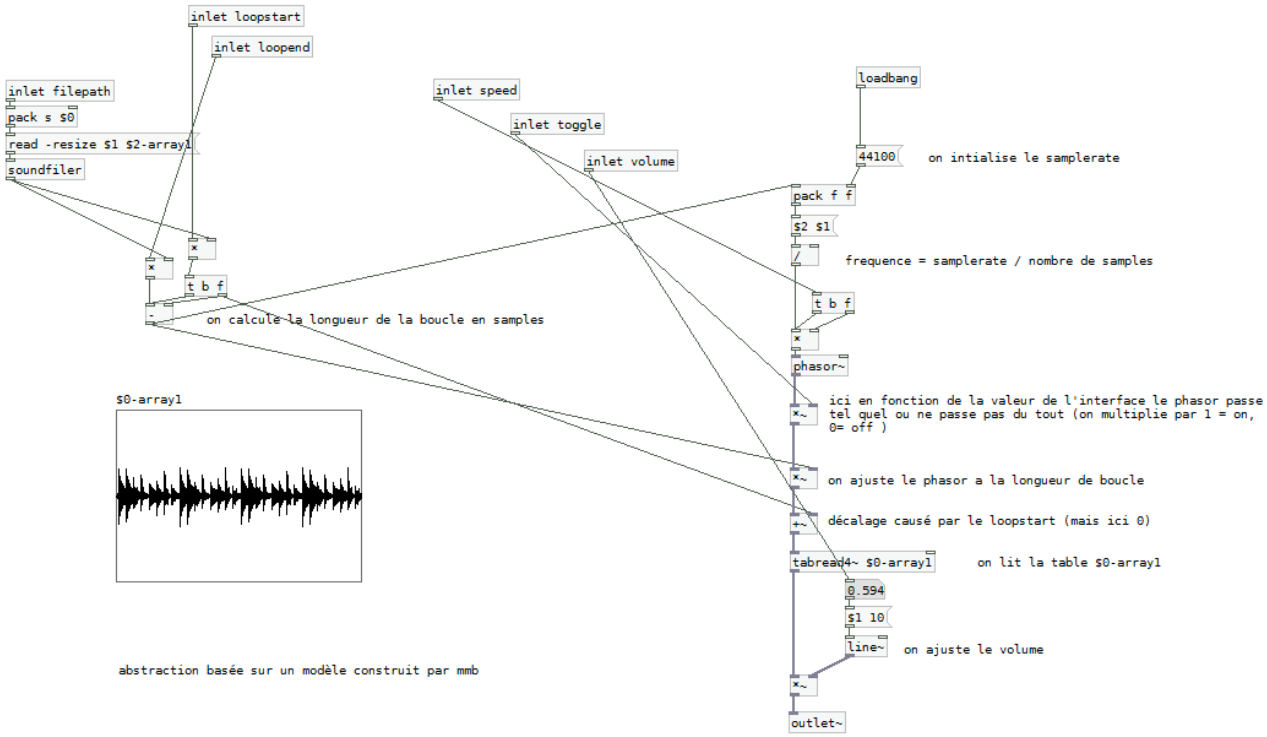
En résumé on va charger un banque de sons, c'est-à-dire 3 ou 4 sons dans des tables à l'aide de l'objet [soundfiler], c'est la méthode la plus souple (et non la plus simple) pour lire des sons. Pour lire 4 sons on va en réalité utiliser 5 tables car on va utiliser une table témoin qui va permettre d'envoyer un signal à la fin de chaque boucle pour la synchronisation. Les autres tables quant à elles vont devoir d'abord être « armées » dans l'interface une fois armées elles attendent le signal de synchronisation et commencent à se lire quand ce signal est reçu. Si l'interrupteur est désactivé immédiatement elle ne se lira qu'une fois sinon elle bouclera.



L'ensemble des messages reçus par chaque sampler.

Chaque lecteur de boucle est « encapsulé » dans ce que l'on appelle une abstraction (c'est en fait un autre patch Pure Data qui se situe dans le même dossier) on peut alors créer un objet portant le nom de

cette abstraction. Cela est très utile car cela permet d'utiliser la notation $0- devant les noms de messages (c'est une notion assez avancée mais très utile). A l'ouverture du patch tous les préfixes « $0 » seront remplacés par un identifiant unique à l'abstraction. Par exemple un message du type « $0-bang » deviendra « 1088-bang » dans une abstraction et « 1158-bang » dans celle d'à côté. Cela permet donc de copier/coller des bouts de codes sans qu'il y ait d'interférences entre les messages qu'ils utilisent.



Le contenu de l'abstraction [mon_sampler]

Cette abstraction permet de charger donc de charger une boucle en mémoire pour la jouer plus tard. Nous n'avons pas nécessairement d'en savoir plus, si ce n'est que l'entrée de gauche reçoit le chemin vers le fichier sonore à lire sur le disque dur, les trois entrées suivantes sont fixées par défaut au chargement du patch, la quatrième reçoit un bang qui va enclencher la lecture de la boucle (le bang aura au préalablement été synchronisé avec le « master-bang ») et la cinquième entrée reçoit le volume compris entre 0 et 1.

4.3- RESSOURCES (FICHIERS A TELECHARGER + LIENS)

4.3.1- LES LIENS :

- http://www.freesound.org/ : une banque de sons collaborative, qui pourra vous permettre de trouver des pour remplacer ceux par défaut.

- http://www.musicradar.com/news/tech/sampleradar-13154-free-sample-downloads-217833/1 : des boucles musicales en téléchargement gratuit.

- http://radio.rumblesan.com/ : une radio générative sur le net, jouant des patchs Pure Data, normalement les patchs sont disponibles au téléchargement, mais le site est actuellement en travaux.

4.3.2-LES FICHIERS :

- interface-help : contient des patchs Pure Data qui vous aideront à comprendre la récupération des informations de interface de votre smartphone (accéléromètre, touchscreen ...).

- Christchurch.pd : est une composition Pure Data sous la forme d'un patch, vous n'avez qu'à l'ouvrir et écouter, puis farfouiller à l'intérieur pour comprendre comment ça marche. Cette composition a été créée par Andy Farnell qui a réalisé beaucoup d'exemple et de tutoriaux, vous pouvez trouver le lien vers son site dans la section Ressources

- Mengine-Ambiant-Std-Ed.zip : est aussi un patch musical. Celui-ci permet de générer de la musique en continu à travers un ensemble de contraintes. A chaque temps, les notes qui vont être jouée sont soumises à des probabilités. Il est possible de choisir des gammes, des structures harmniques, des instruments et de fixer toutes les probabilités une à une. Il est aussi possible de créer des presets de styles musicaux. N'hésitez pas à me contacter directement si vous avez des questions.

Note : pour s'exécuter convenablement ces deux patchs musicaux nécessiteront l'installation de Pd-extended.

5-EPISODE 2 : LES EFFETS

Cette fois nous allons apprendre à utiliser l'entrée microphone de notre téléphone et nous allons la filtrer avec des effets la modifier en temps réel.

5.1- FONCTIONNEMENT (VIDEOS + SNAPSHOTS + FICHIERS A TELECHARGER)

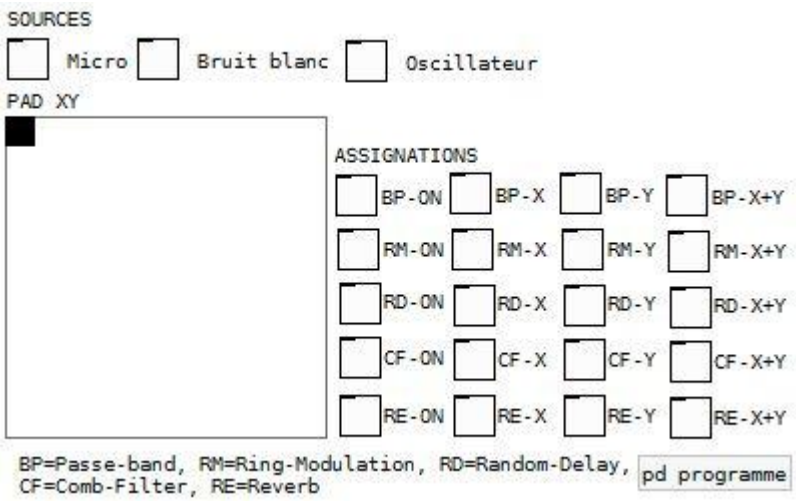
Les applications proposée ce mois ci vont vous permettre d'utiliser des effets audio avec votre téléphone mobile. Pour chacune des deux applications proposées vous pourrez disposez de trois sources sonores (l'entrée micro de votre smartphone, un générateur de bruit blanc et un oscillateur) les signaux vont alors être transformés à travers une chaîne d'effets avant d'être ré-émis par les haut-parleurs. N'héistez surtout pas à installer Pure Data sur votre ordinateur pour étudier la façon dont ces patchs ont été crées, vous trouverez à l'intérieur des commentaires complémentaires).

Le patch pour le Scenplayer d'android ou l'application RJDJ d'iOS va diriger le signal de votre choix à travers trois effets : un filtre passe-bande, suivi d'un « shaper » qui va permettre de modifier l'enveloppe du son, puis enfin un réseau de délais variables. Les mouvements du téléphone vont être captés par l'accéléromètre et vont permettre de contrôler les paramètres suivants : sur l'axe x la résonance su filtre passe-bande et en y la fréquence de coupure de ce même filtre en z cela sera la profondeur de l'enveloppe qui sera appliquée par le « shaper ». Il faut appuyer sur l'écran avec deux doigts pour changer la source ; lorsque la source sélectionnée et l'oscillateur on peut changer sa fréquence en promenant son doigt sur l'écran tactile.

Note : l'application pour iphone et android sont légèrement différentes ce mois ci. Je me suis rendu compte d'un problème avec l'objet [expr] sous iphone. Cet objet permet de rentrer des formules mathématiques, ce qui est parfois assez utile. Du coup une nouvelle version est disponible pour iphone n'utilisant pas cet objet. Le fonctionnement est similaire et les effets relativments proches de la version android. Le patch spécial iphone est également utilisable sur Android sans problèmes.

http://rjdj.me/sharescene/effectedg/

Le patch pour Droidparty est une sorte de Kaoss pad. Il va permettre de contrôler des effets à l'aide d'une surface de contrôle. Il suffira alors de déplacer son doigt sur cette surface pour modifier les paramètres. Une matrice d'assignation permet d'activer ou non différent types d'effets et de choisir quel donnée est envoyée à l'effet (la position en x, la position en y ou la moyenne des deux). Les effets de la chaîne sont dan l'ordre : un filtre passe-bande, un modulateur en anneau, un délais pseudo-aléatoire, un filtre en peigne et une reverb.



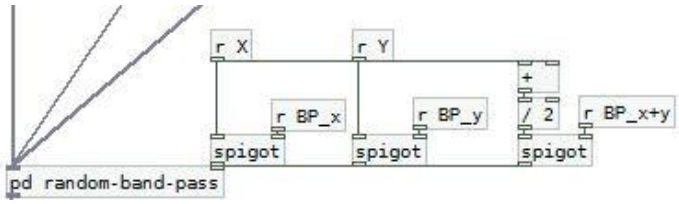
Fenêtre principale de XY_pad_droid

Comme d'habitude le dossier finissant par « .rj » peut être utilisé sur android et iOS, l'autre dossier contient le patch Pure Data pour ordinateur et android.

5.2- GRANDES LIGNES DU CODE (SNAPSHOTS)

Un effet est une unité de traitement ou de transformation audio, de façon simple c'est un sous-patch ou une abstraction, qui va utiliser des entrées et des sorties audio ([inlet~] et [outlet~]). Entre l'entrée et la sortie on va effectuer des traitements à l'aide de divers objets comme [*~] (multiplication de deux signaux), le couple [delwrite~] / [delread~] (inscription d'une entrée audio dans un buffer pour être relus plus tard).Vous pourrez en apprendre plus dans la section ressources, avec des pages wikipédia, des articles, et des patchs à télécharger.

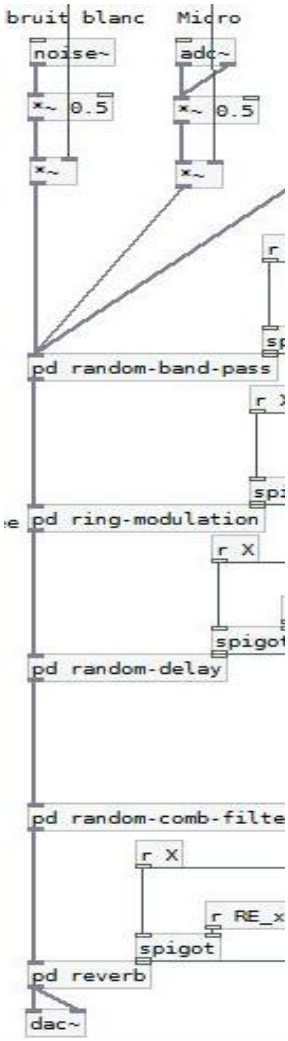
5.2.1 -COMMENT EMPECHER DES DONEES DE PASSER (AUDIO ET MESSAGES)



L'objet [spigot] permet de bloquer ou de laisser passer des données. A l'entrée de gauche on connecte le flux de données à l'entrée de droite on connecte un message (ici un objet [toggle] qui envoie un message 0 si il n'est pas activé, et 1 s'il est coché). Si spigot reçoit un 0 il bloque, s'il reçoit un 1 les messages passent.

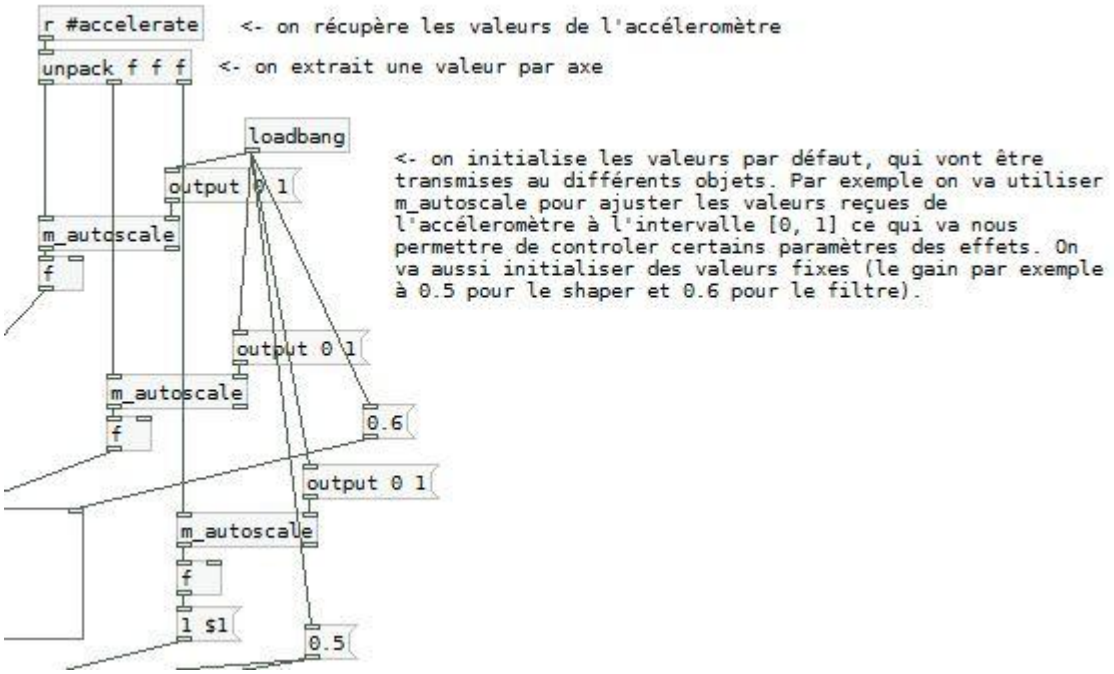
Il existe un objet [spigot~] qui permet de faire la même chose pour les signaux audio (les cables plus épais). Je préfère cependant utiliser une autre technique qui consiste à multiplier le signal par un O ou un 1 en utilisant l'objet [*~] (mutliplication audio). Cette objet permet aussi de réaliser un contrôle de volume linéaire, dont la valeur minimale sera 0 et la maximale 1. C'est la technique qui a été utilisé dans l'épisode précédant pour réaliser le mini-mixer de boucles.

5.2.2- UNE CHAINE D'EFFETS



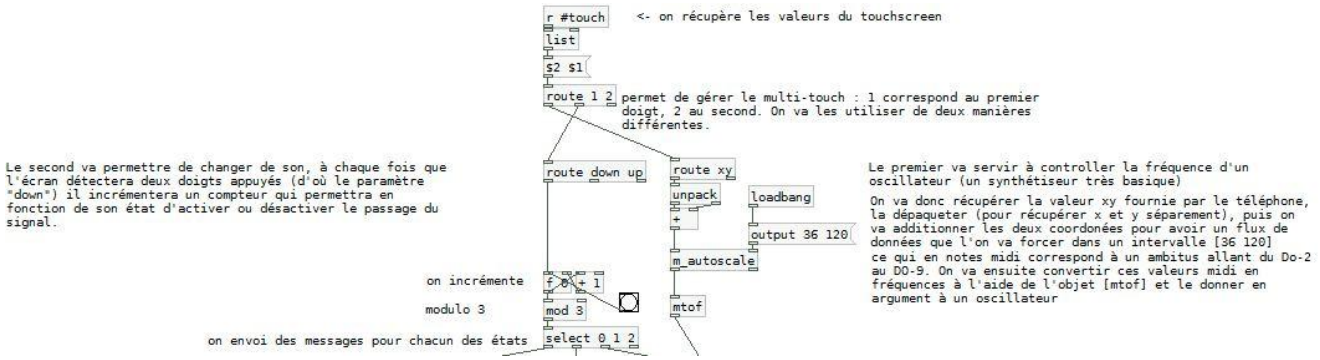
Voici la chaîne d'effet complète de l'application XY-pad-droid, on reçoit tout en haut les signaux, qui ensuite passent dans chacun des effets, l'entrée d'un effet étant la sortie d'un autre, jusqu'à la sortie audio [dac~].

5.2.3- LES VALEURS ENVOYES PAR LE TELEPHONE [R# TOUCH], [R# ACCELERATE] ET [M_AUTOSCALE]



[m_autoscale] permet d'échelonner des valeurs entrantes sur un intervalle de sortie donné automatiquement. C'est-à-dire qu'en entrée on va recevoir un flux de données (des nombres) quelconque qui va être automatique rééchelonné à l'intervalle [0,1] pour contrôler des effets pour cela on va utiliser un message [output 0 1(, on peut ajouter un troisième argument permettant de choisir le type d'interpolation (0=sensible aux petits mouvements, 1=linéaire, 2=sensible aux plus grands mouvements)

L'accéléromètre



Le touch screen

Dans PdDroidParty on utilise l'abstraction [touch] et des interupteurs [toggle] pour avoir des interfaces. On récupère leurs valeurs en définissant un nom d'envoi (send). Click droit-> propriétés , pour un gui

classique de pure data. Et pour les abstractions de Chris via la création de l'objet par exemple [touch 50 50 hello] va créer une surface de 50px X 50px dont les informations de déplacement seront envoyées au receveur hello donc en créant un objet [r hello] suivi d'un [unpack f f] pour séparer les valeurs en x et en y.

5.3- RESSOURCES

Une collection d'objets proposant un bon nombre d'effets :

http://puredata.hurleur.com/sujet-1982-diy2-effects-sample-players-synths-sound-synthesis

Une explication détaillée de comment réaliser une texture à l'aide de délais, se référer à l'index 3.4.2.7 Texture :

http://www.pd-tutorial.com/

La collection d'abstraction du laptop orchestra de Virginia Tech :

http://l2ork.music.vt.edu/main/?page_id=25

Quelques articles théoriques de wikipédia sur quelques filtres et effets utilisés dans les patchs :

http://fr.wikipedia.org/wiki/Filtre_passe-bande

http://fr.wikipedia.org/wiki/Ring_Modulator

http://fr.wikipedia.org/wiki/Delay_(effet)

http://fr.wikipedia.org/wiki/Filtre_en_peigne

http://fr.wikipedia.org/wiki/R%C3%A9verb%C3%A9ration_(acoustique)

Si vous êtes plus aventureux, vous pouvez consulter cet article détaillant la construction d'une reverb sous Pure Data, présenté cet été à la Convetion Internationale Pure Data organisée à l'université Bauhaus de Weimar (Allemagne) :

http://www.uni-weimar.de/medien/wiki/PDCON:Conference/Reverb_Design

Une Scène supplémentaire à télcharger LoopIT:

Cette scène permet d'enregistrer des boucles synchronisées à partir de l'entrée micro de votre smartphone. Il suffit d'appuyer sur l'écran pour lancer l'enregistrement, laisser appuyé le temps de l'enregistrement et relâcher pour le stopper. Si la boucle vous convient enregistrez en une autre, sinon secouez 3 fois le téléphone pour l'effacer. A chaque point de bouclage, il apparait un petit rectangle rouge sur l'écran pendant quelques millisecondes.

LoopIt.rj :http://rjdj.me/sharescene/loopit/

Un Kaoss pad pour WebPD :

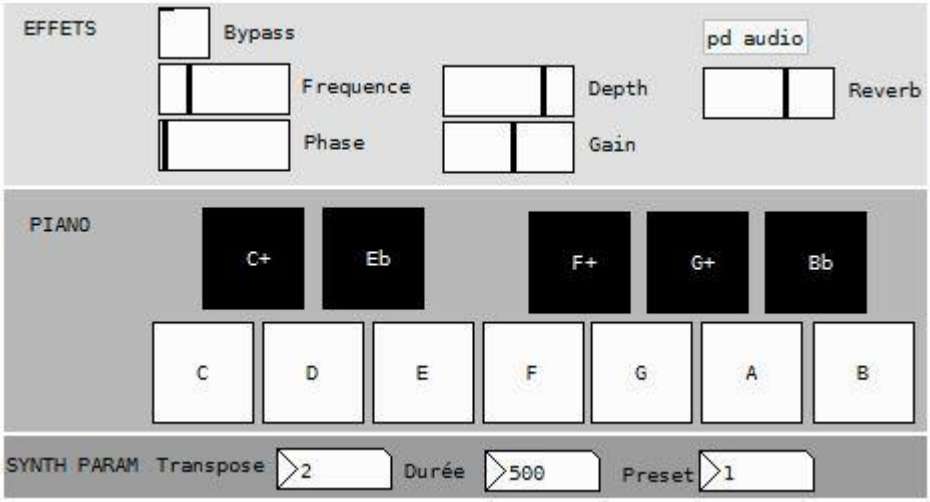
Ce patch est un équivalent au patch PdDroidParty, mais avec une interface en HTML.

6-EPISODE 3 : LES SYNTHETISEURS

La synthèse sonore … comment sont crées les sons synthétiques ? pour ce troisième épisode nous allons créer un patch permettant de créer une mélodie jouable sur plusieurs gammes et avec plusieurs sons pour RJDJ et un mini-piano avec effets pour PdDroidParty

6.1- FONCTIONNEMENT (VIDEOS + SNAPSHOTS + FICHIERS A TELECHARGER)

Pour cet épisode nous allons nous intéresser aux synthétiseurs. Comme d'habitude deux patchs sont proposés : le premier pour l'application RJDJ ou ScenePlayer est un séquencer sans interface, il vous suffit d'appuyer sur l'écran pour que petit à petit une mélodie se mette en place, vous pouvez changer le son de synthèse en appuyant à deux doigts, et secouer pour changer les gammes qui seront jouées, des effets répondent aussi à l'accéleromètre, les synthétiseurs sont issus de patchs d'Andy Farnell ; le second est un petit piano avec quelques effets incorporés et différents sons disponibles, le synthétiseur est lui une adaptation d'un patch réalisé par Tom Erbe.



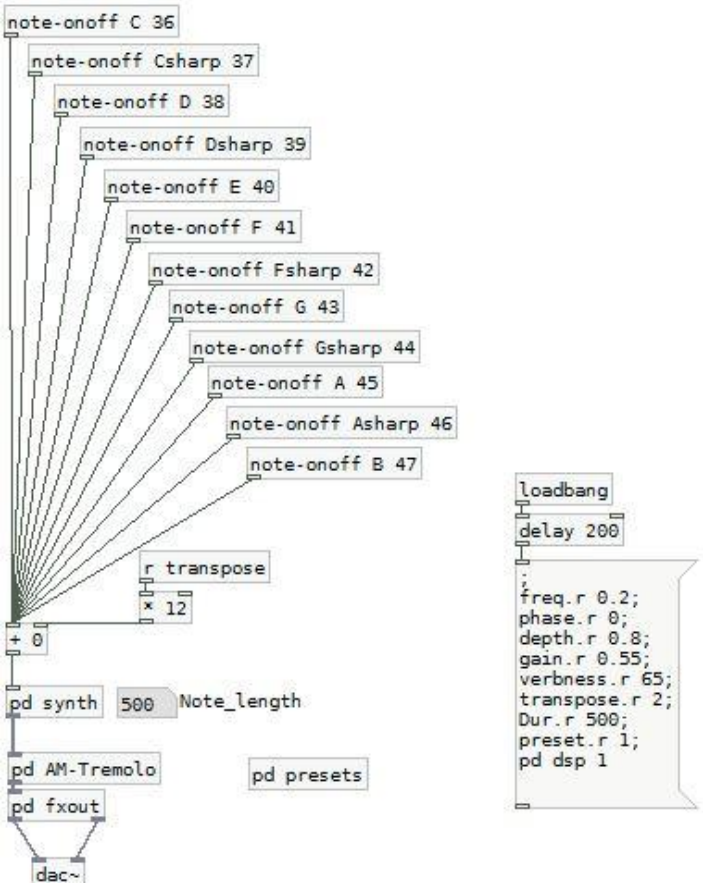
Le patch PdDroidParty

Vous trouverez des liens sur Andy Farnell et Tom Erbe dans la section ressources.

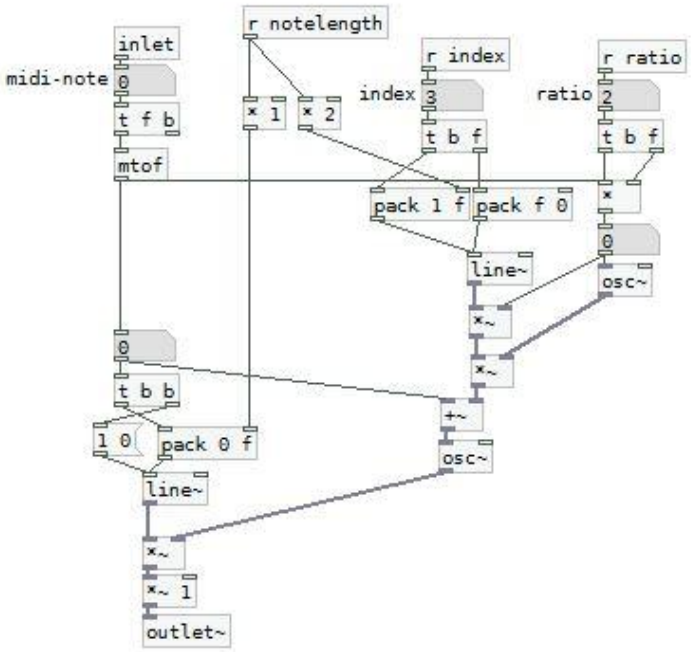
http://rjdj.me/sharescene/tapititap-3/

6.2- GRANDES LIGNES DU CODE

Hormis la nouveauté des la synthèse sonore plusieurs points importants sont abordés dans ces patchs. Référez vous aux ressources mais aussi aux commentaires inscrits à l'intérieur même des patchs.

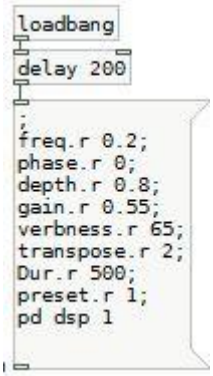


L'intérieur du patch PdDroidParty



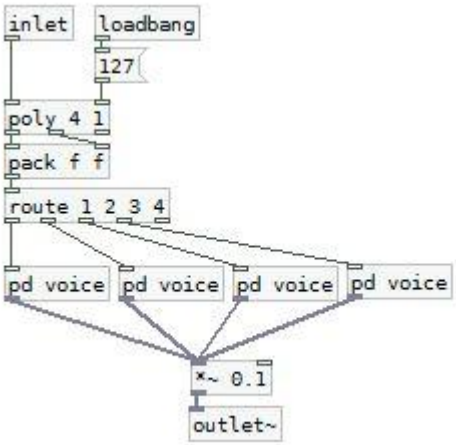
Le moteur de synthèse FM du patch : on peut créer facilement des presets en envoyant des messages de type [ratio 3(, ou [index 4(.

6.2.1- INITIALISATION DES PARAMÈTRES



[loadbang] va envoyer un bang au chargement du patch on peut donc en profiter pour envoyer un liste de paramètres qui vont être reçus à tous les endroits où c'est nécessaire dans le programme.Ici on envoi des messages d'initialisation aux effets, mais on fixe aussi une valeur par défaut de transposition de durée et un numéro de preset à charger

6.2.2- POLYPHONIE



La gestion de la polyphonie peut se faire de différentes façons. Ici on va utiliser l'objet [poly] qui accepte deux arguments ; le premier argument va renseigner le nombre de voix qu'il va prendre en compte , le second va déterminer le « voice stealing », c'est-à-dire que si toutes les voix sont utilisées, il est autoriser à en réquisitionner une. [poly] va accepter un couple (note, vélocité) en entrée (attention si la vélocité n'est pas renseignée [poly] ne passera rien en sortie), et va retourner un triplet (numéro_de_voix, note, vélocité). On va alors encapsuler toutes ces valeurs (en laissant de côté la vélocité puisque le touchscreen n'est pas sensible à différents niveaux de pression) pour pouvoir les diriger vers les différentes voix à l'aide de l'objet [route]. [route] va se séparer du numéro de voix pour ne laisser passer que la note midi.

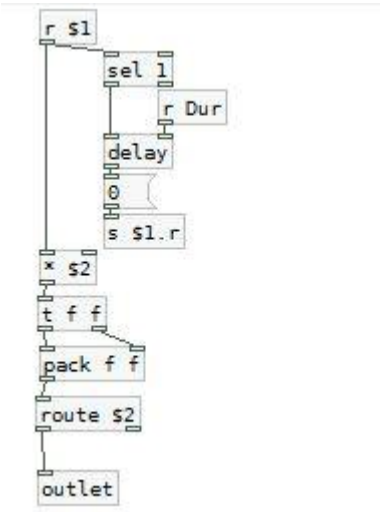
6.2.3- ABSTRACTION ET SOUS-PATCH

Une abstraction est un patch que l'on va enregistrer dans le même dossier que le patch principale, on pourra ensuite l'appeler comme n'importe quel objet de pure data, c'est-à-dire en créant un objet et en

indiquant le nom de cette abstraction. C'est le cas de [note-onoff] dans la première capture d'écran de cette partie. C'est pratique lorsque l'on va utiliser ce code plusieurs fois à différents endroits, cela évite les copier/coller et si on est amené à modifier cette abstraction, les changements seront répercutés partout dans le patch.

Un sous-patch va être un bout de code que l'on va enfermer dans un container. Pour cela il suffit de créer un objet [pd mon_sous_patch] une fenêtre vide s'appelant « mon_sous_patch » va alors s'ouvrir, il n'y a plus qu'à y écrire du code. Cela est pratique notament pour rendre le code plus compact et plus lisible.

6.2.4- UTILISATION DES $0,$1, $2 DANS LES ABSTRACTIONS ET SOUS-PATCHS



Utilisation d'arguments dans l'abstraction [note-onoff]

L'argument $0 dans Pure Data a une place spéciale, il est utilisé pour les abstractions, pour que lorsque l'on envoie un message à l'intérieur d'une abstraction, il soit aussi reçu à l'intérieur de cette abstraction (et non pas envoyé à une autre). On utilise alors $0 en début de chaque message, car au chargement celui là sera remplacé par un nombre aléatoire (non redondant). Voir dans l'épisode 1 la gestion des tables dans l'abstraction mon_sampleur. Si on utilise pas $0 toutes les abstractions recevront le même message (ci-dessus [r Dur] sera reçu de la même façon par toutes les abstractions)

Les symboles $1, $2, $3 etc. eux permettent de spécifier des arguments de création. Dans le cas de [note-onoff], il y a un $1 et un $2. Si je créée un objet [note-onoff C 60] alors dans le code ci-dessus $1 sera remplacé par « C » et $2 par « 60 ». Ici $1 va donc nous permettre de spécifier des noms de messages pour communiquer avec l'interface : on va recevoir le message « C » lorsque l'on appuiera sur l'interrupteur C de l'interface et après un délais d'une durée « Dur » on lui enverra un 0 (pour qu'il se désactive) à travers le message « C.r ».

6.3 RESSOUCES

6.3.1 DES LIENS

La bible du design sonore avec Pure Data, sur le site d'Andy Farnell (ressource incontournable à mon sens) : http://obiwannabe.co.uk/tutorials/html/tutorials_main.html

Le livre associé (si vous pensez à l'acheter mais que vous hésitez, venez me rencontrer lors des workshops, ou au bar de Stereolux, vous pourrez le feuilleter) : http://aspress.co.uk/ds/about_book.php

Modulation de fréquence (voir patchs joints) : http://fr.wikipedia.org/wiki/Modulation_de_fr%C3%A9quence

Modulation d'amplitude (voir patchs joints) : http://fr.wikipedia.org/wiki/Modulation_d%27amplitude

Des patchs de Tom Erbe : http://musicweb.ucsd.edu/~tre/wordpress/

Le lien vers la page de soundhack (projets gratuits de Tom Erbe, en bas à droite se trouvent les liens vers les externals pour Pure Data) : http://soundhack.henfast.com/freeware/

L'ensemble des patchs que j'utilise sous Pure Data, avec des effets, des synthétiseurs et bien plus : http://code.google.com/p/pdlive/

6.3.2 DES FICHIERS A TELECHARGER

Maintenant que l'on sait comment faire un effet, et comment faire un synthétiseur on peut créer un vocodeur. Ces patch sont fournis sans commentaires et comme ils sont.

Talkbox.rj tranforme votre voix en voix robotique (il faut un téléphone de milieu de gamme pour le faire fonctionner sous Android, par exemple l'HTC Legend sera trop juste, par contre il fonctionne sur l'iphone3G du fait des optimisations qui ont pu être mise en œuvre de part la plateforme fermée. Il est intéressant aussi de voir la différence de latence audio entre iOS et Android. Voire note dans la partie de présentation des outils)

http://rjdj.me/sharescene/talkbox-2/

Vocoder.rj un vocoder permettant d'activer différents accords en fonction des orientations du téléphone.

http://rjdj.me/sharescene/vocoder-4/

Des patchs de présentation de la synthése FM et AM ainsi que la technique de waveshapping

Pour les plus téméraires un article très complet sur la synthèse sonore.

7-EPISODE 4 : LES SEQUENCEURS

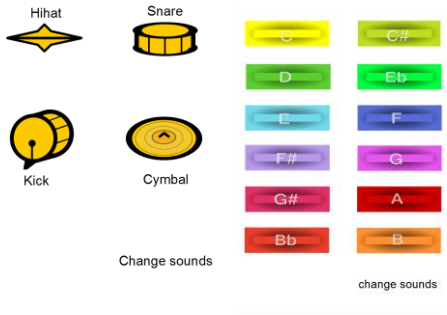
Un des points cruciaux de la musique électronique. Dans ce quatrième épisode marqué par Scopitone nous allons commencer à construire des patchs beaucoup plus évolués, avec des interfaces graphiques plus complexes permettant de créer un réel contenu musical.

Découvrez la composition orginale sms, créée avec le sequenceur Groove_droid présenté ci-dessous et avec un prototype de l'application pour l'épisode 6 ! (c'est l'outil utilisé pour créer toutes les textures)

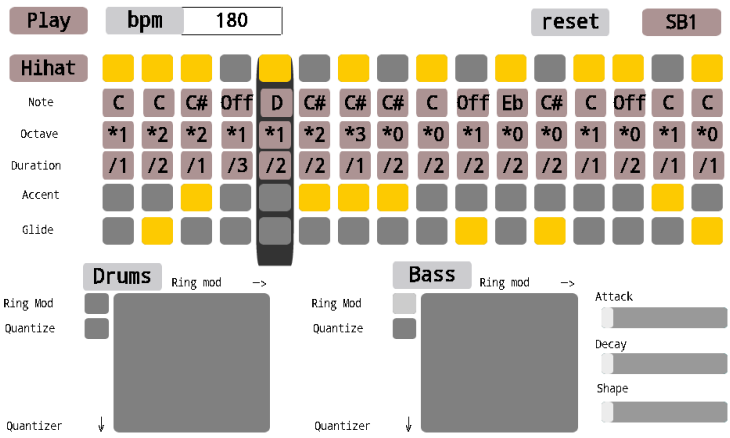
7.1- FONCTIONNEMENT (VIDEOS + SNAPSHOTS + FICHIERS A TELECHARGER)

Aujourd'hui nous allons nous intéresser aux séquenceurs. Un séquencer est outil qui permet de jouer automatiquement et aussi en boucle de la musique. D'une façon générale un séquenceur à un certain nombre de pas et permet de déclencher un certains nombres d'actions par pas en rythme avec un métoronome. Deux applications vous sont aujourd'hui proposées, elle récapitulent un peu tout ce que l'on a découvert jusqu'à maintenant (nous allons utiliser des samples, des synthétiseurs et des effets) pour obtenir de réels petits instruments de musique électronique.

L'application RJDJ pour iphone uniquement, va se décomposer en trois pages cette fois : une première page pour jouer des sons de batterie, une seconde pour des nappes de synthétiseurs et une troisième pour ajouter une ligne de basse ou des mélodies. Sur la dernière page il faut appuyer en haut à droite pour changer les sons. Secouer pour effacer les séquences ! à télécharger ici : http://rjdj.me/sharescene/multi-tap/



L'application PdDroidParty est une boîte à rythme couplé avec une basse à sonorités acid. Un vrai outil permettant de construire un base rythmique pour un morceau électro. Après les récentes mises à jours PdDroidParty permet l'utilisation de fichiers svg pour pouvoir donner une meilleur apparence à son programme. Voyez plutôt :

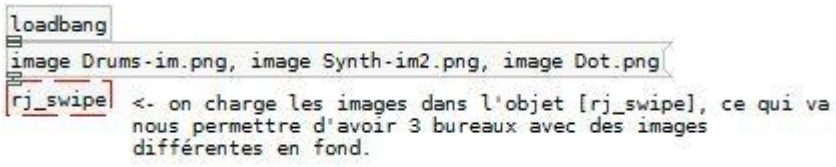


7.2- GRANDES LIGNES DU CODE

MUTLI-TAP.r

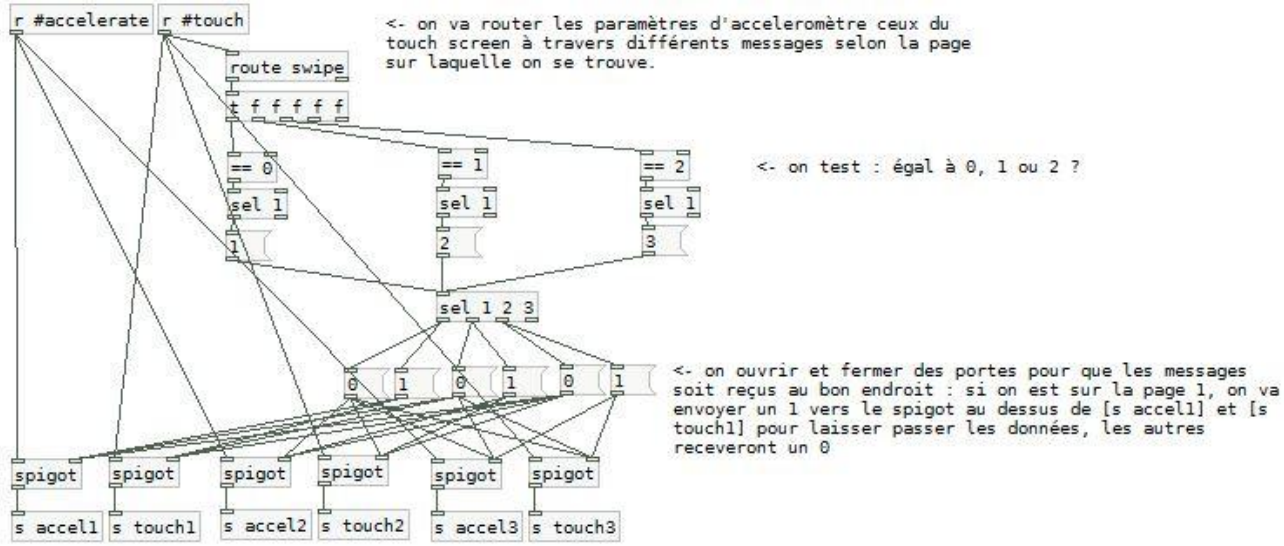
7.2.1- L'EXTERNAL [RJ_SWIPE] (MULTITAP)

Une application pour iphone uniquement car elle utilise un système de pagination spécial. En utilisant l'objet [rj_swipe], on va pouvoir définir plusieurs pages avec chacune leur fonction et chacune leur image de fond.



Utilisation de l'objet [rj_swipe]

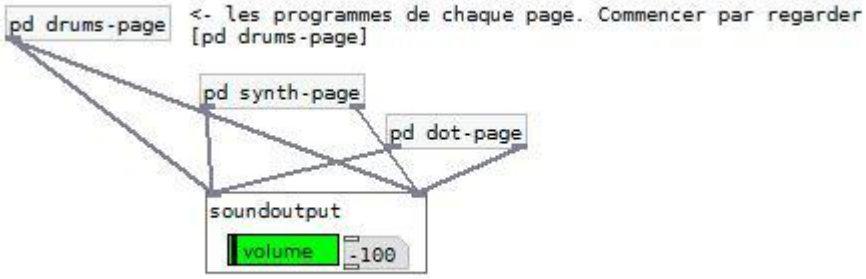
Il nous suffira alors d'établir un système permettant d'orienter les informations venant du touchscreen et de l'accéléromètre dans le programme en fonction de la page sur laquelle on se trouve. Cette méthode est un peu simpliste mais à le mérite de fonctionner.



Réorientation des données vers la page sur laquelle on se trouve.

Les informations sont donc envoyés avec un nom spécial, si on veut récupérer les informations du touchscreen sur le page deux on utilisera l'objet [r touch2].

Chaque page contient un programme à base de séquenceurs. Pour les pages 1 et 2 on va utiliser l'objet [c_taptap] vu à l'épisode précédent. La page 3 va être une adaptation de la scène RJDJ de Chris McCormick CanOfBeats, je vous laisserai donc vous référer à l'originale pour voire comment celle-ci a été adaptée.

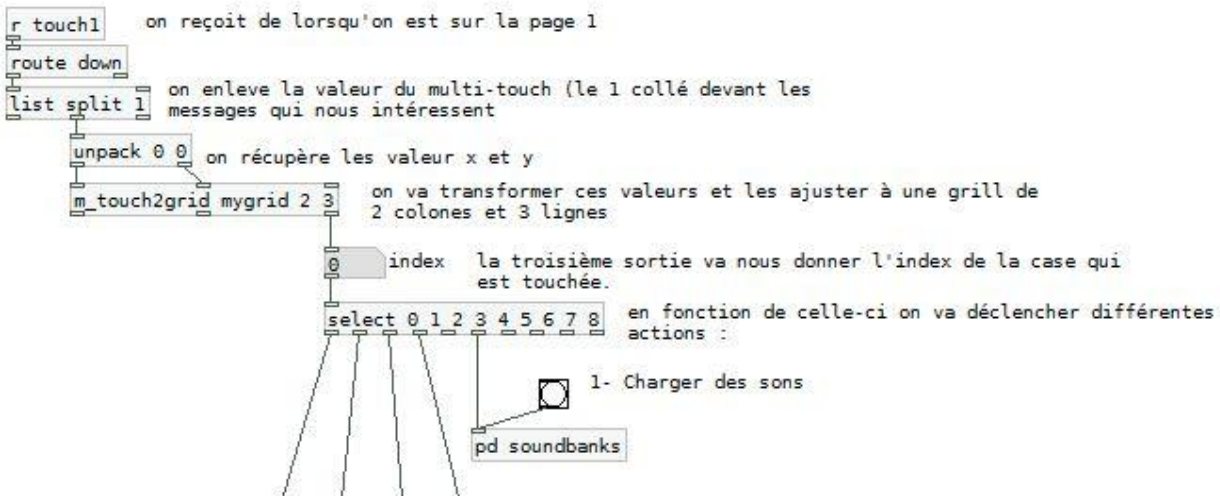


Les programmes de différentes pages sont contenus dans ces trois objets

L'objet [pd synth_page] est une réplique de l'application précédente (épisode 3) on a ajouté la possibilité de secouer pour remettre à zéro les séquenceurs, et d'autres synthétiseurs, mais aussi l'interaction est différente elle utilise [m_touch2grid].

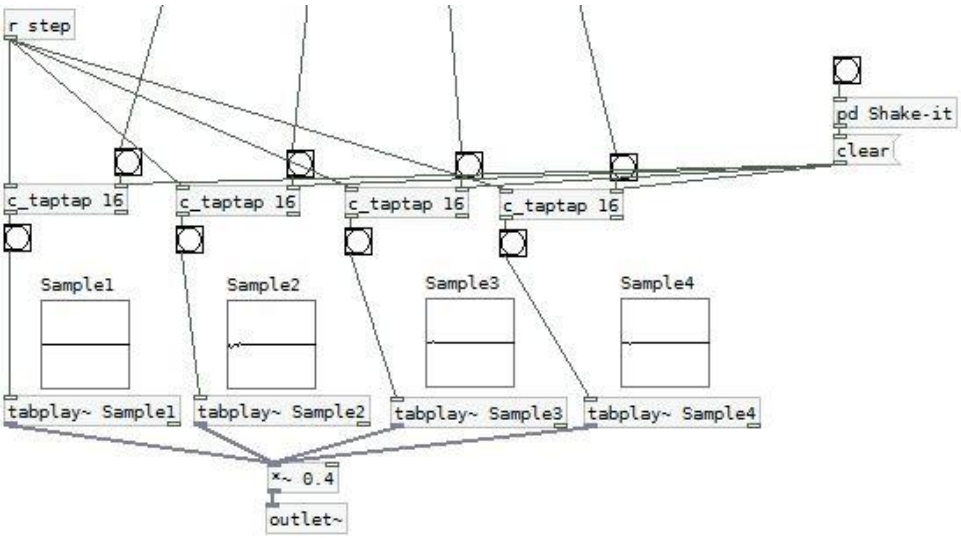
7.2.2- L'ABSTRACTION [M_TOUCH2GRID] (MULTI-TAP)

Mais pour étudier cet objet, nous allons nous intéresser à l'intérieur de [pd drums-page]. Pour récupérer des informations du touchscreen nous allons utiliser l'objet [m_touch2grid] de la librairie RJDJ. Celui-ci permet de réaliser un « mapping » de l'écran vers une grille de résolution plus basse ; c'est-à-dire qu'il va couper l'écran en un certain nombre de lignes et de colones, et qu'il va envoyer le numéro de la zone touchée.



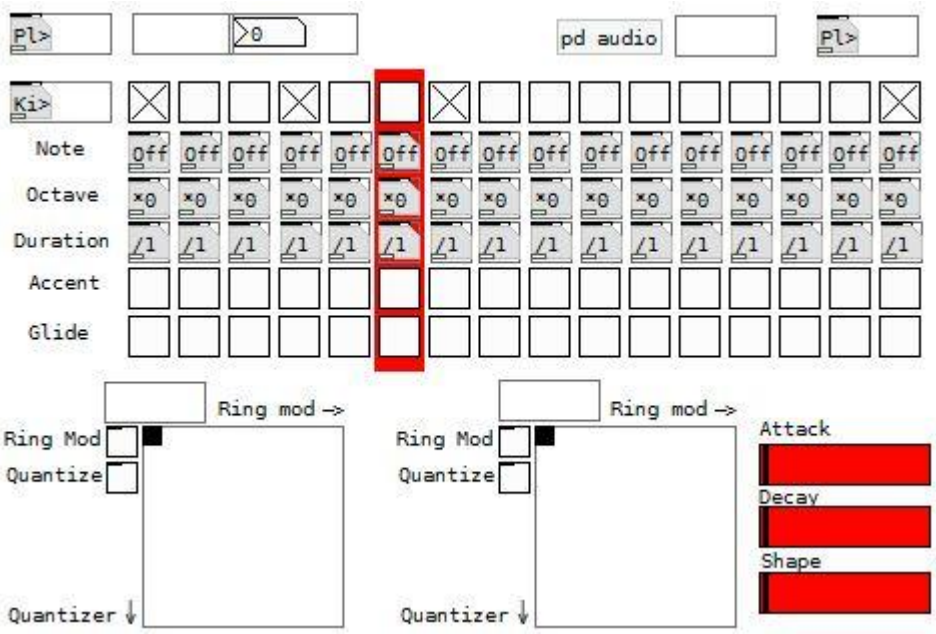
Utilisation de l'objet [m_touch2grid]

Ici on a donc défini six zones avec 2 colonnes et trois lignes ; le zéro est en haut à gauche, le 1 en haut à droite. Les zones 0 1 2 3 vont servir à remplir des séquenceurs et la zone 5 va servir à changer les sons qui sont des samples. Ensuite il suffira alors de construire une image avec des boutons correspondant à ces zones pour construire une interface rudimentaire.



Utilisation de l'objet [c_taptap] pour lire des samples.

GROOVE-DROID



Voici le patch brut tel que vous pourrez le voir sur votre ordinateur. Il est parfaitement fonctionnel mais peu lisible.

Le premier bouton en haut à gauche permet de démarrer le métronome à vitesse donnée (boite de nombre à côté). Tout en haut à droite se trouve un bouton permettant de changer les sons de batteries.

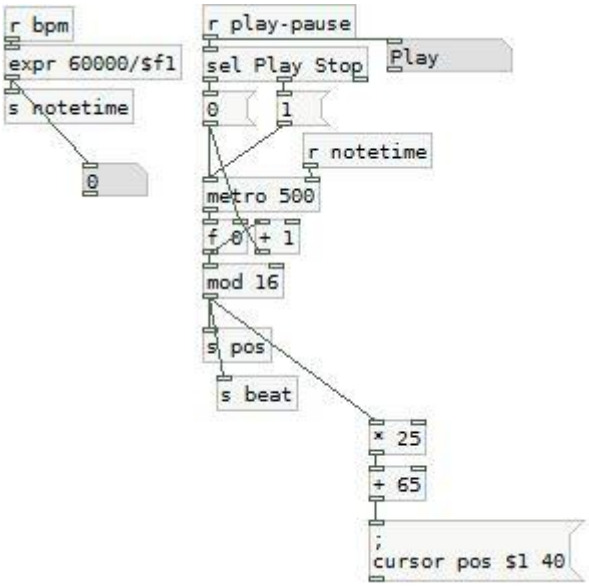
En dessous nous avons le séquenceur de batterie un bouton permettant de changer l'affichage suivant la voix que l'on veut modifier (kick, hihat, snare, cymbale) et donc une ligne d'interrupteurs indiquant si l'on va jouer ou non.

Les 5 lignes du dessous correspondent contrôle du synthétiseur. La première correspond à la note qui va être jouée, la seconde à l'octave dans laquelle elle sera placée (0 grave, 4 aigue), la troisième est une durée donnée en subdivision du tempo principal. La ligne 4 permet d'accentuer une note, et la cinquième de faire un glissando.

En dessous on retouve deux surface tactile pour contrôler des effets sur soit sur la batterie soit sur le synthé. En tout à droite des sliders pour contrôler des paramètres du synthé.

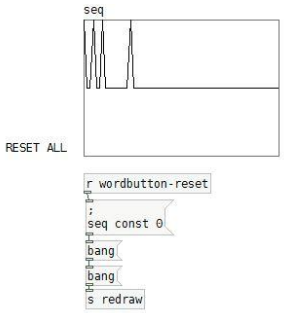
7.2.3- ANATOMIE D'UN SEQUENCEUR DE BATTERIE

Un séquenceur se présente le plus souvent comme un série d'interrupteurs, ici 16, chaque interrupteur contrôle un paramètre de synthèse précis. Ce qu'il faut donc faire pour construire un séquenceur avec Pure Data c'est d'abord avoir une rangée d'interrupteurs ou de nombres, stocker les valeurs de ces objets quelque part (idéalement on utilise l'entrée droite d'un objet [f], l'entrée gauche permettra alors de passer la valeur à la sortie en utilisant un bang) puis d'interroger ces valeurs en rythme avec un métronome. On va donc recevoir du métronome des valeurs allant de 0 à 15 à un pas de un, qui vont entrer dans un objet [select 0 1 2 … 15] chacune de ses sorties étant reliées à l'objet [f] qui contient la valeur voulue. Voici donc le fonctionnement de principe d'un séquenceur très simple.



Voici le compteur utilisé pour cette application, remarquez le message tout en bas qui permet de déplacer dynamiquement un objet dans l'interface.

Ici nous allons compliquer les choses car nous allons stocker les valeurs dans une table qui sera indexée. Ce qui nous permettera à l'aide d'un bouton et d'une seule ligne d'interrupteurs de faire un séquenceur de batterie comportant 4 voix (mais potentiellement beaucoup plus). Il suffira d'appuyer sur un bouton pour changer l'affichage de cette ligne.



La table stockant les valeurs du séquenceur de batterie.

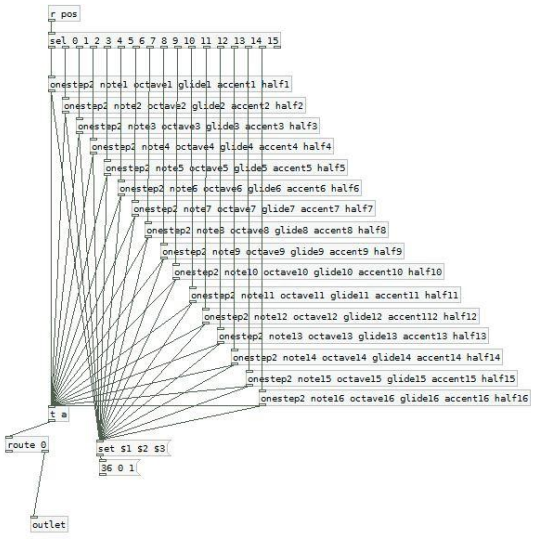
Pour lire la table on utiliser différents objets qui liront de façon synchronisée des indexes différents de la table. Si la valeur est zéro on ne joue pas de sample si la valeur est 1 alors on lit un fichier audio. Cette méthode dépasse un peu le cadre de cette introduction mais méritait d'être mentionnée étant donnée la taille des écrans avec lesquels on travaille. Je vous invite donc à consulter le code pour en savoir plus

7.2.4- ANATOMIE D'UN SEQUENCEUR CONTROLANT UN SYNTHETISEUR.

On va utiliser les même valeurs de synchronisation(le compteur). Mais on va collecter beaucoup plus de paramètres. Une note sera composée d'un pitch (hauteur), d'une véolocité, et d'une durée, valeurs auxquelles on va adjoindre une valeur d'accentuation (0 ou 1) et une valeur de glissando (0 ou 1 : pas de glissando ou glissando sur la durée de la note).

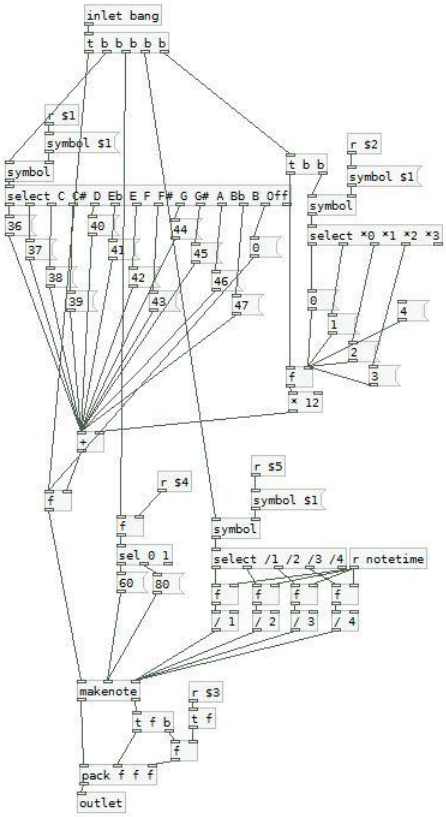
On reçoit donc la valeur pos (envoyé par le compteur) et on interroge une abstraction qui va 'formater' tous les messages provenant de l'interface pour construire un message utilisable pour le synthé.

Pour le séquenceur du synthétiseur, on reste sur le fonctionnement classique à l'aide d'une abstraction on stocke toute les valeurs renseignées dans l'interface dans des floats (=[f]). Au pas correspondant à une abstraction donnée, celle-ci va recevoir un bang et envoyer en sortie toutes les valeurs.



Le séquenceur pour le synthétiseur : il va sortir trois valeurs la note midi, la vélocité de la note et un 1 si il y a glissando un 0 sinon.

Cette abstraction va aussi permettre d'assurer le bon ordre des opérations à l'aide de l'objet [trigger] ou [t], à savoir : lorsque l'on utilise [makenote] il faut renseigner les valeurs de droite à gauche en terminant par la note tout à gauche.



L'abstraction [onstep] qui récupère les données de l'interface pour les formater en données de contrôle du synthétiseur.

Remarquez aussi comment sont récupérées les données provenant des abstractions [taplist] et l'utilisation de symboles. Attention ne pas utiliser le '-' et le '+' dans les symboles ceux-ci seront reconnus comme des séparateurs et tronqueront les messages.

7.2.5- LE SUPPORT SVG DANS DROIDPARTY

Chris le décrit dans sa page anglaise dédiée au projet. Pour résumer, le plus simple est d'utiliser les fichiers .svg présents dans le dossier du programme et de les modifier avec Inkscape (logiciel gratuit supportant le SVG). Le lien entre l'objet et le fichier .svg ce fait par le nom du send attribué à l'objet dans Pure data (clic-droit -> propriétés). Si vous créez un slider vertical en choisissant « sliderX » pour nom de send, il faudra que le fichier correspondant s'appelle Slider-vertical-sliderX.svg. Cela est appliquable pour les abstractions [taplist], [wordbutton], [touch], et pour les toggles, les sliders verticaux ou horizontaux. On peut aussi imposer des illustrations en fond avec background.svg

A l'heure actuelle on peut aussi customiser un objet de type canvas, mais la différenciation par le nom de send ne fonctionne pas, les différents canvas si émulés avec le svg auront tous la même apparence.

7.3- RESSOUCES

7.3.1 DES LIENS

Article wikipédia sur les séquenceurs :

http://fr.wikipedia.org/wiki/S%C3%A9quenceur_musical

Librairies d'abstraction et d'objets musicaux sous Pure-Data :

http://www.lesobjetsvolants.com/music/index.php

Visual Tracker un séquencer géant pour Pure Data :

http://code.google.com/p/vtmodules/wiki/Fast_Start_Video

CanOfBeats de Chris : RJDJ + android market.

RJDJ : http://rjdj.me/music/Chris%20McCormick/CanOfBeats/

Android Market : https://market.android.com/details?id=cx.mccormick.canofbeats

7.3.1 DES PATCH SUPPLEMENTAIRES :

La drum-machine de chris (premier programme à avoir tourné avec le moteur PdDroidParty) :

Simple sequencer (version simplifiée de groove_droid) :

Shake-drums.rj : Secouer pour jouer des rythmes de batterie.

http://rjdj.me/sharescene/shake-drums/

Les différentes pages de mutli-tap.rj en versions compatibles android :

http://rjdj.me/sharescene/dot-tap/

http://rjdj.me/sharescene/drum-tap/

http://rjdj.me/sharescene/synth-tap/

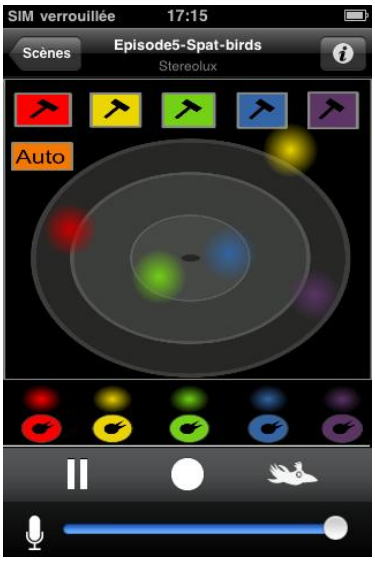
8-EPISODE 5 : SPATIALISATION, ET AUTO-GENERATION

L'idée est de construire un dispositif sonore interactif permettant de gérer une forme de localisation des points dans l'espace sonore. Mais aussi d'avoir des leviers d'interactivité et d'auto génération.

8.1- FONCTIONNEMENT

8.1.1-SPAT-BIRDS.RJ

On va donc construire une cage à oiseau, mais pas n'importe quels oiseaux des oiseaux électroniques de l'espèce FM. On va donc avoir la possibilité d'avoir jusqu'à 5 oiseaux de couleurs différentes. On va pouvoir les déplacer dans l'espace individuellement, ensuite on pourra leur taper individuellement sur la tête avec un marteau (mais gentiment) pour qu'il s'envolent et se mettent à chanter. On peut aussi secouer la cage pour qu'ils s'envolent tous d'un coup. On peut finalement utiliser le mode Auto pour les laisser vaquer à leurs occupations.

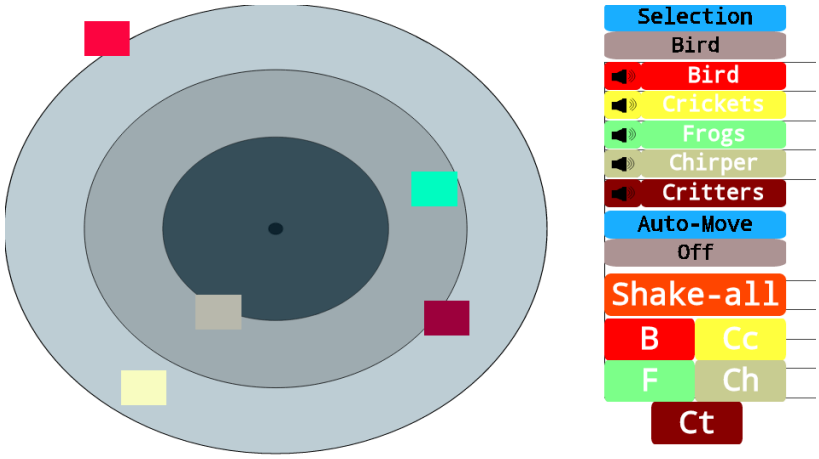


http://rjdj.me/sharescene/spat-birds-b/

8.1.2-SPAT-INSECTS-DROID

On va donc construire une application similaire à spat-birds.rj mais avec quelques différences dues à l'adaptation de l'interface. Les procédés de synthèses sonore sont cependant plus lourds, car on va avoir une richesse de timbres plus large On va donc avoir la possibilité d'avoir jusqu'à 5 sons (1 oiseau avec plusieurs chants, des criquets, des grenouilles, des espèces de grillons exotiques et des cigales) de couleurs différentes. On va pouvoir les déplacer dans l'espace individuellement, ensuite on pourra les

exciter individuellement pour qu'ils se déplacent seuls. On peut finalement utiliser le mode Auto pour les laisser vaquer à leurs occupations.

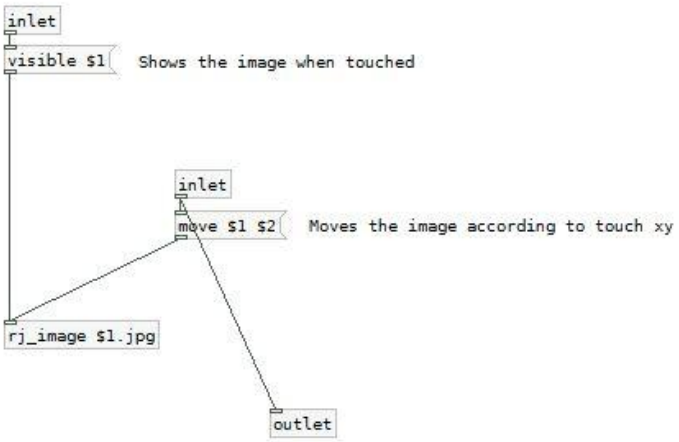


8.2- GRANDES LIGNES DU CODE

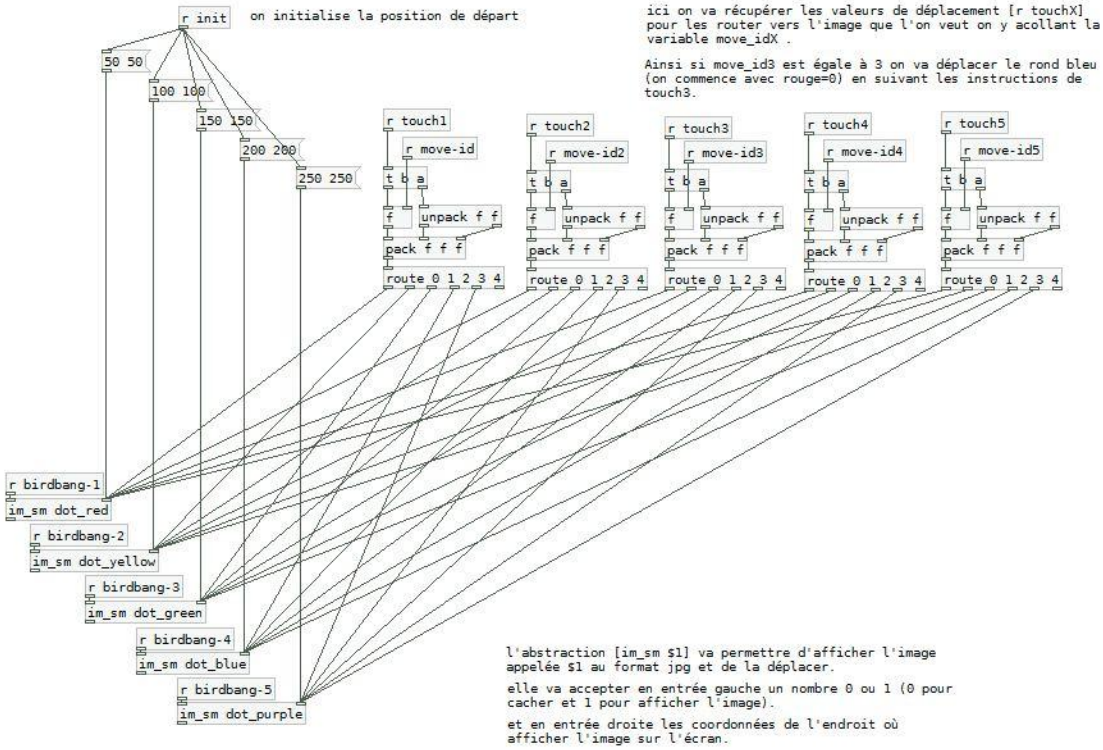
Je vous invite à consulter le commentaire plus exhaustif qui se trouve à l'intérieur des patchs. Ici nous allons aborder la gestion des images et de leur déplacement, ainsi que l' « aléatoirisation » de certains paramètres à l'aide de l'objet [random].

8.2.1- LA GESTION DES IMAGES DANS SPAT-BIRDS.RJ

L'abstraction [im_sim] présente dans le dossier va nous permettre de choisir d'afficher/masquer une image (0 ou 1 à l'entrée de gauche) et de la déplacer sur le touchscreen à une position (x,y) (deux valeurs dans un [pack f f] à l'entrée de droite). C'est une abstraction très simple qui utilise tout simplement le système de message proposée par l'external [rj_image] ; la particularité est qu'elle utilise en argument le nom de l'image à afficher : [im_sim red_dot] va afficher l'image red_dot.jpg présente dans le dossier.



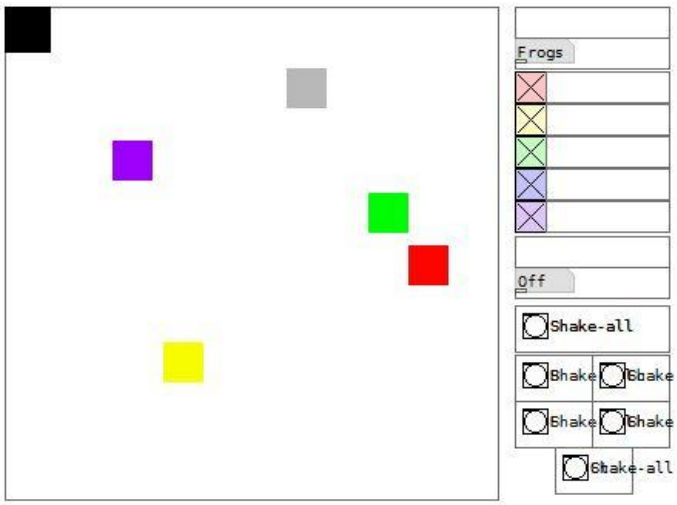
La voici intégrée au reste du code :



Dans l'interface on a donc 5 images qui peuvent s'afficher/se masquer et se déplacer se sont les points colorés. Le reste est une image fixe mappée sur une grille à l'aide de [m_touch2grid].

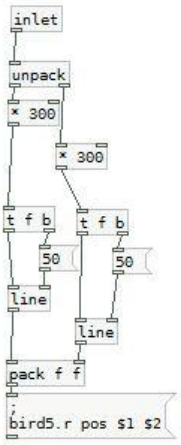
8.2.2- LA GESTION DE L'AFFICHAGE DANS SPAT-INSECTS-DROID

Voici tout d'abord le patch sous sa forme desktop comme dans l'épisode précédent, on va utiliser des widgets SVG pour customiser l'aspect lors de l'utilisation sous android.



Tout d'abord il est important de comprendre que nous ne travaillons plus avec des images, qui n'ont pas de représentation physique dans pure data. Nous allons donc être obligés de travailler avec des objets

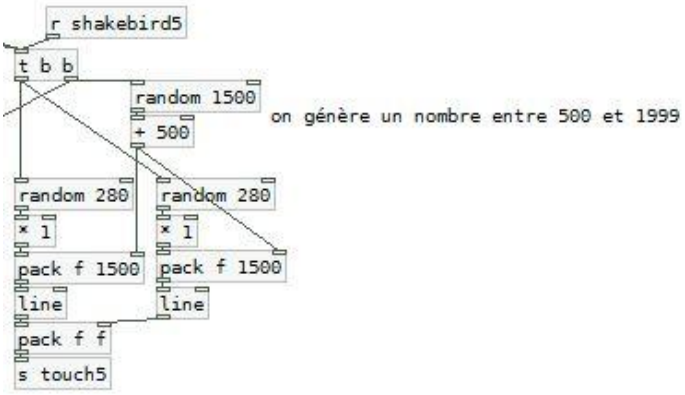
Gui de pure data. Après de nombreuses heures à essayer de débugger les premier tests, il s'avère que la plupart des objets ne peuvent pas encore bouger sous android (cette fonction de pure data n'a pas encore été implémentée pour tous les objets), il faudra donc se contenter de l'objet [canvas] qui lui bouge (on l'a déjà fait à l'épisode 4). On va donc utiliser le même type de code pour déplacer le canvas ; les objets [* 300] permettent d'ajuster le déplacement sur la surface tactile (grâce à l'abstraction [touch 300 300 touch1], de l'intervalle [0,1] à l'intervalle [0,300]).



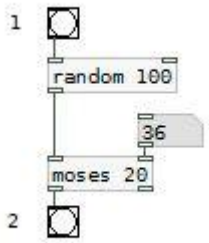
Lors de l'utilisation de la customisation SVG l'objet canvas ne supporte malheureusement pas le fait d'utiliser l'argument send pour attribuer différents fichiers à différents [canvas], il faudra donc se contenter de leur apparence tels qu'ils sont rendus par PdDroidParty sans grande possibilité de savoir à quoi s'attendre avant de charger le patch.

8.2.3- L'OBJET [RANDOM] VERS LES PROBABILITES

[random] est un objet qui va générer des nombres pseudo-aléatoires. Si on créer un objet [random N] il va générer un nombre compris entre 0 et N-1. On peut donc utiliser différents opérateurs mathématique derrière pour forcer la sortie à correspondre à un intervalle d'arrivée satisfaisant pour le paramètres que l'on va controler. Ici on génère une valeur de temps entre 0 et 1999 ms et deux valeurs de position entre 0 et 279 à l'aide de [line] on va donc créer un déplacement de la position précédente à une nouvelle position sur un temps donné.



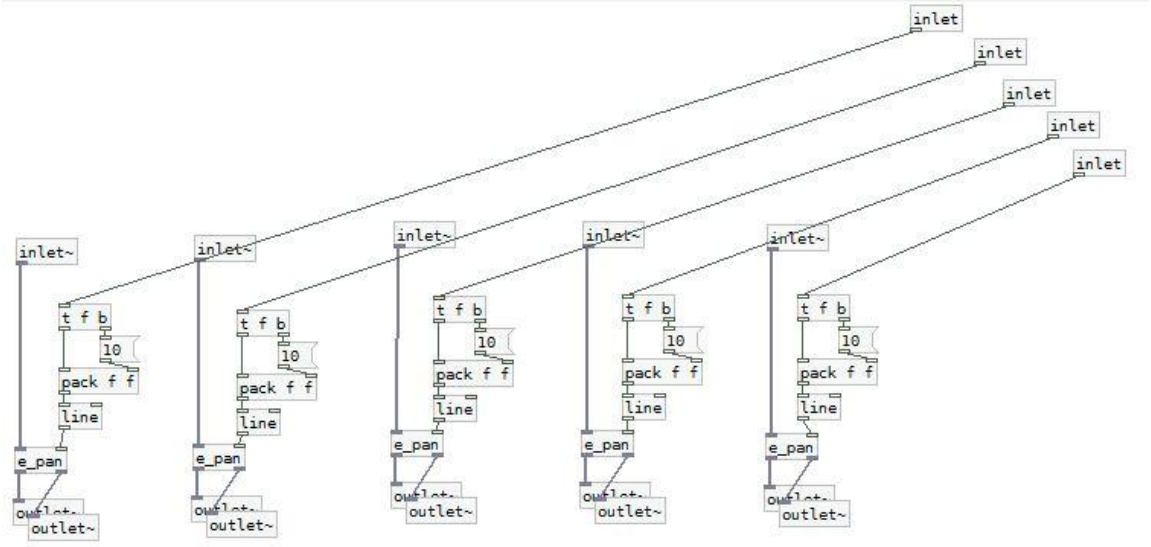
Il devient alors assez facile d'avoir une abstraction permettant de transmettre un bang selon une probabilité fixée par l'utilisateur. Il suffit d'associer [random 100] à [moses ma_proba]. L'exemple ci-dessous à 20% de chances de transmettre une information si l'on change cette valeur à l'aide de l'inlet de droite et d'un objet [numberbox] alors on peut la faire varier (ici 36% de chances).



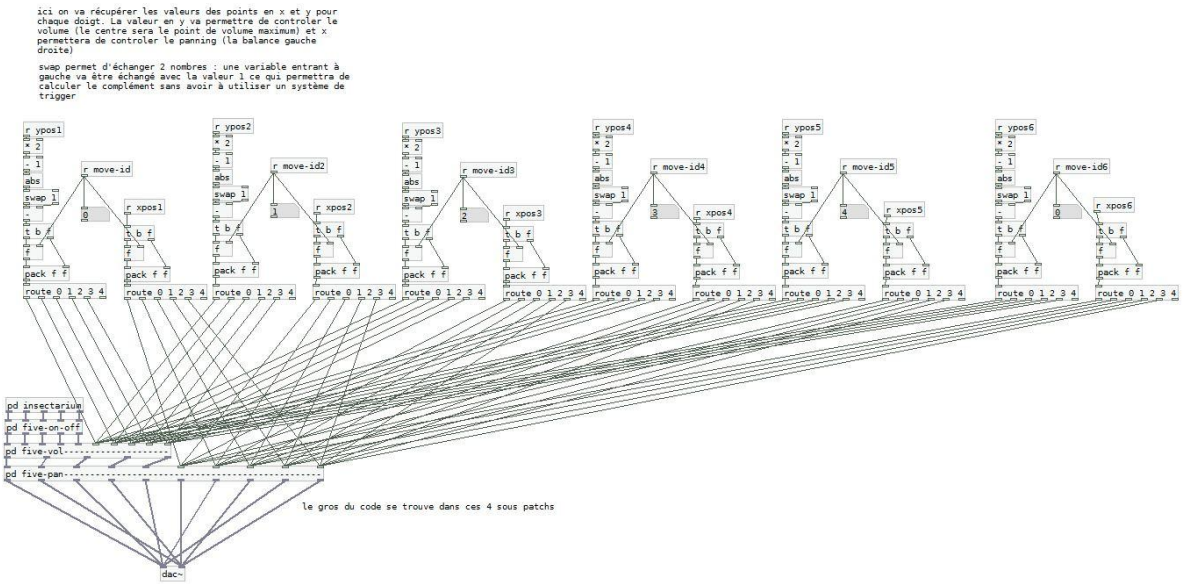
Dans la section ressource de l'épisode 1 nous avions déjà mentionné Mengine.pd, un patch de musique générative qui nous avait permis de créer des boucles sonores. Ici vous pourrez découvrir qu'il utilise une généralisation de ce principe pour l'auto-génération de musique.

8.2.4- L'ABSTRACTION [E_PAN]

C'est une abstraction faisant partie de la collection RJDJ. C'est un panoramique stéréophonique classique qui prend en entrée un signal mono et le transforme en signal stéréo en fonction d'un argument en entrée droite compris entre 0 (gauche) et 1 (droite).



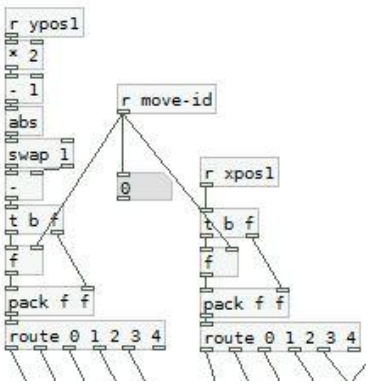
Ici on en utilise un pour chaque objet sonore, il recevra les valeurs approprié grâce à un système de routage comme on en a déjà vu pour recevoir les bonnes valeurs au bon endroit voire ci-dessous.



Chaques valeurs de déplacement xpos1 et ypos1 sont routées vers les bonnes entrées à l'aide de l'information move_id. Les déplacements en x sont pour la gestion du pannoramique et y pour la gestion du volume.

8.2.5- L'UTILISATION DE SWAP DANS LA GESTION DU VOLUME : LE PRINCIPE DES ENTREES CHAUDES ET FROIDES.

Pour le volume on va récupérer une valeur comprise entre 0 et 1 on veut aussi une valeur entre 0 et 1 pour le contrôler, mais au vue de l'interface, il faut que le son soit maximal au milieu et nul en haut et en bas, alors que si on applique directement les valeurs que l'on récupère le son sera maximal en bas et nul en haut, on va donc devoir coder un peu :



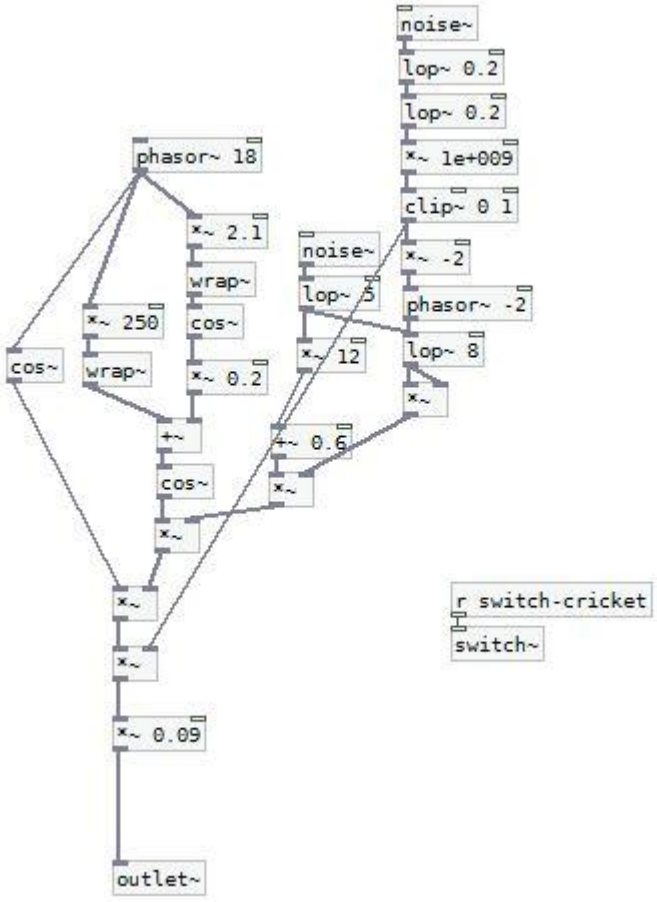
De (0,1) on passe à (0,2) en multipliant par deux, puis à (-1,1) en soustrayant un. Maintenant lorsqu'on est en haut le volume est à moins -1, au milieu il est à 0 et en bas à 1. Ensuite on prend la valeur absolu : toujours 0 au milieu mais 1 en haut et en bas on y est presque il suffit alors de prendre le complément à un de cette valeur. Pour cela on va utiliser [swap 1] qui va simplement échanger les sorties par rapport aux entrée : s'il reçoit 0.4 en entrée et que son argument est 1. La sortie gauche va être évaluée à 1 et la

droite à 0.4. En utilisant cette technique on calcul facilement le complément à 1 de tout nombre se présentant. On arrive alors à avoir 1 au milieu de la surface et 0 en haut et en bas. On ne pourrait pas le faire aussi facilement avec la boîte [-] car on aurait besoin de croiser les fils et de systématiquement envoyer une valeur de 1 à l'entrée de gauche dès qu'une valeur se présente pour ré-évaluer la sortie. C'est le principe des entrées chaude (hot) ou froide (cold). Une entrée chaude va ré-évaluer la sortie (souvent le première entrée à gauche) un entrée froide va juste stocker la valeur en attendant que la sortie soit ré-évaluée.

8.2.6- L'UTILISATION DE SWITCH~

[switch~] va permettre de stopper les calculs de DSP (Digital Signal Processing) dans un sous patch. C'est utile pour économiser de la mémoire et avoir donc des patchs plus efficaces. Par exemple si on a un objet [noise~] dans un patch celui-ci va toujours effectuer des calculs que le son soit coupé ou non, [switch~] permet d'y remédier.

On va simplement le mettre dans le sous patch à couper et lui envoyer un message : 0 pour arrêter et 1 pour calculer.



Voici de nouveaux concepts qui nous permettront d'avoir des interface enrichies avec du mouvement et un peu plus d'interactivité. L'utilisation de random permet de générer des systèmes musicaux autonomes qui ne se répètent pas.

8.3- RESSOURCES

Dans la documentation intégrée à pure data vous pourrez trouver le patch d'aide de [random], vous y trouverez des informations intéressantes sur le seeding et sur la façon dont fonctionnent les algorithmes pseudo-aléatoires utilisés pour la génération de valeur dans random.

8.3.1- DES LIENS

La spatialisation effectuée ici est très basique, pour plus d'informations sur les méthodes de spatialisation possible avec pure data vous pouvez explorer ce lien en anglais, qui est un workshop en anglais sur la spatialisation par Georg Holzman qui présente différentes méthodes (nécessitera pd-extended).

http://grh.mur.at/misc/PdSpatialization.tar.gz

8.3.2- PATCHES SUPPLEMENTAIRES

Animals_Bird_Hansm.pd

Le patch d'oiseau utilisé dans Spat-insects-droid, mais cette fois avec une interface exploitable sur ordinateur et bien plus de paramètres apparents et manipulables.

Animals_Insects.pd

La colonie d'insectes d'Andy Farnell dans un seul patch.

BirdsUseStar.pd

Les oiseaux FM à la base du patch RJDJ, un patch de game.

Mengine-Ambiant-Std-Ed.zip

Mengine, une généralisation de la génération de musique à l'aide de probabilités. Il est en effet possible de créer des presets avec des probabilités pour chaque pas de chaque séquenceur et donc de décider si l'on joue une note ou non. De plus il est aussi utilisé pour gérer l'harmonisation du morceau à la fois dans le choix des gammes et des notes jouées mais aussi dans la structure harmonique globale. Il possède 3 voix différentes (bass, chords et lead) et une voix de batterie .

8.3.3- DES WIDGETS SVG

Les fichiers images crées pour les applications ci-dessus. Ceux-ci peuvent s'utiliser très facilement avec Inkscape, un logiciel gratuit de création et d'édition de graphismes vectoriels.

http://inkscape.org/?lang=fr

9-EPISODE 6 : APPLICATIONS

Dans ce sixième et dernier épisode nous allons créer deux applications relativement complexes. Un petit jeu vidéo dans lequel vous serez un vaisseau spatial errant dans une galaxie lointaine et une application musicale qui permettra de générer des rythmes et des mélodies de façon « intelligente ».

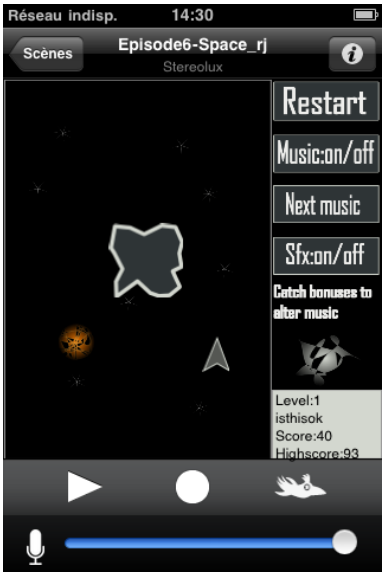
9.1- FONCTIONNEMENT

Dans cet épisode nous allons revoir une bonne partie des choses que nous avions déjà étudié jusqu'à présent, nous allons cependant voire de nouvelles choses comme : utiliser des tests logiques pour détecter des collision entre des objets affichés et utiliser des chaînes de markov pour analyser des rythmes et des mélodies afin de générer des variations musicales.

9.1.1-SPACE-TRAVELLER.RJ

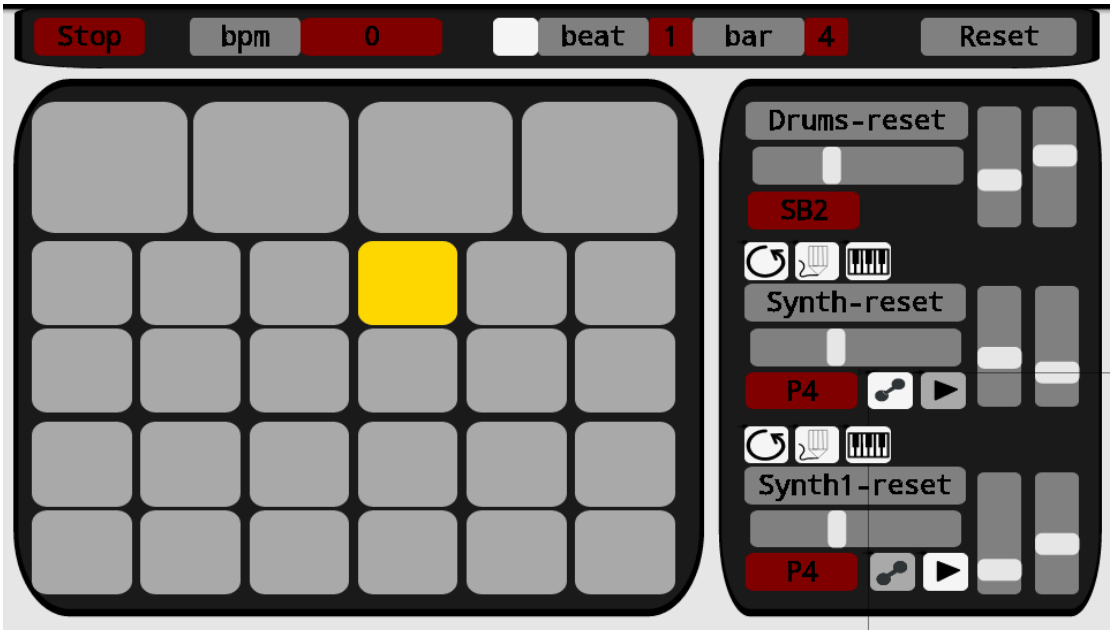
Space-traveller.rj est un petit jeu pour RjDj, dans lequel vous êtes un vaisseau spatial. Vous devrez éviter ou détruire des astéroïdes de différentes tailles, et attraper des bonus. Les bonus vous permettront (en plus de marquer des points supplémentaires) d'altérer la musique du jeu (en modifiant les notes qui la composent) ou en modifiant la part d'aléatoire dans la génération des notes. L'ensemble des graphismes sont composés d'images fixes qui se déplacent sur l'écran, et l'ensemble du son est généré en temps réel.

Un moteur de collision rudimentaire a donc été développé. Il faut relever, que cette réalisation relève plutôt de l'exercice de style RjDj et Pure Data n'étant pas faits pour réaliser ce type de choses, le déplacement des images et la détection des collisions peuvent être parfois un peu aléatoires.



9.1.2-MUTLI-TAP-DROID

Cette application est dérivée de l'application Multi-tap.rj de l'épisode 4. L'interface est maintenant sur mesure pour le projet droidparty et ce portage ajoute aussi de nombreuses fonctionnalités et quelques modifications sur l'utilisation. Elle devient maintenant un instrument de musique complet avec différents modes de jeux, et un retour visuel attrayant.



Nous avons maintenant le possibilité de changer le tempo (a un tempo très rapide l'application permet notamment de créer des très belles textures). L'écran peut être de jeu peut-être séparé en trois zones distinctes : la barre du haut (mise en route du séquenceur, tempo, compteur et bouton pour tout remettre à zéro), la zone de gauche (composée de carrés gris) qui est la zone de jeu, et la partie de droite qui va permettre de choisir les sons /régler leurs volumes/ choisir les différents modes de jeux et contrôler les effets.

La première chose à faire est de choisir un tempo puis de lancer le séquenceur, celui-ci va alors se mettre à défiler comptant 4 mesures de 4 temps. Une fois cela fait vous pouvez appuyer sur les boutons de la zone de jeux. En fonction des modes de jeux les effets seront différents.

Le mode piano permet de jouer comme si le programme était un simple piano, le mode crayon permet d'enregistrer les notes que vous jouez en mémoire, le mode boucle permet de jouer en boucle les notes qui auront été enregistrées. Les différentes combinaisons de ces trois modes peuvent vous permettre de créer une boucle puis d'improviser par-dessus sans la modifier par exemple.

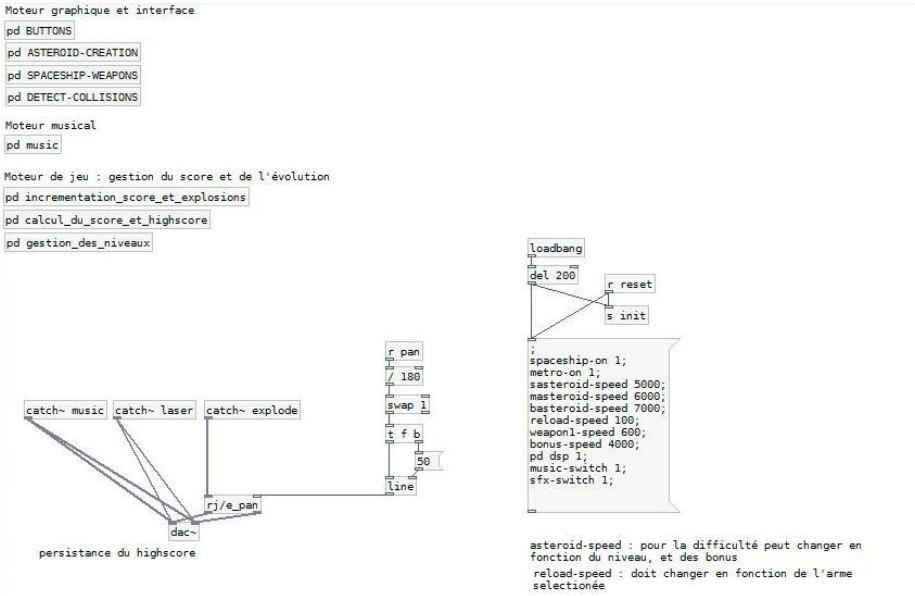
Deux autre modes sont disponibles : le mode entrainement et le mode improvisation. On peut donc ainsi générer des mélodies grâce à des tables de probabilités crées automatiquement par l'analyse du jeu de l'utilisateur. En utilisant cette méthode sur la hauteur des notes mais aussi sur le rythme on arrive à créer des variations de mélodies existantes.

Les sliders verticaux sont pour les effets, tout à droite des reverbs individuelles, et à côté un bitcrusher pour la section rythmique et des delais pour les synthétiseurs. Les sliders horizontaux contôlent le volume de chaque instrument.

9.2- LES GRANDES LIGNES DU CODE

La plupart des éléments qui composent le code de ces deux application ont déjà été éprouvés dans les épisodes précédents, nous n'y reviendrons donc pas en détail.

9.2.1.1- CREATION DES OBJETS AFFICHES ET GESTION DES NIVEAUX. (SPACE-TRAVELLER)



La fenêtre principale du programme

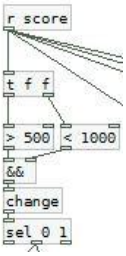
Dans cette application l'évolution du score est utilisé comme gestionnaire des niveau. Le niveau de difficulté va générer les arrivées des asteroids et bonus. Pour cela on va envoyer des messages composées des probabilités d'afficher chacun des éléments à un moment donné, en fonction de la tranche de score dans laquelle on se trouve.

Il y a septs éléments différents qui peuvent arriver du haut de l'écran :

- 3 types d'asteroids (petit, moyen ou grand)
- 3 bonus de types note (note1 en rouge, note2 en vert, note3 en violet)
- 1 bonus de type random.

Il faut détruire les asteroids un petit nous rapporte 10 points, un moyen 20 points et un gros 30 points. Si on attrape un bonus on marque 50 points. Par ce biais on indique clairement que le but du jeu est d'attraper les bonus et donc de faire évoluer la musique plutôt que de détruire des cailloux.

On va ensuite utiliser des test pour déterminer dans quel tranche de score on se trouve. A chaque actualisation du score on va le comparer à une borne supérieure : objet [<] et une borne inférieure : objet [>] (fonctions du niveau) si le score se trouve entre ces deux valeurs alors on envoit la valeur correspondant à ce niveau : objet [&&]. Le niveau 1 correspond à un score entre 0 et 500, le niveau 2 à un score entre 500 et 1000.



Test logique pour la détermination du niveau (ici niveau 2)

En fonction du niveau on va envoyer un message composé d'une liste de probabilités, qui va s'appliquer à l'affichage de chacun des éléments. Un message [15 30 45 55 65 75 100(signifie :

-15% de chances d'avoir un petit asteroid.

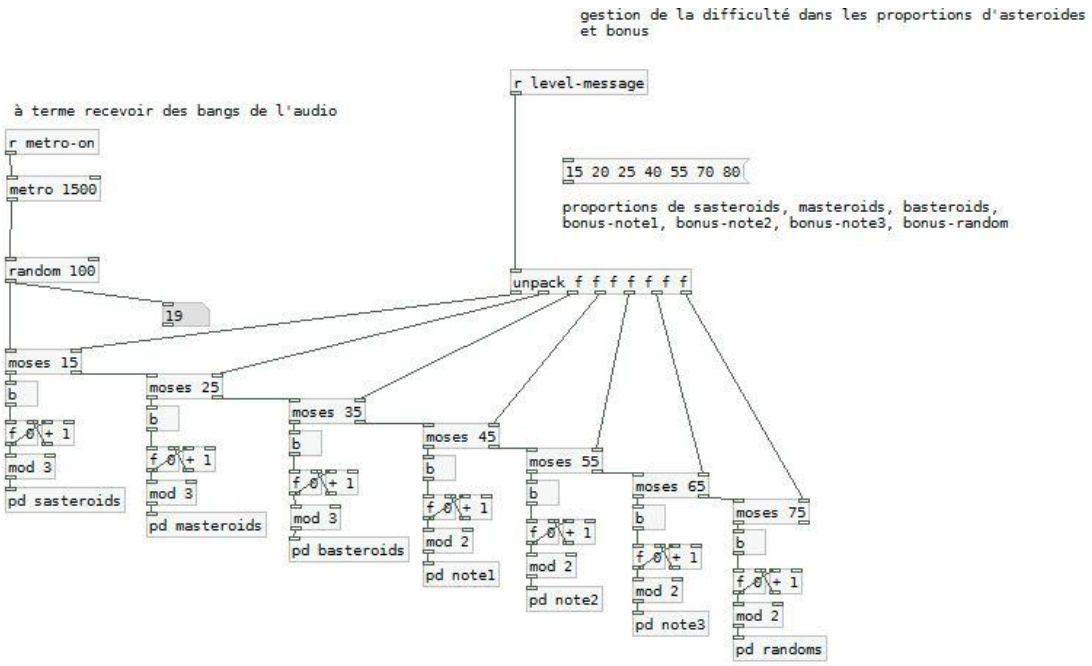
-15% de chances d'avoir un asteroid moyen. (30-15=15)

-15% de chances d'avoir un gros asteroid. (45-30=15)

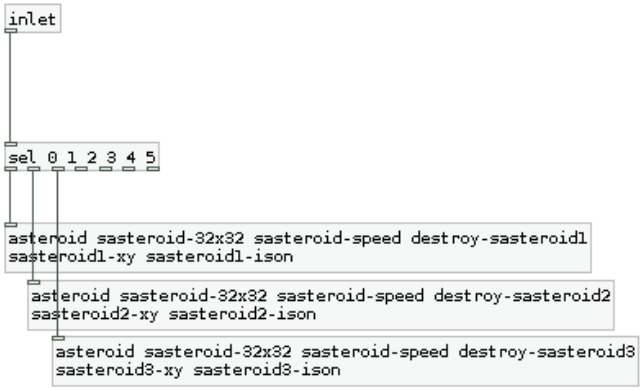
-10% de chances d'avoir un bonus note1, note2 ou note3.

-25% de chances d'avoir un bonus random.

Cette liste va alimenter une chaîne d'objets [moses], qui va en fonction d'un nombre aléatoire généré toutes les 1500 secondes va décider quel objet créer. Nous avons décidé d'avoir 3 affichages du même objet au maximum à l'écran. On va utiliser une abstraction pour afficher ces objets comme à l'épisode précédent. L'abstraction [asteroid] est ainsi assez similaire à l'abstraction [im_sm] de l'épisode précédent, elle permet d'afficher une image en fonction de son nom, de calculer son déplacement et d'envoyer ces valeurs.



La chaîne de moses utilisée pour créer le bon objet



Le contenu du sous patch [pd sasteroid]

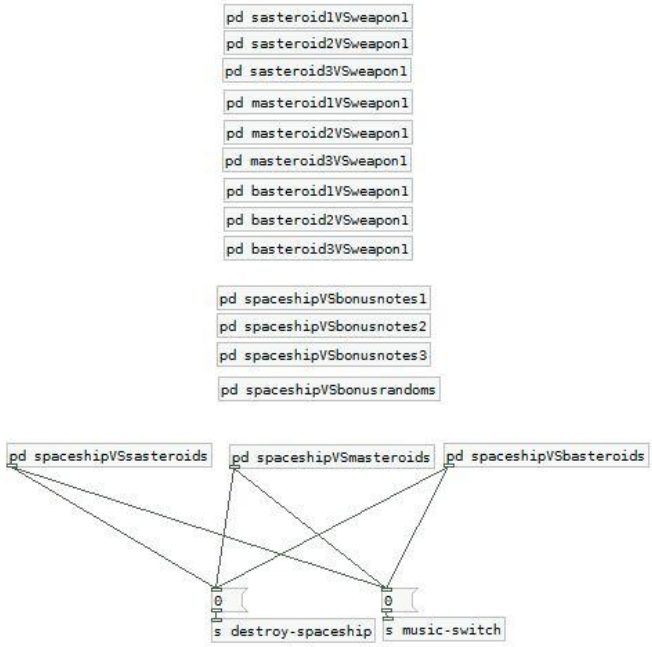
Ce sous patch permet de gérer l'affichage des petits asteroids, les autres sous-patchs sont construits sur le même modèle. Trois abstraction de type [asteroid], qui nécessitent 5 arguments pour leur création : le nom de l'image à charger (sateroid-32x32), le deuxième n'est plus utilisé dans cette version mais permet de spécifier une vitesse de déplacement (il est remplacé par la génération d'une valeur aléatoire comprise entre 1500 ms et 8500ms ce qui signifie que l'asteroid mettra entre 1500 et 8500 ms pour traverser l'écran de haut en bas), le troisième est un nom specific pour recevoir une information lorsqu'il est détruit, le quatrième envoi les valeurs de déplacement, et le dernier est une variable permettant de savoir si l'objet est actuellement affiché ou non (pour l'optimisation du moteur de collision).

9.2.2- LES TESTS LOGIQUES : CREATION D'UN MOTEUR DE COLLISION. (SPACE-TRAVELLER)

Pour construire un moteur de collision, on doit savoir à tout moment où sont tous nos objets et comparer leur coordonnées en permanence. On utilise des système de [spigot] pour ne pas effectuer tout le temps toutes les opérations, surtout lorsque les objets concernés ne sont pas affichés.

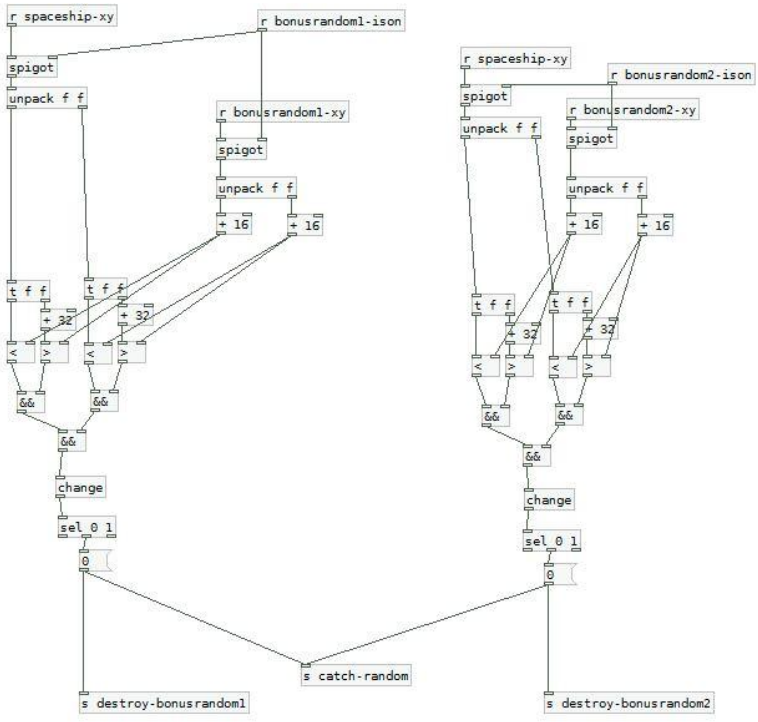
On doit s'occuper des collisions entre différents éléments :

- Vaisseau vs Asteroids : qui va entrainer la destruction du vaisseau et la fin du jeu.
- Vaisseau vs Bonus : qui va entrainer la disparition du bonus, l'altération de la musique et une augmentation de points.
- Asteroids vs Armes : qui va entrainer la destruction de l'asteroid et une augmentation de points.



L'intégralité des sous-patchs du moteur de collision

De la même façon on va comparer avec [>], [<] et [&&] toutes les positions qui nous sont envoyés par messages, par exemple [s spaceship-xy] pour la position du vaisseau.



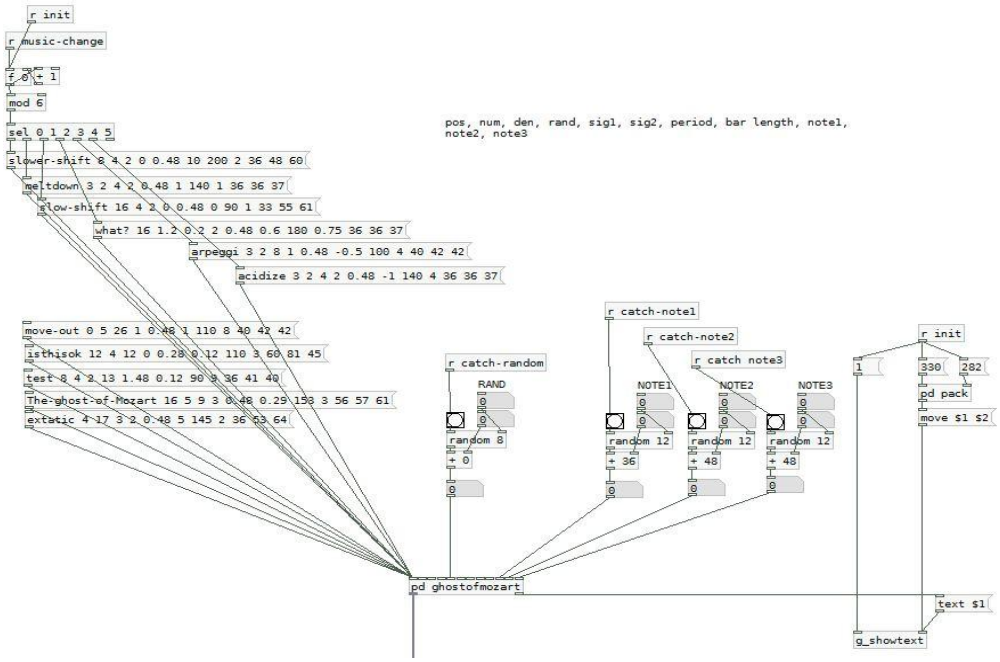
Comparaison des positions du vaisseau et des bonus de type random

Chacun des autres sous-patch est construit sur ce modèle. Vous pourrez remarquer l'objet [+ 36] situé après la réception des coordonnées du bonus, cela va nous permettre de définir le centre de notre objet. D'une façon similaire après les coordonnées du vaisseau on ajoute parfois 32 à ces valeurs. De cette façon on va agrandir la

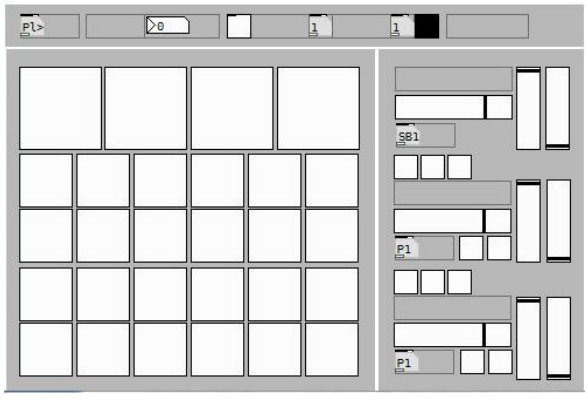
surface sur laquelle on va vérifier la collision : au lieu de vérifier si deux points sont au même endroit on va vérifier si un point (le centre du bonus) est à l'intérieur d'un carré (l'icône du vaisseau).

9.2.3- L'HISTOIRE DU MOTEUR MUSICAL DE SPACE-TRAVELLER : CHRONIQUE DE L'EVOLUTION D'UN PATCH

Le moteur de musique utilisé provient (une fois n'est pas coutume) d'Andy Farnell. On le nourrit on lui fournissant une liste de paramètres. Je l'ai un peu modifié et j'ai créee de nouvelles preset pour pouvoir correspondre à un esprit 8-bit un peu rythmé, mais c'est un moteur permettant une bonne variété musicale dans les sons 8-bits. Les paramètres de randomisation et de changement de notes permettent d'obtenir des variations plus subtiles. C'est un vrai moteur de musique procédurale. Je vous invite à le disséquer et créer vos propres presets. Vous le trouverez dans la partie ressources avec la version originale d'Andy, une première modification qui a été produite pour Patchwerk radio (radio streaming de musique générative utilisant pd) et cette modification intégrée dans un moteur de jeu.



9.2.4- LES CHAINES DE MARKOV : LA RICHESSE DE L'AUTOGENERATION. (MUTLI-TAP)

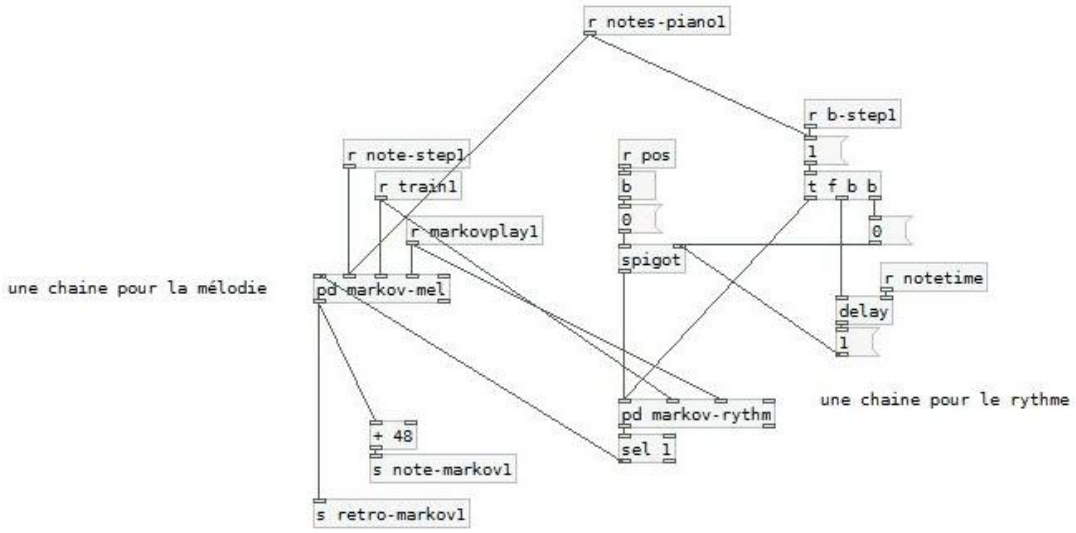


La fenêtre de Multi-tap dans Pure-Data.

Le mode entrainement permet « d'écouter » toutes les notes qui sont jouées par les synthés, une fois le bouton de-enclenché le programme va faire une analyse dite de Markov, ce qui va permettre d'ordonner les notes : connaitre la probabilité qu'une note soit suivie par une autre note ; par exemple si on vient de jouer un do on pourra savoir après analyse qu'il y a 30% de chances que la note suivante soir un ré, 15% de chances que ce soit un si et 55% que ce soit un autre do. En analysant une mélodie on peut ensuite avoir une matrice donnant pour n'importe quelle note la probabilité qu'elle soit suivie par une autre note. En lisant cette matrice on pourra ensuite générer de nouvelles mélodies basées sur ces même probabilités. On pourra aussi implémenter des probabilités rythmiques à savoir la probabilité qu'une note donnée soit suivie d'une autre note ou d'un silence…

On va utiliser les abstractions rjdj concernant les analyse de markov pour les mettre en œuvre, à savoir [c_markov] et [m_markovanal]. (les pages d'aides sont disponibles, je vous invite donc à les consulter).

Il va falloir créer deux implémentations de cette méthode : une pour le rythme et une pour la mélodie. Pour la mélodie on va recevoi à l'entrée de gauche des signaux de synchronisation (bangs en rythme), la deuxième entrée va recevoir les notes jouées puis les troisièmes et quatrièmes entrée vont permettre d'activer les modes de jeux d'abord « entrainement » puis « improvisation » (0 ou 1 pour off ou on). Pour le rythme on va recevoir un 1 si une note est jouée un 0 sinon, le reste des entrées sera le même.

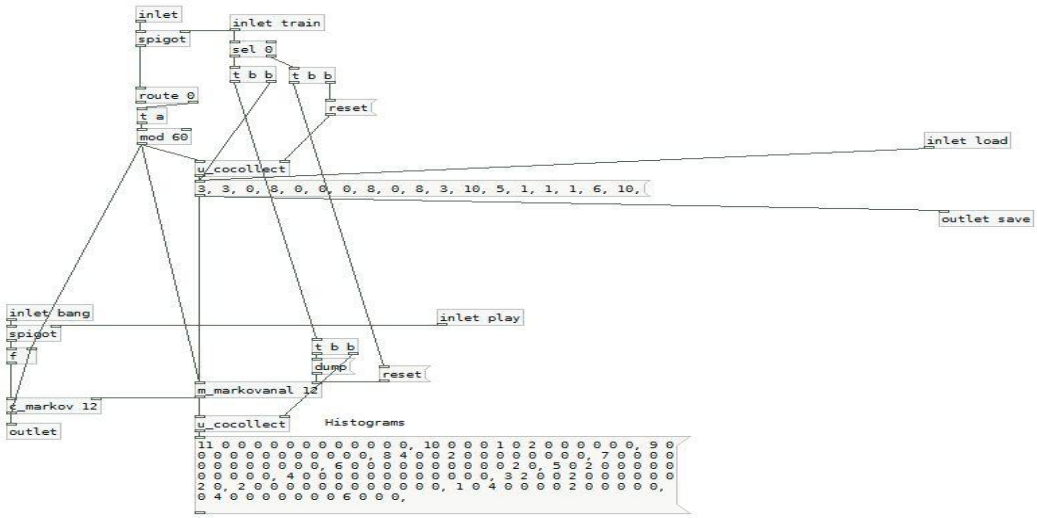


Les deux sous-patchs de chaînes de Markov.

[m_markovanal], permet d'analyser une liste de valeurs et de nous fournir en retour une liste d'histogrammes de probabilités pour chaque note de la liste. En mode entrainement on va donc agréger dans une liste toute les notes qui vont être jouées (quelque soit le mode de jeu), lorsque l'on relâche le bouton d'entrainement la liste et fournie à [m_markovanal] et hop ! les histogrammes sont là prêts à être utilisés par [c_markov] dans l'entrée droite. (on peut envoyer un message [reset(dans l'entrée de droite pour remettre à zéro).

[c_markov] va lui permettre de générer des valeurs aléatoires en fonction de la liste précédemment établie et en entrée gauche l'état sur lequel on se base pour déterminer la note suivante. On doit donc stocker la dernière note jouée par notre chaîne de markov pour la ré-injecter à l'entrée chaude lorsqu'il faudra calculer la prochaine note.

On va utiliser ce couple d'objet deux fois par synthétiseur : une fois pour l'harmonie une fois pour le rythme. Soit quatre fois en tout.



Le couple d'objet [m_markovanal] et [c_markov] ainsi que la liste de note fournie, et l'histogramme

9.3- RESSOURCES

9.3.1- DES LIENS A LIRE

Les chaînes de Markov sur Wikipédia

http://fr.wikipedia.org/wiki/Cha%C3%AEne_de_Markov

Un blog sur la composition algorithmique en utilisant Pure-Data, avec des tutoriaux.

http://www.algorithmiccomposer.com/search/label/puredata

9.3.2- DES FICHIERS

- Le patch d'Andy Ghost of Mozart

http://obiwannabe.co.uk/html/toys/ghostofmozart/ghostofmozart.html

- Ghost of Mozart modifié pour patchwerk radio (inclus au zip)

- Pong.pd : un autre exemple de jeu entièrement construit en Pure Data (inlus au zip). (Il existe aussi un Tetris quelquepart sur internet…)

- svg files : dossier contenant l'ensemble des fichiers svg crées pour ces applications.

9.3.3- DES APPLICATIONS SMARTPHONE.

- autopop de Joe Newlin

http://joenewlin.net/pd.html

- Pure Data Latency Tester : pour tester la latence audio de votre smartphone (inclus dans le zip)

- Sweaty.rj : sonification de gestes sous forme d'une scène rjdj, accompagné d'un artcile en anglais détaillant l'implémentation.

10-QUELQUES RESSOURCES

Pure Data les logiciels issus de Pure Data

http://puredata.info/community/projects/software : téléchargement des version vanilla et extended pour son ordinateur personnel.

http://mccormick.cx/projects/PdDroidParty/ : téléchargement de la dernière version de droidparty.

Pure Data sur les plateformes mobiles forums et tutoriaux

http://blog.rjdj.me/pages/pd-utilities (anglais)

http://noisepages.com/groups/pd-everywhere/forum/

http://www.youtube.com/watch?v=lr-khifcl-U

Pure Data forums et tutoriaux :

http://puredata.info/docs/workshops/ArtAndCodePdDemo

http://www.obiwannabe.co.uk/

http://www.pd-tutorial.com/english/index.html

http://en.flossmanuals.net/pure-data/

http://puredata.hurleur.com/index.php

http://codelab.fr/pure-data (une page de ressources très complète est présente en début de forum, n'hésitez pas à vous y référer)

Pure Data des patchs à découvrir :

- Une fois pd installé on peut aller voir ce lien ou je stocke une version régulièrement actualisée de la collection de patchs que j'utilise sous pd http://code.google.com/p/pdlive/

- Patchwerk radio, une radio diffusant de la musique générative en temps réel grâce à Pure Data (les patchs sont disponibles au téléchargement) : http://radio.rumblesan.com/

Création d'interface :

- Inkscape : un logiciel d'édition de graphismes vectoriels. Il va vous permettre de créer des images, de modifier des fichiers .svg pour la customisation d'interface avec PdDroidParty : http://inkscape.org/?lang=fr

11-QUELQUES NOTIONS

Open-source

Désigne un logiciel dont le code source est disponible publiquement, généralement par l'entremise d'une license d'exploitation qui accorde certains droits aux utilisateurs. Il existe plusieurs types de license open-source, dont nous retiendrons les principales:

License publique: Aucune restriction n'est imposée sur le code source qui peut être modifié et même commercialisé

License BSD: La seule restriction qui s'applique est de citer (donner du crédit à) les développeurs du code source

License Publique Générale Gnu (GPL) (voir "Logiciel libre"): Oblige les utilisateurs qui modifient et redistribuent le logiciel à redistribuer eux-même le code source

License propriétaire: Certains logiciels propriétaires peuvent ouvrir leur code source tout en empêchant toute utilisation

Logiciel libre

Le logiciel libre, aussi appelé "copyleft" (par opposition à "copyright") est un logiciel open-source qui offre quatre libertés fondamentales aux utilisateurs. La license de logiciel libre la plus répandue est la License Publique Générale Gnu. Par exemple, le système d'exploitation Linux est un logiciel libre.

Les quatre libertés, tirées de Wikipedia:

Liberté 0 : La liberté d'exécuter le programme — pour tous les usages ;

Liberté 1 : La liberté d'étudier le fonctionnement du programme — ce qui suppose l'accès au code source ;

Liberté 2 : La liberté de redistribuer des copies — ce qui comprend la liberté de vendre des copies ;

Liberté 3 : La liberté d'améliorer le programme et de publier ses améliorations — ce qui suppose, là encore, l'accès au code source.

DIY

Le DIY ("Do-It-Yourself") est un terme utilisé par différentes communautés centrée sur l'idée de créer soi-même sans avoir recours à l'aide de professionnels. Le terme a pris un sens nouveau avec l'apparition d'internet qui permet d'échanger facilement des informations, des recettes, des idées pour s'approprier ses moyens de production.

Open-hardware

Définit les ordinateurs et le matériel électronique développés selon des principes de la culture du logiciel open-source. Il est donc conçu pour se rapprocher du modèle du logiciel open-source, c'est-à-dire qu'il offre à l'utilisateur l'accès aux informations nécessaires afin de reproduire soi-même le matériel.

Puisque la nature du matériel est différente de celle du logiciel et parce que le concept est relativement nouveau, il n'existe pas encore de définition générale de ce qu'est un matériel "open-source". Le plus couramment, on désigne ainsi les produits dont la part immatérielle (les informations concernant leur conception, les programmes utilisées, etc.) est ouverte.

Creative Commons

Les licences Creative Commons constituent un ensemble de licences régissant les conditions de réutilisation et/ou de distribution d'œuvres (notamment d'œuvres multimédias diffusées sur Internet). Elles ont été créées en partant du principe que la propriété intellectuelle était fondamentalement différente de la propriété physique, et du constat selon lequel les lois actuelles sur le copyright étaient un frein à la diffusion de la culture. Leur but est de fournir un outil juridique qui garantit à la fois la protection des droits de l'auteur d'une œuvre artistique et la libre circulation du contenu culturel de cette œuvre, ceci afin de permettre aux auteurs de contribuer à un patrimoine d'œuvres accessibles dans le « domaine public » (notion prise au sens large).