CREATE MOBILE APPLICATIONS WITH PURE DATA

(IOS ET ANDROID)

STEREOLUX 2011

Foreword :

The goal of this project is to discover new tools to create mobile applications. I would like to thank the whole team at Stereolux, which was dedicated to the project, and enabled me to work in great conditions, write this documentation and set everything up for the workshop.

This project would not have been possible without the help of the Pure Data community, numerous examples come from work from the development team, and members of the community, errors remaining are purely my own.

Let me know if you see something wrong in this document (spelling, things to add or to explain better), moreover if you some code of yours and it wasn't properly mentioned. I intend to complete this document with the feedback I'll receive. (brerenger.recoules@gmail.com)

A web version of this documentation (with videos) can be found at the following adress (french only) :

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

1-NUMERICAL ARTS, WHY ENGAGE PEOPLE TO DEVELOP AND USE FREE SOFTWARE AND FREE HARDWARE ("OPEN SOURCE" AND "OPEN HARDWARE") ?

Open source software have begin to developp in the early 80’s with the creation of the first open license Gnu by Richard Stallman. Stallman was opposed to embezzlement of source code and software by companies. Hide the source code and make it illegal to modify it seemed inconsistent with the very nature of source code, which can theoretically be shared, without any fees (as opposed to objects as a bicycle, or a car- that’s why there are rentals for those). In the Gnu manifesto it is written :

“ Extract money from users of a software, by restraining their use of a program is destructive because at the end it reduces the amount of resources humanity can get from the program {...] since I do not like the situation which results in a general restraining of information, I consider immoral acting like that” [1]

As opposed to proprietary software, free softwares give the liberty to execute, study and distribute copies of a program.

In arts, free software have grown since the 90’s, starting with Pure Data. Several other have made their appearance since then and are used by artists such as : Processing, Arduino, Toonloop, Gephex, Freej etc. These software are used by artists because they are free of charge, powerfull, and easy to use. As they are open they facilitate and encourage, their appropriation by artists, and contribute to research and creation of new art forms. More and more artists get the deployment of free software, and free materials into their workflow: creating their own tools, instead of working with existing ones.

In this spirit, will develop this project aiming at developing small creative applications for your smartphone.We will use open tools to promote their use, hence developing their documentation.

2-THE MULTI-FACETED PURE DATA

Pure Data is free, open source software that runs on (Linux, Windows,OS, android, iOS etc.). It will enable you to compose music in new ways, and help to create multimedia interactive pieces, installations, scenographies and really anything you can imagine. It’s a visual programming language.

Pd is distributed in several versions :

- the **vanilla** version, developed by Miller Puckette (creator of the program).

- the **extended** version which will contain contribution of the Pd community (via externals).

- and recently it has been available as a library: **Libpd**, that will enable you to use and communicate with pd from and to several other programmation environnement.Libpd will thus enable us to program with pd for our smartphones. Thanks to the work of Peter Brinkmann.

Appart from being an asset to many artists, Pd has already been used in “commercial” applications : such as **RJDJ** or the Inception application available on the appstore, or even the game Spore developed by EA Games. EA Games has even developed it’s own proprietary library from the source code of PD.

This project shows many assets for anyone willing to start creating in the numerical field:

- a free, open source, and mutli-platform software, with a very active and friendly community.

- real-time sound processing, and real-time sound synthesis, which would enable a better link between user actions and audio rendering. This should improve user experience.

- possibility to build custom sound engines, exactly adapted to you needs.

- no dependancies.

- lightweight library.

3-THE PROJECT FOR STEREOLUX

Over a six month period we will try to discover Pure Data through mini-application for smartphones (android and iOS)in a DIY (Do It Yourself) spirit. We will develop comments in the code, and a large variety of resources available online.

We will discover several themes ranging from sound synthesis to Gui (graphical user interfaces) design; the heart of the project is attached to open source technologies, resources sharing thanks to the Pure Data community and the several projects it developed over last few years.

At first, articles will published online every month, to demonstrate applications running on smartphones and discuss some technical points. Source code will be available for download, use and distribution as ressources corresponding to technical concepts used and encouraging you to go further in your understanding of those.

The main goal is not to learn how to use Pure Data, as many resources and tutorials are already available online (please refer to the resource page), we will not try to re-write those, but we will try to make our work as easy as possible to understand. You don’t need to have any knowledge of Pure Data to use and run those programs (you just need to know how to use a computer and a smartphone). Nevertheless if you wish to go further and don’t know of Pure Data, we encourage you to start to read the Pure Data floss manual (http://flossmanuals.net/puredata/), the introduction and getting started sections should give you enough basic concept to start studying examples shown in here.

A second phase of the project will introduce a few workshops (free of charge and open to anyone) at the Stereolux premises in Nantes (France). You’ll just need to come with your smartphones to get started using the applications and develop your own. Your contributions could also be available for download on the Stereolux website.

Finally we will develop and interactive installation using those technologies.

THE TOOLS

Here's the link to the offcial Pure Data website, you'll find links to download the different versions of Pd, as well as many ressources there : http://puredata.info/

Here's the direct link to the download page :

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

The projects that will be of interest to us are the following :

- Pure Data « **vanilla** »,is the version of Pure Data to install on your peronal computer to be able to build programs and then transfering them to our smartphones. Thanks to Miller Puckett.

- **ScenePlayer** (android) or **RJDJ** (iOS available through the appstore) : it will enable us to load a Pure Data patch and get our device's Touchscreen and sensors data. We will also be able to record sound from it, and build basic interfaces using images.The patches should be in folder called myScene.rj, and the main patch should be call _main.pd. Thanks to Peter Brinkmann.

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

- **PdDroidPArty** (android only) : it will enable us to load a Pure Data patch as i twas developped with a native emulation of the different GUI elements available in Pure Data.The patches should be in folder, and the main patch should be called droidparty_main.pd. You just need to keep your Gui separated from the rest of your code using sends and receives; the guts of the patch should be in a subpatch. Thanks to Chris McCormick.

http://droidparty.net

- **pd-webkit** (android only) : it will enable us to link a Pure Data patch to a html file.An audio engine built in Pd will communicate to a visual interface programmed in HTML through a simple set of messages. The patches and HTML files should be in folder called myProgram.wpd, the main patch called _main.pd, and the html file index.hmtl. Thanks to Chris McCormick.

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

When you study the patches on you computer, remember that you can enter objects by right click -> open, and check the embedded documentation for any object by right click -> Help.

ANDROID

With android whatever the project is the principle is always the same :

1- download of an application as a .apk, and installing it to your phone.

2- building patches with Pure Data.

3- copying the patches on the Sd card of the smartphone (with a usb cable or with an app enabling to share your SD card via Wifi as File Expert : https://market.android.com/details?id=xcxin.filexpert)

4- the application will automatically find patches on startup.

If you want more detailed steps you check out this page :

http://puredata.hurleur.com/sujet-6140-running-your-patches-android-usind-pddroidparty-steps

On android you can also download the files straight from the website and use something like Linda manager to extract zip files and move it anywhere on your SD card, and also install .apk files https://market.android.com/details?id=com.lindaandny.lindamanager

IOS

On iOS we will advise you to download the RJDJ app from the appstore (it's free), and the follow the direct links available on each episode page.

Alternatively you can use the local server system "rjzserver"(available for OS, linux and windows) to copy you patches on the phone.http://blog.rjdj.me/pages/pd-utilities

If your phone is jailbroken you can use something like iphone explorer (http://www.macroplant.com/iphoneexplorer/ - windows and mac only) to directly copy the myscene.rj folderin the RJDJ's application folder. It's a bit tricky to find this one, you'll need to go to /Root Directory/var/mobile/Applications/ in this last folder names are encrypted you'll need to find the right one ! The folder we are looking for contains a folder called « RJDJ.app ». Once you found it you'll need to enter /Documents/rjdj_scenes/ you can finally copy tour folder in here.

ANDROID VS IOS

These two operating systems have different (opposed) philosophies :

iOS is proprietary (software and hardware), it is stable and effective; thanks to it's controlled hardware it's possible optimized set of instruction for a specific program thus enabling to "boost"performances. However it lacks a bit of "openness"to enable to do everything we want to easily (e.g. simply copy files from computer to smartphones). Even if it is proprietary it won't prevent us to use open source code .

Android is open. But due to the variety of hardware available to run this operating system, it is sometimes difficult to be certain that all application will be able to run without flaws on each of the devices available. It's not possible to have optimization for each piece of hardware thus you will most certainly experience a bit of latency.

In this domain, technical performances evolve very fast, some of the oldest smartphones may lack of CPU power (or FPU) to run every application smoothly.

4-EPISODE 1: HOW TO USE SAMPLES ?

This first episode will help us to get familiar with Pure Data its interface and basically how it does work. We will learn to play audio files synched on to another using code snippets from the community, and how to control the volume of the playback. We will also discover how we can interact with a smartphone with the tools provided.

4.1- HOW DOES IT WORK ?

This little application allows you to mix 3/4 audio loops together, 3 for the RJDJ version and 4 for PdDroidParty version. The Rjdj version maps each loop volume to the accelerometer sensor, moving the phone will then change the mix. For the PdDroidParty version a little interface is available (see snapshot), and for each loop you can toggle it and adjust the volume through a slider. There's also the possibility to have several loop packs, and switch between them (touch with two finger on RJDJ, select through a number box for PdDroidParty).

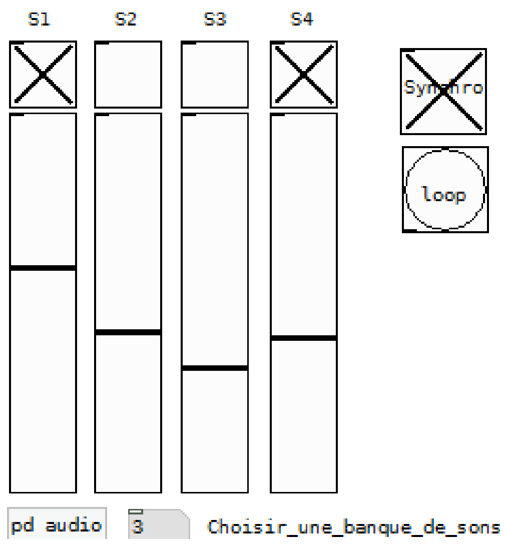
Loops were (for most of them) created from Pd patches available for download in the resource section. The app has been to designed to run on entry level smartphones, if you have a powerfull smartphone, a bigger screen etc. don't hesitate to modify the patch to add more and more loops. Audio files need to be at a samplerate of 22050hz to be played back.

You can either download the patches and play with it on you computer, use the RJDJ app or the PdDroidParty app considering your device.

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

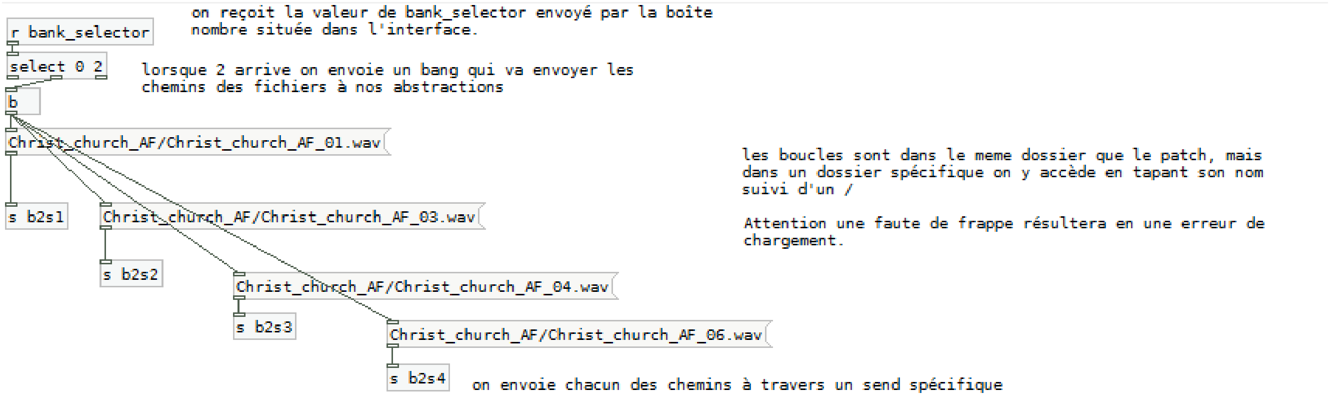
4.2- MAIN CONCEPTS

For further details, please refer to the comments included in the patch. The goal is not to understand everything but to have an idea how to deal with samples and keep them in synch.



Main window of the PdDroidParty patch

The principle is simple, we need to play a pack of pre-recorded audio loops and play them in synch; that is to say that if we want to launch the next loop we will need to wait for the playing one to loop (same principle as in Abbelton Live).



```
r bank_selector
select 0 2
b
Christ_church_AF/Christ_church_AF_01.wav
s b2s1
Christ_church_AF/Christ_church_AF_03.wav
s b2s2
Christ_church_AF/Christ_church_AF_04.wav
s b2s3
Christ_church_AF/Christ_church_AF_06.wav
s b2s4
```

on reçoit la valeur de bank_selector envoyé par la boîte nombre située dans l'interface.

lorsque 2 arrive on envoie un bang qui va envoyer les chemins des fichiers à nos abstractions

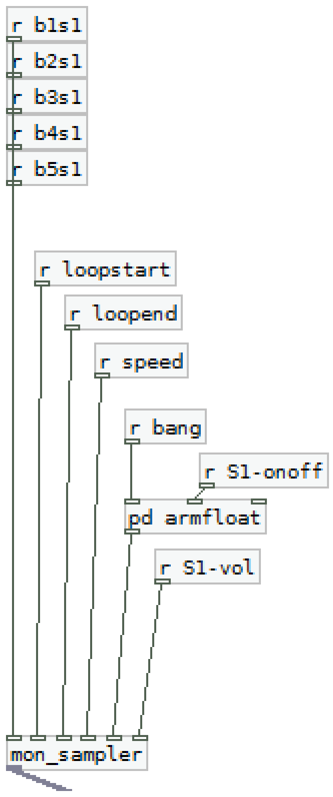
les boucles sont dans le meme dossier que le patch, mais dans un dossier spécifique on y accède en tapant son nom suivi d'un /

Attention une faute de frappe résultera en une erreur de chargement.

on envoie chacun des chemins à travers un send spécifique

Content of the subpatch which loads sounds

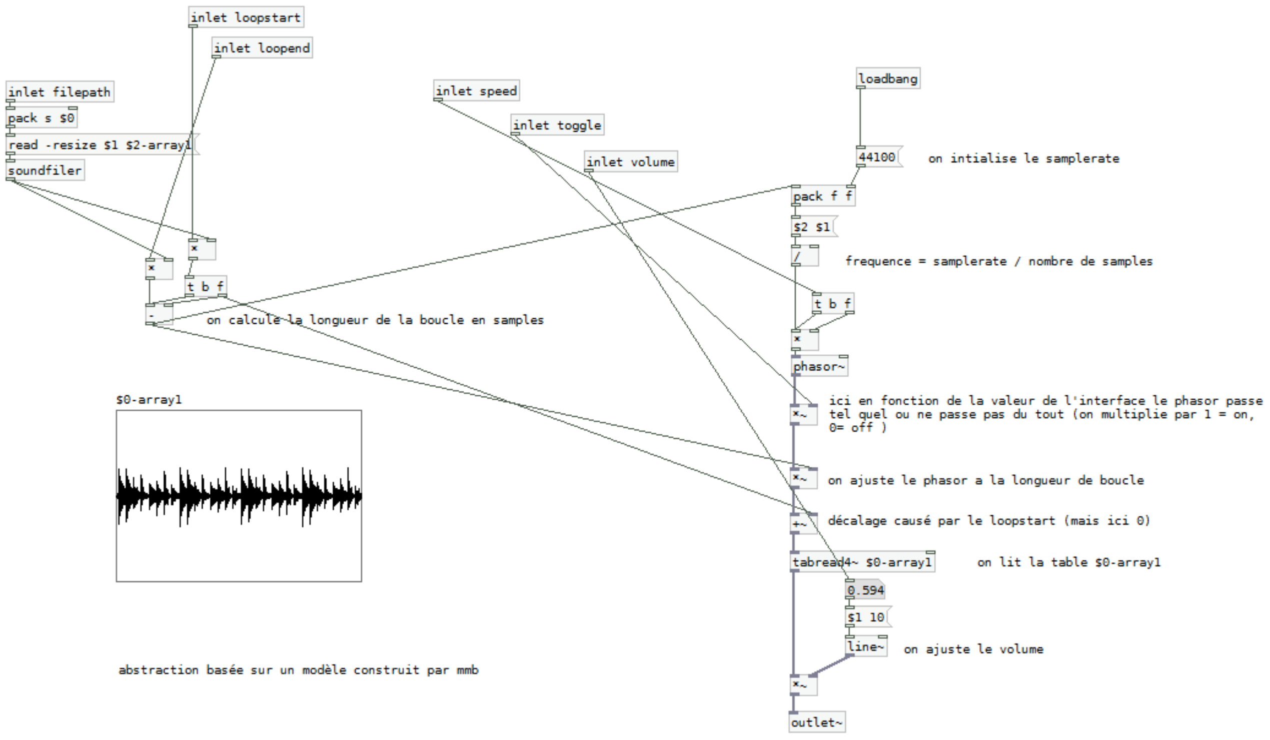
We will load a pack of sounds (3 for RJDJ and 4 for PdDroidParty) in tables using the [soundfiler]. To play back 4 sounds we will use 5 tables, as we will use a table to synch the playback of others, this control table will send a bang each time it loops that will be received by others. We'll use the toggle to “arm” the playback that will start only when this bang is received.



```
r b1s1
r b2s1
r b3s1
r b4s1
r b5s1
r loopstart
r loopend
r speed
r bang
r S1-onoff
pd armfloat
r S1-vol
mon_sampler
```

Messages received by the sampler abstraction

Each loop will use a sampler abstraction (it's simply another Pure Data patch being in the same folder as your main patch) this abstraction can be loaded by creating an object and naming it by its name. It will enable us to use the “$0-“ messaging system, which will be replaced by an unique identifier on creation. For instance a “$0-bang” will be replaced by “1088-bang” in an abstraction and by a “1055-bang” in another; it’s usefull to copy/paste chunck of codes without them interfering with one another


inlet loopstart
inlet loopend
inlet filepath
pack s $0
read -resize $1 $2-array1
soundfiler
inlet speed
inlet toggle
inlet volume
loadbang
44100
on intialise le samplerate
pack f f
$2 $1
/
frequence = samplerate / nombre de samples
t b f
×
phasor~
×
×
t b f
-
on calcule la longueur de la boucle en samples
$0-array1
ici en fonction de la valeur de l'interface le phasor passe tel quel ou ne passe pas du tout (on multiplie par 1 = on, 0= off)
×~
×~
on ajuste le phasor a la longueur de boucle
+~
décalage causé par le loopstart (mais ici 0)
tabread4~ $0-array1
on lit la table $0-array1
0.594
$1 10
line~
on ajuste le volume
×~
outlet~
abstraction basée sur un modèle construit par mmb

The content of the sampler abstraction.

Something important is also to have well prepared samples. They should be the exact same duration, and recorded (or converted) at a 22050Hz, in .wav format.

4.3- RESSOURCES

4.3.1- LINKS:

- http://www.freesound.org/ : a collaborative soundbank to find new sounds easily, to replace those used by default.
- http://www.musicradar.com/news/tech/sampleradar-13154-free-sample-downloads-217833/1 : free musical loops.
- http://radio.rumblesan.com/ : an online generative radio playing Pure Data patches. All patches are downloadable.

4.3.2- FILES:

- interface-help : is an help patch to understand how to get data from the touchscreen and accelerometer of your phone.

- Christchurch.pd : is a Pure Data composition. It’s a patch that you just have to open to hear it out. Check inside to understand how it’s done. This patch has been developped by Andy Farnell.

- Mengine-Ambiant-Std-Ed.zip : is also a musical patch. This one allows you to generate endless music according to sets of constraints. It’s based on a generalized probabilistic approach. For each beat, each note will be played according to probability sets. It’s possible to choose musical scales, harmonic structures, instruments that play, and set every probability one by one. It’s also possible to create preset for styles, don’t hesitate to contact me directly by mail if you have any question. (documentation is planed but not ready yet).

Note : for those 2 patches you may need to install Pd-extended.

3.2.4- Contributions

Don’t be shy and test away, hack it and send us your modifications. Change the sounds to put your own in. Another patch will be edited further to included the best contributions.

5-EPISODE 2 : WHAT ABOUT EFFECTS ?

Let's look at the microphone input from our smartphone, we will learn how to alter the audio coming through the mic in real time.

5.1- HOW DOES IT WORK ?

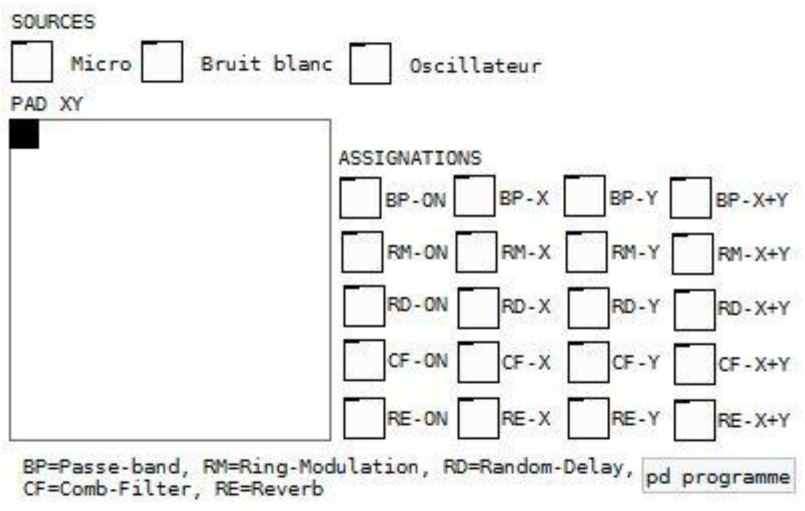
This month we will focus on audio effects. For each application you can choose between 3 sound sources (microphone input, noise generator and a basic oscillator), sound signals will then be routed through a chain of effect, then to the speaker (or headphones)

The RJDJ patch will route the signal through three effects : a band-pass filter, a shaper, and finally a variable delay network. Accelerometer values will be used to control some of the parameters : the x-axis will control the band-pass resonance factor, the y-axis the cut-off frequency, and the z-axis will control the depth of the shaper. You'll need to press the touch screen with 2 fingers to change the sound source. When the oscillator is selected you can change its frequency by moving your finger on the touchscreen.

Note : the patch for the iphone is slightly different. For some reason the patch as it is for android will not work on iphone, so another version is available for iphone users, the principle is the same. This iphone version is fully working on android though.

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

The PdDroidParty is a kind of Kaoss pad. It will enable you to use effects with a control surface (the [touch] abstraction by Chris). You'll just have to move your finger along the pad to modify the values sent. An assignation matrix is available, enabling you to toggle on and off effects and choose which value is sent to control this effect (x position value, y position , or x+y/2). The chain of effect is composed from top to bottom of a band-pass, a ring modulator, a random delay, a comb filter and finally a reverb.



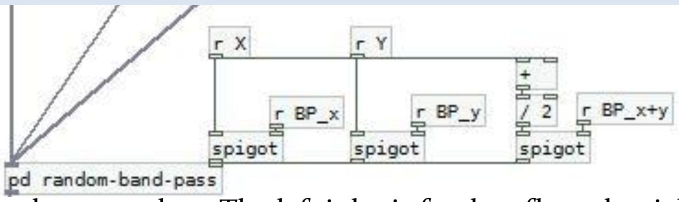
BP=Passe-band, RM=Ring-Modulation, RD=Random-Delay,
CF=Comb-Filter, RE=Reverb

Main window of XY_pad_droid

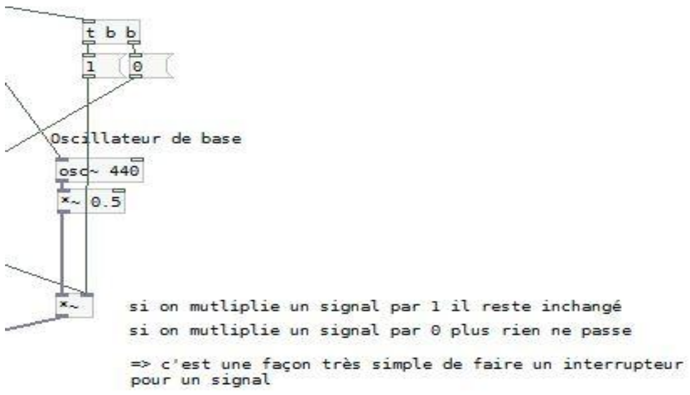
5.2- MAIN CONCEPTS

An effect is used to alter an audio stream going through it. In pd it can be seen as a subpatch or an abstraction using audio inlets [inlet~] and outlets [outlet~]. In between the audio signal will be modified using objects like [*~] to multiply an audio signal by another (or even a number), the couple of objects [delread~] et [delwrite~] to write the audio signal in a buffer to be read later on etc. Check the resources section, to learn more on the theorical aspects of the effects used in this chapter.

5.2.1- HOW TO PREVENT DATA FROM PASSING THROUGH ?



[spigot] will help you to block or pass data. The left inlet is for data flow, the right inlet receives à 0 or a 1 (0 blocks data, 1 lets the pass). A toggle object in the Gui send those messages for each effect. The data flow comes from the [touch] abstraction, created this way [touch sizeX sizeY XY] you receive messages by an object [r XY], then you just have to link this object to a [unpack f f] to get X and Y values separately.

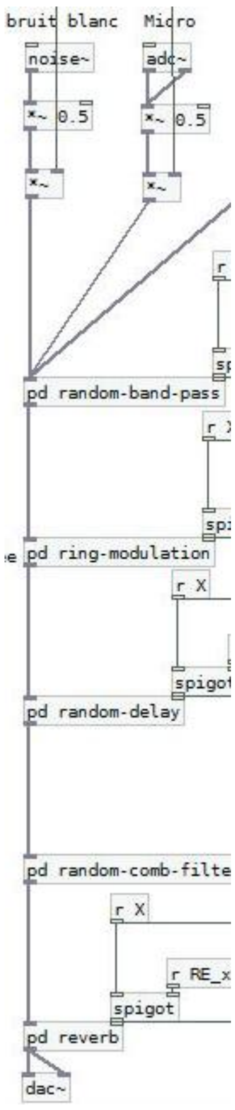


si on mutliplie un signal par 1 il reste inchangé
si on mutliplie un signal par 0 plus rien ne passe

=> c'est une façon très simple de faire un interrupteur pour un signal

A [spigot~] object exists to deal with audio signals. We'd rather use a different technique to do so : we'll use the [*~] object instead, in the left inlet comes an audio signal and the right inlet will receive a 0 or 1. If you multiply 0 you get nothing, if you multiply by 1 then you get the exact same message. This is also useful to control output volume of a patch, link a slider ranging from 0 to 1 to the left inlet, and you get a linear control of the volume moving your slide (we used this in the first episode to build the mini loop mixer).

5.2.2- AN EFFECT CHAIN

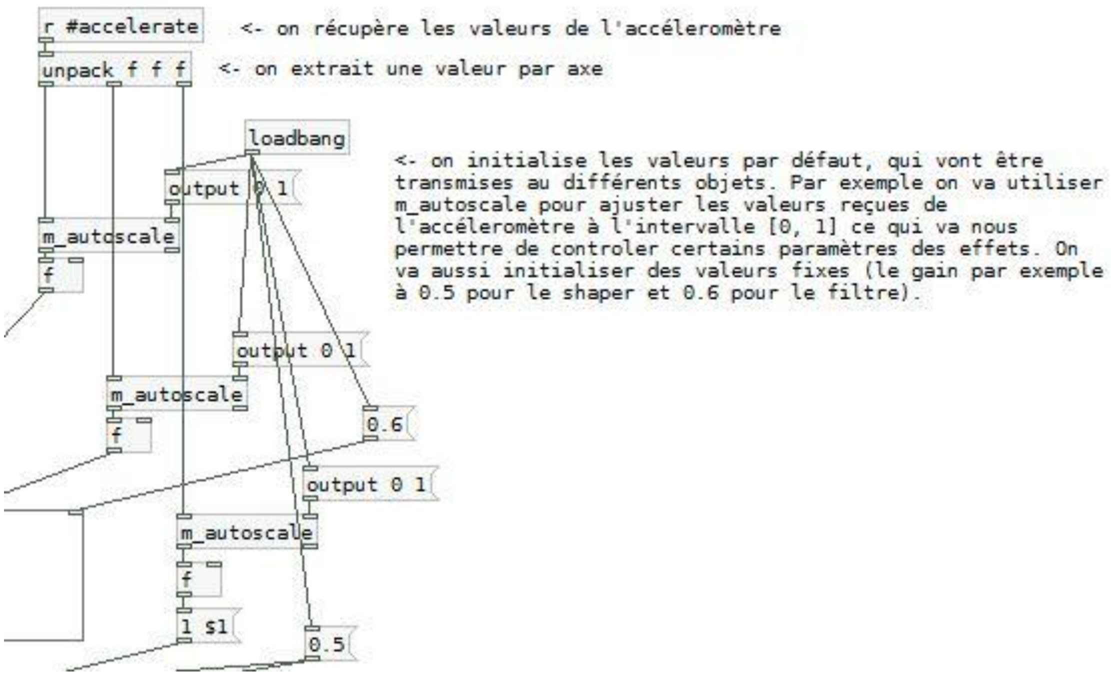


This is the complete effect chain from the XY-pad-droid, we receive audio signals at the top [noise~] and [adc~](=mic input) objects, those signals will go through each effect to the output.

ADC = analog digital converter.

DAC = digital analog converter.

5.2.3- SCALING DATA SENT FROM THE PHONE [R# TOUCH], [R# ACCELERATE] AND [M_AUTOSCALE]



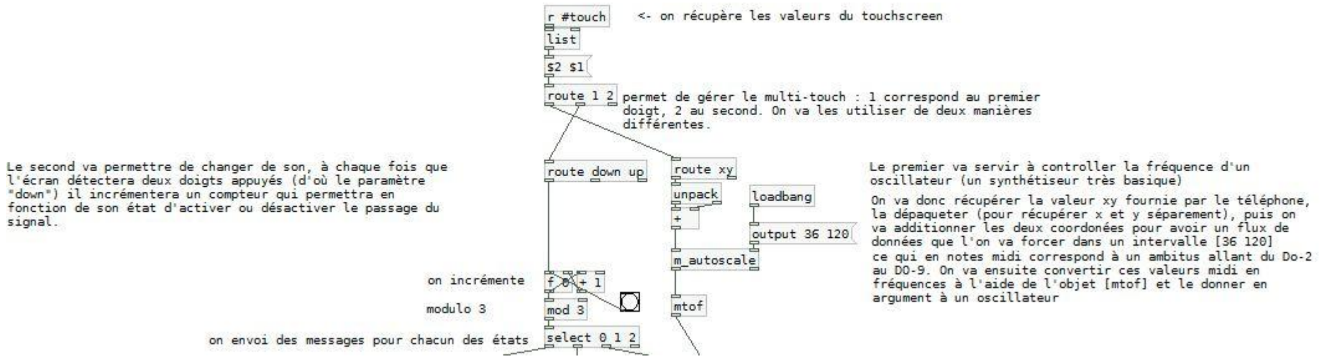
<- on récupère les valeurs de l'accéleromètre

<- on extrait une valeur par axe

<- on initialise les valeurs par défaut, qui vont être transmises au différents objets. Par exemple on va utiliser m_autoscale pour ajuster les valeurs reçues de l'accéleromètre à l'intervalle [0, 1] ce qui va nous permettre de controler certains paramètres des effets. On va aussi initialiser des valeurs fixes (le gain par exemple à 0.5 pour le shaper et 0.6 pour le filtre).

Receiving accelerometer values and scaling them

[m_autoscale] from the RJDJ set of abstractions, will enable you to scale any flow of values. The left input will receive the data flow (numbers) which will be rescaled to the [0,1] interval. This is set by a message [output 0 1(, you can scale to [0,127] by initializing the object with the following message [output 0 127(. You can also choose the kind of interpolation you want with a third argument [output 0 1 4(. 0 = sensitive to small movements, 1=linear, 2=less sensitive to small movements. See the help patch for more details.



<- on récupère les valeurs du touchscreen

permet de gérer le multi-touch : 1 correspond au premier doigt, 2 au second. On va les utiliser de deux manières différentes.

Le second va permettre de changer de son, à chaque fois que l'écran détectera deux doigts appuyés (d'où le paramètre "down") il incrémentera un compteur qui permettra en fonction de son état d'activer ou désactiver le passage du signal.

Le premier va servir à controller la fréquence d'un oscillateur (un synthétiseur très basique)

On va donc récupérer la valeur xy fournie par le téléphone, la dépaqueter (pour récupérer x et y séparement), puis on va additionner les deux coordonées pour avoir un flux de données que l'on va forcer dans un intervalle [36 120] ce qui en notes midi correspond à un ambitus allant du Do-2 au DO-9. On va ensuite convertir ces valeurs midi en fréquences à l'aide de l'objet [mtof] et le donner en argument à un oscillateur

on incrémente

modulo 3

on envoi des messages pour chacun des états

Receiving data from the touchscreen

You can receive data from the touchscreen with [r #touch] then use [route down up xy] according to what you want, finally you can use [route 1 2] to get first finger information or second finger ones.

In Droidparty things are a little different. We use the [touch] abstraction provided by Chris, and [toggles] from the pd gui elements. To make a gui element send its value you need to specify a "send-symbol", by right-clicking the object, selecting "properties" and specify the name the value will be sent to entering "mysend" in the send-symbol field will enable you to receive the value of the toggle (0 or 1) anywhere in your patch creating a simple [r mysend] object. For Chris' [touch] abstraction you need to create an object [touch 50 50 hello], this will create a surface of 50px X 50px which will send touch events (ie position of a finger in this surface) anywhere in the patch by creating a [r hello] object . Then you can use [unpack f f] to get x and y values.

5.3- RESSOURCES

A big collection of objects, including a number of effects, synths and other stuff :

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

A detailed explanation on how to implement a network of variable delays to create a texture (check the following index : 3.4.2.7 Texture)

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

Abstraction collection from the Virginia Tech laptop orchestra (all vanilla !)

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

Some theory : (article on the french wikipedia, it won't be to hard to find the corresponding ones in english)

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)

If you are brave enough check this one out, it's a detailed article on how to build a reverb with Pure Data presented at the PdCon 2011 at the Bauhaus university of Weimar :

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

A bonus scene to download, LoopIt:

This is a basic loop on loop application. Touch your screen and keep pressed to record a loop, lift your finger when your done. If the recording is ok, record another if not shake to erase and start again. At the looping point a small red square is shown on the main screen.

A Kaoss pad for WebPD :

Similar to the PdDrodParty patch, but using an HTML interface.

6-EPISODE 3 : SYNTHESIZERS

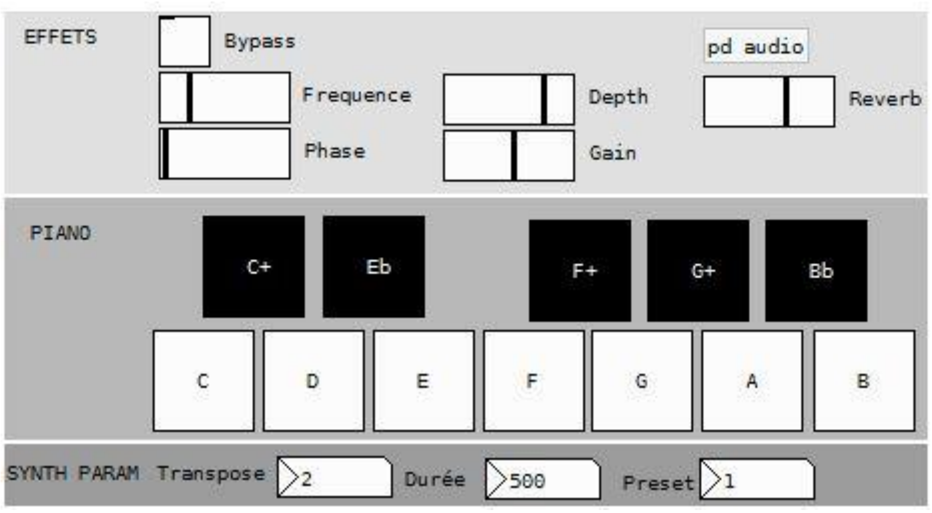
Sound synthesis ... For this third episode we will create two apps. First the RjDj patch will help us build a melody according to a scale, being able to switch between scales and sounds. The DroidParty example will be a mini-piano.

6.1- HOW DOES IT WORK ?

For this episode will learn a bit about synthesizers. As usual two patches are presented.

The RJDJ patch is a sequencer withour interface, you just need to press the touchscreen to build a melody a step at a time. Press with two fingers to change the sound produced, shake to change the musical scales that are played. Some effects are also mapped to accelerometer movements. Both synthesizers available are from Andy Farnell patches.

The PdDroidParty patch is a small piano with a GUI, some effects are incorporated to the controls, you can also choose the transposition factor, the note duration, and you can test out the nine sound presets that are available. This synthesizer is adapted from a patch by Tom Erbe.



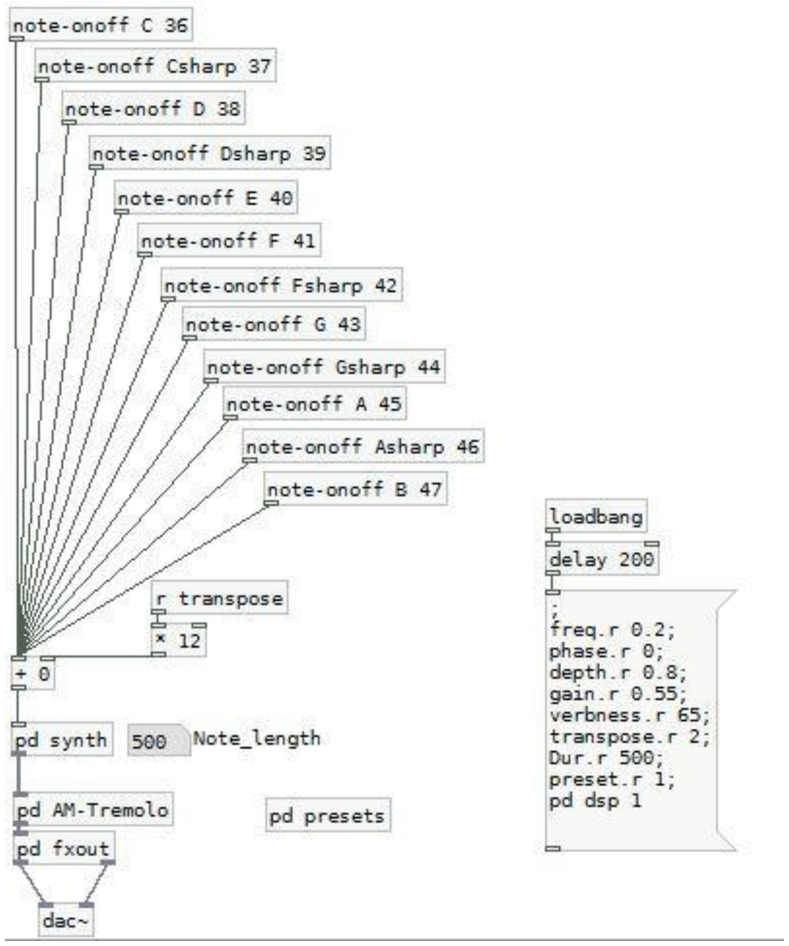
PdDroidParty patch interface.

You can find more information about Andy Farnell and Tom Erbe in the ressource section.

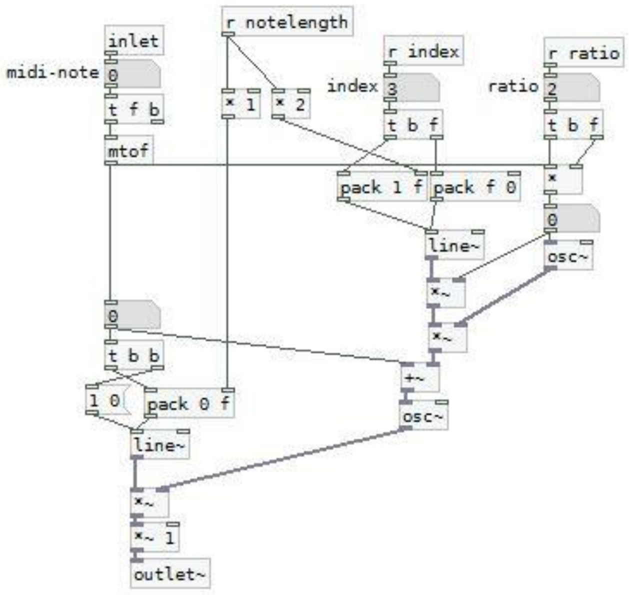
http://rjdj.me/sharescene/tapititap-3/

6.2- MAIN CONCEPTS

Appart from sound synthesis several important points will be explained there (as a synthetizer needs to reveive informations to play). Please refere to comments inside the patches and to the ressources section.

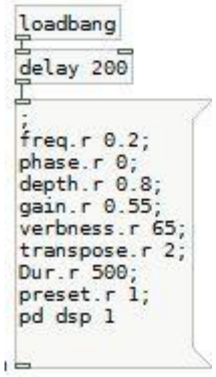


Guts of the PdDroidParty patch



FM audio engine

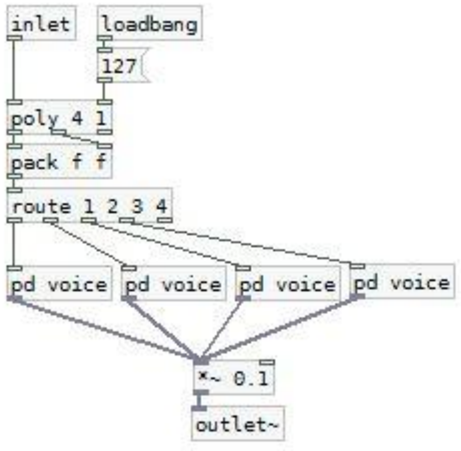
6.2.1- INITIALIZING PARAMETERS



```
loadbang
delay 200
;
freq.r 0.2;
phase.r 0;
depth.r 0.8;
gain.r 0.55;
verbness.r 65;
transpose.r 2;
Dur.r 500;
preset.r 1;
pd dsp 1
```

[loadbang] will send a bang on the startup of the patch, it's usefull to initiate a lot of parameter such as synth parameters, or effects values etc.

6.2.2- POLYPHONY



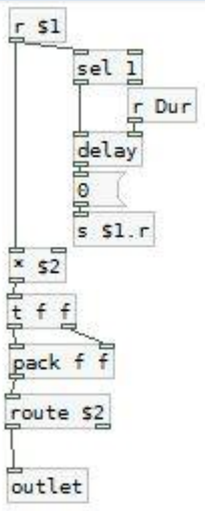
Polyphony can be achieved by various ways, it's much more elegant with dynamic patching, but we will use a simple way of doing it with the [poly] object. [poly] uses two arguments, the first one will indicate the number of voices you want the second one is the voice-stealing argument (that is to say, if all your voices are busy, it will "steal"one to output the new incoming value). [poly] receives a couple of value (note,velocity) respectively left and right inlet (beware, if velocity is 0 it will not output anything, and also the order of argument has it's importance it should receive velocity first), and will return a triplet (voice_id, note, velocity). Then using [pack f f f] and [route 1 2 3 4] (because we 4 for voices : [poly 4 1]) we will be able to send (note,velocity) couples to voices abstractions or subpatches. In this example we leave the velocity values aside, as the touchscreen has no pressure informations. Furthermore our synth are initialized with a default velocity. An important point is that [route] will get rid of the first number of the list to pass it on : if it receives a message [1 60 127(the ouput of the first outlet will be [60 127(.

6.2.3- ABSTRACTIONS ET SUBPATCHES

As we already discussed it a litle an abstraction is a Pure Data patch, saved in the same folder as the main patch it can be created by just creating an object with the very same name. It's the case of the [note-onff] patch (shown in the first snapshot of those explanations). This is useful when we use the same bit of code several times in one patch, if you modify save an abstraction changes will apply to all the places it is used. We can also use arguments when developing abstractions, see the next paragraph for that.

A subpatch is useful to make the code more readable and compact. You just need to create called « pd my_subpatch », a new blank page will popup, you just need to write your code in it, creating inlets and outlets or send and receive will help you to communicate with the rest of your code. You can also just copy/paste a bit of code in it.

6.2.4- USE OF ARGUMENTS IN ABSTRACTIONS AND SUBPATCHES ($0, $1, $2, …)



Use of arguments in the [note-onoff] abstraction

The $0 argument is a bit of a special one. It is used INSIDE abstractions to help us deal with messages inside the abstraction, so that a message sent inside one abstraction stays in it, we will then use $0- as a prefix to those messages. When loading the abstraction $0 will be replaced by a random number (not-redundant), that is to say that one abstraction will then use a "1088-bang" whether the next one will use a "1158-bang". In the first episode we use this in the [mon_sampleur] abstraction, in the example above we receive [r Dur] it will the same value (coming from the Gui) for all copies of this abstraction.

Using $1, $2, $3 etc is different. It enables you to specify creation arguments. Above in the [note-onoff] abstraction we use $1 and $2. If you create un object called [note-onff C 60], $1 will be replaced by "C" and $2 by "60". In this case $1 will be used to specify messages names to communicate with the GUI : a 1 will be received when you'll toggle the toggle labeled "C", after a delay specified by "Dur"a 0 will be sent back to it to reinitiate it using "C.r". The same goes for $2 which will be replaced by "60".

6.3 RESSOUCES

6.3.1 LINKS

Great examples of sound design with Pd (a truly amazing and interesting ressource) : http://obiwannabe.co.uk/tutorials/html/tutorials_main.html

There's a full book associated with those, it's worth the price : http://aspress.co.uk/ds/about_book.php

Frequency modulation (see the attached patch too) : http://fr.wikipedia.org/wiki/Modulation_de_fr%C3%A9quence

Amplitude modulation (see attached patch too) : http://fr.wikipedia.org/wiki/Modulation_d%27amplitude

Some educational patches made by Tom Erbe : http://musicweb.ucsd.edu/~tre/wordpress/

The soundhack page (externals developed by Tom Erbe as DAW plugins at first the ported to Pure Data (at the bottom of the page on the right are the Pure Data externals) : http://soundhack.henfast.com/freeware/

My personal collection of abstractions, found through researches over the web, it contains synths, effects and more : http://code.google.com/p/pdlive/

6.3.2 FILES TO DOWNLOAD

Now that we know how to use effects and synth we can build a vocoder ! Here are to bonus scenes to study (or to have fun)

Talkbox.rj : « I am a robot » (you need a somehow powerfull machine to run it ; for instance HTC Legend will a bit ligth for this, but it works fine on an iphone 3G. This patch is also a good example to see the latency issue with android compared to iOS (see the note in 3.1.3- Android vs iOS)

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

Vocoder.rj : is an example vocoder put together by the RJDJ team enabling to play differents chords according to device orientation and mix it with you voice.

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

Pure Data implementation of FM, AM and waveshapping synthesis techniques.

For the brave, a full article on sound synthesis.

7- EPISODE 4 : LES SEQUENCEURS

In this episode we will start building more evolved programs, using graphical interfaces, and those applications will be able to create real musical content. The sequencing concept is very important to electronic music as it will help you to create instruments that will share the same time line.

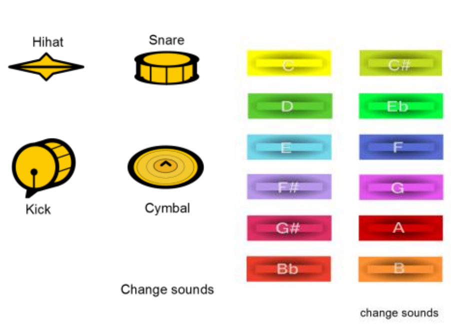
7.1- HOW DOES IT WORK ?

So what is a sequencer ? A sequencer is a tool to help you synch events between one or more instruments. Simply put a sequencer will cycle through a certain number of steps (ie count from 0 to 15 for a 16-step sequencer), each step you will be able to send one or several values to other parts of your program. If several sequencer are used they can count in synch if they are linked to a common metronome ([metro] object in pd)

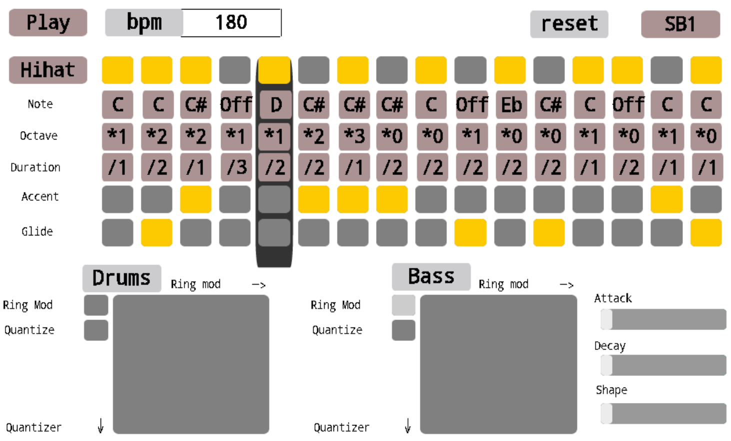
Both applications proposed here, will synthesize what we've been discovering in the first three episodes: we will play audio files, use synthesizers, and effects, all together to have a glimpse of what an autonomous application on a smartphone can do.

The RjDj app for iphone only, will use a 3 page set up. A first page will be used to sequence drum sounds, a second one will be used to play synthetized leads, the third page will enable us to play bass sounds. A metronome is running behind the scenes and will synch all those instruments. Shake to clear sequences !

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



The DroidParty application is a groove box : drum sequencer with an acid bass instrument. This tool will enable you to build a real rhythmic base for an electro track. After the recent updates on the DroidParty project, we can now use svg templates to improve the visual aspects of our programs. How is that for a change ?

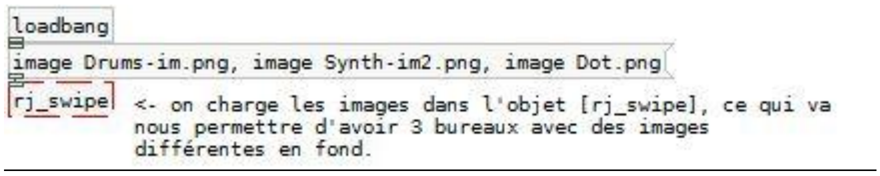


7.2- MAIN CONCEPTS

MULTI-TAP.RJ

7.2.1- USING [RJ_SWIPE] (MULTI-TAP)

This one is for iphone only as it uses a specific pagination system, that isn’t included in the ScenePlayer app for android (you will be able to dow,load each page seperatly for android looking in the ressources section). Using [rj_swipe] will enable us to implement several pages having their own interactions and their background image.

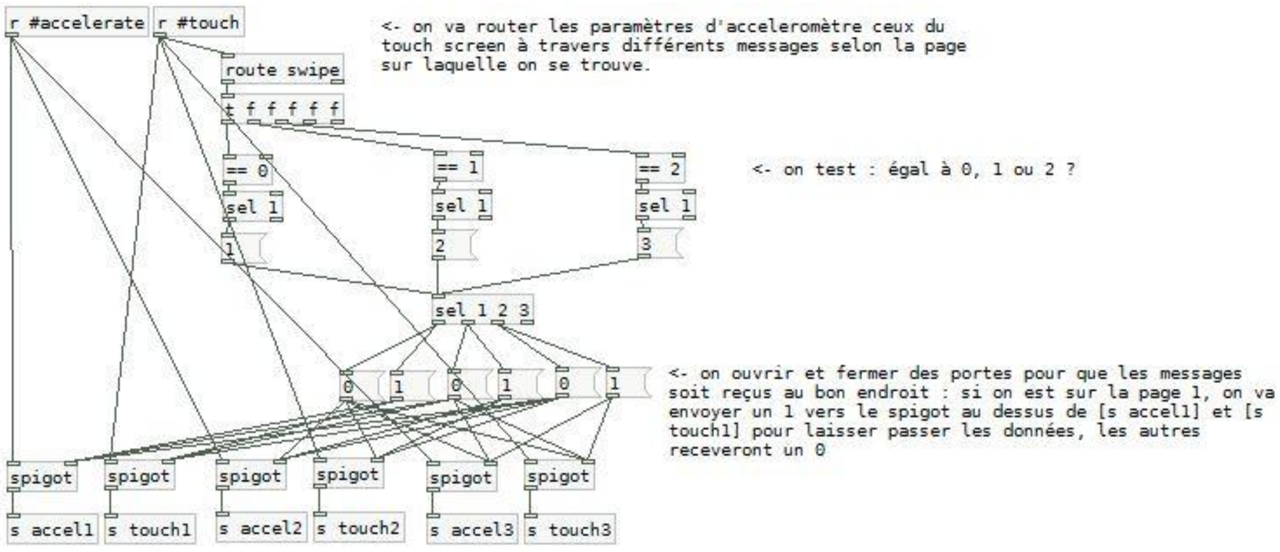


<- on charge les images dans l'objet [rj_swipe], ce qui va nous permettre d'avoir 3 bureaux avec des images différentes en fond.

Use of the [rj_swipe] object

Those three object will be enough to declare our three pages with three different images in the png format.

After that will have to build a system to route the data from the touchscreen and accelerometer to the right part of the program according to the page we are on. This is a basic but working implementation.



<- on va router les paramètres d'acceleromètre ceux du touch screen à travers différents messages selon la page sur laquelle on se trouve.

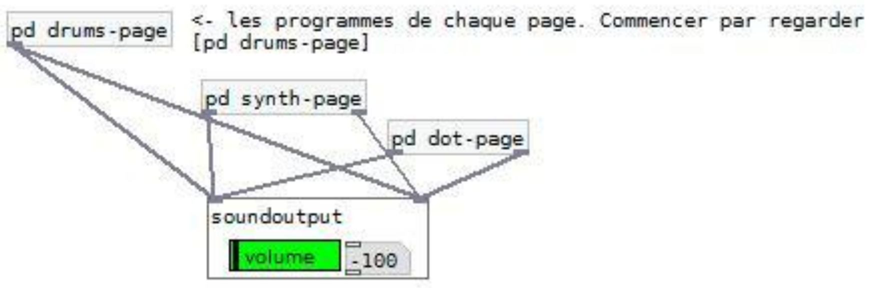
<- on test : égal à 0, 1 ou 2 ?

<- on ouvrir et fermer des portes pour que les messages soit reçus au bon endroit : si on est sur la page 1, on va envoyer un 1 vers le spigot au dessus de [s accel1] et [s touch1] pour laisser passer les données, les autres receveront un 0

Routing of data according to the page we are on.

With this snippet of code data is then sent to a specific name. If we are on the first page ([==0] is true) so we enable data to pass through spigot to [s accel1] and [s touch1].

Each page will contain un program based on sequencers. The first and the second page will use [c_taptap] that we have already studied last chapter. The third page is an adaptation of Chris McCormick CanOfBeats, so I’ll let you check the original to see how it’s been adapted. (This is a good exercice)



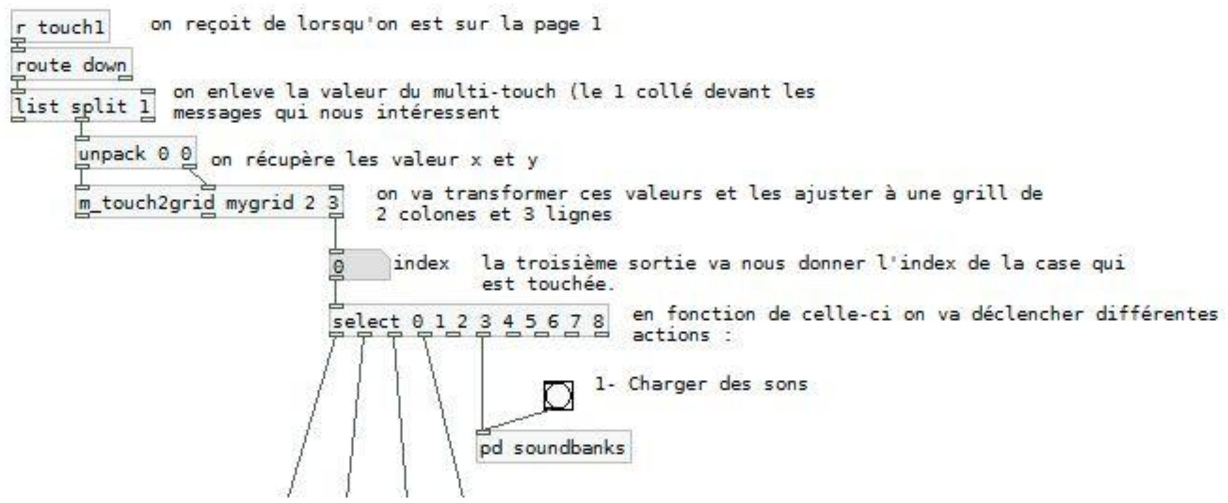
<- les programmes de chaque page. Commencer par regarder [pd drums-page]

The three programs for each page contained in three sub-patches for more clarity

The [pd synth-page] subject is the same as the third chapter, I just added the possibility to shake to put the sequencer back to zero, and changed the synthesizers available. As we build an image for the background the interaction implemented will use [m_touchgrid], so let's have a look at this.

7.2.2- THE [M_TOUCH2GRID] ABSTRACTION (MULTI-TAP)

We will look at the implementation made inside the [pd drum-page] subpatch. To get data we will use the [m_touch2grid] object from the RjDj library. It will help us to map our screen to a lower resolution grid; that is to say it will spilt the screen in as much subdivision that you want by specifying a number of rows and columns… [m_touch2grid] will output at its right most outlet the index of the area that was touched.



on reçoit de lorsqu'on est sur la page 1

on enleve la valeur du multi-touch (le 1 collé devant les messages qui nous intéressent

on récupère les valeur x et y

on va transformer ces valeurs et les ajuster à une grill de 2 colones et 3 lignes

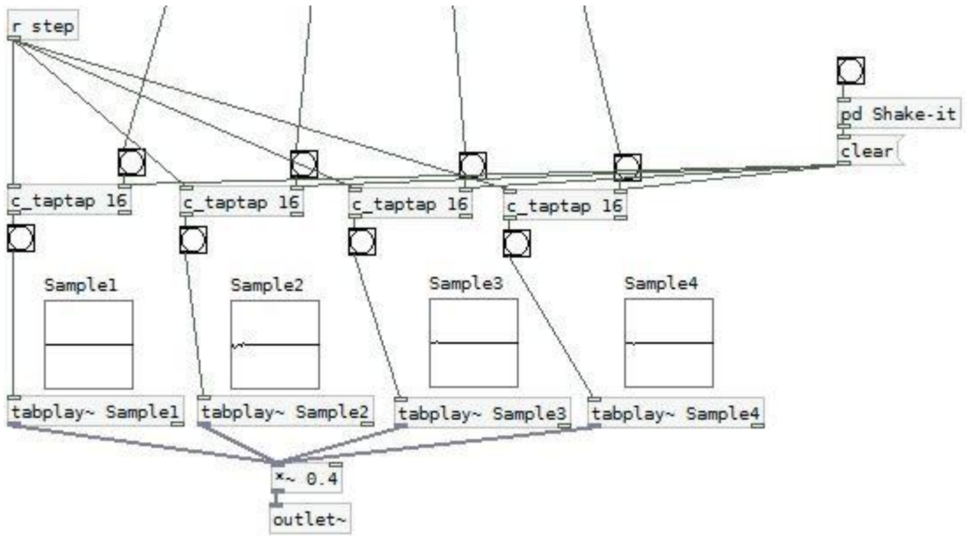
la troisième sortie va nous donner l'index de la case qui est touchée.

en fonction de celle-ci on va déclencher différentes actions :

1- Charger des sons

Use of the [m_touch2grid] object

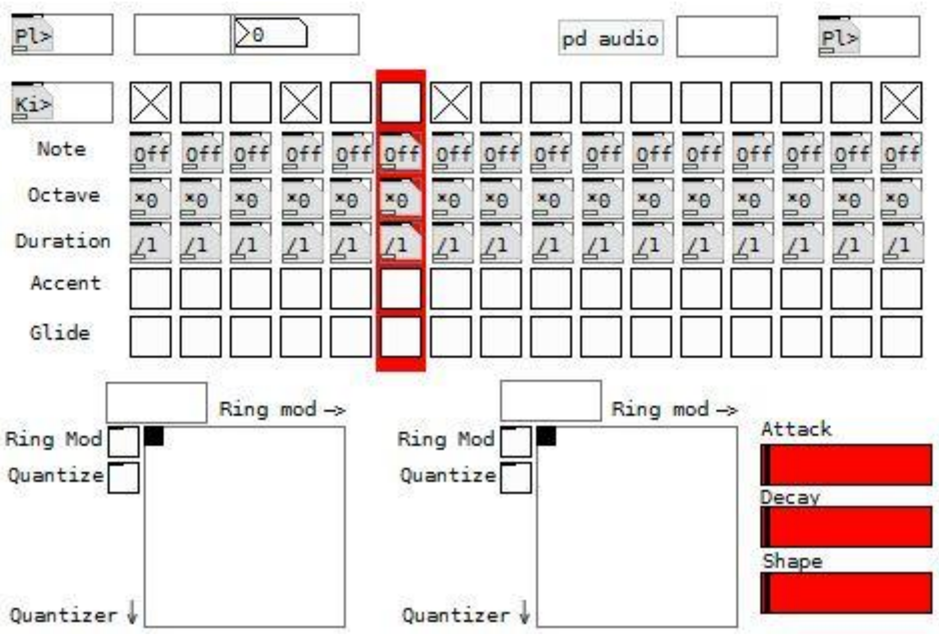
In this example we have defined six areas on the screnn (2 columns * 3 lines); the index zero matches to the upper left corner of the screen, one is the upper right corner and so on. In this example the indexes 0, 1, 2, 3 will be used to fill sequencers to play drum samples et the 5th area will help us to change between the available set of sounds. Once it's done you will have to create an image with the corresponding buttons, and you'll have a simple interface to control your program.



Use of [c_taptap] to read audio samples.

Note that the [r step] object is the count from the main metronome that will drive each page of the program.

GROOVE-DROID



Here's the raw DroidParty patch as it appears in Pure Data on a desktop machine.

The button of the upper left corner is used to start a metronome at a chosen speed (through the number box next to it). On the upper right corner the button will enable you to cycle through the soundbankx available for drum sounds.

The line below is our first sequencer that will be used to trigger drums. The button on the left will help you to choose which voice you want to edit between the four available (kick, hihat, snare, cymbal), if you hit the button the voice displayed will change so that you can now edit it. Each toggle represent a step, and there are 16 of them so it is a 16-step sequencer.

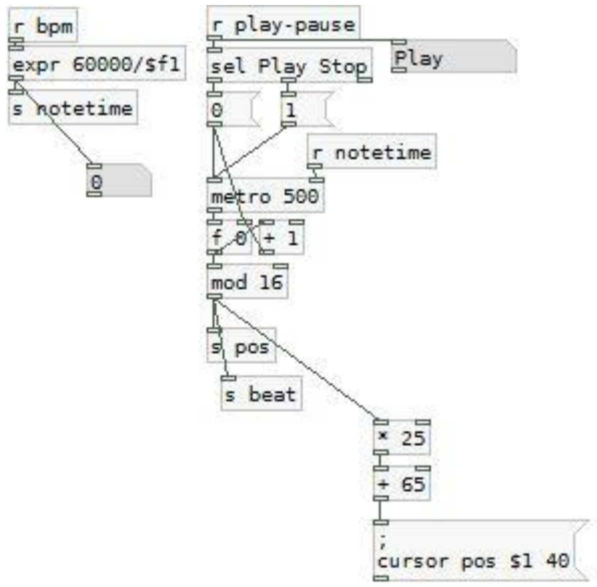
The group of five lines controls a bass synthesizer. The first one will control the note that is played (C, Csharp, D ...), the second one will adjust the octave of this note (0 bass, 4 high), the third line will let you choose the duration of the note for instance "/2" will play a half note (the length is computed from the tempo you set earlier), the fourth line will enable you to accentuate a note, and the fifth will create a glissando between the previous note and this one.

Right below there is to 'touch' surfaces to apply effects on both the drums and the bass synth. On the right there are three sliders to help you change the sound of the synth.

7.2.3- A DRUM SEQUENCER ?

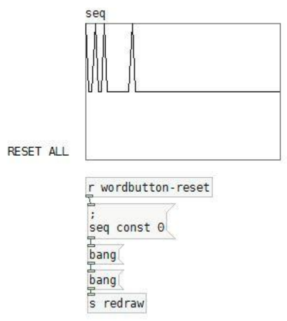
A sequencer is in most case a series of instructions that will be played in synch with a metronome. A sequencer has a number of step a 16 step sequencer will count from 0 to 15, so you will have sixteen steps to edit synthesis parameters for a given sound.

In pd terms we will be have to build a line of sixteen controllers (ie toggles, sliders or whatever...) to input values, we will then need an object to store each step's value (usually the right inlet of a [f]object), this value will then be output only when requested by a the combination of a counter (group of object counting from 0 to 15 at a specified tempo) and the object [select 0 1 2 ... 15]. So we only have to link each output of the [sel] object to the left inlet of the [f] corresponding to its step and the bang output by [sel] will pass on the value to the output of [f].



Here's an example of the counter built for this application. The message in the lower right hand corner will allow us to move the red canvas in the main interface.

In the drum sequencer example we will complicate things a little. As we will store values into an indexed table. This will help us to use less space on our display to control the drumming part. We will only have one line of toggles and a button will enable us to cycle between four voices (potentially a lot more). So every time we press the button we will be able to edit another voice.



The table used to store drum sequencing events.

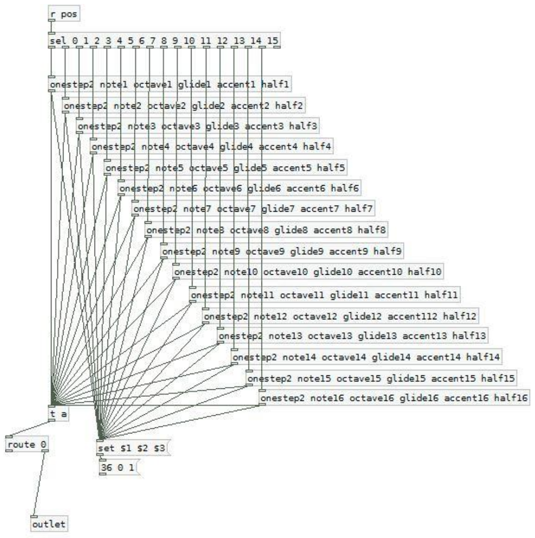
To read the table we will have to use several objects, to read from the different indexes. If the retrieved value is zero we will not read the sample if it's one we will. This method is a little too advanced for this introduction but it was worth mentioning considering the size of the screens we work on. I will not extend on the subject but check out the code to know more. You will learn more about 'classic-sequencer' in the next paragraph.

7.2.4- WHAT IF I WANT TO CONTROL A SYNTHESIZER?

For this example will use the same counter but we will collect a lot more parameters than just an order to play or stay quiet : a full note will be composed of a pitch, a velocity, and duration (those are three values used in the MIDI protocol) but we will add an accent value (0 or 1 wether the not will be louder or not) and a glissando value (0 no glissando or 1 glissando over the specified duration).

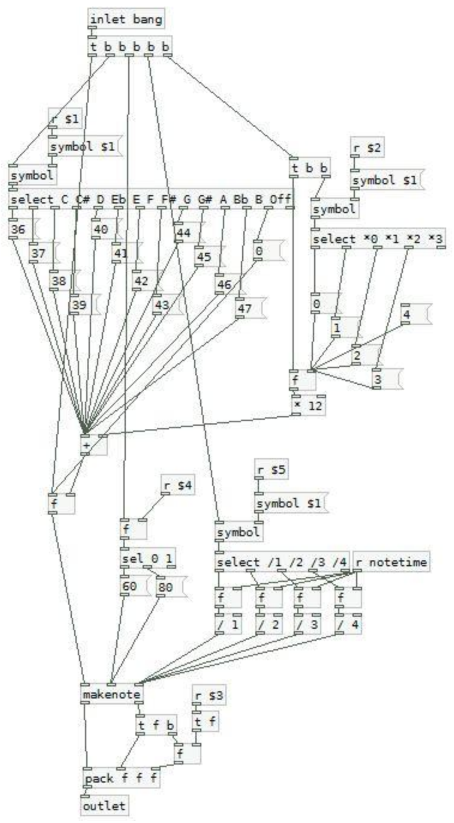
We will receive the 'pos' value sent by the counter, and we will "bang" an abstraction which will conform every message coming from the interface to build a new one that our synth will be able to understand.

So the principle is classic, as explained, an abstraction will catch all messages from the interface using floats (=[f]) for a given step. When it's "banged" it will output all those values in message.


r pos
sel 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
onestep2 note1 octave1 glide1 accent1 half1
onestep2 note2 octave2 glide2 accent2 half2
onestep2 note3 octave3 glide3 accent3 half3
onestep2 note4 octave4 glide4 accent4 half4
onestep2 note5 octave5 glide5 accent5 half5
onestep2 note6 octave6 glide6 accent6 half6
onestep2 note7 octave7 glide7 accent7 half7
onestep2 note8 octave8 glide8 accent8 half8
onestep2 note9 octave9 glide9 accent9 half9
onestep2 note10 octave10 glide10 accent10 half10
onestep2 note11 octave11 glide11 accent11 half11
onestep2 note12 octave12 glide12 accent112 half12
onestep2 note13 octave13 glide13 accent13 half13
onestep2 note14 octave14 glide14 accent14 half14
onestep2 note15 octave15 glide15 accent15 half15
onestep2 note16 octave16 glide16 accent16 half16
t a
route 0
set $1 $2 $3
36 0 1
outlet

The synth sequencer : outputting the midi pitch, velocity and duration plus a 0 or a 1 to indicate glissando

This abstraction will also force the right order of operation using [trigger] or [t]. For instance when using [makenote] we have to give the value in order from right to left. It's imperative to finish by giving the note argument in last for the object to work properly. But we will learn more about that in the next episode.



This is the abstraction [onestep] it stores every value from the interface to conform them to standard being able to control the synth.

It's important to notice how the data coming from the [taplist] abstractions is retrieved using the [symbol] object to convert the type of data we are receiving. Beware not to use '-' and '+' symbols as those will be recognized as indicators to truncate the message.

7.2.5- THE SVG SUPPORT

Chris is explaining everything very well in the page dedicated to DroidParty. So in short, the simplest solution would be to use the .svg files packed with the program and edit them with any software you want (Inkscape did the trick for me, free and open source). The link between the object and the svg file is made through the 'send-name' attributed to this object in pd (right-click -> properties).

For instance if you create a vertical slider, you specify its send-name to "sliderX", the corresponding svg file will need to have a name like "Slider-vertical-sliderX.svg". This is also right for [taplist], [wordbutton], [touch] as well as for toggles, sliders. You cans also change the background of your app using background.svg.

7.3- RESSOURCES

7.3.1 DES LIENS

Wikipedia article on sequencers

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

Abstractions and musical objects :

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

Visual Tracker a big sequencer for 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 MORE PATCHS :

Chris's drum machine (first program ever that used the droidparty engine) :

Simple sequencer (simplified version of groove_droid) :

Shake-drums.rj : Shake to play drums !

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

mutli-tap.rj in three scenes, this time compatible with android scene player

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

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

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

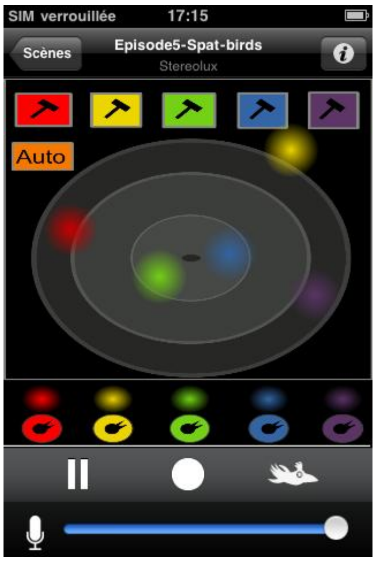
8-EPISODE 5 : SPATIALISATION, AND AUTO-GÉNÉRATION

In this episode we will build an audio interactive application using auto generation and spatialization methods.

7.1- HOW DOES IT WORK ?

8.1.1- SPAT-BIRDS.RJ

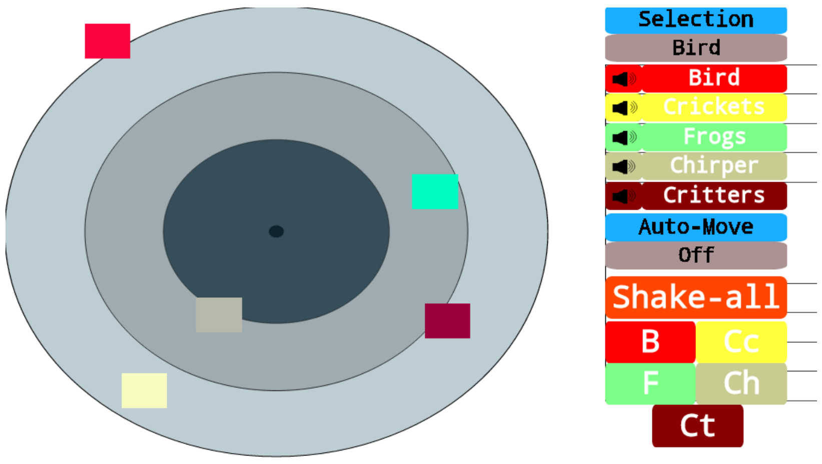
We will build a bird-cage, but of course those will be electronic from the FM species. So we will have five birds, we will be able to place them in the sound field, hit them on the head with hammer so that they sing and fly away. You can also shake their cage so that they all start flying away in all direction, or you can click on 'Auto' to let them do whatever they want.



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

8.1.2- SPAT-INSECTS-DROID

The android application is almost the same a few differencies are due to the way I chose to adapt the interface. The audio synthesis part will be more CPU consuming as we will extend the sound possibilities to several insects. We will control one bird (with several voices), crickets, frogs, critters and circadas. The interactions will be the same, but we will select which bird to move with a cycling button.

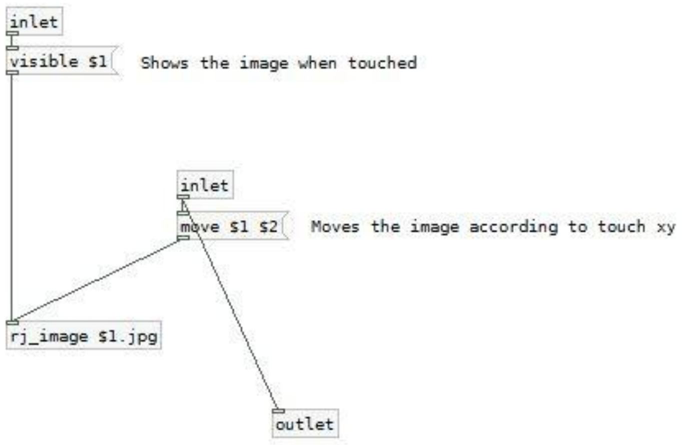


8.2- MAIN CONCEPTS

Once again please refer to the comment in the code to have further information . In this section we will focus on how to use images and how to move them across the screen aswell as randomizing some parameters of our program.

8.2.1- IMAGES IN SPAT-BIRDS.RJ

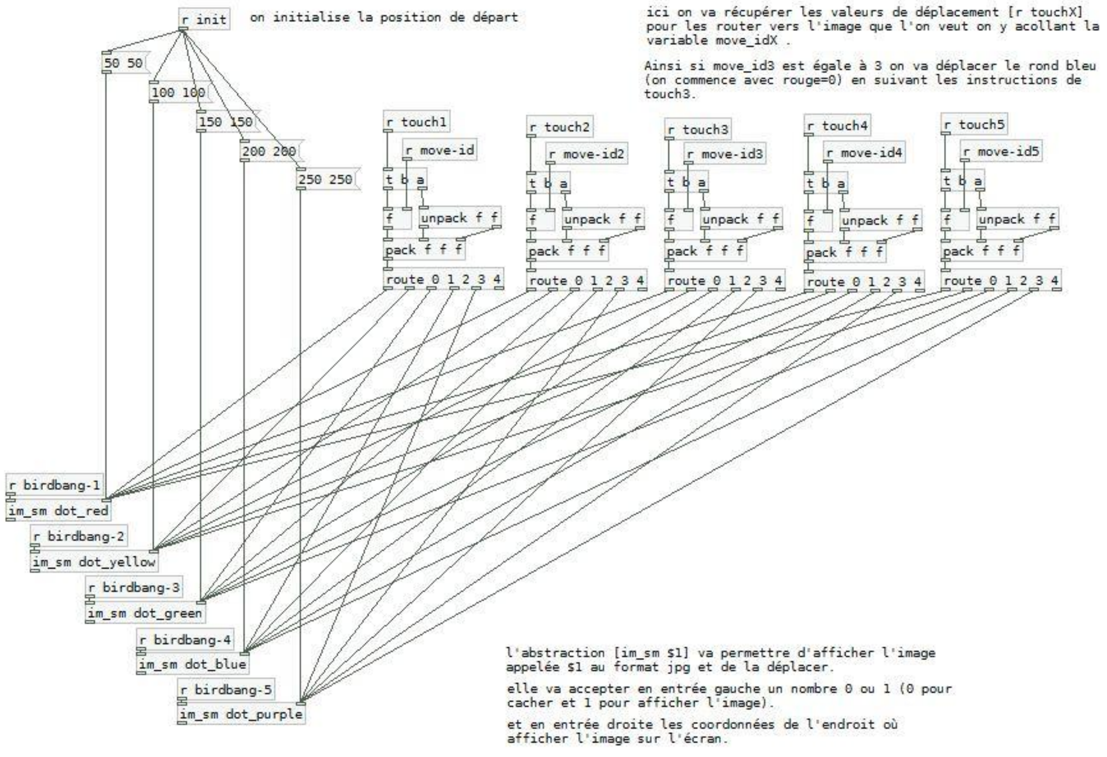
I've built an abstraction to deal with images it's called [im_sim] and you'll find it in the code folder. It's a very simple abstraction that will enable us to display or hide an image (with an variable : 0 or 1 in the left inlet) and to move it across the touch screen (by inputting two values in pixels in the right inlet). This abstraction uses as an argument for its creation the name of the file you want to display : if you create an object [im_sim red_dot] the program will load the file red_dot.jpg that should be in the same folder as the patch.



```
inlet
visible $1      Shows the image when touched
inlet
move $1 $2      Moves the image according to touch xy
rj_image $1.jpg
outlet
```

The code of the abstraction (two inlets, and one argument ($1) to set up the image to load)

Here's the abstraction inserted in the code :


on initialise la position de départ

ici on va récupérer les valeurs de déplacement [r touchX]
pour les router vers l'image que l'on veut on y acollant la
variable move_idX .

Ainsi si move_id3 est égale à 3 on va déplacer le rond bleu
(on commence avec rouge=0) en suivant les instructions de
touch3.

l'abstraction [im_sm $1] va permettre d'afficher l'image
appelée $1 au format jpg et de la déplacer.

elle va accepter en entrée gauche un nombre 0 ou 1 (0 pour
cacher et 1 pour afficher l'image).

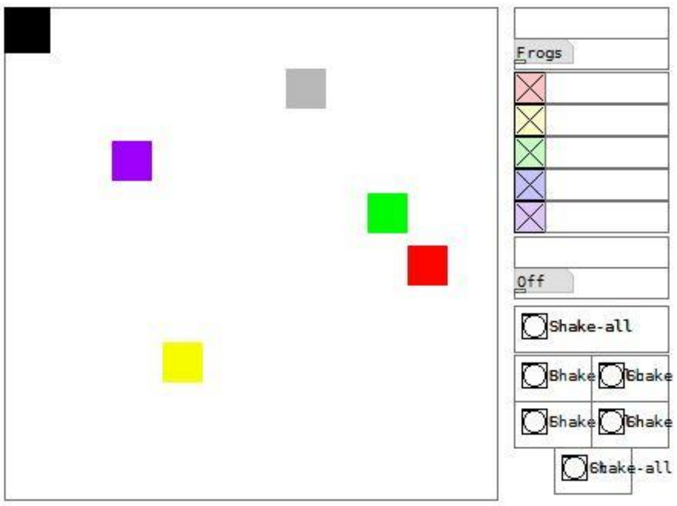
et en entrée droite les coordonnées de l'endroit où
afficher l'image sur l'écran.

In this example we have five images (five colored dots), each can be displayed or hidden, each can be moved by specific messages, and they have initial position on startup (fixed by the five messages in the upper left corner).

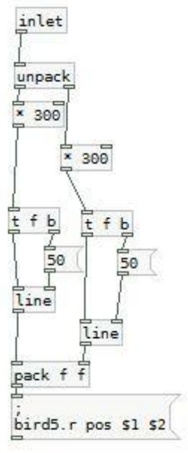
Those are the only part of the interface that are interactive, other buttons and elements are a drawn on a still image mapped to a grid with [m_touch2grid] as in the previous episode.

8.2.2- THE DISPLAY IN SPAT-INSECTS-DROID

First a snapshot of the patch as is on a desktop machine. As before we will use the svg support to customize the looks under android.



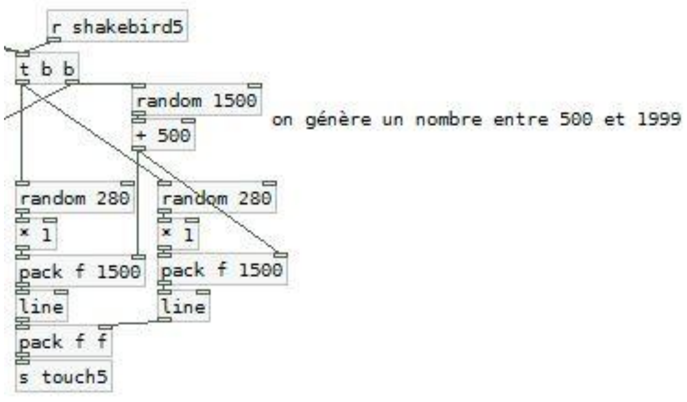
Firstofall it's important to understand that we will not work with something like an image As the engine works for now we have to map an image to a gui object in pure data. After sometime trying to move the objects available I can safely state that at the time most of the object cannot be moved under android (this function of pd hasn't been ported for droidparty yet). We will only depend on the [canvas] object for this kind of interaction (as we have successfully moved one in the 4th episode). We will use the same code : we use a [*300] object to adjust the output of a [touch] abstraction to the size of our surface.



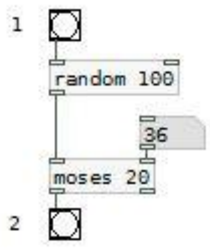
The canvas doesn't support yet the customization of several objects with several svg files. We will have to keep them as rendered by PdDroidParty.

8.2.3- USING [RANDOM]: TOWARDS PROBABILITIES

[random] is an object that will allow us to generate pseudo-random numbers. If you create a [random N] object, it will generate a number between 0 and N-1. You can then use several mathematical operations afterwards to force the output to be mapped to a specific interval. In the example below we will generate a time value between 0 and 1999ms and to position values between 0 and 279 with the help of a [line] we will then be able to create a movement from the previous position to a new one over a specified time.



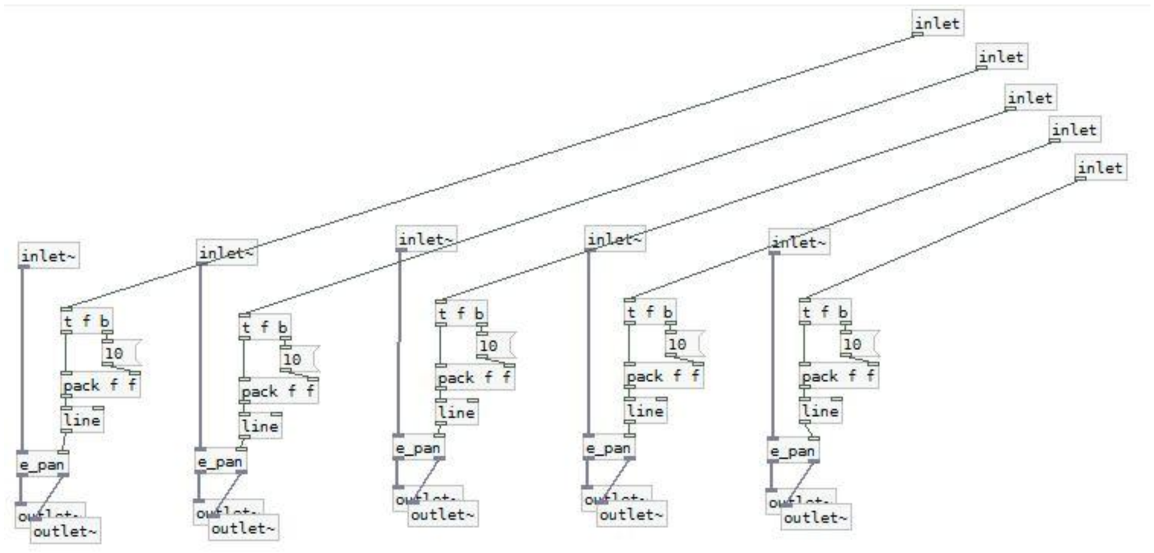
It is now easy to build a group of object that will be able to output a bang according to a probability set by the user. You just need to link a [random 100] to a [moses my_probability]. The example below illustrates this principle if you click on the bang on top you'll have a 20% chance that the second bang will light up, unless you have changed the number inside the numberbox (which I did) so now it's a 36% chance.



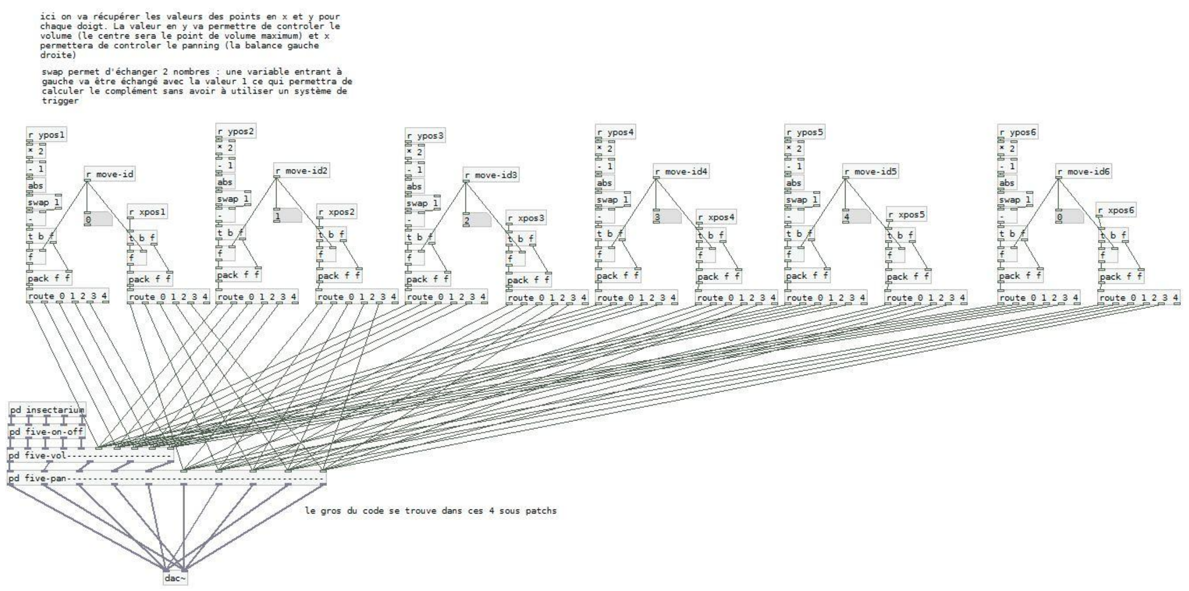
In the resource section of the first episode we already mentioned the patch Mengine.pd which is a generative music patch from which I extracted music loops. If you have a look at the code you can see that it uses this method a lot…

8.2.4- THE [E_PAN] ABSTRACTION

This abstraction come once again from the RJLib. It's a classic stereo panoramic effect : in the left inlet you can enter a monophonic signal that will be panned to a stereophonic signal using a variable in the right inlet 0 is for left and 1 is for right.



We use one abstraction for each object, it will receive values from the movements of the interface through the routing system illustrated right below.


ici on va récupérer les valeurs des points en x et y pour chaque doigt. La valeur en y va permettre de controler le volume (le centre sera le point de volume maximum) et x permettera de controler le panning (la balance gauche droite)

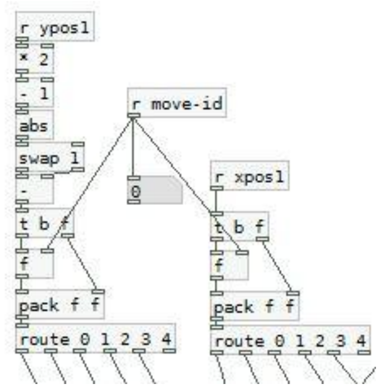
swap permet d'échanger 2 nombres : une variable entrant à gauche va être échangé avec la valeur 1 ce qui permettra de calculer le complément sans avoir à utiliser un système de trigger

le gros du code se trouve dans ces 4 sous patchs

Each movement parameter xpos1 and ypos1 (for instances) are routed to the right inlets using the move_id parameters. The movement on the x-axis will control stereo placement, and the y-axis will control the volume.

8.2.5- USING [SWAP] : COLD AND HOT INLETS

For the volume we already get a value from 0 to 1 from the [touch] abstraction, but it outputs a 0 when we are on top of the screen and a 1 at the bottom. What we want is to have the maximum volume (ie 1) in the middle of the screen and 0 on top and at the bottom. We will need to write some code to achieve this :



From (0,1) we multiply by 2 to get to (0,2) then we substract 1 to have a range of (-1,1). So after two objects we now have -1 on top and at the bottom, and 0 when we are in the middle of the screen. We can use [abs] to get the absolute value so now we have 1 on top and at the bottom , and still 0 in the middle. We are almost there... since we just have to calculate ones' complement of this value. We can use [swap 1] to achieve this. [swap] will exchange inputs and outputs the number in the left inlet will be output to second outlet and vice versa.

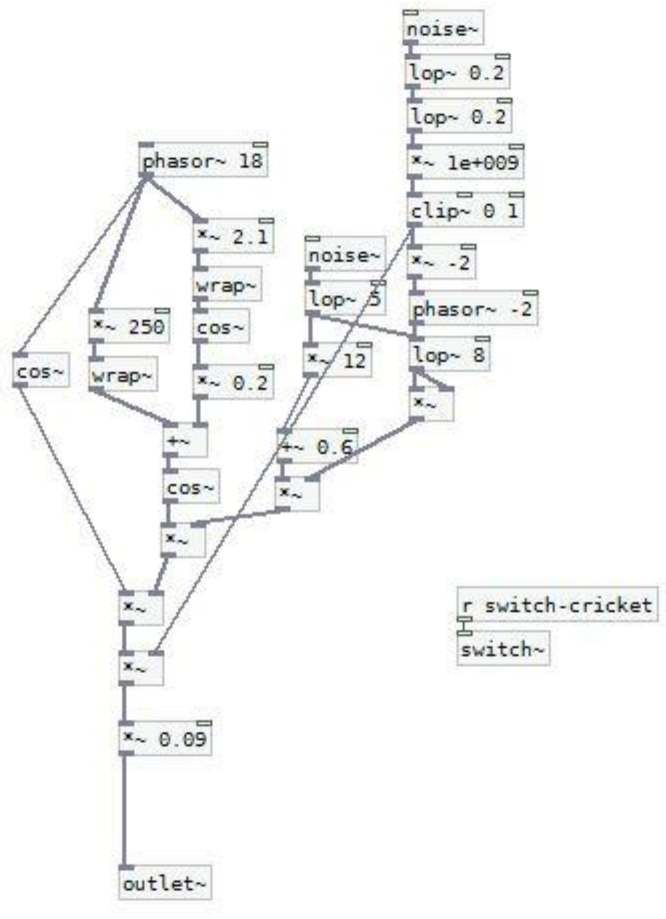
For instance if we receive 0.4 at the left inlet and the argument is 1 we will have the number 1 output at the left outlet and 0.4 at the right outlet. We just have to to link those two outlet to both inlet of a [-] object to get the complement.

We need to use [swap] to do this. This is because of what we call hot inlets and cold inlets. A hot inlet will re-evaluate the output each time a variable is throw at it (it's generally the first inlet on the left), a cold inlet will just store the value waiting for the output to be evaluated. We already use this with [f] : for instance we store a number using the right inlet but it's not passed to the output right away, it will ba only when a bang is received on the left inlet.

8.2.6- USING [SWITCH~]

[switch~] is used to stop DSP (Digital Signal Processing) calculations within a subpatch or an abstraction. It's usefull to save some CPU cycles when a subpatch is idle. For instance if we have a [noise~] object it will keep on calculating until we use [switch~] to stop it. You can use a basic audio gate to prevent sound from passing, but this will not stop calculations within [noise~] or others objects.

You only have to create the object in a subpatch and send it a 0 or a 1. Zero will stop calculations, one will enable them.



So we have discovered a few new concepts that will enable us to build more complex interfaces, creating movement, with several levers for interactivity. The use of [random] will be precious in the creation of generative systems.

8.3- RESSOURCES

As usual you can check out the help patch for [random], that will provide insights in the process of seeding and the way pseudo-random algorithms generate those values.

8.3.1- LINKS

The way we spatialised sound is very simple. For more information about spatialisation in Pure Data you can check this link. It's the files from a workshop taught by Georg Holzman (you'll need pd-extended though).

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

8.3.2- A FEW MORE PATCHES

Animals_Bird_Hansm.pd

The bird patch used in Spat-insect-droid this time with a nice desktop interface to tweak parameters.

Animals_Insects.pd

The full insectarium of Andy Farnell in a single small patch.

BirdsUseStar.pd

FM birds used in the RjDj patch, originally made by game.

Mengine-Ambiant-Std-Ed.zip

Mengine is a generative music engine using probabilities at every step of the process. You cand set presets with probabilities for each step of each sequencer to decide if it will play a note or not. Probabilities are also used to choose wich notes are played within a scale.

8.3.3- SOME SVG WIDGETS

Every graphic features of those application were created using inkscape which is a free open source software to create vector based graphics.

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

9-EPISODE 6 : APPLICATIONS

This will be our final episode. So we will create more complex applications. A small video game in which you'll play a spaceship travelling along galaxies and musical app that will be able to generate rhythm and melodies in a 'smart' way.

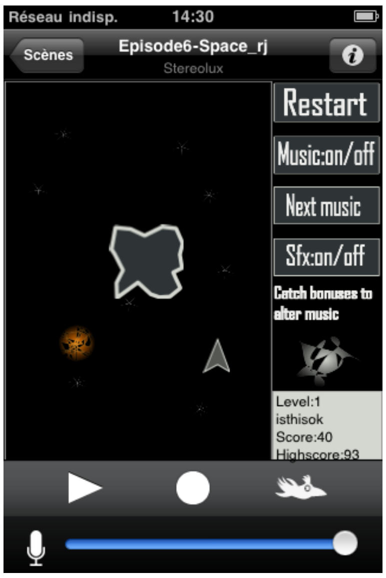
9.1- HOW DOES IT WORK ?

We will use most of the things we've discovered so far. But we will go further by learning how to use logical tests to detect collisions between two objects displayed on the screen, or using markov chains to analyse rhythm and melodies played on a keyboard to generate musical variations of a melody played.

9.1.1-SPACE-TRAVELLER.RJ

Space-traveller.rj is a little game for Rjdj. You are a spaceship drifting along, to score points you will need to destroy asteroids coming at you and catch bonuses. If you catch bonuses not only will it increase your score but it will also alter the music played in realtime by changing the notes played or changing the radomisation of events in the generation of notes. All the graphics are build with still graphics built with inkscape that moves around the screen and all the sound is achieved with real time synthesis.

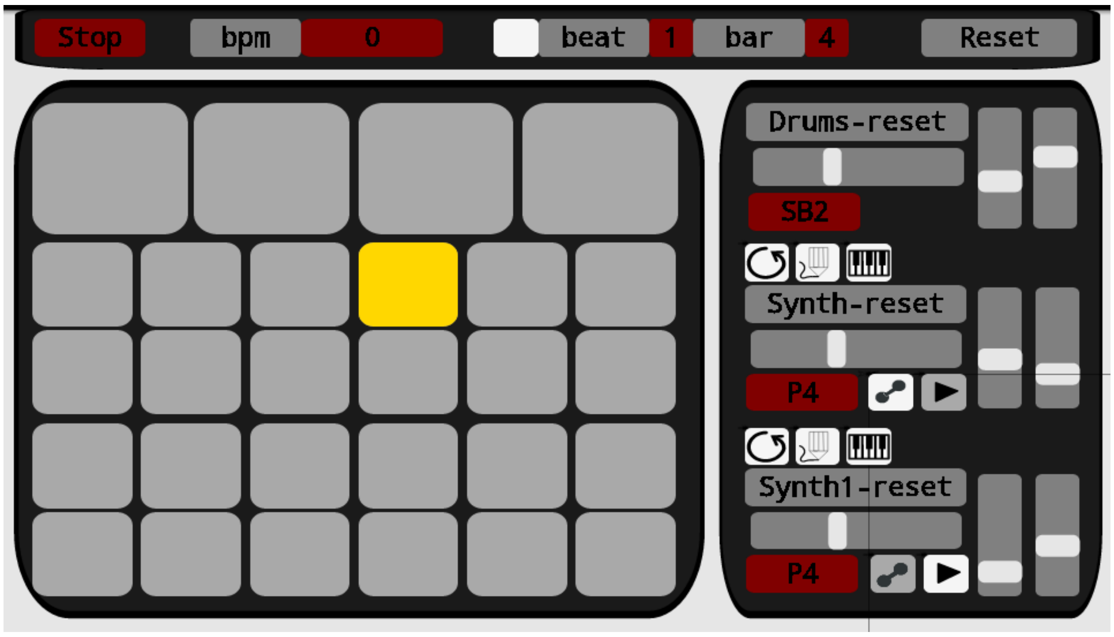
A simple collision engine has been implemented. This example is about pushing the limits of the rjdj framework as it was not created to build games. All the process are quiet heavy but fits somehow to the level of performance of first gen iphone. Some of the collision detection may seem a little buggy and sometimes the game slows down on this device.



Level:1
isthisok
Score:40
Highscore:93

9.1.2-MUTLI-TAP-DROID

This application is an extent of the Multi-tap.rj from the fourth episode. The new interface offers a lot more controls over the musical output, and introduces a few new features such as several way of playing with it and a nice visual feedback.



We have the possibility to change the tempo (at very high speed it can create nice textures). The screen can be split in three areas : the bar on top of the screen (switch the sequencer on/off, enter the speed, a audio/visual feedback of the metronome, and a button to reset everything, the left hand side of the screen (grey squares a line of four for 4 drum sounds, and 2*2 lines of six covering each a full octave for a both synths available) which is the playing area, and the right side of the screen which will enable us to control volumes/choose sounds/ choose the play mode and control effects.

The first thing is to choose a tempo, then start the sequencer. Once it's a done a sequencer will start running, you can then play by hitting the grey square in the left area. According the play modes selected this will produces different behaviours.

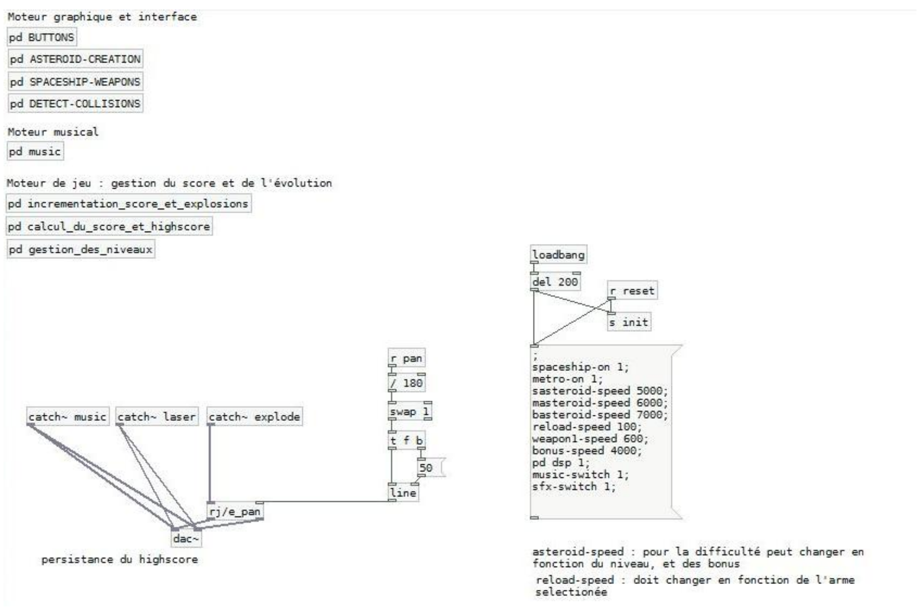
The piano mode will let you play as the programm was a piano, the pencil mode will write the patterns you tap into a table, and the loop mode will play those notes as a loop. Combining those modes will help you to build a melody loop it and improvise on it without modifying its content.

Two other modes are available : the training mode and the improvisation mode ▶. This will control the markov chain stuff. What it means is that we can generate new melodies based on probability tables created by the analysis of the user input.

The vertical sliders are for the effects, the right ones are reverbs, the left one is a bitcrusher for the drum section, for both other instruments it's a delay. Horizontal sliders control the volume of each instrument.

9.2- MAIN CONCEPTS

9.2.1.1- GRAPHIC DISPLAY, LEVELS AND SCORE. (SPACE-TRAVELLER)



```
Moteur graphique et interface
Moteur musical
Moteur de jeu : gestion du score et de l'évolution

;
spaceship-on 1;
metro-on 1;
sasteroid-speed 5000;
masteroid-speed 6000;
basteroid-speed 7000;
reload-speed 100;
weapon1-speed 600;
bonus-speed 4000;
pd dsp 1;
music-switch 1;
sfx-switch 1;

persistance du highscore

asteroid-speed : pour la difficulté peut changer en
fonction du niveau, et des bonus
 reload-speed : doit changer en fonction de l'arme
 selectionée
```

This is the patch's main window

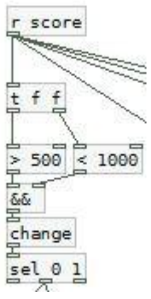
In this application score evolutions will be used to define what level we are ate and will thus control the number of asteroids and bonuses coming from the top of the screen. To do this we will use logical test to know at which level we are at, according to the level we will send a message that will set the probability for each graphic element (asteroids or bonuses) to be displayed.

We have seven types of graphics that can come down from the top of the screen :

- 3 types of staeroids (small, medium or big)
- 3 types of bonus (the red is called note1, green is note2, purple is note3)
- 1 random bonus (weird looking orange stuff)

The score is calculated as follow : a small asteroids gets you 10 points, un medium 20 points, and a big 30 points. If you catch a bonus you will score 50 points. The goal of the game is catch bonus to alter the music rather than destroying rocks...

So we had all those elements in a variable, and compare it to predefined boundaries. Each time the score changes we will receive it and compare it with the help of [<] and [>], if the score is in between those two values then we are at this level. Level 1 is between 0 and 500 and level 2 between 500 and 1000.



A logical test for the level 2 according to the current score

For each level as I mentioned before we will send a list of probabilities, one probability for each graphic elements we can display. A message like [15 30 45 55 65 75 100(means :

-15% chance of displaying a small asteroid

-15% chance of displaying a medium asteroid (30-15=15)

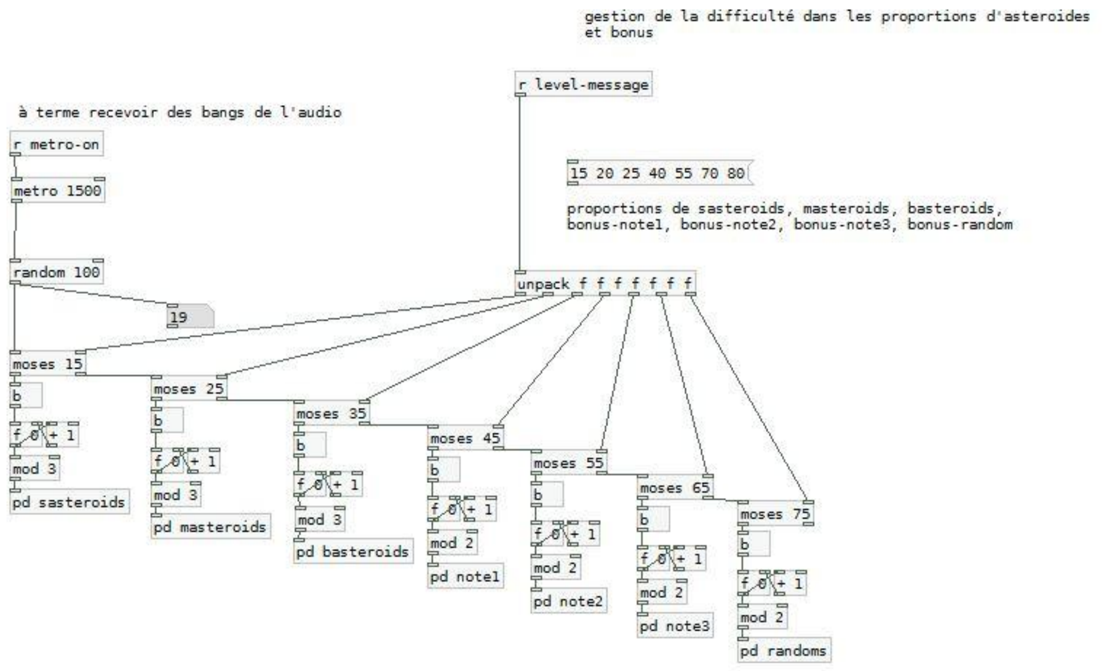
-15% chance of displaying a big asteroid (45-30=15)

-10% chance to have either a note1, note2 or note3 bonus

-25% chance to have a random bonus

This list will feed a chain of [moses] objects that will route a random generated number to bang the right abstraction to create a graphic. There is a hardcoded limitation of instance for one object : that is to say that you cannot create more than 3 representation of the same of object displayed on the screen.

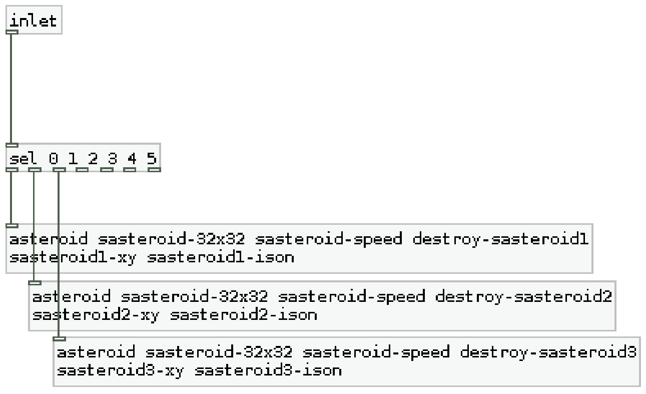
As we did in the last episode we will use an abstraction similar to [im_sim] to do that. It's called [asteroid] and it will enable us to display an image according to its name but also generate a stream of data for its movement according to random parameters, the movement data will be sent with a specific name.


gestion de la difficulté dans les proportions d'asteroides
et bonus

à terme recevoir des bangs de l'audio

proportions de sasteroids, masteroids, basteroids,
bonus-note1, bonus-note2, bonus-note3, bonus-random

Here is the chain of moses



```
inlet
sel 0 1 2 3 4 5
asteroid sasteroid-32x32 sasteroid-speed destroy-sasteroid1
sasteroid1-xy sasteroid1-ison
asteroid sasteroid-32x32 sasteroid-speed destroy-sasteroid2
sasteroid2-xy sasteroid2-ison
asteroid sasteroid-32x32 sasteroid-speed destroy-sasteroid3
sasteroid3-xy sasteroid3-ison
```

The subpatch [pd sasteroid]

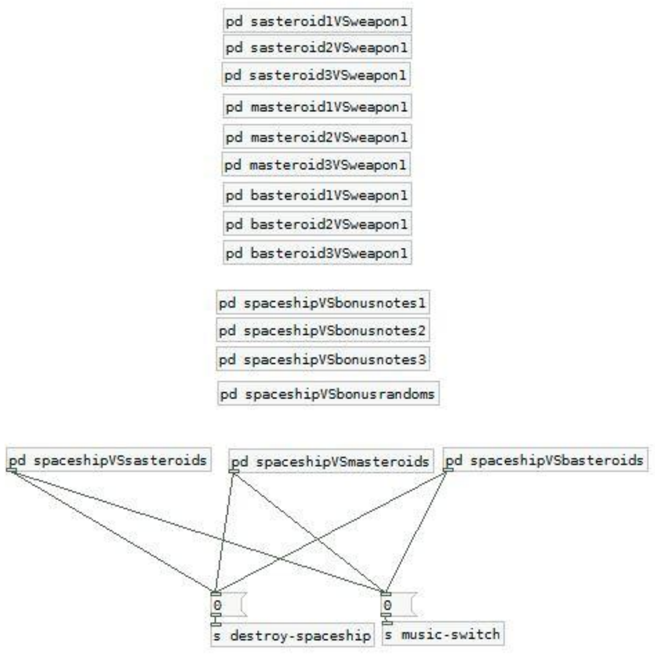
This subpatch will deal with the display of small asteroids, the other subptaches are built one the same model. Three [asteroid] abstraction are used, each of them uses five arguments for its creation : the name of the image to be loaded (sasteroid-32x32), the second is not used in this version but allows us to specify a speed parameters for the movements (I've replaced it with à random number between 1500 and 8500, that means that each time an small asteroid is created it will take it between 1500 and 8500 ms to travel from the top of the screen to the bottom), the third argument is a specific name to receive a bang when the asteroid is destroyed (destroy-sasteroid1), the fourth send movement data packed, the last one is a variable letting us know if the object is displayed or not (it's used for optimization).

9.2.2- LOGICAL TESTS : DESIGN OF A SIMPLE COLLISION ENGINE. (SPACE-TRAVELLER)

To build a collision engine, we need to be aware at each moment where all the objects are, and compare their coordinates if two objects are more or less in the same place (according to their size) then they collide. We will use [spigot] to help the program calculate only what's needed : if an object is not displayed there's no use to compare its coordinates to the other.

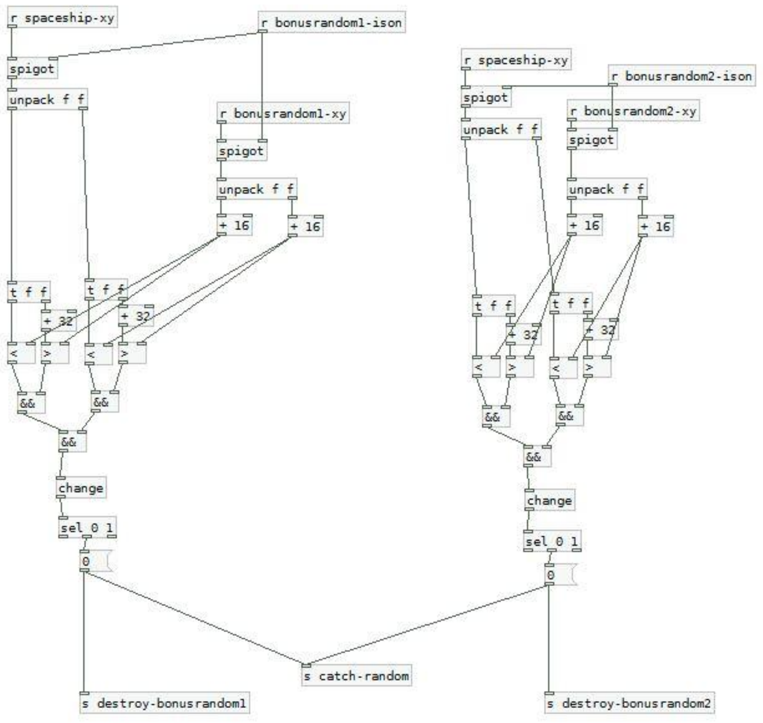
We need to check collision between several elements :

- Spaceship vs Asteroids : which destroy the spaceship and end the game.
- Spaceship vs Bonus : which will destroy the bonus, change the music and increase the score.
- Asteroids vs Weapons : which will destroy the asteroid, increase the score and generate a sound.



The full code for the collision engine

In the same way as for the score, we will compare with [<], [>] and [&&] every coordinates of every objects, by receiving messages from the [asteroid] abstractions. For instance to receive the coordinates of the spaceship we can do this by creating a [r spaceship-xy] object.

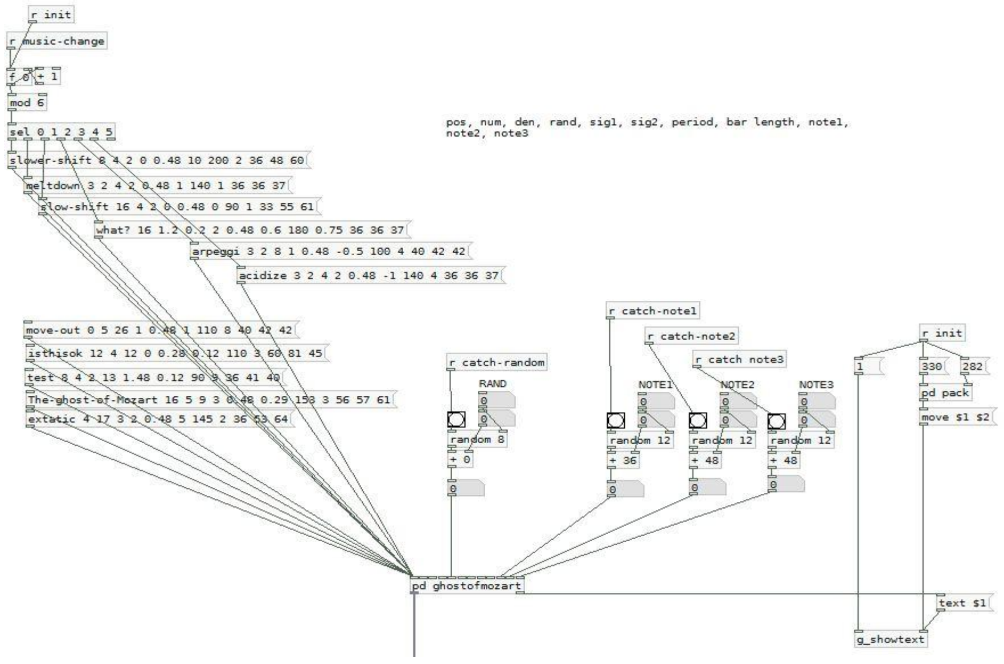


This is how we compare the spaceship vs the random bonus

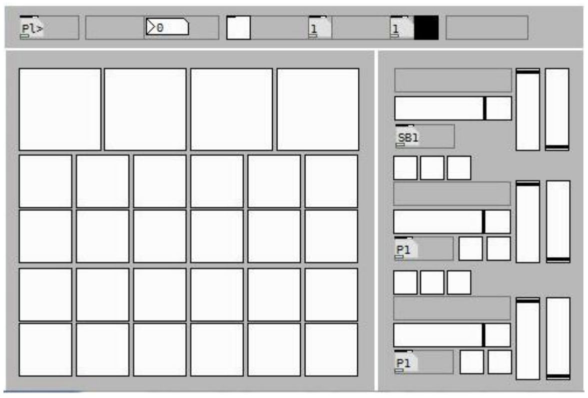
Every subpatch is built like this one. Notice the object [+ 36] after receiving position, this will help us to define the center of the object to state if there is a collision or not. The value '36' changes according to the size of the object created. In the same way after receiving the spaceship position we add 32 for some of the position. It is made this way to enlarge the area of collision instead of checking if to points are on the same place, we check if a point (center of the bonus) is inside a square (the spaceship icon).

9.2.3- THE MUSIC ENGINE : CHRONICLE OF THE USE AN RE-USE OF A PATCH

The music engine is based on a patch from Andy Farnell. You can seed an algorithm by giving it a list of parameters. I modified it a bit, creating new presets, boosting the output and choosing some parameters to be changed according to the game design. I encourage you to study it create your own presets. In the resources section you'll find the original version by Andy, a first modification I did for the patchwerk radio project aswell as this modification.


r init
r music-change
f 0
+ 1
mod 6
sel 0 1 2 3 4 5
slower-shift 8 4 2 0 0.48 10 200 2 36 48 60
meltdown 3 2 4 2 0.48 1 140 1 36 36 37
slow-shift 16 4 2 0 0.48 0 90 1 33 55 61
what? 16 1.2 0.2 2 0.48 0.6 180 0.75 36 36 37
arpeggi 3 2 8 1 0.48 -0.5 100 4 40 42 42
acidize 3 2 4 2 0.48 -1 140 4 36 36 37
pos, num, den, rand, sig1, sig2, period, bar length, note1, note2, note3
move-out 0 5 26 1 0.48 1 110 8 40 42 42
isthisok 12 4 12 0 0.28 0.12 110 3 60 81 45
test 8 4 2 13 1.48 0.12 90 9 36 41 40
The-ghost-of-Mozart 16 5 9 3 0.48 0.29 153 3 56 57 61
extatic 4 17 3 2 0.48 5 145 2 36 53 64
r catch-random
RAND
0
0
random 8
+ 0
0
r catch-note1
r catch-note2
r catch note3
NOTE1
NOTE2
NOTE3
random 12
random 12
random 12
+ 36
+ 48
+ 48
0
0
0
r init
1
330
282
pd pack
move $1 $2
pd ghostofmozart
text $1
g_showtext

9.2.4- MARKOV CHAINS (MUTLI-TAP)

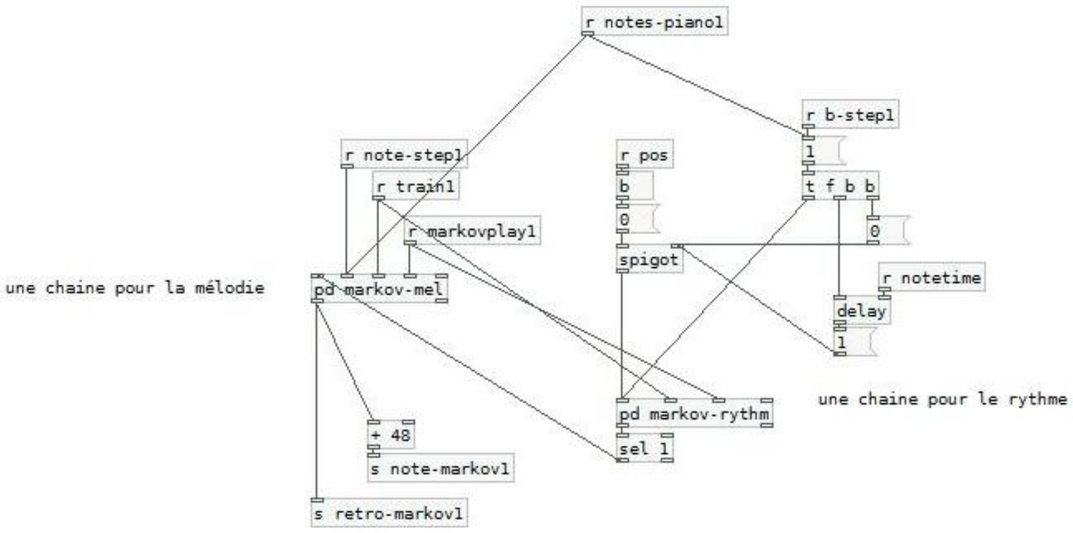


Multi-tap opened on a desktop version of pd

The training mode will record all the notes played by the synthesizer once the button is pressed, when released the program will perform markov analysis. That is to say that it will organize notes to know the probability that one note will be followed by another note. For instance after analysis if you input a C in a markov chain it will have 30% chance to output a D, 15% chances to output a B and 55% of another C. Analysing a melody with a markov chain will give the probability for each base note to be followed by any other, using this data we will then be able to generate musical variation of those melodies. It becomes really interesting when using a markov chain to analyse harmonicity and a second one for rhythm.

It can sound a little complicated, but we will use abstractions from the Rjlib [c_markov] et [m_markovanal]. The help patches are available so check it out for more information.

So we will need two implementations one for the rhythm and one for the melody. For the melody we will receive through the left inlet synchronization data (rythmics bangs to play a melody), then the second inlet will receive the notes played, the third and fourth inlets will turn on or off the “training” and “play” modes (0 or 1). Concerning the rhythm we will receive through the left inlet a 1 if a note is played a 0 if no note is played.

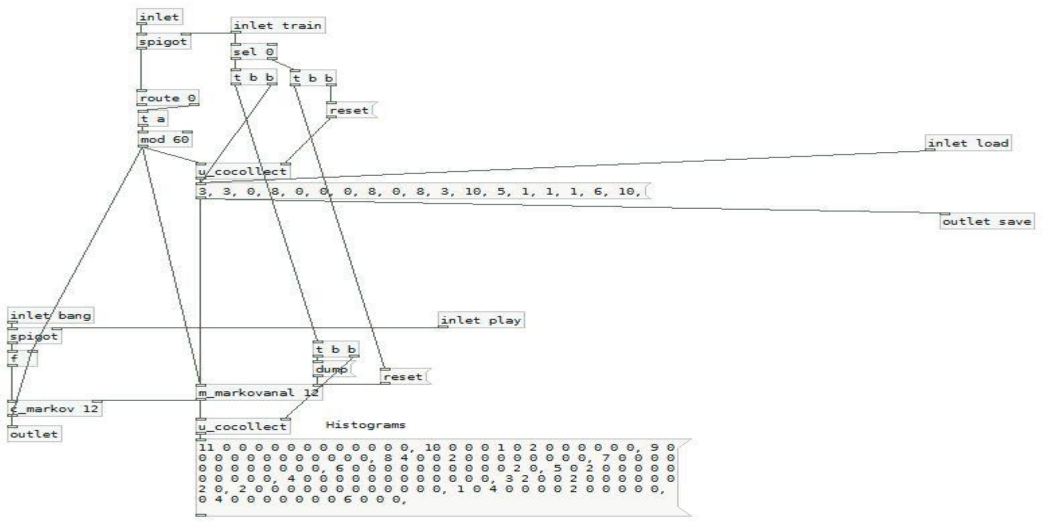


Both subpatches used for the markov chain implementation

[m_markovanal] will analyze a list of values and give us a histogram of the probabilities for each note in the list. In the training mode we will append all thoses notes in a message, once the training mode is off this list is dumped to [m_markovanal] that will output all the histograms ready to be used in the right inlet of [c_markov]. A [reset(message can be used to clear everything.

[c_markov] will help us to generate values according to those histogram we calculated. You just have to input a note in the left inlet and it will output a new note based on the probabilities. We will need to store the last note played to re-inject it at the hot inlet when we will need to calculate the next note.

We will use both objects two time for each synth : once for the rhythm and one for the harmony, that is to say four times in the whole patch.



```
3, 3, 0, 8, 0, 0, 0, 8, 0, 8, 3, 10, 5, 1, 1, 1, 6, 10,
```

```
11 0 0 0 0 0 0 0 0 0 0 0 0, 10 0 0 0 1 0 2 0 0 0 0 0 0, 9 0
0 0 0 0 0 0 0 0 0 0 0 0, 8 4 0 0 2 0 0 0 0 0 0 0 0, 7 0 0 0 0
0 0 0 0 0 0 0 0, 6 0 0 0 0 0 0 0 0 0 0 2 0, 5 0 2 0 0 0 0 0
0 0 0 0 0, 4 0 0 0 0 0 0 0 0 0 0 0 0, 3 2 0 0 2 0 0 0 0 0 0
2 0, 2 0 0 0 0 0 0 0 0 0 0 0 0, 1 0 4 0 0 0 0 2 0 0 0 0 0,
0 4 0 0 0 0 0 0 0 6 0 0 0,
```

The couple [m_markovanal] and [c_markov] together, with the list of notes and the histogram

9.3- RESSOURCES

9.3.1- LINKS

Markov chain on Wikipédia :

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

A full blog on algorithmique using pd :

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

9.3.2- FILES

- Andy's patch Ghost of Mozart

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

- Ghost of Mozart for patchwerk radio

- Pong.pd : a full game on desktop version of pd (there's a tetris and and arkenoid somewhere on the internet)

- svg files : all the graphics used in the applications.

9.3.3- OTHER APPLICATIONS

- autopop by Joe Newlin

http://joenewlin.net/pd.html

- Pure Data Latency Tester :

- Sweaty.rj : sonification of gestures as a rjdj scene, with an article

10-SOME RESSOURCES

Pure Data, and software made around Pure Data

http://puredata.info/community/projects/software : download Pd vanilla and Extended.

http://mccormick.cx/projects/PdDroidParty/ : download the last version of PdDroidParty.

Pure Data on smartphones, tutorials and forums :

http://blog.rjdj.me/pages/pd-utilities

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

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

Pure Data forums et tutorials

http://puredata.info/docs/workshops/ArtAndCodePdDemo (tuto)

http://www.obiwannabe.co.uk/ (tuto)

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

http://en.flossmanuals.net/pure-data/ (manual)

http://puredata.hurleur.com/index.php (english forum)

http://codelab.fr/pure-data (french forum, with a very detailed ressource page, as a sticky post)

Pure Data patches to discover:

- My big collection of abstraction, many from the web and some I did by myself : http://code.google.com/p/pdlive/

- Patchwerk radio, an online web radio playing Pure Data patches (patches are available for download) : http://radio.rumblesan.com/

Graphic creation

- Inkscape : create svg files to use with droidparty http://inkscape.org/?lang=fr

11- GLOSSARY

<u>Open-source</u>

Indicates that the source code of a software is publicly available, generally using a license which will grant certain rights to users. Several open-source license are available for instance :

Public license : no resitriction on the source code, which can be modified and even used for commerce.

BSD : the only restriction is to give credit the source code developers.

GPL : force users that modify and redistribute the source code to redistribute the source code themselves.

Proprietary license : some proprietary software can authorize the consultation of the code but prevent their use.

<u>Free software</u>

An Open-source software offers 4 crucial liberties to users. From wikipédia :

- Freedom 0: The freedom to run the program for any purpose.
- Freedom 1: The freedom to study how the program works, and change it to make it do what you wish.
- Freedom 2: The freedom to redistribute copies so you can help your neighbor.
- Freedom 3: The freedom to improve the program, and release your improvements (and modified versions in general) to the public, so that the whole community benefits.

<u>DIY</u>

DIY is a term used to describe building, modifying, or repairing of something without the aid of experts or professionals. In recent years, the term DIY has taken on a broader meaning that covers a wide range of skill sets. Central to the ethic is the empowerment of individuals and communities, encouraging the employment of alternative approaches when faced with bureaucratic or societal obstacles to achieving their objectives.

Rather than belittling or showing disdain for knowledge or expertise, DIY champions the average individual seeking knowledge and expertise for him/herself. Instead of using the services of others who have expertise, a DIY oriented person would seek out the knowledge for him/herself.

Open-hardware

Consists of physical artifacts of technology designed and offered in the same manner as free and open source software (FOSS). Open source hardware is part of the open source culture movement and applies a like concept to a variety of components. The term usually means thatinformation about the hardware is easily discerned.

Creative Commons

Creative Commons licenses are several copyright licenses that allow the distribution of copyrighted works. The licenses differ by several combinations that condition the terms of distribution. They were initially released on December 16, 2002 by Creative Commons, a non-profit corporation founded in 2001. Their licenses allow creators to communicate which rights they reserve, and which rights they waive for the benefit of recipients or other creators.

Creative Commons has been described as being at the forefront of the copyleft movement, which seeks to support the building of a richer public domain by providing an alternative to the automatic "all rights reserved"copyright, dubbed "some rights reserved." Beyond that, Creative Commons has provided "institutional, practical and legal support for individuals and groups wishing to experiment and communicate with culture more freely."